Localization and Mapping based on Semantic and Multi-Layer Maps Concepts

Por

André Silva Pinto de Aguiar

Orientador: José Boaventura Ribeiro da Cunha

Co-orientador: Filipe Baptista Neves dos Santos

Co-orientador: Armando Jorge Miranda de Sousa

Tese submetida à
UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO
para obtenção do grau de
DOUTOR
em Engenharia Electrotécnica e de Computadores, de acordo com o disposto no
DR – I série–A, Decreto-Lei n.º 74/2006 de 24 de Março e no
Regulamento de Estudos Pós-Graduados da UTAD
DR, 2.ª série – Deliberação n.º 2391/2007

Localization and Mapping based on Semantic and Multi-Layer Maps Concepts

Por

André Silva Pinto de Aguiar

Orientador: José Boaventura Ribeiro da Cunha

Co-orientador: Filipe Baptista Neves dos Santos

Co-orientador: Armando Jorge Miranda de Sousa

Tese submetida à

UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO

para obtenção do grau de

DOUTOR

em Engenharia Electrotécnica e de Computadores, de acordo com o disposto no

DR – I série–A, Decreto-Lei n.º 74/2006 de 24 de Março e no

Regulamento de Estudos Pós-Graduados da UTAD

DR, 2.ª série – Deliberação n.º 2391/2007

Orientação Científica :

José Boaventura Ribeiro da Cunha

Professor Associado com Agregação do
Departamento de Engenharias
Escola de Ciências e Tecnologia
da Universidade de Trás-os-Montes e Alto Douro

Filipe Baptista Neves dos Santos

Investigador Sénior do
Centro de Robótica Industrial e Sistemas Inteligentes
Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência (INESC TEC)

Armando Jorge Miranda de Sousa

Professor Auxiliar do
Departamento de Engenharia Eletrotécnica e de Computadores
Faculdade de Engenharia da Universidade do Porto

"The greatest challenge to any thinker is stating the problem in a way that will allow a solution."

Bertrand Russell

UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO
Doutoramento em Engenharia Electrotécnica e de Computadores

Os membros do Júri recomendam à Universidade de Trás-os-Montes e Alto Douro a aceitação da tese intitulada "**Localization and Mapping based on Semantic and Multi-Layer Maps Concepts**" realizada por **André Silva Pinto de Aguiar** para satisfação parcial dos requisitos do grau de **Doutor**.

fevereiro 2023

Presidente: **Vitor Manuel De Jesus Filipe**,
Professor Associado com Habilitação do
Departamento de Engenharias da Escola de Ciências e Tecnologia
da Universidade de Trás-os-Montes e Alto Douro

Vogais do Júri: **Professor Doutor Miguel Armando Riem de Oliveira**,
Professor Auxiliar do
Departamento de Engenharia Mecânica
da Universidade de Aveiro

Professor Doutor Carlos Baptista Cardeira,
Professor Auxiliar do
Departamento de Engenharia Mecânica
do Instituto Superior Técnico

Professor Doutor António Paulo Gomes Mendes Moreira,
Professor Associado do
Departamento de Departamento de Engenharia Eletrotécnica e de Computadores
da Faculdade de Engenharia da Universidade do Porto

José Boaventura Ribeiro da Cunha,
Professor Associado com Agregação do
Departamento de Engenharias
do Escola de Ciências e Tecnologia
da Universidade de Trás-os-Montes e Alto Douro

Filipe Baptista Neves dos Santos,
Investigador Sénior do
Centro de Robótica Industrial e Sistemas Inteligentes
do Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência (INESC TEC)

Professor Doutor António Luís Gomes Valente,
Professor Associado com Agregação do
Departamento de Engenharias da Escola de Ciências e Tecnologia
da Universidade de Trás-os-Montes e Alto Douro

,

Localization and Mapping based on Semantic and Multi-Layer Maps Concepts

André Silva Pinto de Aguiar

Submetido na Universidade de Trás-os-Montes e Alto Douro
para o preenchimento dos requisitos parciais para obtenção do grau de
Doutor em Engenharia Electrotécnica e de Computadores

Resumo

A Agenda 2030 para o Desenvolvimento Sustentável da ONU visa promover uma agricultura sustentável. A procura de recursos naturais e de alimentos é expectável que continue a crescer devido ao contínuo crescimento da população mundial. Por isso, existe a necessidade de tornar a agricultura mais eficiente e sustentável. A robótica pode desempenhar um papel fundamental na agricultura dado que os robôs podem adquirir informação em tempo real, desempenhar operações autonomamente, desencadear planos de ação em caso de pragas, entre outros. A agricultura impõe diversos desafios à robótica. Nomeadamente, a elevada extensão dos ambientes agrícolas, as irregularidades do terreno, a elevada simetria em culturas caracterizadas por corredores naturais, e as inclinações íngremes em culturas de montanha. Em termos de robótica para a agricultura, a perceção semântica e o mapeamento são conceitos essenciais. Através da semântica, os robôs podem aprender a atribuir significado aos dados, aprender a coexistir com humanos, e providenciar informação útil para o posterior uso humano. Para implementar sistemas de navegação autónoma nestas condições, os robôs devem ser capazes de operar totalmente autonomamente durante largos períodos de tempo. Por isso, algoritmos de Localização e Mapeamento Simultâneos devem ser capazes de ser executados em condições de operação de larga escala e longo termo. Um dos desafios é manter recursos de memória reduzidos durante o mapeamento de ambientes extensos. Este trabalho propõe uma nova solução chamada VineSLAM que permite que os robôs se localizarem em ambientes desafiantes enquanto criam diferentes tipos de mapas da cultura - métricos, semânticos, e topológicos. A primeira camada de

mapeamento (métrica) utiliza sensores *Light Detection And Ranging* para mapear pontos e polígonos correspondentes ao plano do chão e aos planos das canópias. A segunda camada de mapeamento (semântica) utiliza deteção de objetos baseada em *Deep Learning* para detetar elementos naturais em diferentes fases de crescimento, e mapeia-os. Nesta tese, testou-se este método de mapeamento através da deteção de troncos de videira e cachos de uvas. Finalmente, a terceira camada de mapeamento (topológica) possibilita aos robôs trabalhar continuamente na cultura sem restrições temporais ou de escala, através de um mapa topológico baseado num grafo. Para além desta arquitetura de mapeamento, o VineSLAM providencia um algoritmo de localização baseado num Filtro de Partículas que é capaz de utilizar diferentes tipos de características do ambiente e mapas para calcular os seis graus de liberdade da pose do robô. A utilização de diferentes sensores em conjunto para localização e mapeamento é possível através de um algoritmo de calibração extrínseca LiDAR-para-câmera proposto nesta tese. Como uma ferramenta, o VineSLAM abre a porta para robôs operarem autonomamente em condições agrícolas desafiantes através de um algoritmo de localização robusto combinado com um algoritmo de mapeamento em multicamada que providencia uma representação rica e variada da cultura. Em comparação com algoritmos do estado de arte, o VineSLAM apresenta uma performance genérica melhor, dado que consegue localizar o robô com uma precisão semelhante ou superior, e consegue operar durante periodos de tempo ilimitados. Os resultados demonstram que, em termos de perceção semântica, obteve-se uma precisão média na deteção de troncos de 84.16% e de 44.93% na deteção de cachos de uvas. O algoritmo de localização e mapeamento foi testado em diversos ambientes, tendo-se obtido, por exemplo, um erro absolute de pose de 0.9 metros numa sequência realizada numa vinha de montanha com 337.1 metros com uma variação de altitude de 7 metros. Foi demonstrado que esta abordagem permite a navegação autónoma sem restrições de tempo ou de escala, tendo sido o algoritmo testado em robôs agrícolas em diversos ambientes reais, permitindo a sua navegação de forma autónoma.

Palavras Chave: SLAM, Semântica, Topologia, Robôs Agrícolas.

Localization and Mapping based on Semantic and Multi-Layer Maps Concepts

André Silva Pinto de Aguiar

Submitted to the University of Trás-os-Montes and Alto Douro
in partial fulfillment of the requirements for the degree of
Philosophiae Doctor in Electrical Engineering and Computers

Abstract

The ONU's 2030 Agenda for Sustainable Development aims to promote sustainable agriculture. The demand for natural resources such as food is expected to continue to grow due to the continuous increase of the world's population. Thus, there is the need to make agriculture more efficient and sustainable. Robotics can play a key role in agriculture since robots can collect real-time information, perform autonomous operations, trigger action plans in case of diseases, among others. Agriculture imposes several challenges to agriculture. Namely, the high extension of the crops, the terrain irregularities, the high symmetry in crops characterized by natural corridors, and harsh inclinations in mountain crops. In this environments, semantic perception and mapping are essential concepts. With semantics, robots can learn how to assign meaning to data, learn to coexist with humans and provide useful information for further human use. To implement autonomous navigation systems in these conditions, robots should be able to operate fully autonomously during large periods.Thus, Simultaneous Localization and Mapping algorithms should be able to work in large-scale and long-term operating conditions. One of the main challenges is maintaining low memory resources while mapping extensive environments. This work proposes a novel solution called VineSLAM that allows robots to localize themselves in challenging environments while creating different types of maps of the crop - metric, semantic, and topological. The first mapping layer (metric) uses Light Detection And Ranging sensors to map feature points and polygon-based features such as the ground plane or the canopy walls. The second mapping layer (semantic) uses Deep Learning-based object detection models to detect natural elements in

different growth stages in images, and maps them. In this thesis, this semantic perception system was tested through the detection of vine trunks and grapes bunches. Finally, the third mapping layer (topological) enables robots to work continuously in the crop without time or scale restrictions through a graph-based topological map. On top of the entire mapping architecture, VineSLAM provides a localization algorithm based on a modular Particle Filter that is able to use the different sources of features and maps to compute the six degrees of freedom of the robot's pose. The usage of different sensors together for localization and mapping is possible through a LiDAR-to-camera extrinsic calibration algorithm proposed in this thesis. As a framework, VineSLAM opens the door for robots to operate autonomously in challenging agricultural conditions through a robust localization algorithm combined with a multi-layer mapping approach that provides a rich and varied representation of the crop. Compared with state-of-the-art algorithms, VineSLAM presents an overall better performance since it can localize the robot with similar or higher precision, and can operate for unlimited time frames. Results show that in terms of semantic perception, was obtained an average precision of 84.16% for trunk detection and 44.93% for grape bunch detection. The localization and mapping algorithm was tested in multiple environments, demonstrating, for example, an absolute pose error of 0.9 meters in a sequence placed in a mountain vineyard with 337.1 meters long and an altitude differential of 7 meters. It was shown that this formulation allows the autonomous navigation without time or scale restrictions. The algorithm was tested in agricultural robots in different real environments, allowing them to move autonomously.

Keywords: SLAM, Semantics, Topological, Agricultural Robots

Graphical Abstract

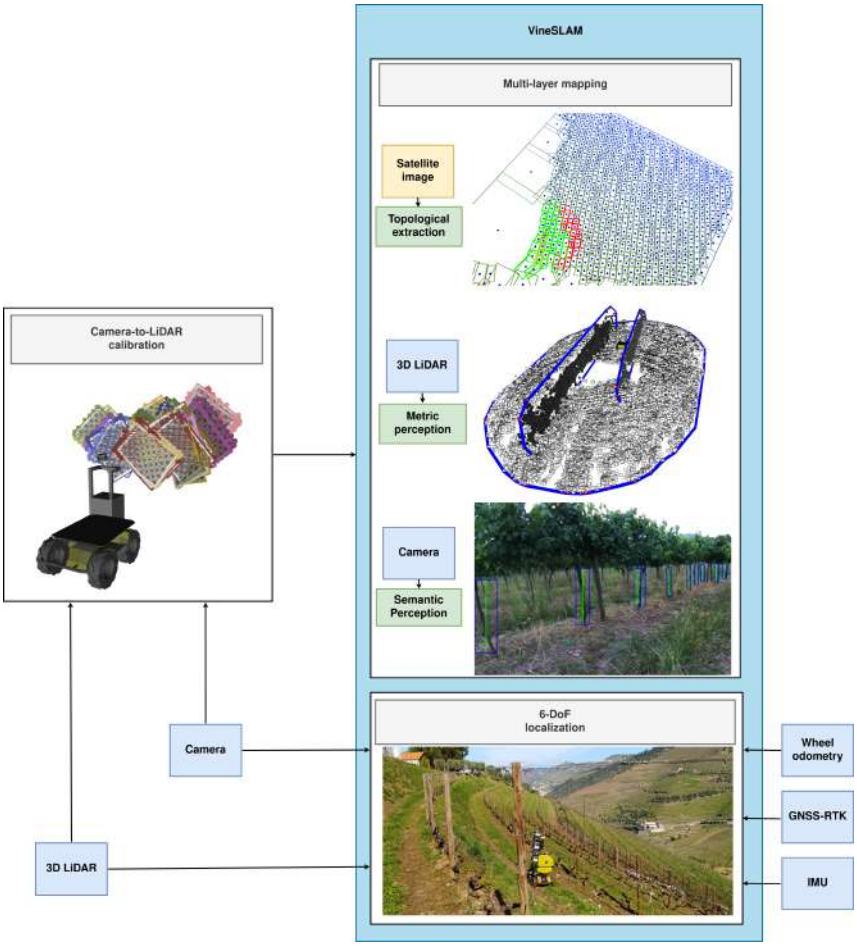


Figure 1 – Graphical abstract of the systems proposed in this thesis and their relationship.

Acknowledgments

UTAD, André Silva Pinto de Aguiar
Vila Real, 01/10/2022
I would like to thank to every person that guided and helped me during this journey.
To my supervisor Professor José Boaventura and co-supervisor Armando Sousa for their support and supervision. To my co-supervisor Filipe Neves dos Santos for his close guidance during the entire thesis. His friendship, knowledge and vision were crucial for the success of this work.
To my co-workers Luis Santos, Héber Sobreira, and Tatiana Pinho and my colleagues Miguel Riem Oliveira, Eurico Pedrosa, João Valente, Luis Castro, and Rui Martins that contributed to this thesis.
To all my remaining co-workers André Baltazar, Pedro Moura, Sandro Magalhães, José Sarmento, Daniel Silva, Francisco Terra, Rui Coutinho, André Bianchi, Domingos Bento, Vitor Tinoco, and Alexandro Magno.
To my family for all the love, support, and guidance.
To my fiancée and best friend for always believing in my dreams and encouraging me in the difficult moments.
Thank you.

I would like to acknowledge the financing of this work by the ERDF - European Regional Development Fund, through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme under the Portugal 2020 Partnership Agreement, within project ROMOVI, with reference POCI-01-0247-FEDER-017945. This work has also received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101004085. Finally, I would like to thank the FCT for the PhD. Grant DFA/BD/5318/2020.

Contents

List of figures

Glossary, acronyms and abbreviations

Acronyms list

Acronym	Expansion
CNN	*Convolutional Neural Network*
CPU	*Central Processing Unit*
DL	*Deep Learning*
DoF	*Degrees of Freedom*
EKF	*Extended Kalman Filter*
GNSS	*Global Navigation Satellite System*
GPS	*Global Positioning System*
GPU	*Graphical Processing Unit*
IMU	*Inertial Measurement Units*
LiDAR	*Light Detection And Ranging*
PF	*Particle Filter*

Acronym	**Expansion**
RANSAC	*Random Sample Consensus*
RTK	*Real-time Kinematic*
SLAM	*Simultaneous Localization and Mapping*
TL	*Transfer Learning*
TPU	*Tensor Processing Unit*
UNESCO	*United Nations Educational, Scientific and Cultural Organization*
USB	*Universal Serial Bus*
YOLO	*You Only Look Once*

1 Introduction

1.1 Context and motivation

The strategic research agenda for robotics in Europe 2014-2020 (Robotics, 2014) emphasizes the importance of developing robotic solutions for agriculture. The automation of agricultural processes is expected to have a positive impact on the environment by reducing waste and increasing food security, maximizing the utilization of resources (Duckett et al., 2018; Billingsley et al., 2008). Additionally, robotics will highly decrease the intensive human labor that is still characteristic of the agricultural sector.

Agriculture presents several challenges to the implementation of robotics. One particular challenging scenario are the Oporto vineyards located in the Douro Demarched Region, the oldest controlled winemaking region in the world, a United Nations Educational, Scientific and Cultural Organization (UNESCO) heritage place (Andresen et al., 2004). These vineyards are built in steep slope hills, which brings several challenges to the development of robotic solutions in this context. The hill's characteristics cause a signal blockage that decreases the accuracy of signals emitted by the Global Navigation Satellite System (GNSS), making unreliable the

(a) AgRob robot

(b) Weta robot

(c) Modular-E robot

Figure 1.1 – Autonomous robotic platforms used in the scope of this work.

use of, for example, the Global Positioning System (GPS). Also, the terrain highly characterized by irregularities leads to a decrease in accuracy of sensors like wheel odometry and Inertial Measurement Units (IMU)s. These vineyards are an example of the harsh conditions that are usually present in agriculture, which open intense, interesting and challenging research topics for robotics. This work formulates a solution that is general enough to work in different types of environments, and focus on testing and deploying them in agricultural robots to implement autonomous navigation in agriculture.

Given the above, the development of autonomous robots can revolutionize the agricultural sector, but faces many challenges. With them, precision agriculture can be implemented with the automation of several tasks such as application of fertilizers, nutrients and water, harvesting, monitoring and planting. The main motivation of this work is the belief that ground robots can be efficiently automatized to perform precision agriculture tasks even in harsh conditions. Autonomous robots

can improve the efficiency of the agricultural sector, improve the quality of human-labor in this area and perform many operations that are increasingly unwanted by humans.

1.2 Thesis statement and goals

Three main components are required for implementing autonomous robots: localization and mapping, path planning, and motion control (Fahimi, 2009). This work focuses on the localization and mapping components. The statement that this thesis defends is: **it is possible to localize a robot and simultaneously map the environment in long-term and large-scale considering metric, semantic, and topological information by the fusion of different types of sensors.**

This work makes use of the vast knowledge acquired during the development of the SLAM (Durrant-Whyte and Bailey, 2006; Bailey and Durrant-Whyte, 2006) concept over the years and extends it to the agricultural area. The main goals aimed in this thesis are:

- **G1**: An approach for automatic extrinsic calibration method to calibrate the main sensors used in the proposed localization and mapping algorithms - cameras and 3D LiDARs.
- **G2**: An approach for a perception system that can recognize and detect semantic elements in agricultural environments.
- **G3**: An approach for a 6-DoF localization approach based on 3D metric information considering the geometry of the environment.
- **G4**: An approach for semantic mapping to create maps of agricultural environments with meaningful information in different growth stages.

- **G5**: An approach for a multi-layer mapping algorithm that is suitable for long-term operations and large-scale environments without time and scale limitations.

This thesis presents a novel system that pushes forward the implementation of robotics in agriculture. In particular, the main focus is placed on the exploration of a robust 3D localization system that is interdependent of a multi-layer mapping architecture. The 6-DoF of the robot's pose are estimated due to the harsh inclinations, terrain irregularities, and differentials of altitude that characterize agricultural environments. Also, in this work we consider that the 3D mapping should be decomposed in three layers: metric, semantic and topological. The mapping should find different ways of representing the environments and dealing with the typical large dimension of agricultural crops working without time and scale limitations. The environments' dimension is a problem that triggers memory issues on SLAM systems since, if they are not optimized, processors will not have enough resources to allocate all the necessary memory to consider the entire environment. The formulated approach proposes an efficient way to deal with this issue, adding a mapping layer - topological - to manage the memory resources. It is worth noting that this multi-layer architecture is modular, in the sense that the proposed approach can be used without the memory management mapping layer and operate only using the feature-based layers.

1.3 Scientific contributions

The work developed and presented in this thesis was applied in the following R&D European projects: ROMOVI, SCORPION and NOVATERRA. The close relation with researchers working on agricultural robotics contributed to the achieved scientific contributions. During this work the following scientific contributions were accomplished:

- Creation of a camera-to-LiDAR extrinsic calibration approach that is fully

general on the amount of sensors to calibrate retrieving a precise homogeneous transformation between all the sensors considered in the calibration process;

- A Deep Learning- (DL)-based semantic perception system to detect vine trunks and grape bunches through the creation dataset of vineyard images with grape bunches in different growth stages and in different stages of the year. The dataset was made available for the community and includes the image annotations.
- Creation of a SLAM algorithm for agricultural robots that comprises:
  - A Particle Filter- (PF)-based localization and mapping algorithm based on point feature extraction and semiplane segmentation (polygon-based ground plane and vegetation walls) from 3D LiDAR data;
  - A clustering procedure to refine the PF estimation using stereo camera systems;
  - A topological mapping approach that is responsible for managing the memory resources and allowing the algorithm to run without time and scale restrictions;
  - A semantic mapping pipeline that is able to map vine trunks and grape bunches in different growth stages using a camera-LiDAR fusion technique.

Resultant from the listed contributions, the following 9 journal articles were published:

- Aguiar, A. S., dos Santos, F. N., Cunha, J. B., Sobreira, H. and Sousa, A. J. (2020) Localization and mapping for robots in agriculture and forestry: A survey. Robotics, 9. URL: https://www.mdpi.com/2218-6581/9/4/97.
- Aguiar, A., M. Oliveira, E. Pedrosa, and F. Santos, A Camera to LiDAR calibration approach through the Optimization of Atomic Transformations, Expert Systems with Applications (2021) p. 114894, ISSN: 0957-4174, DOI: https://doi.org/10.1016/j.eswa.2021.114894,2021

- A. S. Pinto de Aguiar, F. B. Neves dos Santos, L. C. Feliz dos Santos, V. M. de Jesus Filipe, and A. J. Miranda de Sousa, "Vineyard trunk detection using deep learning – an experimental device benchmark," Computers and Electronics in Agriculture,vol. 175, Aug. 2020, doi:10.1016/j.compag.2020.105535.
- Aguiar, A. S., Santos, F. N. D., De Sousa, A. J. M., Oliveira, P. M. and Santos, L. C. (2020) Visual trunk detection using transfer learning and a deep learning-based coprocessor. IEEE Access, 8, 77308–77320.
- Aguiar, A. S., Monteiro, N. N., Santos, F. N. d., Solteiro Pires, E. J., Silva, D., Sousa, A. J. and Boaventura-Cunha, J. (2021c) Bringing semantics to the vineyard: An approach on deep learning-based vine trunk detection. Agriculture, 11. URL: https://www.mdpi.com/2077-0472/11/2/131.
- Aguiar, A. S., Magalhães, S. A., dos Santos, F. N., Castro, L., Pinho, T., Valente, J., Martins, R. and Boaventura-Cunha, J. (2021b) Grape bunch detection at different growth stages using deep learning quantized models. Agronomy, 11. URL: https://www.mdpi.com/2073-4395/11/9/1890.
- Aguiar AS, Neves Dos Santos F, Sobreira H, Boaventura-Cunha J, Sousa AJ. Localization and Mapping on Agriculture Based on Point-Feature Extraction and Semiplanes Segmentation From 3D LiDAR Data. Front Robot AI. 2022 Jan 28;9:832165. doi: 10.3389/frobt.2022.832165. PMID: 35155589; PMCID: PMC8831384.
- Aguiar, A. S., dos Santos, F. N., Sobreira, H., Cunha, J. B. and Sousa, A. J. (2021a) Particle filter refinement based on clustering procedures for high-dimensional localization and mapping systems. Robotics and Autonomous Systems, 137, 103725. URL: https://www.sciencedirect.com/science/article/pii/S0921889021000105.
- Aguiar, A.S., dos Santos, F.N., Santos, L.C., Sousa, A.J. & Boaventura-Cunha, J. (2022) Topological map-based approach for localization and mapping memory optimization. Journal of Field Robotics, 1– 20. https://doi.org/10.1002/rob.22140.

Additionally, the following manuscript was submitted and is being peer reviewed at the moment:

- Semantic Mapping of Grape Bunches and Stems using Sensor Fusion and a Robust Localization Algorithm [submitted to the Journal of Robotics and Autonomous Systems at 19-08-2022].

Finally, this thesis produced:

- An open-source repository[1] containing the VineSLAM C++ implementation.
- A publicly available dataset[2] for DL object detection model training containing vineyard images with trunk and grape bunch annotations in different growth stages. This dataset, called VineSet, was recognized by the ROS Agriculture community[3] as "A Large Vine Trunk Image Collection and Annotation using the Pascal VOC format".
- A set of publicly available datasets[4] in the ROSBag2 format containing navigation data (cameras, LiDARs, IMUs, RTK-GNSS, odometry) of agricultural robots in vineyards and orchards.

1.4 Thesis organization

This document contains six chapters. Chapters 2, 3, 3, 4 and 5 and their corresponding sections present a small introduction followed by one or multiple articles. In summary, the document is structured as follows:

Chapter 1: Presents the introduction and is composed of the context and motivation of the work, the thesis statement and its goal, the scientific contributions made during the developed work and the thesis organization.

Chapter 2: Explores the state-of-the art presenting a survey on localization and mapping algorithms for robots in agriculture and forestry. Besides performing an extensive review of the literature, this section also approaches fundamental concepts of SLAM.

Chapter 3: Approaches the extrinsic calibration method for cameras and 3D LiDARs. This method appears as the first implementation chapter since it is a basis for the simultaneous use of cameras and LiDARs in the remaining chapters. Thus, the calibration procedure is of extreme importance for the proper fusion of

[1] https://gitlab.inesctec.pt/agrob/vineslam_stack/vineslam
[2] https://zenodo.org/record/5362354
[3] http://wiki.ros.org/agriculture
[4] https://zenodo.org/communities/scorpion-h2020/

both modalities of sensors and, consequently, for the localization and multi-layer mapping steps.

Chapter 4: Describes the research around the created semantic perception system for vineyards. This chapter is composed of four different papers that tackle the use a coprocessor for performing the DL-based object detection; the experimental benchmark between two devices to perform vine trunk detection; and a DL-based approach to detect vine trunks and grape bunches at different growth stages. In this chapter it is possible to observe an evolution on the semantic perception system with the constant increase of the dataset size, the improvement of the detection precision and recall, and the consideration of a temporal dimension on the detection using images of the crop in different stages.

Chapter 5: Presents VineSLAM, the localization and mapping algorithm for agricultural robots, in four different papers (two of them still in peer reviewing process). This chapter describes all the contributions made in terms of 6-DoF localization and multi-layer mapping, providing three papers describing each mapping layer (metric, semantic and topological), and another describing a refinement of the PF-based localization algorithm using clustering concepts.

Chapter 6: Provides the main conclusions obtained from this work and research topics for possible exploration in future work.

2 Related work

This chapter presents a general overview of the current state-of-the-art of localization and mapping algorithms applied to robots that operate in agricultural and forestry environments. In this scope, this chapter is composed of a survey paper entitled *Localization and Mapping for Robots in Agriculture and Forestry: A survey* (Aguiar et al., 2020a).

2.1 Localization and mapping for robots in agriculture and forestry: A survey

The article presented in this section was published in the special issue Advances in Agriculture and Forest Robotics[1] of the MDPI Robotics journal. The main goal of this work is to achieve a localization and mapping approach that can estimate efficiently the robot's 6-DoF pose while considering multiple types of features and maps. Thus, this paper presents a brief overview of the localization and mapping problem and the common approaches to solve it. Then, it explores the mapping strategies presented in the literature focusing on metric, topological, and semantic maps. To review the works that approach these issues, the article looks for the

[1] https://www.mdpi.com/journal/robotics/special_issues/agriculture_forest_robotics

following criteria: application, localization approach, mapping approach, accuracy, scalability, and availability. These topics help to characterize the applicability of SLAM approaches to agricultural and forestry environments and were taken in consideration during the development of this thesis.

To achieve a localization and mapping algorithm that uses multiple types of sensors, and that considers metric, semantic and topological information, a more in-depth literature review was made during the work on each topic. The remaining articles presented in this thesis contain their own target literature review.

robotics

MDPI

Review

Localization and Mapping for Robots in Agriculture and Forestry: A Survey

André Silva Aguiar [1,2,*], Filipe Neves dos Santos [1], José Boaventura Cunha [1,2], Héber Sobreira [1] and Armando Jorge Sousa [1,3]

[1] INESC TEC—INESC Technology and Science, 4200-465 Porto, Portugal; filipe.n.santos@inesctec.pt (F.N.d.S.); jboavent@utad.pt (J.B.C.); heber.m.sobreira@inesctec.pt (H.S.); asousa@fe.up.pt (A.J.S.)
[2] School of Science and Technology, University of Trás-os-Montes e Alto Douro, 5000-801 Vila Real, Portugal
[3] Faculty of Engineering, University of Porto, 4200-465 Porto, Portugal
* Correspondence: andre.s.aguiar@inesctec.pt

Received: 8 October 2020; Accepted: 18 November 2020; Published: 21 November 2020

Abstract: Research and development of autonomous mobile robotic solutions that can perform several active agricultural tasks (pruning, harvesting, mowing) have been growing. Robots are now used for a variety of tasks such as planting, harvesting, environmental monitoring, supply of water and nutrients, and others. To do so, robots need to be able to perform online localization and, if desired, mapping. The most used approach for localization in agricultural applications is based in standalone Global Navigation Satellite System-based systems. However, in many agricultural and forest environments, satellite signals are unavailable or inaccurate, which leads to the need of advanced solutions independent from these signals. Approaches like simultaneous localization and mapping and visual odometry are the most promising solutions to increase localization reliability and availability. This work leads to the main conclusion that, few methods can achieve simultaneously the desired goals of scalability, availability, and accuracy, due to the challenges imposed by these harsh environments. In the near future, novel contributions to this field are expected that will help one to achieve the desired goals, with the development of more advanced techniques, based on 3D localization, and semantic and topological mapping. In this context, this work proposes an analysis of the current state-of-the-art of localization and mapping approaches in agriculture and forest environments. Additionally, an overview about the available datasets to develop and test these approaches is performed. Finally, a critical analysis of this research field is done, with the characterization of the literature using a variety of metrics.

Keywords: localization and mapping; autonomous navigation; agriculture; forestry

1. Introduction

There have been several developments in the research and applications of robotic solutions for the agriculture sector and novel contributions in the near future are expected [1,2]. The need for automatic machines in this area is increasing since farmers increasingly recognize its impact in agriculture [3]. Robots are now used for a variety of tasks such as planting, harvesting, environmental monitoring, supply of water and nutrients, and others [4]. In this context, developing solutions that allow robots to navigate safely in these environments is essential. These advances impose a main requirement: localizing the robots in these agriculture and forestry environments. The most common solution is to use Global Navigation Satellite System (GNSS) standalone-based solutions [5,6]. However, in many agricultural and forestry places, satellite signals suffer from signal blockage and multi-reflection [7,8], making the use of GNSS unreliable. In this context, it is extremely important to research and develop intelligent solutions that use different modalities of sensors, as well as different sources of input

data, to compute the robot localization. Simultaneous localization and mapping (SLAM) [9,10] is the state-of-the art approach to do so. This technique consists of estimating the state of a robot using input sensor data, while simultaneously building a map of the surrounding environment [11]. The robot model usually comprises its pose, and, in some cases, its velocity, calibration parameters, sensor offsets, among others. The map is a multi-dimensional representation of the agents observed by the on-board robot sensors, that are used as references in the localization procedure. The map creation is usually important to provide information about the environment. In agriculture and forestry, they can be used by human operators to report information about the cultures. Furthermore, maps can be saved and loaded, being reused and updated by the robotic platforms each time they operate in the terrain. When mapping the environment is not desired, alternatives to SLAM are also approached. One of the most common is visual odometry (VO) [12]. As stated by Scaramuzza et al. [13], VO is the process of estimating the motion of the on-board camera (s) using only image data as input.

Given all of the above, one can conclude that there is a huge dependency on GNSS-free localization systems from autonomous mobile robots working in agriculture and forestry. So, this leads to the main question: is robotics localization a solved topic? The answer to this question depends on many aspects: the context where the robot operates, the quality of the on-board sensors, and, the desire performance of the localization system. For example, 2D LiDAR-based SLAM in indoor environments is a mature research field, with many high-quality state-of-art methods [14,15] reporting high-performance results. On the other hand, SLAM in harsh outdoor environments (as agriculture and forestry) is still a growing research topic. These are highly dynamic environments that change drastically over the year, which makes long term mapping a difficult task. To overcome this, the concept of 4D mapping, i.e., spatio-temporal reconstruction of the environment [16], is becoming popular. Additionally, the characteristics of illumination and terrain irregularities lead to a more unstable motion and to a more difficult perception of the environment. Furthermore, both SLAM and VO suffer from the well known drift problem, and usually in these environments, it is intended that robots perform long term operations. So, as these localization algorithms tend to accumulate error over time, for long term operations the drift can be quite significant. To overcome this, many SLAM algorithms are endowed with the capability of recognizing previously visited places. With this, they can detect loop closures [17,18] and correct the drift issue.

Even though so many solutions have been proposed to solve localization and mapping main issues in agriculture and forestry, it is clear that there are still several working lines to be improved. The difficulty of this problem leads to the creation of new solutions and the development of new concepts to localize outdoor robots and make them autonomous. This work explores the research field of localization and mapping in agriculture and forestry, highlighting the solutions that are present in the literature to solve this problem. In a first stage, in Section 2, the general approaches to solve SLAM and VO are detailed, and the main issues inherent to them are described. In this, the solutions to solve the theoretical SLAM problem are detailed, the mapping approaches and data association problems are discriminated, and VO is also presented. Then, Section 3 presents the methodology used in this work to collect the set of works presented in the article, related to localization and mapping in agriculture and forestry. Following this description, Sections 4 and 5 present the respective works. Next are described the existing datasets that contain sensor data acquired in agriculture or forestry that can be used in localization and mapping. Finally, the main conclusions of this work are outlined in Section 7.

2. The Localization and Mapping Problem

Localization and mapping can be approached in several different ways. The most common technique is to perform both tasks simultaneously, as performed in SLAM. Additionally, localization can be performed singly. On one hand, maps previously built can be loaded and used to localize the robot inside it. On the other, localization can be performed without registering the observed keypoints, as in VO. This section presents a description of all these approaches.

2.1. The SLAM Method

SLAM was originally proposed by Smith and Cheeseman [19] in 1986. By this time, robotics started to be addressed in innovative ways, considering probabilistic viewpoints. This new way of thinking robotics lead to the consideration on how to manipulate geometric uncertainty [20], which was crucial to the development of mapping techniques. An example is [21] that came up with the concept of a stochastic map, that is a map as we know it by now, considering relationships between objects and their uncertainty, given the available sensor data information. All these concepts provided the sufficient knowledge to the creation of the well known SLAM problem: estimating the map objects location along with the robot pose in a single joint state [22]. So, from the very beginning, SLAM was formulated as a probabilistic problem, as will be described further.

2.1.1. Solving the SLAM Problem

The SLAM problem, as referenced in [23], can be defined as: an autonomous mobile robot starts its travel in a known starting point, and explores an unknown environment. The SLAM offers a solution to estimate the unknown robot motion while building a map of the unknown environment. This solution had been formulated statistically, in many different ways. Let us denote each time instant as k, the robot pose at each instant as x_k, and the full robot trajectory until the time instant k as

$$X_k = \left\{ x_0, x_1, ..., x_k \right\}. \tag{1}$$

The robot pose encodes the robot location and orientation, and can describe both 2D and 3D spaces. Then, SLAM uses two sources of input information. The first are the controls u_k that usually represent odometry, i.e., relative information from wheel encoders between two consecutive robot poses. Additionally, other sources of controls can be used such as inertial sensors. The historic of control inputs is here represented as

$$U_k = \left\{ u_0, u_1, ..., u_k \right\}. \tag{2}$$

The other source of input information are observations taken from on-board sensors z_k. With these measurements, a map m is built using registration methods. Similar to robot states and controls, observations can be stored and their historic can be defined as

$$Z_k = \left\{ z_0, z_1, ..., z_k \right\}. \tag{3}$$

Figure 1 shows a graphical representation of SLAM, as well as all the previously mentioned variables involved in it.

In this, the relationships between the entities are well defined. It is possible to observe the sequence of robot poses, as well as the interconnection of the localization and mapping procedures.

Given all of the above, probabilistic SLAM can be addressed in two different ways. The first estimates the posterior distribution

$$p(X_k, m | Z_k, U_k, x_0), \tag{4}$$

i.e., the full robot trajectory and the environment map, given the entire set of observations, all the control inputs, and the first robot pose. The second approach does not consider the entire robot path, but only its current pose, as follows

$$p(x_k, m | Z_k, U_k, x_0). \tag{5}$$

This approach performs an incremental estimation and discards the historic of poses. To solve Equations (4) and (5), two main groups of SLAM approaches have been created and are still under development: filter-based SLAM and optimization-based SLAM.

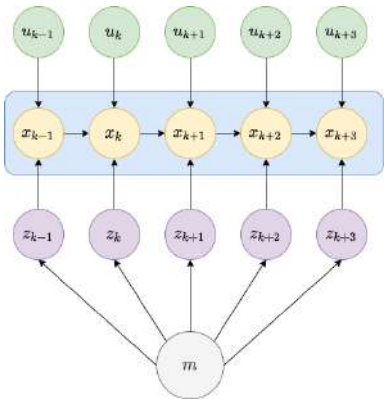


Figure 1. Graphical representation of simultaneous localization and mapping (SLAM). The robot moves from x_{k-1} to x_{k+3} through a sequence of controls. At each time instant, features are observed by the on-board robot sensors, and are registered on the global map m.

Filter-based SLAM is derived from the Bayesian filtering and is composed of two main steps: prediction step and update step. In the first, the robot pose is updated through a motion model that considers the control inputs. This is defined as

$$p(x_k|x_{k-1}, u_k). \tag{6}$$

With this information, the joint posterior can be updated integrating the motion model over the previously joint state as follows

$$p(x_k, m|Z_{k-1}, U_k, x_0) = \int p(x_k|x_{k-1}, u_k) \cdot p(x_{k-1}, m|Z_{k-1}, U_k, x_0) dx_{k-1}. \tag{7}$$

Similarly, the update step uses an observation model defined as the probability of making an observation z_k given the robot pose and map configuration, as follows

$$p(z_k|x_k, m). \tag{8}$$

Usually, SLAM assumes that once the robot pose and map configuration are known, there is a conditional independence between observations. With this information, the update step is formulated as the following conditional probability

$$p(x_k, m|Z_k, U_k, x_0) = \frac{p(z_k|x_k, m) \cdot p(x_k, m|Z_{k-1}, U_k, x_0)}{p(z_k|Z_k, U_k)}. \tag{9}$$

With all these fundamentals, several filter-based approaches are used to solve the SLAM problem.

1. The extended Kalman filter (EKF): EKF-SLAM [24,25] is based on the well known group of Kalman filters (KFs) [26]. Since both SLAM-based motion and observation models are, in most cases non-linear, the original KF is not suitable for such systems. In this context, EKF is usually used, since it linearizes the models. Additionally, this approach considers that the state uncertainty is approximated by a mean and a covariance, i.e., a Gaussian distribution. EKF-SLAM state is formulated as $f(x_{k-1}, u_k) = [x_k, m]^T$, i.e., the robot pose and the map built so far. This means that the motion model is described as

$$p(x_k|x_{k-1}, u_k) = f(x_{k-1}, u_k) + w_k, \tag{10}$$

where $f(.)$ is a non linear representation of the robot motions, and w_k represents Gaussian motion noise with covariance $\boldsymbol{Q_k}$. This formulation can lead to a high-dimensional filter, as long as the map grows, that can have serious impact on the SLAM algorithm runtime performance. Since the SLAM update time depends quadratically on the size of the state vector, EKF-SLAM is usually not suitable for large scale environments. To solve this issue, some approaches developed the sub-map concept [27,28]. In these, the global map is partitioned in local sub-maps, that can share information, still becoming conditionally independent. In this way, the state vector dimension is reduced, and the global map can still be recovered by the composition of the local sub-maps. To introduce the observations, EKF-SLAM formulates the observations model in a similar way of what is done in the motion model, by linearizing a non linear function $h(x_k, m)$ that is introduced in the model as follows

$$p(z_k|x_k, m) = h(x_k, m) + v_k, \tag{11}$$

where $h(.)$ describes the observations and v_k is zero mean Gaussian noise with covariance $\boldsymbol{R_k}$. Both $f(x_{k-1}, u_k)$ and $h(x_k, m)$ are formulated accordingly with the input data and the type of mapping approach used.

2. The information filter (IF) in the IF [29], also known as inverse covariance filter, the covariance matrix is replaced by the information matrix, i.e., its inverse. In comparison with the KF and EKF, in case of complete uncertainty, i.e., when the covariance is infinity, the information matrix is zero, which is easier to work with. Additionally, the information matrix is usually sparser than the covariance matrix. On the other hand, if the system presents non-linearities, to recover the mean is required to solve a linear system of equations. In the literature, IF is not as popular as the EKF to solve the SLAM problem. Even so, some works adopt this approach [30,31], and it is especially popular in multi-robot SLAM systems [32].

3. The particle filter (PF): PFs are based on Monte Carlo sampling, and are widely used in the SLAM context. In these, the system state is sampled by a well-defined number of particles with a given likelihood. each particle encodes information about the robot pose, and can also contain the map data. The PF overcomes the limitation of KF-based approaches, by not restricting the motion and observation models noise to zero mean Gaussians. This formulation can approximate the on-board sensors characteristics in a more realistic way. In this context, FastSLAM [33] was a huge mark in probabilistic SLAM research. This algorithm considers $\boldsymbol{N}$ particles where each one contains the robot trajectory X_k and a set of 2-dimensional Gaussians representing each landmark on the map, solving the full-SLAM problem represented in Equation (4). However, the high dimension that characterizes SLAM systems can lead PFs to become computationally unfeasible, as the number of required particles increases to perform a good approximation of the state-space. To overcome this problem, the Rao–Blackwellization became popular in the SLAM context, where the Rao–Blackwellized PFs gained impact [15,34,35]. In these, the joint state is represented as $\{w_{k-1}^{(i)}, X_{k-1}^{(i)}, p(m|X_{k-1}^{(i)}, Z_{k-1}^{(i)})\}_i^N$, where $w_{k-1}^{(i)}$ is the weight of the *ith* particle. To solve the probabilistic SLAM problem, this approach factorizes the posterior distribution in the following way

$$p(X_k, m|Z_k, U_k, x_0) = p(m|X_k, Z_k) \cdot p(X_k|Z_k, U_k, x_0). \tag{12}$$

This formulation is the key factor that speeds up the filter, since the posterior over maps $p(m|X_k, Z_k)$ can be solved analytically given the robot path and the observations until the current instant.

4. Graph-based optimization SLAM: Graph-SLAM [36] is a whole different way of looking and approaching the SLAM problem, comparing to filter-based techniques. The basic principle of this approach is as follows: the robot and map features can be represented as nodes in a graph-way procedure. Then, arcs exist between consecutive robots poses x_{k-1}, x_k representing the information given by the input controls u_k, and between robot poses and map features. Figure 2 represents this configuration.

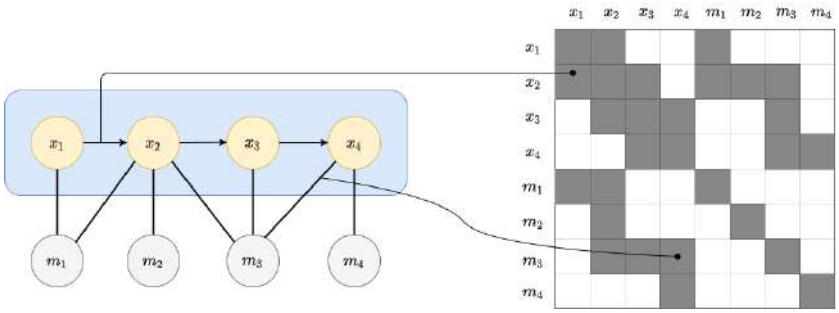


Figure 2. Illustration of graph construction present in Graph-SLAM. Adapted from [23].

In this figure, it is possible to observe that, for example, when the robot in the pose x_1 observed a map feature m_1, an arc is added between these two agents. This relation is also encoded in a matrix-like representation, where a value is added between the x_1 and m_1 components. When the robot moves (Figure 2b), the control input u_2 leads to an arc between x_1 and x_2. As the input data increase, the graph representation also increases. Even so, this representation is considered to be sparse, in that each robot pose node is connected to few other nodes. This leads to a great advantage in comparison with EKF-SLAM, since the update time of the graph is constant, and does not grow quadratically as in EKF-SLAM. As stated in [37], this graph-based approach can solve for the robot pose and map structure, by finding a minimum of a cost function of the following form

$$F(X_k, m) = \sum_{ij} e_{ij}(X_k, m)^T \, \Omega_{ij} \, e_{ij}(x_k, m), \tag{13}$$

where X_k is the vector containing the robot trajectory until the time instant k, m is the map, e_{ij} is an error function that relates the prediction and the observations, and Ω_{ij} is graph-based matrix representation referenced before. Thus, the optimal values (X_k^*, m^*) are obtained minimizing as

$$(X_k^*, m^*) = \underset{X_k, m}{argmin} \; F(X_k, m). \tag{14}$$

To solve (14), the state-of-the-art methods can be used, such as Gauss–Newton, Levenberg–Marquardt, and others.

2.1.2. Mapping the Environment

The mapping procedure is crucial in the localization performance. The accuracy of the on-board sensors, and the quality of the data post-processing algorithms dictate the quality of the perception of the surrounding environment by the robotic platform. In agriculture, this is specially true, since in most cases, the environment present harsh conditions for robotics localization and mapping. Figure 3 shows an example of a working place of an agricultural robot.

In this are visible the long corridors that constitute the vineyard, as well as the absence of structured objects, such as walls. All these characteristics complicate the mapping procedure. To build precise maps of these environments, the robotic platform present in Figure 3 uses high-range 3D LiDARs and stereo cameras. In this way, 3D dense reconstruction with color information is achieved using the vision systems, and high range omnidirectional maps are constructed using the 3D LiDAR sensors. To address the mapping procedures present in the SLAM context, in this article we start by describing the well known problem of data association. Then, the mapping approaches are divided in two main sets: metric maps, topological maps, and semantic maps.

Figure 3. Agricultural robot working place on an agricultural environment [38]. The image shows the complexity of the mapping procedure, and the importance of the projection of an adequate system of sensors.

1. The data association problem: The data association in SLAM [39–41] is the process of associating a sensed feature in a given time instant, with another one observed in a different instant, ensuring that both correspond to the same physical point in the world scene. This procedure can constitute a problem in SLAM, since the number of possible associations between features grow exponentially over time, since the map becomes larger at each observation [42]. If data association is unknown, the estimation of the posterior distribution has to consider a high density of modes, which can lead to an infeasible computational cost. Moreover, the detection of loop closures is essentially a data association problem, in that associations between the global map and a local map are searched to find previously visited places by the robot.

The data association problem has been addressed in several works present in the literature [43,44]. The most common procedure is to perform incremental maximum likelihood association, i.e., choose the most likely association hypothesis. Other algorithms support the multi-association hypothesis, where in cases of high association uncertainty, a feature can have multiple associations until one of them has a higher likelihood. This, besides improving the success rate of data association, can lead to the growth of the number hypothesis exponentially [45]. In more advanced approaches, machine learning is used to compute data association algorithms, performing decisions for groups of features at once [46]. In general, considering its importance, data association is a topic which is always under research and improvement, considering new types of sensors, data, and features.

2. Metric maps: The metric representation is a structure that encodes the geometry of the environment [11]. Metric maps can be represented in several ways, depending on the input sensor data, and the dimension of the mapping procedure. One of the most common approaches is landmark-based mapping procedures. In these, the environment is represented as a set of 3D landmarks, in a sparse way. The landmarks encode relevant features of the scene, such as planes, lines or corners [47], and are represented as unique signatures (also called descriptors). A good descriptor is crucial for the proper performance of the mapping procedure. The more unique is the descriptor, the easier is the data association procedure. Additionally, metric maps can represent the scene in a more structured way, using occupancy grid maps. Typically, these maps represent the environment in 2D, and sample the geometric space into cells with an associated probability of occupation [48].

3. Topological maps: The concept of partitioning the geometric space into local maps gained strength in the SLAM mapping procedure. In this context, topological maps come up as a logical solution. These algorithms represent the global map as a set of connected local maps stored in nodes [49]. As stated by Lowry et al. [50], a topological map is conceptually a biological cognitive map, where nodes represent possible places in the world and edges the possible paths that connect these nodes. In the SLAM context, the routes between nodes represent the robot navigation path. This context has the advantage of storing local maps in nodes, allowing the robot to load only the local maps of interest at each point in time. In agriculture, this is specially interesting, since usually maps are dense and long-term, leading to high memory consumption. The partitioning of the map allows greater memory efficiency, in that only portions of the map memory are loaded at each instant. On the other side, this technique can have implementation issues. For example, the robot has to be able to associate physical places and routes to nodes and paths. Moreover, the extraction of the topological map can also be challenging, and usually needs the definition of a metric occupancy grid map of the environment.

4. Semantic maps: In many cases, adding semantic information to maps can enhance the quality of the localization systems [51]. This is a whole new challenge for perception systems, that have to be able to recognize and classify objects and places. To build a semantic map, several aspects have to be taken into account. For example, the detail of semantic concepts is important, and depends mainly on the robot task. If the robot wants to move from some corridor into another, the semantic classification can be as high-level as recognizing corridors. On the other hand, if the robot is intended to harvest, the semantic classification should be able to recognize trunks, leaves, etc. Another important concept is the definition

of the semantic properties, since a single object can be characterized by an unlimited number of concepts. This process is crucial for the mapping procedure, and can be viewed as a dictionary of the environment.

5. Hybrid maps: All the previously described maps can be merged and fused, creating the so-called hybrid map architectures. For example, objects or places can be semantically characterized and represent nodes of a topological map that holds local metric maps. For example, in [52] is proposed, a probabilistic map representation that considers objects and doors, and classifies places using prior information about the objects. Figure 4 shows another representation of an hybrid map, applied to an agricultural context.

In this, the vineyard is semantically characterized as rows and corridors, in a topological map that has paths interconnecting each node.

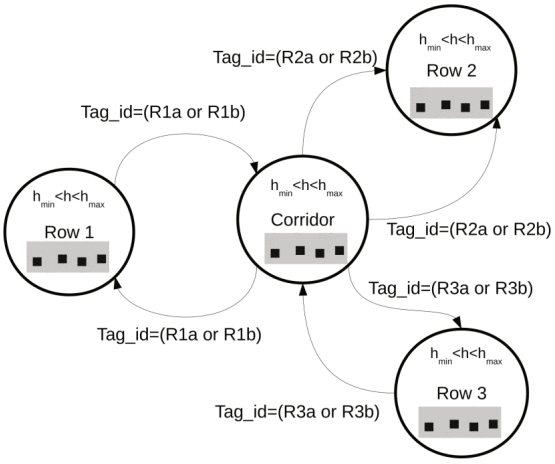


Figure 4. Semantic and topological map of a vineyard [53]. Nodes represent well defined places and contain semantic information: they either represent a vineyard row or corridor. Local maps of artificial landmarks are stored in each node. Edges encode the geometric path between nodes.

2.2. The Visual Odometry Method

VO's ultimate goal is to localize the absolute camera pose in relation with a given initial reference pose, as represented in Figure 5.

Contrary to the SLAM problem, VO does not maintain a map of the environment. This approach receives a set of consecutive images, computes the relative transformation between them, and integrates each transformation to recover the absolute camera pose. This algorithm is a particular case of structure for motion, that solves the problem of 3D reconstruction of the environment and the camera poses using a sequence of unordered image frames [54]. To compute the relative transformation between images, VO is composed of a sequence of steps. Firstly, the input image is processed so that features can be extracted from it [55]. This is done so that robust feature matching algorithms can be executed between images. Then, matches between features are used to estimate the motion between images. This motion estimation can be computed in several ways, such as 2D to 2D, where the features are specified in 2D coordinates for both images. Moreover, 3D to 3D motion estimation can be calculated, using depth information of stereo cameras, specifying the images features in the 3D space. In some cases, bundle adjustment [56] techniques are applied to refine the local estimate of the trajectory. This method is implemented aiming to optimize the camera parameters and the 3D landmark parameters simultaneously.

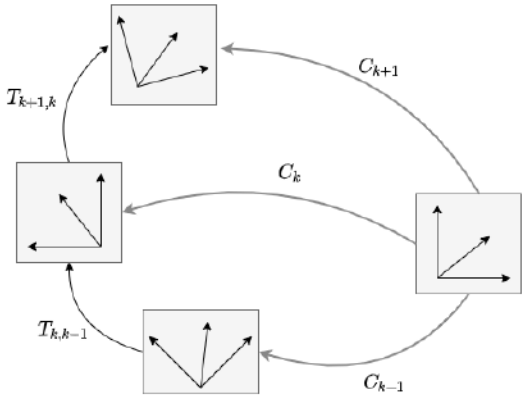


Figure 5. The Visual Odometry problem. Just like in wheel odometry, each relative transformation between camera poses $T_{k,k-1}$ is integrated in order to recover the absolute camera pose C_k, regarding an initial coordinate frame. Adapted from [13].

3. Methodology

The previous sections provided an overview of the variety of methods and solutions for robotics localization and mapping. This work pretends to evaluate the performance of these approaches in agricultural and forest environments. The main goal is to understand if this is still an open issue in the scientific community, and, if so, what are the target research areas to overcome the limitations of the science in this field. To do so, a deep literature analysis was performed. The result was the collection of 15 works on agricultural fields, and 9 on forest environments. To evaluate them, the following criteria were considered:

- *Application*: Agricultural or forest application of the desired autonomous system.
- *Localization approach:* The methods and sensors used to localize the robot.
- *Mapping approach:* The methods and sensors used to map the environment.
- *Accuracy:* Evaluation of how accurate the robot localization is. In the ideal case the accuracy show be less that some value (usually 20 cm [37,57]).
- *Scalability:* Evaluation of the capacity of the algorithm to handle large-scale paths.
- *Availability:* Evaluation of the possibility of the algorithm to present reliable localization right away, without need for building prior maps of the environment.

Given the immature status of this research field, let us not consider more advanced metrics such as recovery, updatability, or dynamicity [37]. Figure 6 shows the distribution of the years of creation of the works collected.

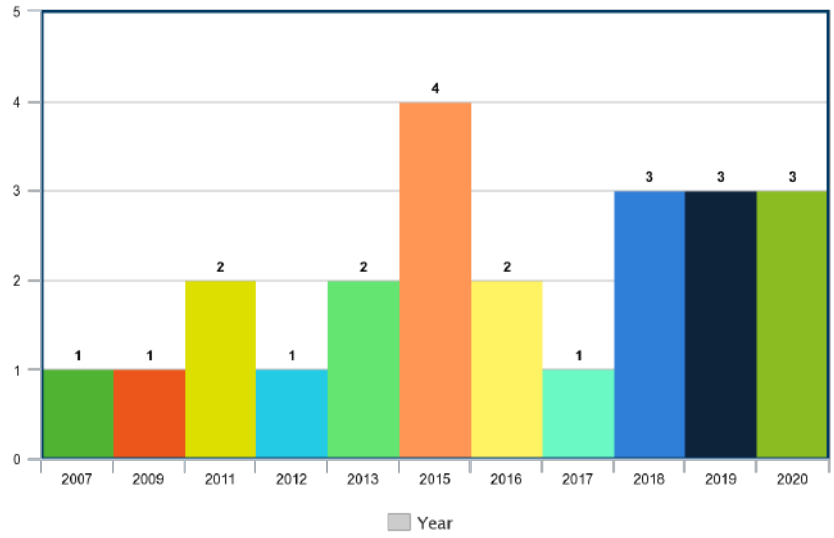


Figure 6. Histogram representing the years of creation of the works collected of localization and mapping in agriculture and forestry. The great majority of them were developed since 2011.

From this, it is possible to observe that this research field is still recent, since the majority of works were developed since 2011. Finally, dataset collections in agriculture and forest for autonomous driving were also searched in the literature. This resulted in the collection of 5 works. To evaluate them, the following topics were considered:

- *Description:* Agricultural or forestry area where the data was collected, as well as sensor information.
- *Large-scale:* Whether or not the data were collected in large-scale environments, and large-scale paths.
- *Long-term:* Whether or not the data were collected in different seasons of the year, and different times of the day.

It is worth noting that all these works were developed after 2016.

4. Localization and Mapping in Agriculture

Recent works approach the problem of localization and mapping in agriculture. This work performed an intense literature review in this area. Table 1 presents a summary of the works collected and their main characteristics.

Table 1. Summary of the works collected on localization and mapping related to the agricultural field.

Ref.	Agricultural Application	Localization Approach	Mapping Application	Tested in Real Scenario	Accuracy	Scalability	Availability
Freitas et al. [58,59] (2012)	Precision agriculture in tree fruit production	(2D) EKF-based. Wheel odometry and laser data. Uses points and lines features to match withs previously built map.	Offline orchard metric mapping for artificial landmark detection.	**Yes**. Three experiments with more than 1.8 km each.	Low error for flat and dry terrains (0.2 m). High errors in steep terrains (up to 6 m).	**Yes**. Long-term experiments performed.	**No**. The method requires a previously build map.
Duarte et al. [60] (2015)	Autonomous navigation on steep slope vineyards.	(2D) PF-based. Fusion of GPS, wheel odometry, and previously mapped landmarks.	Offline metric mapping. Use of wireless sensors to compute landmarks location.	**Yes**. However, tests were done in an urban environment.	Beacons mapped with 1.5m of average error. Robot pose estimation not evaluated quantitatively.	**Not tested.**	**No**. No proper performance without a previously built metric map.
Zaman et al. [61] (2019)	Precision agriculture.	(2D) VO algorithm based on a cross correlation approach.	-	**Yes**. Tested in soil, grass, concrete, asphalt, and gravel terrains.	Normalized cumulative error of 0.08 mm for short paths.	**No**. System performance degrades when incremanting path lenght.	**Yes**. No need for first passage of theslgorithm in the agricultural field.
Habibie et al. [62] (2017)	Monitoring of ripe fruit.	(2D) Use of the state-of-the-art Gmapping [15] and Hector SLAM [14] approaches.	Combination of metric maps. Occupancy grid map generated by the SLAM approach, and fusion with tree/fruit detection.	**No**. Experiments only performed in simulation.	Localization not quantitatively evaluated. Accurate detection of simulated fruits, and trees.	**Not tested**.	**Not tested**.
Younse et al. [63] (2007)	Greenhouse spraying.	(2D) VO algorithm. Use of Kanade-Lucas-Tomasi (KLT) Feature Tracker.	-	**Yes**. Short-term tests in indoor environments, and outdoor with different ground surfaces.	12.4 cm translation error for a short path of 305 cm, and 8° orientation error for a 180° rotation.	**Not tested**.	In theory.
Bayar et al. [64] (2015)	Autonomous navigation in orchards.	(2D) Fuses wheel odometry and laser range data. Assumes that orchards rows length is known. Localization only relative to the rows' lines.	-	**Yes**. Experiments performed in several orchards rows.	Low errors relative to the rows' lines. No quantitatively average values provided.	**Partially**. Only under the assumption that the row length is known, and the localization is relative to the rows' lines.	**No**. Requires the previously mentioned assumptions.
Le et al. [65] (2018)	General agricultural tasks.	(3D) Localization based on non-linear optimization techniques. Uses the Lavenberg-Marquardt algorithm.	3D LiDAR mapping. Uses edge and planar features extracted directly from the point cloud.	**Yes**. Tested both on real and simulated scenarios.	Less than 2% translation error for a 500m trajectory.	**Yes**. A successful long-term experiment was performed. Loop closure supported.	**Yes**. The system is able to perform online SLAM without any priormap.
Chesin et al. [66] (2011)	Autonomous navigation on olive groves.	(2D) Extended IF-based SLAM. Uses a laser sensor and a monocular vision system.	Metric map composed of olive stems. Support Vector Machine used to detect stems, and laser data to map them.	**Yes**. Tests performed in a real olive scenario.	Successful reconstruction of the entire olive. Consistent SLAM approach. Error does not exceed 0.5m.	**Yes**. The method is able to be long-term consistent operating in the entire olive.	**Yes**. The system is able to perform online SLAM without any prior map.
Chebrolu et al. [67] (2019)	Precision agriculture in crop fields.	(2D) PF-based. Fuses wheel odometry with camera visual data.	Offline mapping. Metric-semantic map of landmarks build from aerial images.	**Yes**. Real experiments performed on sugarbeet field.	Maximum error of 17 cm on a >200 m path.	**Yes**. Experiments show good performance on long-term paths.	**Partially**. Only if a previously extracted aerial map is available.
Blok et al. [68] (2019)	Autonomous navigation in orchards.	(2D) PF-based: uses a laser beam model. KF-based: uses a line-detection algorithm.	-	**Yes**. Tests on two real orchard paths.	Lateral deviation errors <10 cm and angular deviation <4°.	**Yes**. Successful tests in orchards paths in 100 m.	In theory. Does not need any prior mapping information.
Pyatthilaka et al. [69] (2011)	General agricultural tasks.	(2D) EKF-based. Fusion of a VO approach with a stereo vision range measurement system.	-	**Yes**. Experiments performed in real outdoor environment.	Average error of 53.84 cm on the tested sequence.	**No**. High cumulative error for long paths.	In theory. Does not need any prior mapping information.
Iqbal et al. [70]	Phenotyping, plant volume and canopy height measurement.	(2D) EKF-based. Fusion 2D laser, an IMU, and GPS data.	2D point cloud map built by successive registration.	**No**. Experiments performed in the Gazebo simulator.	2.25 cm error on 32.5 m path, and 7.78 cm on 38.8 m path.	**Not tested.**	**Yes**. Does not require prior mapinformation.
Bietresato et al. [71]	Volume reconstruction and mapping of vegetation.	(2D) Fusion of sonar, IMU and RTK GPS.	Plant volume calculation using LiDAR sensors and optical-data to obtain normalized difference vegetation index (NDVI) maps.	**Yes**. Mapping approach tested in indoor and outdoor environments.	Localization system not tested.	**Not tested.**	**Yes**. Does not require prior map information.
Ubstumo et al. [72]	In-row weed control.	(2D) EKF-based. Uses a forward facing monocular camera and a GPS module.	Support Vector Machine (SVM) used to extract environment features and create a spray map.	**No**. Localization and mapping not tested.	Localization system not tested.	**Not tested.**	**Not tested.**
Santos et al. [8] (2020)	Autonomous navigation on steep slope vineyards.	(2D) PF-based. Fuses wheel odometry with a vision system.	Metric map composed of high-level landmarks detected using Deep Learning techniques.	**Yes**. One experiment done on a real vineyard.	Average error of 10.12 cm over the tested sequence.	**Not tested.**	**Yes**. The approach does not need aprreviously built map.

Autonomous navigation is being studied in tasks like precision agriculture in tree fruit production, greenhouse spraying; and in fields like steep slope vineyards, orchards, and olive groves. The implementation of autonomous robotic platforms in agriculture has been growing, in particular in tasks like weeding, seeding, disease and insect detection, crop scouting, spraying, and harvesting [73]. The localization algorithms of these works are, in their great majority, based on PFs, KFs (EKF and IF), or VO. From all the works collected, only Le et al. [65] developed a system able to localize the robot in 3D. This work uses a non-linear optimization technique based on the Lavenberg–Marquardt algorithm to solve for the robot pose. From the evaluation performed, we concluded that this work is the most suitable for autonomous navigation in agriculture, since it meets all the criteria: was tested in the real scenario, presents a high accuracy, scalability, and availability. Cheein et al. [66], also present a very interesting work. This is the only one based on an IF, and performs 2D localization with accuracy, availability and scalability. It is worth noting that the majority of works use either laser sensors, or visual sensors such as monocular or stereo camera systems. Wheel odometry is also commonly used to give filter-based approaches the control inputs. Few works use GNSS-based receivers, which shows the recurrent unavailability of satellite signals in agricultural environments, and so, the necessity of the creation of GNSS-free localization and mapping algorithms. In this context, the real time kinematic (RTK) satellite-based localization systems are rising up due to their high accuracy. In particular, Bietresato et al. [71] use this technology, fusing it with a sonar and an IMU to localize a robotic platform. This work is proposed in order to reconstruct the volume of plants and to map the vegetation, using laser data and NDVI information. On the negative side, the authors do not provide any experiments on the localization and mapping approach. VO methods tend to use only visual data, and have the advantage being, in theory, always available, in that they do not require prior mapping information. On the other hand, these families of localization methods are not, in general, scalable, since they accumulate error over the time, due to the integration of relative motions between image frames. As an example, Zaman et al. [61] propose a VO approach, tested in real scenarios such as soil, grass, and other terrains, that presents accurate results for short paths (0.08 mm error), but that degrades when increasing the path length. Similarly, Younse et al. [63] present a VO approach, only tested for a path with 305 cm extension, which does not prove to have practical availability for long-range motion types.

In terms of mapping, one can conclude that not all the methods perform this task. In particular, VO approaches do not map the environment. Furthermore, some works use localization-only approaches with prior information. For example, Bayar et al. [64] assume that the orchards rows length is known, and localization is performed relatively to the rows' lines. In other cases, mapping is performed offline [58–60]. In this context, Freitas et al. [58,59] perform an offline orchard metric mapping using artificial landmark detection, and use this information in the localization procedure. In this, points and line features are detected and matched to the prior map information to perform the robot pose estimation. From all the works collected, the one proposed by Chebrolu et al. [67] stands out as being the only one that performs semantic mapping. This work implements an offline mapping procedure, using aerial images to build a map of landmarks. Moreover, Santos et al. [8] use an interesting approach of computing 2D landmark location using deep learning techniques. In general, metric maps are used, by means of probabilistic occupancy grid maps, point clouds, or feature-based approaches.

From this study, many conclusions can be taken. Firstly, the fact that almost all the works were tested in real scenarios is exciting, in that it shows the ambition of researchers to actually implement autonomous navigation in agriculture. From these, we highlight [58,59] that perform tests in real sequences with more than 1.8 km each. In terms of accuracy, the majority of works present good results. For example, Le et al. [65] present a system with 2% translation error for a trajectory with approximately 500 m. Furthermore, Chebrolu et al. [67] propose a work that achieves a maximum error of 17 cm on a sequence with a path higher than 200 m. However, besides many of the works presenting acceptable performance, the problem of 3D localization in agriculture is still an area with

very low research impact. Only one work focuses on this problem, which shows that much work and development is yet to come. In particular, steep slope agricultural environments will require localization system able to accurately estimate all the 6-DoF of the robot pose. Additionally, for robots to operate safely in agricultural environments, longer term experiments should be carried out. This is due to the requirement of automatizing procedures during long periods, which imposes the need of long-term autonomous operability. In the same context, all-weather experiments are required, since the navigation systems should work under a vast range of meteorological conditions. In the current state-of-the-art, the majority of works focus on low- or mid-term experiments, and do not take into account the all-weather challenge. In addition, the mapping approaches are, in general, based on metric maps. Topological and semantic mapping are concepts almost nonexistent in this area. This also shows that much work can be done in this area. Semantic mapping can be important in the classification of agricultural agents, obstacle detection on these fields, etc. In this area, deep learning can have an important role, where high-qualified deep neural networks can be trained and used to provide an intelligent perception of the environment. All this information can be further used by human operators for example, for monitoring tasks, detection of anomalies, and others. Moreover, topological mapping can represent a huge advance in these area, when considering scalability. Global maps can be partitioned in local maps in a graph-like procedure, which can solve memory issues, allowing the robot to navigate safely during longer periods of time. Finally, not all the works present scalability and availability, which are concepts required before others which are more complex can arise such as recovery, updatability, or dynamicity.

5. Localization and Mapping in Forestry

Recent works approach the problem of localization and mapping in forestry. This work performed an intensive literature review in this area. Table 2 presents a summary of the works collected and their main characteristics.

Table 2. Summary of the works collected on localization and mapping related to the forestry field.

Ref.	Forestry Application	Localization Approach	Mapping Application	Tested in Real Scenerio	Accuracy	Scalability	Availability
Qian et al. [74] (2016)	Accurate forest stem mapping.	(2D) Fusion of GNSS/INS with scan-match-based approach solved using the Improved Maximum Likelihood Estimation.	Metric occupancy grid map built from laser scan data.	**Yes**. Real field experiments performed.	Positioning accuracy of 13 cm for the field data sequence.	**Yes**. Successful results in long-term sequence with 800 m.	**Yes**. The algorithm does not required prior map information.
Hussein et al. [75] (2015)	Autonomous navigation in forests.	(2D) Localization based on a scan-matching procedure.	Metric map of trees generated by on-board LiDAR sensors. Map matching with a global map generated from aerial orthoimagery.	**Yes**. Experiments performed on real forest.	Average error of <2 m for robot pose.	**Partially**. Long-term experiments performed, but with considerable errors.	**No**. Requires a map generation from aerial images.
Li et al. [76] (2020)	Autonomous harvesting and transportation.	(2D) Map matching localization approach based on Delaunay triangulation.	3D LiDAR-based stem mapping.	**Yes**. Real experiment using a forestry dataset.	Location accuracy of 12 cm on the tested sequence.	**Yes**. Successful results in a long-term path (200 m).	**No**. Requires a previously built stem map.
Pierzchała et al. [77] (2018)	3D forest mapping.	(3D) Graph-based SLAM. Uses the Levenberg-Marquardt method.	3D point cloud map generated using LiDAR odometry, with graph optimization through loop closure detection.	**Yes**. Data recorded by authors' robot in a forest.	SLAM system provides tree positioning accuracy—mean error of 4.76 cm.	**Yes**. Successful long-term real experiments in sequence with 130.7 m.	**Yes**. The method performs online SLAM without need of prior map.
Rossmann et al. [78] (2013)	Autonomous navigation in forests.	(2D) PF-based. Fusion of a laser sensor and a GPS.	Offline generation of forest tree map.	**Yes**. However, no demonstration of results available.	Authors measure the location error in sample points in time. They claim to obtain mean error of 0.55 m.	**Not tested**.	**Not tested**.
Miettinen et al. [79,80] (2007)	Forest harvesting.	(2D Feature-based SLAM. Computed using laser odometry.	Metric feature map, built by fusing laser data and GPS information.	**Yes**. Experiments on real outdoor environment.	**Not tested**, due to the unavailability of ground truth.	**Not tested**.	**No tested**.
Heikki et al. [81,82] (2013)	Stem diameter measure.	(3D) Laser-odometry approach, fused with an IMU.	Metric landmark map composed of stem detections and ground estimation.	**Yes**. Long-term real experiment performed (260 m).	7.1 m error for a 260 m path.	**No**. High localization errors reported for a long-term path.	**Yes**. Does not need prior map information.
Tang et al. [83] (2015)	Biomass estimation of forest inventory.	(2D) Scan-matching based SLAM. Uses the Improved Maximum Likelihood Estimation algorithm.	Metric occupancy grid map built using laser data.	**Yes**. Long-term real experiment performed (500 m).	Obtained positioning error <32 cm in the real world experiment.	**Yes**. Successful performance in long-term experiment.	**Yes**. Does not need prior map information.

The main goals for robotics in forest environments are stem mapping, harvesting, transportation, and biomass estimation. Similarly to the methods collected in the agricultural context, a great majority of the developed localization systems for forest environments only compute the robot pose in 2D. Only Pierzchała et al. [77], aiming to perform 3D mapping of a forest environment, propose a graph-based SLAM approach, that computes 6-DoF robot pose using the Levenberg–Marquardt algorithm. In addition, this work presents other contributions, such as built in-house datasets that were used to test and evaluate the SLAM pipeline. Moreover, Heikki et al. [81,82] propose a 3D SLAM algorithm where the translation components are calculated by a laser-odometry technique, and the rotational ones using an IMU. It is worth noting that the SLAM process also estimates the drifting bias of the yaw angle. On the negative side, this work reports high errors for long-term trajectories. All the other works propose 2D-based SLAM algorithms. From these, we highlight [74] that, in order to build accurate stem maps, fuses GNSS/INS with a scan-match approach solved using the improved maximum likelihood estimation method. This work presents an accuracy of 13 cm in a real experiment with 800 m, presenting scalability and availability, since it does not require prior mapping information. Furthermore, Li et al. [76] propose a very innovative localization approach based on map matching with Delaunay triangulation. Here, the Delaunay triangulation concept is used to match local observations with previously mapped tree trunks. This is an innovation since, unlike the majority of works that use point cloud-based matching approaches, the authors propose a topology-based method. This work achieved an accuracy of 12 cm in a long-term path with 200 m. On the negative side, it requires a previously built map so that it can operate. Similarly, Hussein et al. [75] propose a localization and mapping approach that requires a prior map. The interesting innovation of this work is the creation of a stem map using aerial orthoimagery. Thus, this approach does not require that the robot goes one first time to the field before it can perform autonomously. Miettinen et al. [79,80] use the interesting concept of laser odometry to localize the robot. This approach is related with VO in that it computes the relative transformation between consecutive sensor observations. The major difference is that, in laser odometry, a scan-matching approach is used to align consecutive observations and extract the relative transformation matrix. Moreover, using scan-matching concepts, Tang et al. [83] propose a 2D SLAM algorithm that solves the Improved maximum likelihood estimation method to localize the robot, with the goal of estimating the biomass forest inventory. Considering the overall collection of works, it is worth noting that visual sensors are not so used as in agriculture. Laser range sensor and GNSS-based receivers are the ones used in this field.

In terms of mapping, all the works perform this task, contrary to the case of agriculture. From all the works, we highlight [77], since it performs 3D point cloud map generated using LiDAR odometry, with graph optimization through loop closure detection. Moreover, only two works compute offline mapping. The first, proposed by Li et al. [76], requires a prior visit to the working environment, in order to create a global 3D stem map using a LiDAR sensor. In a more innovative way, Hussein et al. [75] build the global map using aerial orthoimagery, which enables the navigation with prior map information without mandatory previous visits to the working environment. In terms of mapping, another major conclusion is that metric maps play an important role in many SLAM algorithms. This is due to the frequent use of LiDAR sensors to build tree trunk (stem) maps. This approach is common since stems constitute highly trustworthy landmarks to use in SLAM algorithms. In this context, Miettinen et al. [79,80] use their SLAM algorithm not only to localize the robot, but also to build precise stem maps, with specific information of the forest structure. In this work, during the mapping procedure, tree diameters, positions and stand density are measured. In the same way, Heikki et al. [81,82] propose a mapping approach where the stem diameters are measured, and a ground model is built. This work reports a trunk diameter error of 4 cm for short range observations (less than 8 m). Given all this, we can conclude that, in forestry, mapping seems to be more evolved, with intense research focus on stem detection and mapping.

From this collection of works, many conclusion can be taken. After an intensive search, only nine proper works were found, and most of them were proposed after 2013. This shows that this

research line is still quite new, not yet developed, and has many open topics. From these topics, we can highlight the issue of availability. From the reported works, only half of them showed the ability of working without prior map information, which shows that online SLAM is still under-developed in this field. Moreover, as mentioned before, 3D localization is still quite rare in forest environments. This can deny autonomous navigation in steep forests with considerable inclinations, and different levels of altimetry. On the other side, mapping seems a more advanced area, with some works focused on creating 3D maps of the forests, and others developing methods for stem mapping. Even so, just like in the agricultural field, metric maps are quite abundant. So, more advanced mapping concepts such as semantic and topological perception of the environment are still quite under-developed. Moreover, many works propose the detection of tree trunks to use as landmarks in the localization procedure. However, none of them currently use deep learning concepts to detect such agents in the forest fields. Since the tree trunk diameter measurement is one of the fields of interest in forestry, semantic segmentation can play an important role in the future. These algorithms use neural networks to segment images or point clouds into semantic objects, at the pixel or point level. The use of visual sensors not common in forestry is also related to this. The implementation of visual perception algorithms can also improve the localization and mapping procedures. On the bright side, just like in agriculture, the works were tested in real scenarios, which shows that resources are being channeled to this research field.

6. Datasets for Localization and Mapping in Agriculture and Forestry

In order to have reliable data to develop new concepts and algorithms related with localization and mapping in agriculture and forestry, the creation of open-source datasets is essential. Table 3 shows a collection of 5 works done in this area, all of them developed after 2016.

Chebrolu et al. [84] propose an agricultural dataset for plant classification and robotics navigation on sugar beet fields. This work provides a variety of sensors, such as RGB-D cameras, 3D LiDAR sensors, GPS, and wheel odometry, all calibrated extrinsically and intrinsically between each other. The main interesting feature of this work was the recording of data during a period of three months, which can be used to test localization and mapping approach with long-term considerations. Kragh et al. [85] propose a collection of raw data from sensors mounted on a tractor operating in a grass mowing scenario. This work provides precise ground-truth localization using a fusion of IMU and GNSS-based receiver. The main limitation of this proposal is the lack of data from different times of the day, or year. In [86] a dataset for test and evaluation of visual SLAM approaches in forests is presented. In this, data are provided from four RGB cameras, an IMU, and a GNSS receiver, all of them calibrated and synchronized. Data were recorded in summer and winter seasons, and different times of the day. In [87] is presented a dataset with six sequences in soybean fields. This works has in consideration repetitive scenes, reflection, rough terrains, and other harsh conditions. Data were collected on two different days, but at the same period of the day. Finally, Reis et al. [88] present a set of datasets with information from a variety of sensors, suitable for the localization and mapping purposes. In this work, a state-of-the-art SLAM approach was tested under the different datasets recorded in different forests. Overall, all the methods present a large-scale dataset, with long distances travelled. In the near future, with the intense growth of deep learning models in robotic systems, more advanced datasets should be created. In particular, sensor data can be put together with object labelling. In this context, visual data can be provided along with landmark annotation, both at object and pixel level. Furthermore, 3D LiDAR data can also be annotated in order to enable researchers to train and test their own semantic segmentation models.

Table 3. Summary of the works collected presenting datasets for use in autonomous navigation in agriculture and forestry.

Ref.	Description	Large-Scale	Long-Term
Chebrolu et al. [84] (2017)	Agricultural dataset for plant classification and robotics navigation on sugar beet fields. Provides data from RGB-D camera, 3D LiDAR sensors, GPS, and wheel odometry. All the sensors are calibrated extrinsically, and intrinsically.	**Yes**.	**Yes**. Recorded over a period of three months, and, on average, three times a week.
Kragh et al. [85] (2017)	Raw sensor data from sensors mounted on a tractor in a grass mowing scenario. It includes stereo camera, thermal camera, web camera, 360° camera, LiDAR, and radar. Precise vehicle localization obtained from fusion of IMU and GNSS.	**Yes**. Data recorded on a large field with 2 ha.	**No**. The dataset has approximately 2 h, all at the same day.
Ali et al. [86] (2020)	Dataset for visual SLAM on forests. The vehicle is equipped with four RGB cameras, an IMU, and a GNSS receiver. Sensor data is calibrated and synchronized.	**Yes**. Range of distance travelled varies from 1.3 km to 6.48 km.	**Yes**. Data recorded on summer and winter conditions. Also, different times of the day were considered.
Pire et al. [87] (2019)	Dataset with six sequences in soybean fields. Considers harsh conditions such as repetitive scenes, reflection, rough terrain, etc. Contains data from wheel odometry, IMU, stereo camera, GPS-RTK.	**Yes**. Total length trajectory around 2.3 km.	**Partially**. Data recorded in two separate days, but in the same time of the year, and the day.
Reis et al. [88] (2019)	Dataset containing data from different sensors such as 3D laser data, thermal camera, inertial units, GNSS, and RGB camera in forest environments.	**Yes**. Data recorded in three different large-scale forests.	No information.

7. Conclusions

This work proposed the analysis of the current state-of-the art of localization and mapping in agriculture and forestry. After an intensive search, 15 works were collected related to localization and mapping in agriculture, 9 were collected in the context of forestry, and 5 works that present datasets for test and evaluation of these approaches were characterized. The works were characterized in terms of their agricultural/forestry application, the localization and mapping approaches, and their accuracy, availability and scalability. This study leads to the main conclusion that this research line is still premature, and many research topics are still open. In particular, 3D localization is quite rare in these environments. Furthermore, advanced mapping techniques that are now present in other areas such as topological and semantic mapping are yet to develop in agriculture and forestry. This being said, we believe that this area has a lot of potential to grow, and that, in the near future, many works and innovations will arise to bridge the current faults of the state-of-the-art.

Author Contributions: Conceptualization, A.S.A. and F.N.d.S.; methodology, A.S.A. and F.N.d.S. and J.B.C.; validation, F.N.d.S. and J.B.C. and H.S. and A.J.S., F.N.d.S. and J.B.C., H.S., A.J.S.; investigation, A.S.A. and F.N.d.S.; resources, F.N.d.S. and J.B.C.; writing—original draft preparation, A.S.A.; writing—review and editing, F.N.d.S. and J.B.C.; supervision, F.N.d.S. and J.B.C. and H.S. and A.J.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work is funded by funds through the FCT—Fundação para a Ciência e a Tecnologia, I.P., within the framework of the project "WaterJPI/0012/2016". The authors would like to thank the EU and FCT for funding in the frame of the collaborative international consortium Water4Ever financed under the ERA-NET Water Works 2015 cofounded call. This ERA-NET is an integral part of the 2016 Joint Activities developed by the Water Challenge for a changing world joint programme initiation (Water JPI).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Auat Cheein, F.A.; Carelli, R. Agricultural Robotics: Unmanned Robotic Service Units in Agricultural Tasks. *IEEE Ind. Electron. Mag.* **2013**, *7*, 48–58. [CrossRef]
2. Skvortsov, E.; Skvortsova, E.; Sandu, I.; Iovlev, G. Transition of Agriculture to Digital, Intellectual and Robotics Technologies. *Econ. Reg.* **2018**, *14*, 1014–1028. [CrossRef]
3. Billingsley, J.; Visala, A.; Dunn, M. Robotics in Agriculture and Forestry. In *Springer Handbook of Robotics*; Siciliano, B., Khatib, O., Eds.; Springer: Berlin/Heidelberg, Germany, 2008; pp. 1065–1077. [CrossRef]
4. Roldán, J.J.; del Cerro, J.; Garzón-Ramos, D.; Garcia-Aunon, P.; Garzón, M.; de León, J.; Barrientos, A. Robots in Agriculture: State of Art and Practical Experiences. In *Service Robots*; InTech: London, UK, 2018. [CrossRef]
5. Perez-Ruiz, M.; Upadhyaya, S. GNSS in Precision Agricultural Operations. In *New Approach of Indoor and Outdoor Localization Systems*; InTech: London, UK, 2012. [CrossRef]
6. Guo, J.; Li, X.; Li, Z.; Hu, L.; Yang, G.; Zhao, C.; Fairbairn, D.; Watson, D.; Ge, M. Multi-GNSS precise point positioning for precision agriculture. *Precis. Agric.* **2018**, *19*, 895–911. [CrossRef]
7. De Aguiar, A.S.P.; dos Santos, F.B.N.; dos Santos, L.C.F.; de Jesus Filipe, V.M.; de Sousa, A.J.M. Vineyard trunk detection using deep learning—An experimental device benchmark. *Comput. Electron. Agric.* **2020**, *175*, 105535. doi:10.1016/j.compag.2020.105535. [CrossRef]
8. Santos, L.C.; Aguiar, A.S.; Santos, F.N.; Valente, A.; Ventura, J.B.; Sousa, A.J. Navigation Stack for Robots Working in Steep Slope Vineyard. In *Advances in Intelligent Systems and Computing*; Springer International Publishing: New York, NY, USA, 2020; pp. 264–285. [CrossRef]
9. Durrant-Whyte, H.; Bailey, T. Simultaneous localization and mapping: Part I. *IEEE Robot. Autom. Mag.* **2006**, *13*, 99–110. [CrossRef]
10. Bailey, T.; Durrant-Whyte, H. Simultaneous localization and mapping (SLAM): Part II. *IEEE Robot. Autom. Mag.* **2006**, *13*, 108–117. [CrossRef]
11. Cadena, C.; Carlone, L.; Carrillo, H.; Latif, Y.; Scaramuzza, D.; Neira, J.; Reid, I.; Leonard, J.J. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. *IEEE Trans. Robot.* **2016**, *32*, 1309–1332. [CrossRef]

12. Nister, D.; Naroditsky, O.; Bergen, J. Visual odometry. In Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004, CVPR 2004, Washington, DC, USA, 27 June–2 July 2004. [CrossRef]
13. Scaramuzza, D.; Fraundorfer, F. Visual Odometry [Tutorial]. *IEEE Robot. Autom. Mag.* **2011**, *18*, 80–92. [CrossRef]
14. Kohlbrecher, S.; Meyer, J.; Graber, T.; Petersen, K.; Klingauf, U.; von Stryk, O. Hector Open Source Modules for Autonomous Mapping and Navigation with Rescue Robots. In *RoboCup 2013: Robot World Cup XVII*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 624–631. [CrossRef]
15. Grisetti, G.; Stachniss, C.; Burgard, W. Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters. *IEEE Trans. Robot.* **2007**, *23*, 34–46. [CrossRef]
16. Dong, J.; Burnham, J.G.; Boots, B.; Rains, G.; Dellaert, F. 4D crop monitoring: Spatio-temporal reconstruction for agriculture. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017. doi:10.1109/icra.2017.7989447. [CrossRef]
17. Hess, W.; Kohler, D.; Rapp, H.; Andor, D. Real-time loop closure in 2D LIDAR SLAM. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016. [CrossRef]
18. Williams, B.; Cummins, M.; Neira, J.; Newman, P.; Reid, I.; Tardós, J. A comparison of loop closing techniques in monocular SLAM. *Robot. Auton. Syst.* **2009**, *57*, 1188–1197. [CrossRef]
19. Smith, R.C.; Cheeseman, P. On the Representation and Estimation of Spatial Uncertainty. *Int. J. Robot. Res.* **1986**, *5*, 56–68. [CrossRef]
20. Durrant-Whyte, H. Uncertain geometry in robotics. *IEEE J. Robot. Autom.* **1988**, *4*, 23–31. [CrossRef]
21. Smith, R.; Self, M.; Cheeseman, P. Estimating Uncertain Spatial Relationships in Robotics. In *Autonomous Robot Vehicles*; Springer: New York, NY, USA, 1990; pp. 167–193. [CrossRef]
22. Leonard, J.J.; Durrant-Whyte, H.F. Simultaneous map building and localization for an autonomous mobile robot. In Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems (IROS '91), Osaka, Japan, 3–5 November 1991; Volume 3, pp. 1442–1447.
23. Thrun, S. Simultaneous Localization and Mapping. In *Robotics and Cognitive Approaches to Spatial Mapping*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 13–41. [CrossRef]
24. Bailey, T.; Nieto, J.; Guivant, J.; Stevens, M.; Nebot, E. Consistency of the EKF-SLAM Algorithm. In Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China, 9–15 October 2006. [CrossRef]
25. Paz, L.; Tardos, J.; Neira, J. Divide and Conquer: EKF SLAM in $O(n)$. *IEEE Trans. Robot.* **2008**, *24*, 1107–1120. [CrossRef]
26. Kalman, R.E. A New Approach to Linear Filtering and Prediction Problems. *J. Basic Eng.* **1960**, *82*, 35–45. [CrossRef]
27. Pinies, P.; Tardos, J.D. Scalable SLAM building conditionally independent local maps. In Proceedings the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Diego, CA, USA, 29 October–2 November 2007. [CrossRef]
28. Pinies, P.; Tardos, J. Large-Scale SLAM Building Conditionally Independent Local Maps: Application to Monocular Vision. *IEEE Trans. Robot.* **2008**, *24*, 1094–1106. [CrossRef]
29. Maybeck, P. *Stochastic Models, Estimation, and Control*; Academic Press: New York, NY, USA; London, UK; Paris, France, 1982.
30. Walter, M.R.; Eustice, R.M.; Leonard, J.J. Exactly Sparse Extended Information Filters for Feature-based SLAM. *Int. J. Robot. Res.* **2007**, *26*, 335–359. [CrossRef]
31. Eustice, R.; Singh, H.; Leonard, J.; Walter, M.; Ballard, R. Visually Navigating the RMS Titanic with SLAM Information Filters. In *Robotics: Science and Systems I*; Robotics: Science and Systems Foundation; Massachusetts Institute of Technology: Cambridge, MA, USA, 2005. [CrossRef]
32. Thrun, S.; Liu, Y. Multi-robot SLAM with Sparse Extended Information Filers. In *Springer Tracts in Advanced Robotics*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 254–266. [CrossRef]
33. Montemerlo, M.; Thrun, S.; Koller, D.; Wegbreit, B. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In Proceedings of the AAAI National Conference on Artificial Intelligence/IAAI, Edmonton, AB, Canada, 28 July–2 August 2002; p. 593598.

34. Montemerlo, M.; Thrun, S.; Koller, D.; Wegbreit, B. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In Proceedings of the IJCAI, Acapulco, Mexico, 9–15 August 2003; pp. 1151–1156.
35. Grisettiyz, G.; Stachniss, C.; Burgard, W. Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation, Barcelona, Spain, 18–22 April 2005. [CrossRef]
36. Lu, F.; Milios, E. Globally Consistent Range Scan Alignment for Environment Mapping. *Auton. Robot.* **1997**, *4*, 333–349. [CrossRef]
37. Bresson, G.; Alsayed, Z.; Yu, L.; Glaser, S. Simultaneous Localization and Mapping: A Survey of Current Trends in Autonomous Driving. *IEEE Trans. Intell. Veh.* **2017**, *2*, 194–220. [CrossRef]
38. Aguiar, A.S.; Santos, F.N.D.; Sousa, A.J.M.D.; Oliveira, P.M.; Santos, L.C. Visual Trunk Detection Using Transfer Learning and a Deep Learning-Based Coprocessor. *IEEE Access* **2020**, *8*, 77308–77320. [CrossRef]
39. Bar-Shalom, Y.; Fortmann, T.E.; Cable, P.G. Tracking and Data Association. *J. Acoust. Soc. Am.* **1990**, *87*, 918–919. [CrossRef]
40. Cox, I.J. A review of statistical data association techniques for motion correspondence. *Int. J. Comput. Vis.* **1993**, *10*, 53–66. [CrossRef]
41. Montemerlo, M.; Thrun, S. Simultaneous localization and mapping with unknown data association using FastSLAM. In Proceedings of the 2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422), Taipei, Taiwan, 14–19 September 2003. [CrossRef]
42. Hähnel, D.; Thrun, S.; Wegbreit, B.; Burgard, W. Towards Lazy Data Association in SLAM. In *Springer Tracts in Advanced Robotics*; Springer: Berlin/Heidelberg, Germay, 2005; pp. 421–431. [CrossRef]
43. Neira, J.; Tardos, J. Data association in stochastic mapping using the joint compatibility test. *IEEE Trans. Robot. Autom.* **2001**, *17*, 890–897. [CrossRef]
44. Thrun, S.; Burgard, W.; Fox, D. A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots. *Mach. Learn.* **1998**, *31*, 29–53. [CrossRef]
45. Zhou, W.; Cao, Z.; Dong, Y. Review of SLAM Data Association Study. In Proceedings of the 2016 International Conference on Sensor Network and Computer Engineering, Xi'an, China, 8–10 July 2016; Atlantis Press: Amsterdam, The Netherlands, 2016. [CrossRef]
46. Tardós, J.D.; Neira, J.; Newman, P.M.; Leonard, J.J. Robust Mapping and Localization in Indoor Environments Using Sonar Data. *Int. J. Robot. Res.* **2002**, *21*, 311–330. [CrossRef]
47. Shan, T.; Englot, B. LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018. [CrossRef]
48. IEEE Standard for Robot Map Data Representation for Navigation. In Proceedings of the IROS2014 (IEEE/RSJ International Conference on Intelligent Robots and Systems) Workshop on "Standardized Knowledge Representation and Ontologies for Robotics and Automation", Chicago, IL, USA, 14–18 September 2014; pp. 3–4. [CrossRef]
49. Yi, C. Map Representation for Robots. *Smart Comput. Rev.* **2012**. [CrossRef]
50. Lowry, S.; Sunderhauf, N.; Newman, P.; Leonard, J.J.; Cox, D.; Corke, P.; Milford, M.J. Visual Place Recognition: A Survey. *IEEE Trans. Robot.* **2016**, *32*, 1–19. [CrossRef]
51. Walter, M.; Hemachandra, S.; Homberg, B.; Tellex, S.; Teller, S. Learning Semantic Maps from Natural Language Descriptions. In *Robotics: Science and Systems IX*; Robotics: Science and Systems Foundation; Technische Universität Berlin: Berlin, Germany, 2013. [CrossRef]
52. Vasudevan, S.; Gächter, S.; Nguyen, V.; Siegwart, R. Cognitive maps for mobile robots—an object based approach. *Robot. Auton. Syst.* **2007**, *55*, 359–371. [CrossRef]
53. dos Santos, F.B.N.; Sobreira, H.M.P.; Campos, D.F.B.; dos Santos, R.M.P.M.; Moreira, A.P.G.M.; Contente, O.M.S. Towards a Reliable Monitoring Robot for Mountain Vineyards. In Proceedings of the 2015 IEEE International Conference on Autonomous Robot Systems and Competitions, Vila Real, Portugal, 8–10 April 2015. [CrossRef]
54. Yousif, K.; Bab-Hadiashar, A.; Hoseinnezhad, R. An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics. *Intell. Ind. Syst.* **2015**, *1*, 289–311. [CrossRef]
55. Fraundorfer, F.; Scaramuzza, D. Visual Odometry: Part II: Matching, Robustness, Optimization, and Applications. *IEEE Robot. Autom. Mag.* **2012**, *19*, 78–90. [CrossRef]

56. Agarwal, S.; Snavely, N.; Seitz, S.M.; Szeliski, R. Bundle Adjustment in the Large. In *Computer Vision—ECCV 2010*; Daniilidis, K., Maragos, P., Paragios, N., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 29–42.
57. Ziegler, J.; Bender, P.; Schreiber, M.; Lategahn, H.; Strauss, T.; Stiller, C.; Dang, T.; Franke, U.; Appenrodt, N.; Keller, C.G.; et al. Making Bertha Drive—An Autonomous Journey on a Historic Route. *IEEE Intell. Transp. Syst. Mag.* **2014**, *6*, 8–20. [CrossRef]
58. Freitas, G.; Zhang, J.; Hamner, B.; Bergerman, M.; Kantor, G. A Low-Cost, Practical Localization System for Agricultural Vehicles. In *Intelligent Robotics and Applications*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 365–375. [CrossRef]
59. Libby, J.; Kantor, G. Deployment of a point and line feature localization system for an outdoor agriculture vehicle. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011. [CrossRef]
60. Duarte, M.; dos Santos, F.N.; Sousa, A.; Morais, R. Agricultural Wireless Sensor Mapping for Robot Localization. In *Advances in Intelligent Systems and Computing*; Springer International Publishing: New York, NY, USA, 2015; pp. 359–370. [CrossRef]
61. Zaman, S.; Comba, L.; Biglia, A.; Aimonino, D.R.; Barge, P.; Gay, P. Cost-effective visual odometry system for vehicle motion control in agricultural environments. *Comput. Electron. Agric.* **2019**, *162*, 82–94. [CrossRef]
62. Habibie, N.; Nugraha, A.M.; Anshori, A.Z.; Masum, M.A.; Jatmiko, W. Fruit mapping mobile robot on simulated agricultural area in Gazebo simulator using simultaneous localization and mapping (SLAM). In Proceedings of the 2017 International Symposium on Micro-NanoMechatronics and Human Science (MHS), Nagoya, Japan, 3–6 December 2017. [CrossRef]
63. Younse, P.; Burks, T. Greenhouse Robot Navigation Using KLT Feature Tracking for Visual Odometry. *Agric. Eng. Int. CIGR J.* **2007**, *IX*, 62744503.
64. Bayar, G.; Bergerman, M.; Koku, A.B.; ilhan Konukseven, E. Localization and control of an autonomous orchard vehicle. *Comput. Electron. Agric.* **2015**, *115*, 118–128. [CrossRef]
65. Le, T.; Gjevestad, J.G.O.; From, P.J. Online 3D Mapping and Localization System for Agricultural Robots. *IFAC-PapersOnLine* **2019**, *52*, 167–172. [CrossRef]
66. Cheein, F.A.; Steiner, G.; Paina, G.P.; Carelli, R. Optimized EIF-SLAM algorithm for precision agriculture mapping based on stems detection. *Comput. Electron. Agric.* **2011**, *78*, 195–207. [CrossRef]
67. Chebrolu, N.; Lottes, P.; Labe, T.; Stachniss, C. Robot Localization Based on Aerial Images for Precision Agriculture Tasks in Crop Fields. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019. [CrossRef]
68. Blok, P.M.; van Boheemen, K.; van Evert, F.K.; IJsselmuiden, J.; Kim, G.H. Robot navigation in orchards with localization based on Particle filter and Kalman filter. *Comput. Electron. Agric.* **2019**, *157*, 261–269. [CrossRef]
69. Piyathilaka, L.; Munasinghe, R. Vision-only outdoor localization of two-wheel tractor for autonomous operation in agricultural fields. In Proceedings of the 2011 6th International Conference on Industrial and Information Systems, Kandy, Sri Lanka, 16–19 August 2011. [CrossRef]
70. Iqbal, J.; Xu, R.; Sun, S.; Li, C. Simulation of an Autonomous Mobile Robot for LiDAR-Based In-Field Phenotyping and Navigation. *Robotics* **2020**, *9*, 46. [CrossRef]
71. Bietresato, M.; Carabin, G.; D'Auria, D.; Gallo, R.; Ristorto, G.; Mazzetto, F.; Vidoni, R.; Gasparetto, A.; Scalera, L. A tracked mobile robotic lab for monitoring the plants volume and health. In Proceedings of the 2016 12th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA), Auckland, New Zealand, 29–31 August 2016. [CrossRef]
72. Utstumo, T.; Urdal, F.; Brevik, A.; Dørum, J.; Netland, J.; Overskeid, Ø.; Berge, T.W.; Gravdahl, J.T. Robotic in-row weed control in vegetables. *Comput. Electron. Agric.* **2018**, *154*, 36–45. [CrossRef]
73. Fountas, S.; Mylonas, N.; Malounas, I.; Rodias, E.; Santos, C.H.; Pekkeriet, E. Agricultural Robotics for Field Operations. *Sensors* **2020**, *20*, 2672. [CrossRef]
74. Qian, C.; Liu, H.; Tang, J.; Chen, Y.; Kaartinen, H.; Kukko, A.; Zhu, L.; Liang, X.; Chen, L.; Hyyppä, J. An Integrated GNSS/INS/LiDAR-SLAM Positioning Method for Highly Accurate Forest Stem Mapping. *Remote Sens.* **2016**, *9*, 3. [CrossRef]
75. Hussein, M.; Renner, M.; Iagnemma, K. Global Localization of Autonomous Robots in Forest Environments. *Photogramm. Eng. Remote Sens.* **2015**, *81*, 839–846. [CrossRef]

76. Li, Q.; Nevalainen, P.; Queralta, J.P.; Heikkonen, J.; Westerlund, T. Localization in Unstructured Environments: Towards Autonomous Robots in Forests with Delaunay Triangulation. *Remote Sens.* **2020**, *12*, 1870. [CrossRef]
77. Pierzchała, M.; Giguère, P.; Astrup, R. Mapping forests using an unmanned ground vehicle with 3D LiDAR and graph-SLAM. *Comput. Electron. Agric.* **2018**, *145*, 217–225. [CrossRef]
78. Rossmann, D.I.J. *Navigation of Mobile Robots in Natural Environments: Using Sensor Fusion in Forestry*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 43–52.
79. Miettinen, M.; Ohman, M.; Visala, A.; Forsman, P. Simultaneous Localization and Mapping for Forest Harvesters. In Proceedings of the 2007 IEEE International Conference on Robotics and Automation, Roma, Italy, 10–14 April 2007. [CrossRef]
80. Öhman, M.; Miettinen, M.; Kannas, K.; Jutila, J.; Visala, A.; Forsman, P. Tree Measurement and Simultaneous Localization and Mapping System for Forest Harvesters. In *Springer Tracts in Advanced Robotics*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 369–378. [CrossRef]
81. Hyyti, H.; Visala, A. Feature Based Modeling and Mapping of Tree Trunks and Natural Terrain Using 3D Laser Scanner Measurement System. *IEAC Proc. Vol.* **2013**, *46*, 248–255. [CrossRef]
82. Hyyti, H.; Öhman, M.; Miettinen, M.; Visala, A. Heuristic correlation based laser odometry method for unconstructed environment. In Proceedings of the IASTED International Conference on Robotics and Applications, Cambridge, MA, USA, 2–4 November 2009; pp. 194–200.
83. Tang, J.; Chen, Y.; Kukko, A.; Kaartinen, H.; Jaakkola, A.; Khoramshahi, E.; Hakala, T.; Hyyppä, J.; Holopainen, M.; Hyyppä, H. SLAM-Aided Stem Mapping for Forest Inventory with Small-Footprint Mobile LiDAR. *Forests* **2015**, *6*, 4588–4606. [CrossRef]
84. Chebrolu, N.; Lottes, P.; Schaefer, A.; Winterhalter, W.; Burgard, W.; Stachniss, C. Agricultural robot dataset for plant classification, localization and mapping on sugar beet fields. *Int. J. Robot. Res.* **2017**, *36*, 1045–1052. [CrossRef]
85. Kragh, M.; Christiansen, P.; Laursen, M.; Larsen, M.; Steen, K.; Green, O.; Karstoft, H.; Jørgensen, R. FieldSAFE: Dataset for Obstacle Detection in Agriculture. *Sensors* **2017**, *17*, 2579. [CrossRef]
86. Ali, I.; Durmush, A.; Suominen, O.; Yli-Hietanen, J.; Peltonen, S.; Collin, J.; Gotchev, A. FinnForest dataset: A forest landscape for visual SLAM. *Robot. Auton. Syst.* **2020**, *132*, 103610. [CrossRef]
87. Pire, T.; Mujica, M.; Civera, J.; Kofman, E. The Rosario dataset: Multisensor data for localization and mapping in agricultural environments. *Int. J. Robot. Res.* **2019**, *38*, 633–641. [CrossRef]
88. Reis, R.; dos Santos, F.N.; Santos, L. Forest Robot and Datasets for Biomass Collection. In *Advances in Intelligent Systems and Computing*; Springer International Publishing: New York, NY, USA, 2019; pp. 152–163. [CrossRef]

2.2 Final remarks

The article presented in this chapter consists of a deep literature review regarding localization and mapping in agriculture and forestry. This study showed that in the past years many works approached this research topic, which means that there is an interest in developing new and more robust robotic solutions for agriculture and forestry. Also, this survey article leads to the conclusion that there is still gaps in this area, and much work yet to be developed. Specifically, 3D localization is not approached in the majority of the reviewed works. This means that in environments with steep inclinations or considerable altitude differentials, the algorithms are not able to estimate the roll, pitch and elevation components of the robot's pose. This also means that a proper 3D reconstruction of the environment is not built since not all the 6-DoF of the robot's pose are known. In addition, advanced mapping solutions such as topological or semantic are commonly not used. This thesis formulates several solutions to tackle this state-of-the-art gaps and pushes forward the development of robotics in agriculture.

3 Camera to LiDAR calibration

This chapter presents the approach formulated to solve the camera-to-LiDAR extrinsic calibration problem. This solution allows the use of both sensor modalities in the multi-layer mapping architecture that will be presented in chapter 5.

3.1 A Camera to LiDAR calibration approach through the optimization of atomic transformations

During the development of this work two main sensor types were used for localization and mapping: 3D LiDARs and cameras. As will be described in the next chapters the 3D LiDAR is the main sensor used for the metric mapping and 6-DoF localization. Also, stereo and monocular camera systems were used to detect semantic features in the vineyard. To be able to unify the features extracted from both sensors and to create maps that consider all the modalities of features extracted simultaneously, it is essential to have a precise estimation of the spatial transformation between the sensors present in the robot. One common approach is to manually measure the spatial relationship between all the sensors and register the homogeneous transformation between them. This approach has the disadvantage of being manual and presenting a considerable associated error specially in rotational components.

For this reason, an algorithm for extrinsic calibration of cameras and 3D LiDARs was developed and published in the Elsevier Expert Systems with Applications journal. This paper entitled *A Camera to LiDAR calibration approach through the optimization of atomic transformations* (Aguiar et al., 2021a) is presented bellow. This work is inserted in an open-source extrinsic calibration software framework called ATOM[1] (de Oliveira et al., 2022). The main novelty of the proposed system is the calibration of multiple sensors ($N \geq 2$) simultaneously without changing the topology of the transformation tree of the robot. Most of the state-of-the-art algorithms solve the extrinsic calibration problem in a pairwise fashion, i.e., they estimate all the combinations of transformations between all the sensors to calibrate. The proposed system calibrates all the sensors present in the robot simultaneously in a sensor to calibration pattern manner through the optimization of atomic transformations. ATOM is general enough to accommodate any modality of sensor by simply designing an objective function for it.

[1] https://github.com/lardemua/atom

A Camera to LiDAR calibration approach through the Optimization of Atomic Transformations

André Silva Pinto de Aguiar[a,c], Miguel Armando Riem de Oliveira[b], Eurico Farinha Pedrosa[b], Filipe Baptista Neves dos Santos[a]

[a] *INESC TEC - INESC Technology and Science; 4200-465, Porto, Portugal, {andre.s.aguiar, fbsantos}@inesctec.pt*
[b] *Institute of Electronics and Informatics Engineering of Aveiro, University of Aveiro, Portugal, {mriem, efp}@ua.pt*
[c] *School of Science and Technology, University of Trás-os-Montes e Alto Douro; 5000-801 Vila Real, Portugal*

Abstract

This paper proposes a camera-to-3D Light Detection And Ranging calibration framework through the optimization of atomic transformations. The system is able to simultaneously calibrate multiple cameras with Light Detection And Ranging sensors, solving the problem of Bundle. In comparison with the state-of-the-art, this work presents several novelties: the ability to simultaneously calibrate multiple cameras and LiDARs; the support for multiple sensor modalities; the calibration through the optimization of atomic transformations, without changing the topology of the input transformation tree; and the integration of the calibration framework within the Robot Operating System (ROS) framework. The software pipeline allows the user to interactively position the sensors for providing an initial estimate, to label and collect data, and visualize the calibration procedure. To test this framework, an agricultural robot with a stereo camera and a 3D Light Detection And Ranging sensor was used. Pairwise calibrations and a single calibration of the three sensors were tested and evaluated. Results show that the proposed approach produces accurate calibrations when compared to the state-of-the-art, and is robust to harsh conditions such as inaccurate initial guesses or small amount of data used in calibration. Experiments have shown that our optimization process can handle an angular error of approximately 20 degrees and a translation error of 0.5 meters, for each sensor. Moreover, the proposed approach is able to achieve state-of-the-art results even when calibrating the entire system simultaneously.

Keywords: Computer Vision, Geometric Optimization, Atomic Transformations

1. Introduction

Nowadays, autonomous robotic systems are endowed with high-quality onboard sensors of different modalities, i.e. sensors that output different types of data, such as cameras and Light Detection And Ranging (LiDAR) sensors. To move autonomously while safely avoiding any kind of obstacle, these vehicles need to perform complex tasks such as Simultaneous Localization and Mapping (Durrant-Whyte & Bailey, 2006) and Path Planning (Sariff & Buniyamin, 2006) which require calibrated sensor data. The quality of the onboard sensors data is also crucial, since the robot should have a clear perception of the environment. For example, in agriculture, the automation of tasks such as crop monitoring or harvesting is a complex challenge that requires data of high quality sensors (dos Santos et al., 2016), such as, for example long-range 3D LiDARs. To perform data fusion, and take advantage of all the sensors present in the robotic system, it

is essential to know the spatial relationship between all sensors (Melendez-Pastor et al., 2017; Majumder & Pratihar, 2018). To do so, the most common approach is to perform *extrinsic calibration*, i.e., to compute the transformation between the sensors' reference frames. The standard approach to perform extrinsic calibration is to find associations between data incoming from each sensor to be calibrated. Thus, a cost function that minimizes the error between associations is used. Most of the calibration procedures use a pattern that can be detected independently of the sensor modality, so that data correspondences can be found. Using these concepts, camera to 3D LiDAR extrinsic calibration have been approached in several works. A minority of works perform calibration without using a pattern. In those, the characteristics of the environment are used as features to compute intrinsic and extrinsic calibration. In autonomous driving, for example, lane detection and vanishing point tracking are common approaches (Badue et al., 2021; de Paula et al., 2014; Álvarez et al., 2014).

The great majority of the works found in the literature follow a pairwise calibration between a monocular camera and a 3D LiDAR, a concept introduced by Huang & Barth (2009). In this work, the extrinsic coefficients are computed solving a closed-form equation, and refined with a maximum likelihood estimation. Similarly, Verma et al. (2019) use a standard chessboard to calibrate a perspective/fisheye camera and a 3D LiDAR using a Genetic Algorithm. Wang et al. (2017) propose a work where the corners of the pattern are automatically detected for both a panoramic camera and a 3D LiDAR so that the calibration can be performed. For the LiDAR case, authors propose a detection based on the intensity of reflectance of the beams. Fremont & Bonnifait (2008); Guindel et al. (2017a) use circle-based patterns to perform the extrinsic calibration. Mirzaei et al. (2012) propose the estimation of a 3D LiDAR intrinsic parameters, as well as the extrinsic calibration with a monocular camera, through the minimization of a non-linear least squares cost function. The calibration is used to build photorealistic 3D reconstruction of indoor and outdoor scenes. Pandey et al. (2010) calibrate a 3D LiDAR with an omnidirectional camera also using a standard planar pattern. To calibrate the system, the sensors should observe the pattern from at least three different points of view. With this input, the extrinsic coefficients are calculated with a non-linear optimization technique. With the same purpose, Huang & Grizzle (2020) use a pattern of known dimension and geometry and estimates the pattern to LiDAR pose automatically using a fitting algorithm.

Although all these works perform successful extrinsic calibrations between 3D range sensors and monocular cameras, pairwise calibration is a major shortcoming since most robotic systems present more than two sensors to be calibrated. In a system with more than two sensors, one would have to combine multiple pairwise calibrations. The problem is that the number of the combinations of pairwise calibrations required grow quickly. For example, Zhou et al. (2018) presente a system with a 3D LiDAR and a stereo camera system. However, to calibrate the three sensors (LiDAR and two cameras), two calibrations have to be performed: LiDAR to left camera, and LiDAR to the right camera. In the same way, with the purpose of fusing point clouds of multiple stereo cameras, Dhall et al. (2017) use a 3D LiDAR to perform pairwise calibration with all the cameras in the system. Only after obtaining the transformation between the range sensor and each camera of the stereo system, the transformation between the stereo cameras can be found. Then the point clouds can be fused. Similarly, su Kim & Park (2019) perform six pairwise calibrations between a 3D LiDAR and six monocular cameras mounted in an hexagonal plate that constitute an omnidirectional camera.

To overcome this limitation, few works exist that consider multi-sensor, multi-modal calibration in a non-pairwise fashion. For example, Zuniga-Noel et al. (2019) propose a method to estimate the extrinsic calibration between multiple sensors such as LiDARs, depth cameras and

RGB cameras. The calibration procedure is separated in two parts: a motion-based approach that estimates 2D extrinsic parameters and a method that uses the observation of the ground plane to estimate the remaining ones. It is worth noting that this framework requires the robotic platform to be moving. Liao et al. (2017) propose a joint objective function to simultaneously calibrate three RGB cameras with respect to an RGB-D camera. Rehder et al. (2016), propose an approach for joint estimation of both temporal offsets and spatial transformations between sensors. This approach is one of few that is not designed for a particular set of sensors, since its methodology does not rely on unique properties of specific sensors. It is able to calibrate systems containing both cameras and LiDARs. Pradeep et al. (2014), present a joint calibration of the joint offsets and the sensors locations for a PR2 robot. This method takes sensor uncertainty into account and is modelled in a similar way to the bundle adjustment problem.

The two major shortcomings of the state-of-the art in extrinsic calibration are: most of the methods perform pairwise calibration, which can be exhaustive for a robotic system with many sensors to be calibrated; and the majority of the works are focused on specific sensor modalities, rather than working in a more general way. To overcome these issues, this work proposes an extrinsic calibration framework with the following contributions:

- Support for calibration of multiple sensors (i.e. $N \geq 2$);
- The ability to handle multiple sensor modalities;
- Calibration without changing the topology of the input transformation tree;
- Integration of the system within the Robot Operating System (ROS) framework, with interactive tools to collect data, set the initial estimates, and visualize the calibration.

Our previous works have focused on the calibration of intelligent vehicles. These platforms are often characterized by the large amount of sensors of different modalities mounted onboard. As such, these previous works presented a methodology based on atomic transformations for multi-sensor, multi-modal robotic systems (Guindel et al., 2017b; Oliveira et al., 2020a,b). In this work, we extend our framework - Atomic Transformation Optimization Method (ATOM) [1] (Oliveira et al., 2020a) - to also consider the calibration of 3D LiDARs along with the other supported modalities. Atomic transformations are geometric transformations that are not aggregated, i.e., they are indivisible. As such, this article presents the methodologies implemented to simultaneously calibrate a 3D LiDAR sensor with multiple cameras using a Bundle Adjustment optimization scheme (Agarwal et al., 2010). As our approach is not focused on a single robotic platform, we present the calibration of a different robotic platform in comparison with our previous works - the agricultural robot AgRob V16 (de Aguiar et al., 2020; Santos et al., 2019).

The remainder of this paper is organized as follows: Sec. 2 details the optimization procedure and how it is cast as a Bundle Adjustment problem; Sec. 3 describes the ROS (Quigley et al., 2009) calibration setup, i.e., the steps from the robotic platform configuration until the data collection; Sec. 4 details the experimental results; and finally, Sec. 5 provides conclusions and future work.

[1] `https://github.com/lardemua/atom`

2. Proposed Approach

ATOM is a calibration framework that simultaneously calibrates sensors of different modalities though the optimization of atomic transformations. This concept is supported by a well-defined optimization pipeline, that defines a set of optimization parameters and minimizes a cost function that supports different input modalities. This function f, which depends on the optimization parameters $\mathbf{\Phi}$, is known as the *objective function*. Our approach minimizes f to calibrate all the sensors of generic multi-modal robotic platforms simultaneously. In this process, the definition of a tree graph which contains topological information about the relationship between reference frames is required. In this tree, nodes are reference frames and edges correspond to the transformation between nodes. This data structure allows for the definition of unique paths between the graph nodes, i.e., enables an efficient retrieval of the transformations between any two frames in the tree. Figure 1 represents the robotic platform to be calibrated (AgRob V16) with its respective referential frames, and the transformation tree considered for the calibration.

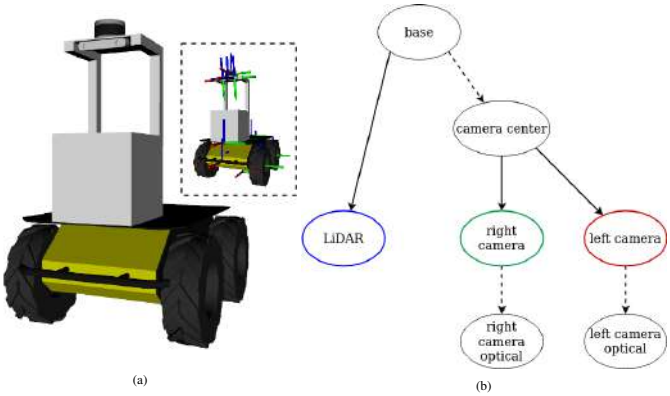


Figure 1: **(a)** AgRob V16 model and the respective referential frames represented as red-green-blue axes. **(b)**: Transformation tree for AgRob V16 robotic platform. The majority of the frames not to be calibrated were omitted for simplicity. Each sensor has an associated atomic transformation, denoted by the solid edges. Dashed edges denote transformations that are not optimized (they can be static or dynamic). Each sensor has a corresponding link to which the data it collects is attached, denoted in the figure by solid thin ellipses. Very few approaches in the literature are capable of calibrating such a system while preserving the initial structure of the transformation graph.

The design of the transformation tree leads to the definition of the optimization parameters to calibrate the system. An extrinsic calibration can be viewed as a pose estimation problem, where the pose of each sensor is estimated. Thus, the set of parameters to optimize $\mathbf{\Phi}$, must together define the pose of each sensor. To perform such a calibration, we propose to maintain the initial structure of the transformation tree, calibrating only one atomic transformation per sensor. Since the system contains camera sensors, it is also possible to introduce the intrinsic parameters of each camera in the set $\mathbf{\Phi}$. In this way, the set of parameters will be composed of different modalities and so, there is the need to design an objective function that is able to characterize sensors of in a multi-modal fashion.

As previously discussed, pairwise approaches for projecting the objective function result in complex graphs with many combinations of relationship definitions. For every existing pair of sensors, these relationships must be established according to the combined modality of the pair of sensors, which leads to a problem of scalability for which there is no solution in the literature. To solve this issue, we formulate our solution in Bundle Adjustment problem, in that the structure of the objective function is designed in *a sensor to calibration pattern* paradigm. Also, for every collection of data, the transformation that takes the corners of the calibration pattern to the world is optimized. In other words, the poses of the pattern are jointly estimated with the poses of the sensors. To perform this iterative procedure, the set of optimization parameters $\mathbf{\Phi}$ must be initialized. The first guess for the pattern pose is obtained w.r.t. one of the cameras to be calibrated, resulting in a transformation ${}^{cam_i}\boldsymbol{T}_p$, where p denotes pattern and ${}^{a}\boldsymbol{T}_b$ represents the transformation from frame b to a. As will be detailed later on, our calibration framework allows the definition of an initial guess for the pose of each sensor and, consequently, for the transformations to be calibrated. With this definition, it is possible to compute the pose of any particular sensor j as an aggregate homogeneous transformation ${}^{w}\boldsymbol{A}_{s_j}$, obtained from the chain of transformations for that particular sensor present in the topological transformation graph:

$$ {}^{w}\boldsymbol{A}_{s_j} = \prod_{i \in R} {}^{i}\boldsymbol{T}_{i+1} = \prod_{i \in K} {}^{i}\boldsymbol{T}_{i+1} \cdot {}^{child}\hat{\boldsymbol{T}}_{parent} \cdot \prod_{i \in L} {}^{i}\boldsymbol{T}_{i+1}, \tag{1} $$

where ${}^{child}\boldsymbol{T}_{parent}$ represents the transformation to be calibrated, ${}^{i}T_{i+1}$ for $i \in K$ represent the prior links to the frame *parent*, ${}^{i}T_{i+1}$ for $i \in L$ the later to the frame *child*, and R is the set that contains all the frames present in the chain of transformation of sensor j. So, to obtain the homogeneous transformation from the pattern to the world, the following calculation is applied:

$$ {}^{w}\boldsymbol{T}_p = {}^{w}\boldsymbol{A}_{cam_m} \cdot {}^{cam_m}\boldsymbol{T}_p \ , \tag{2} $$

where p refers to the pattern, w states for world, and cam_m for the mth camera sensor. So, the set of parameters $\mathbf{\Phi}$ to be optimized is composed of the transformation represented in (2) along with the poses of each sensor to be calibrated, and, in the case of cameras, their intrinsics and distortion parameters:

$$ \begin{aligned} \mathbf{\Phi} = \Big[& \overbrace{\boldsymbol{x}_{m=1}, \boldsymbol{r}_{m=1}, \boldsymbol{i}_{m=1}, \boldsymbol{d}_{m=1}, ..., \boldsymbol{x}_{m=M}, \boldsymbol{r}_{m=M}, \boldsymbol{i}_{m=M}, \boldsymbol{d}_{m=M}}^{\text{cameras}}, \\ & \overbrace{\boldsymbol{x}_{n=1}, \boldsymbol{r}_{n=1}, ..., \boldsymbol{x}_{n=N}, \boldsymbol{r}_{n=N}}^{\text{LiDARs}}, \overbrace{\cdots}^{\text{Other modalities}} , \overbrace{\boldsymbol{x}_{k=1}, \boldsymbol{r}_{k=1}, ..., \boldsymbol{x}_{k=K}, \boldsymbol{r}_{k=K}}^{\text{Calibration pattern}} \Big], \end{aligned} \tag{3} $$

where m refers to the mth camera to be calibrated, n states for the nth LiDAR to be calibrated, k refers to the pattern detection of the kth collection of data, $\boldsymbol{x}$ is a translation vector $[t_x, t_y, t_z]$, $\boldsymbol{r}$ is a rotation represented thought the angle-axis parameterization $[r_1, r_2, r_3]$ (where the vector $\boldsymbol{r}$ is used to represent the axis and its norm represents the angle), $\boldsymbol{i}$ is a vector with each camera intrinsic parameters $[c_x, c_y, f_x, f_y]$, and $\boldsymbol{d}$ is a vector of each camera distortion coefficients $[d_0, d_1, d_2, d_3, d_4]$. The intrinsic and distortion parameters of each camera can be initialized using any camera calibration toolbox, or in some cases, these parameters are also provided by the manufacture. The angle-axis parameterization was chosen because it has three components and three degrees of freedom, which means that it does not introduce more sensitivity than the one inherent to the problem itself (Hornegger & Tomasi, 1999), unlike the rotation matrix which has

nine degrees of freedom for the also three components, or the euler angles that loose a degree of freedom when two axis are aligned. Using angle-axis representation, we have six optimization parameters per sensor that represent the pose of each one, i.e., the geometric transformation that will be calibrated. The optimization procedure, as will be explained, consists of the minimization of an *objective function* by the definition of residuals that are calculated as an error (in pixels for RGB cameras and in meters for 3D LiDARs) between the re-projected position of the calibration pattern, and the position of the pattern detected by each sensor.

2.1. Objective function

To be able to consider multiple modalities in the same optimization process, we propose to structure the *objective function* $F(\mathbf{\Phi})$ as a composition of as many sub-functions $f_i(.)$ as desired modalities. The objective function $F(\mathbf{\Phi})$ from (4) is minimized using a non-linear least squares approach[2]. Least-squares finds a local minimum of a scalar cost function, with bounds on variables, by having an m-dimensional real residual function on n real variables. As such, we choose this minimization approach as its is the best fit for our problem. So, for each new modality added to the calibration, a sub-function associated with it is designed and incorporated in $F(\mathbf{\Phi})$, which allows for the minimization of the error associated with the pose of sensor of that specific modality. This is one of the reasons why the proposed approach is scalable. Thus, the optimization procedure can be defined as:

$$\underset{\mathbf{\Phi}}{arg\,min}\; F(\mathbf{\Phi}) = \underset{\mathbf{\Phi}}{arg\,min}\; \frac{1}{2}\sum_{i} \|f_i(\{\mathbf{\Phi}_i\})\|^2, \tag{4}$$

where $f_i(.)$ is the objective sub-function for the i-th sensor considering the set of k optimization parameters $\{\mathbf{\Phi}_i\}$. Thus, the final cost to be minimized is computed by the sum of the squared sub-function values for each set of optimization parameters. The value for all these sub-functions is a vector with the residuals associated to whit re-projection of the points of the calibrated pattern. For our use-case, the goal is to calibrate a stereo camera system (two cameras) and a 3D LiDAR sensor. So, the *objective function* is composed of the vector values of three sub-functions, two for the cameras and one for the 3D LiDAR. Each sub-function is detailed in the next sub-sections.

2.1.1. Camera modality sub-function

When the sensors to be calibrated are cameras, their calibration is performed as a bundle adjustment (Agarwal et al., 2010), as described in our previous work, Oliveira et al. (2020b). Thus, the created sub-function is based on the average geometric error corresponding to the image distance (in pixels) between a projected point and a detected one. So, the goal of the cost sub-function for camera sensors is to adjust the initial estimate for the intrinsic and distortion parameters, and position of the pattern corners, in order to minimize the re-projection error f_{cam}, given by:

$$f_{cam} = \left[\|\boldsymbol{x}_{c=1} - \hat{\boldsymbol{x}}_{c=1}\|. \ldots, \|\boldsymbol{x}_{c=C} - \hat{\boldsymbol{x}}_{c=C}\|\right], \tag{5}$$

where $\|.\|$ represents the Euclidean distance between two vectors, c is the index of the pattern corners, $\boldsymbol{x_c}$ denotes the ground-truth pixel coordinates of the measured points given by the pattern

[2]In this work we used the least-squares solver provided by SciPy: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.least_squares.html`.

detection, and $\hat{\boldsymbol{x}}$ are the projected points, given by the relationship between a 3D point in the world and its projection onto the image.

To perform such calibration, 3D pattern points have to be found and re-projected onto the image plane. For each collection of data, the camera(s) to be calibrated capture the pattern. By knowing the real size of the pattern, and the size of each square that composes it, the 3D coordinates of the corners can be found in the local pattern reference frame. Then, each corner is located in the plane $z = 0$, since the corners are in the XoY plane of the local pattern reference frame. Thus, each corner in the local pattern referential frame $\boldsymbol{p}^p$ is transformed to the camera referential frame as follows:

$$\boldsymbol{p}^{cam} = {}^{cam}\boldsymbol{T}_w \cdot {}^{w}\boldsymbol{T}_p \cdot \boldsymbol{p}^p. \tag{6}$$

Note that, $\boldsymbol{p}^{cam}$ and $\boldsymbol{p}^p$ are homogeneous vectors of the 3D corner coordinates in each reference frame, so that (6) is valid. Finally, to re-project each 3D corner from camera's reference frame $\boldsymbol{p}^{cam}_{c=i}$ to the image plane, taking into account each camera intrinsic and distortion parameters, the pinhole camera model (Sturm, 2014) is used:

$$\hat{\boldsymbol{x}}_{c=i} = \boldsymbol{K} \cdot \boldsymbol{p}^{cam''}_{c=i}, \tag{7}$$

where $\boldsymbol{K}$ is the matrix that contains the intrinsic parameters $\boldsymbol{i}$ and,

$$\boldsymbol{p}^{cam'}_{c=i} = \boldsymbol{p}^{cam}_{c=i} \cdot \frac{1}{z_{cam}} = \left[\frac{x_{cam}}{z_{cam}},\ \frac{y_{cam}}{z_{cam}},\ 1\right]^T, \tag{8}$$

$$\boldsymbol{p}^{cam''}_{c=i} = \begin{bmatrix} x_{cam'} \cdot (1 + d_0 l^2 + d_1 l^2 + d_4 l^6) + 2d_2 \cdot x_{cam'} y_{cam'} + d_3 \cdot (l^2 + 2x^2_{cam'}) \\ y_{cam'} \cdot (1 + d_0 l^2 + d_1 l^2 + d_4 l^6) + d_2 \cdot (l^2 + 2y^2_{cam'}) + 2d_3 \cdot x_{cam'} y_{cam'} \end{bmatrix}, \tag{9}$$

where $l = \sqrt{(\frac{x_{cam}}{z_{cam}})^2 + (\frac{y_{cam}}{z_{cam}})^2}$, and d_j is the *jth* component of the distortion vector $\boldsymbol{d}$. From (6) to (9), it is possible to conclude that all the desired parameters to optimize are being considered: the pattern to world transformation ${}^{w}\boldsymbol{T}_p$ present in (6) that can be computed as the inverse of (2); the world to camera transformation ${}^{cam}\boldsymbol{T}_w$ also present in (6); and finally, the intrinsic $\boldsymbol{i}$ and distortion parameters $\boldsymbol{d}$ considered in (7)-(9).

The use of these parameters to project the 3D corners in the image plane, together with the minimization of the geometric re-projection error, lead to the parameter configuration that optimize the sub-function f_{cam}. Thus, it is expected that the re-projected points become closer to the ground-truth corners during the optimization. Figure 2 shows the difference between the initial position of the pattern corners, and the final position of these same projected points, after the optimization has been completed.

It is possible to observe that the pixels corresponding to the projection of the final position of the points (dots in Fig. 2) almost perfectly match the ground-truth point (squares in Fig. 2).

2.1.2. 3D LiDAR sub-function

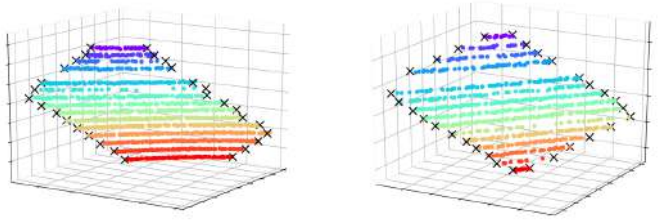
For the case of 3D LiDARs, the sub-function f_{lidar} considers two types of residuals: orthogonal distance and limit points distance. To compute both, this approach also uses the calibration pattern and, in specific, its boundary points. As will be detailed later on, our calibration framework has a semi-automatic labelling procedure that allows to save, for each collection of data, the LiDAR 3D points that are on the pattern. As in case of the camera sensor, this approach formulates the cost sub-function by minimizing the residuals w.r.t. some ground-truth. Here, the

Figure 2: Difference between the initial position of the pattern corners, and the final position of these same projected points, after the optimization has been completed. Squares denote the position of the detected pattern corners; crosses denote the initial position of each projected corner; points denote the current position of the projected points.

ground-truth 3D points are, once again, generated in the pattern reference frame by knowing the three dimensional structure of the pattern, such as its height and width. Thus, by knowing the size of the pattern, the size of each pattern square, and the pattern origin (bottom left corner), the coordinates of the boundary points defined in the local pattern's reference frame are computed. It is important to note that, the size of the board between the pattern grid and the end of the physical pattern had to be measured so that this step could be implemented. Also, as explained before, all the calculated pattern limit points have coordinate $z = 0$, since the pattern's reference frame is in the XoY plane. After calculating the ground-truth boundary points of the pattern, two things are required: the pattern boundary points observed by the 3D LiDAR sensor and the homogeneous transformation that converts 3D points from the LiDAR referential frame to the local pattern reference frame.

Given a set of labelled 3D LiDAR cartesian points on the pattern $\boldsymbol{p}^{lidar} = \left[x_{c=i}, y_{c=i}, z_{c=i}\right]$, the boundary points are calculated using a spherical parameterization for each 3D point. After computing the spherical coordinates of each 3D LiDAR point on pattern $\boldsymbol{p}^{s,lidar} = \left[r_{c=i}, \theta_{c=i}, \phi_{c=i}\right]$, two limit points are calculated considering the set of 3D LiDAR points on pattern belonging to a given horizontal scan of the original point cloud. As the labelled set of 3D LiDAR points on pattern is an unordered point cloud, the horizontal scans are computed by clustering the points considering their θ value. So, points with the same θ value belong to the same horizontal scan. Finally, to extract the two limit points per horizontal scan, the ϕ component maximum and minimum values of each set are computed, resulting in the two most distant points, corresponding to points in the pattern boundaries. The result of this procedure is represented in Fig. 3.

The final step before computing the residuals that constitute the cost sub-function f_{lidar} is to convert the set of labelled 3D LiDAR points on pattern, as well as the computed boundary points, to the patterns' reference frame. This is done using the homogeneous transformations computed



(a) (b)

Figure 3: Two examples of the pattern boundary points extraction from LiDAR data. The colored points represent the clustered LiDAR scans considering the spherical component θ, and the crosses represent the boundary points extracted using the maximum and minimum values of the spherical component ϕ, for each cluster.

in (1) and (2):

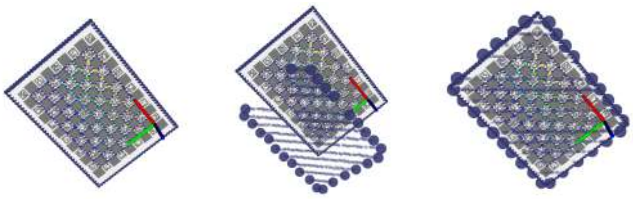
$$\boldsymbol{p}^{p} = {}^{p}\boldsymbol{T}_{w} \cdot {}^{w}\boldsymbol{A}_{lidar} \cdot \boldsymbol{p}^{lidar}, \tag{10}$$

where ${}^{p}\boldsymbol{T}_{w}$ is the transformation matrix from the world to the pattern reference frame, and ${}^{w}\boldsymbol{A}_{lidar}$ the transformation matrix from the world to the LiDAR sensor reference frame. Similarly to the cameras' case, (10) shows that the optimization parameters include the sensor pose and the pattern pose. After this, the two residual types can be computed. The first, orthogonal distance, is the absolute z value of the coordinates of the projected 3D LiDAR points. As they are on patterns' referential frame, it is intended that their z coordinate is zero. Therefore, any value different from zero means that the optimization parameters (sensor pose and pattern pose) are not yet correct. The second residual type is the Euclidean distance of x and y components between the ground-truth pattern boundary points, and the LiDAR 3D points on pattern boundary calculated as described before. For each LiDAR point on the pattern boundary, the residual is computed as the distance between x and y coordinates, in the calibration pattern frame, of the respective LiDAR boundary point and the closest point that belongs to the limit of the physical board that is being detected. This being said, the 3D LiDAR cost sub-function is as follows:

$$f_{lidar} = \left[\left\{ \left| z_{l=1}^{lidar,p} \right|, \| \boldsymbol{p}_{q=1,xy}^{boardlimit} - \boldsymbol{p}_{q=1,xy}^{p} \| \right\}, \ldots, \left\{ \left| z_{l=L}^{lidar,p} \right|, \| \boldsymbol{p}_{q=Q,xy}^{boardlimit} - \boldsymbol{p}_{q=Q,xy}^{p} \| \right\} \right], \tag{11}$$

where $z_{l=i}^{lidar,p}$ is the z coordinate of the *ith* 3D LiDAR point projected into the patterns' referential frame, $\boldsymbol{p}_{q=j,xy}^{boardlimit}$ are the x and y coordinates of the *jth* 3D LiDAR boundary point on the same referential, and $\boldsymbol{p}_{q=j,xy}^{p}$ are the x and y coordinates of the corresponding ground-truth boundary point.

Figure 4 shows the ground-truth pattern boundary points representation (blue lines on the left of Fig. 4), the calculated boundary points at the start of the calibration procedure (blue circles on the middle of Fig. 4), and the result of the optimization procedure with the ground-truth points and the boundary points observed by the LiDAR aligned (right representation on Fig. 4).



(a) (b) (c)

Figure 4: **(a)**: calibration pattern and respective ground truth boundary points represented as blue lines; **(b)**: misaligned boundary points observed by the 3D LiDAR with the ground-truth points at the start of the calibration procedure; **(c)**: optimization result - ground-truth points and projected boundary points aligned. It is noteworthy that the orthogonal distance aligned the z coordinate of the ground-truth pattern and 3D LiDAR points.

2.2. Normalization of multi-modal residuals

We propose a full calibration method where sensors of different modalities contribute to a global vector residual of residuals. While the camera sub-function provides a set of residuals that are expressed in pixels, the LiDAR sub-function provides a set of residual expressed in meters. This mismatch in units of measurements may display highly disparities in error magnitudes, which could result in unwanted behaviours in the optimization processes due to differences in scale. For example, a residual of 1 pixel has higher influence (or weight) in the optimization path than a residual of 0.5 meters. Yet, our knowledge about the system tells us that the opposite should be considered. As result, the parameters that influence the residuals with higher scale will dominate the optimization, while the other parameters are perceived to already be close to their optimal state.

To handle the different scales in multi-model residuals in Equation 4, we employ a normalization factor to the optimization. Let $C = \{\mathbf{c}\}$ be the set of existing residual classes (e.g $C = \{\text{pixels}, \text{meters}\}$) and $c(i) \in C$ is the residual class for the ith sensor, then the optimization is defined as

$$\underset{\mathbf{\Phi}}{arg\,min}\ F(\mathbf{\Phi}) = \underset{\mathbf{\Phi}}{arg\,min}\ \frac{1}{2}\sum_{i}\left\|\frac{f_i(\{\mathbf{\Phi_i}\})}{\eta_{c(i)}}\right\|^2, \tag{12}$$

where $\eta_{c(i)}$ is the normalization factor for residuals created by the sensor sub-function f_i. Note that the normalization factor $\eta_{c(i)}$ is defined per residual class and not per sensor. The normalization values per class are given by the arithmetic mean of the same class residuals before the optimization. For example, the normalization for the class *pixels*, with $\eta_{c(i)} = \eta_{\text{pixels}}$, is given by

$$\eta_{\text{pixels}} = \frac{1}{n}\sum_{j}\|f_j(\{\mathbf{\Phi_j}\})\|_1 : \forall f_j \in \{\text{pixels}\}\ , \tag{13}$$

where n is the total number of residuals that are part of the considered class and $\|.\|_1$ is the L1 norm. Note that the normalization factors are constant values during optimization, calculated once with the residuals that result from the initial guess.

3. Calibration framework

The ROS (Quigley et al., 2009) has become the standard framework for the development of robotic solutions. As referenced before, the proposed calibration procedure requires the creation of a transformation tree, from which atomic transformations are optimized. For this purpose, ROS provides a tree graph referred to as *tf tree* (Foote, 2013). With this tool, it is possible to define a data structure as the one present in Fig. 1. Also, the Robot Operating System Visualization (RVIZ) tool supports additional functionalities, such as robot visualization, collision detection, etc. In fact, this visualization procedure is interactive, in that if any transformation between two links changes, the robotic platforms and sensors affected by these links change its pose accordingly. This interactive procedure is possible since the optimizations' cost function always recomputes the aggregate transformations. Therefore, a change in one atomic transformation in the chain affects the global sensor pose, and consequently, the error to minimize. So, if atomic transformations change due to the calibration procedure, the *tf tree* will automatically adjust the robots and sensors poses accordingly. It should be emphasized that, due to all these functionalities, the calibration procedure should not change the structure of the *tf tree*. Our approach preserves the predefined structure of the *tf tree*, since, during optimization, only the values of some atomic transformations contained in the chain are estimated, securing the topology of the tree. To the best of our knowledge, our approach is one of few which maintains the structure of the transformation graph before and after optimization.

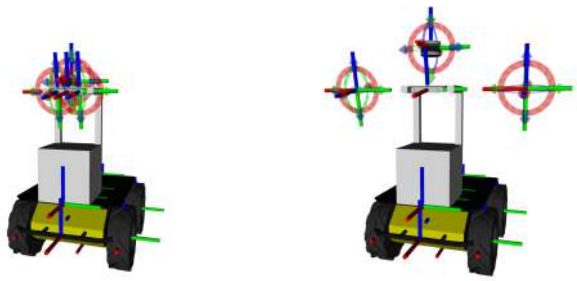
Given all of the above, we state an extensive integration with ROS as a key component of the proposed approach. The ROS calibration framework is segmented in five main components: configuration, initial estimate, data labelling, data collection, optimization procedure. Each will be described in detail in the following sections.

3.1. Calibration configuration

The configuration defines the parameters which will be used throughout the calibration procedure, from the definition of the sensors to be calibrated to a description of the calibration pattern. The proposed approach, detailed in Sec. 2, is based on the optimization of atomic transformations. These were combined through the use of the topological information contained in a tree. The transformation tree is generated from a ROS Unified Robot Description Format (URDF) . Additional information must be given to define which, out of the set of atomic transformations, will be optimized during the calibration procedure. Also, a description of the calibration pattern must be provided. All this information is defined in a calibration configuration file.

3.2. Initial parameter estimation

Optimization procedures suffer from the known problem of local minima. This problem tends to occur when the initial solution is far from the optimal parameter configuration, and may lead to failure in finding adequate parameter values. To avoid this, the setup of the a plausible initial guess for the entire parametric optimization system is essential. Our approach supports different modalities of parameters, as stated in (3). Thus, each modality requires a specific type of initialization. For cameras intrinsic $\boldsymbol{i}$ and distortion $\boldsymbol{d}$ parameters, the initialization is performed using any state-of-the-art camera calibration toolbox, or using the calibration provided by the manufacture. To initialize the transformations of sensors in general, we developed an interactive tool which parses the configuration URDF file and creates a 3D visualization tool for ROS interactive marker associated with each sensor. Figure 5 shows an example of the developed tool.



(a) (b)

Figure 5: Example of the developed interactive tool operation for AgRob V16 system. Here, the reference frames of the cameras that compose the stereo camera system and the 3D LiDAR sensor were moved. When the user positions the sensors in the desired pose, the initial estimate for the pose of each sensor is saved to use in calibration.

Here, we can see the user changing each sensor reference frame, dragging the respective interactive markers. With this tool the user can move and rotate the markers relative to each sensor. This provides a simple, interactive method to easily generate plausible first guesses for the poses of the sensors. Immediate visual feedback is provided to the user by the observation of the 3D models of the several components of the robot model and how they are put together, e.g. where each camera or LiDAR is positioned w.r.t. the vehicle. Also, for multi-sensor systems, it is possible to observe how well the data from a pair of sensors overlap. An example of this procedure can be watched at `https://youtu.be/llg8jYCeAjk`.

Concerning the atomic transformations associated with the calibration pattern ${}^{w}\boldsymbol{T}_{p}$ present in (2), these are initialized by defining a new branch in the transformation tree which connects the pattern to the frame to which it is fixed. For example, for AgRob V16 case, ${}^{w}\boldsymbol{T}_{p}$ is estimated through (2) where ${}^{cam_m}\boldsymbol{T}_{p}$ is estimated solving the Perspectiva-n-Point (PnP) for the detected pattern corners (Gao et al., 2003; Fabbri et al., 2020; Penate-Sanchez et al., 2013), and ${}^{w}\boldsymbol{A}_{cam_m}$ by deriving its topology from the *tf tree* and using the initial values for each atomic transformation in the chain.

3.3. Labeling data

The labeling of data refers to the annotation of the portions of data which captures the calibration pattern. A labeling procedure is executed for the data of each sensor, and can be automatic, semi-automatic or even manual in some cases. The information that is stored depends on the modality of the sensor, but for cameras it is always the pixel coordinates of the corners observed in the pattern.

The standard calibration pattern that is used for camera calibration is a chessboard pattern. The images are labelled using one of the many available image-based chessboard detectors (Czyzewski, 2017). Our system is also compatible with charuco boards (Garrido-Jurado et al., 2016). These have the advantage of being able to detect the pattern even when it is partially occluded. Also in this case we make use of off the shelf detectors, e.g. Romero-Ramirez et al. (2018); Hu et al. (2019).

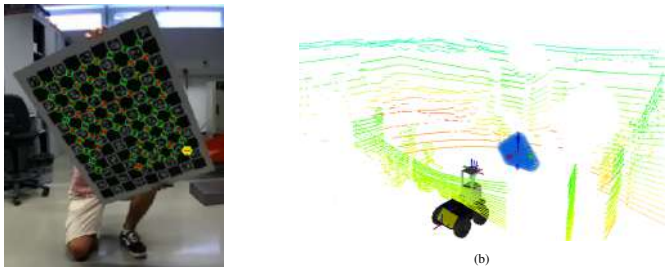
For the calibration of AgRob V16, a labeling algorithm for 3D LiDARs was developed. This method is semi-automatic and is initialized by a setup of a *seed point*. To label the pattern points viewed by the 3D LiDAR, the user drags an interactive marker to a point located in the pattern - the *seed point*. This constitutes the non-automatic stage of the procedure. After that, the algorithm clusters a set of points (that are intended to belong to the pattern) using an Euclidean distance threshold computed using the *a priori* known dimensions of the pattern. Despite being simple and fast, this approach reveals lack of precision, since it includes many outliers in the labeling procedure. The pattern used is rectangular. Thus, the Euclidean distance threshold has to be higher than the smaller side of the rectangle. This means that, if the pattern is close to another object, points from this object will be labeled as pattern points. To overcome this issue, a Random Sample Consensus (RANSAC) (Fischler & Bolles, 1981) algorithm is executed to fit the set of labeled points in a plane (the pattern plane), eliminating the outliers. RANSAC is an iterative algorithm, and it is performed a maximum number of times M. To find points that belong to the plane, the point to plane distance is computed in each iteration i for each point j as

$$D_{ij} = \frac{\left|a_i x_j + b_i y_j + c_i z_j + d_i\right|}{\sqrt{a_i^2 + b_i^2 + c_i^2}} \tag{14}$$

where $\boldsymbol{p}_j = [x_j, y_j, z_j]^T$ is the *jth* point on the cluster. With this, a point is considered as inlier if its distance to the plane D_{ij} is smaller than a given threshold $D_{threshold}$. The final set of inliers corresponds to the one found in the iteration i that gives the higher number of points belonging to the plane.

Figure 6 shows an example of the labeling procedure for cameras and 3D LiDARs proposed in ATOM.

It is worth noting that, our approach works with partial detections, as represented in the figure. This interactive data labeling procedure is showcased in `https://youtu.be/uNPIOCqxb5w`.



(a) (b)

Figure 6: **(a)**: labeling image data using a charuco pattern; **(b)**: labeling 3D LiDAR data on pattern (solid blue points) using the semi-automatic proposed approach. Note that, our approach works with partial detections, i.e., collections of data where the pattern is not fully detected. This figure presents an example of an partial detection for each type of sensor. For the camera, only a portion of the corners of the pattern were detected. For the 3D LiDAR, the pattern is not fully observable due to the low vertical resolution. Even though, our approach is able to calibrate the system with these detections.

3.4. Collecting data

In most robotic systems, the data coming from the sensors is streamed at different frequencies. However, to compute the associations between the data of multiple sensors, temporal synchronization of the sensor data is required. Of course, this only becomes an issue when calibrating multi-sensor robotic systems. For now, the synchronization problem is solved trivially by collecting data (and the corresponding labels) at user defined moments in which the scene has remained static for a certain period of time. In static scenes, the problem of data de-synchronization is not observable, which warrants the assumption that for each captured collection the sensor data is "adequately" synchronized. This can be done using, for example, a tripod to held the pattern before collecting each snapshot of data (Rehder et al., 2016; Furgale et al., 2013). In this work, the problem is approached in a simpler way, where the pattern is hold by the user, as shown in Fig. 2, remaining static by sufficient amount of time to ensure the synchronization between all the sensors.

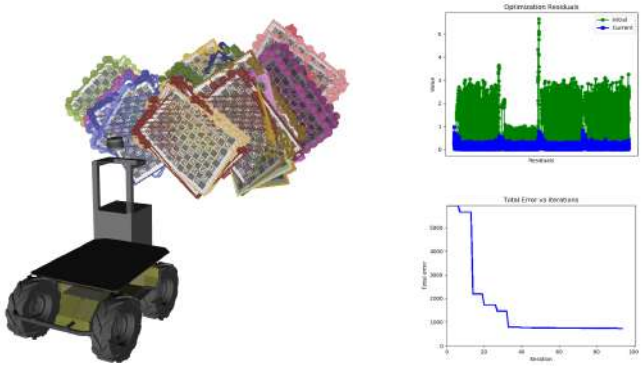
To save the scene data captured by all the sensors in the calibration system, the user can do it with just two mouse clicks on the interactive ROS-based tool (on RVIZ) developed. We refer to these recordings of data as *data collections*. Each one of them contains the values of all atomic transformations that exist in the system at a given timestamp, a copy of the robot configuration file, sensor data and labels, and high level information about each sensor, such as the topological transformation chain, extracted from the transformation tree. This information is stored in a *dataset file* that will be read by the optimization procedure afterwards. Also, a video showing the procedure for collecting data is provided for AgRob V16 calibration here `https://youtu.be/p7TuhSsRMcw`.

It should be pointed out that, the set of collections should contain as many different poses as possible. As such, collections should preferably have different distances and orientations w.r.t. the calibration pattern, so that the calibration returns more accurate results. This concern is common to the majority of calibration procedures.

3.5. Visualizing the optimization

The immediate visualization of the calibration is essential for several reasons: it provides the user the necessary data so that he can detect failures on the calibration, such as outlier data collections; it gives feedback about the cost function residuals minimization, which can serve to detect possible local minima, and to make sure that the optimization procedure is converging. Figure 7 shows the three main visualization features of ATOM.

Our calibration framework provides a simultaneous visualization of all the data collections, as well as immediate feedback of the alignment between ground-truth points and labeled points for optimization, images with the reprojection, robot meshes, the position of the reference frames, etc. Also, optimization graphics are provided with residuals values and the total error for each iteration. This configuration is similar to the standard one which is used during the initial parameter estimation, the data labeling and collection, but contains a couple of key distinctions. As mentioned above, the calibration procedure uses a dataset file which contains information about each of the stored collections. These collections contain data gathered in a a set of sequential instants in time. The ROS calibration configuration publishes data from all collections simultaneously, as if those time instants were packed together and processed as if they had occurred all at the same time. Collisions in topic names and reference frames are avoided by adding a collection related prefix to each designation. Also, the original transformation tree is replicated for each collection. A video with an example of a calibration execution for AgRob V16 is provided here `https://youtu.be/HtTyTsWuMIQ`.



(a) (b)

Figure 7: **(a)**: ROS calibration configuration - simultaneous visualization of all the collections of data and respective alignment between ground-truth points and labeled points; **(b)**: graphics representing the objective function minimization - individual residuals value, and the total error vs iterations.

4. Results

To test and validate the performance of the proposed approach, an extensive evaluation procedure was developed. Our calibration framework, ATOM, was used to calibrate three configurations of the AgRob V16 sensing system. Two of them were pairwise calibrations between two cameras of a stereo system, and a single camera and a 3D LiDAR, which we denote as *ATOM pairwise*. The third was a calibration between all three sensors (two cameras and 3D LiDAR), which we call *ATOM full*, where results for particular pairs of sensors are obtained using a full calibration. In this procedure, three datasets were used, as represented in Tab. 1.

Two of them (*train-1*, *train-2*) were used for training, i.e., to perform the calibrations, and the third one (*test-3*) was used to test the calibration with specific metrics that will be detailed later on. The datasets contain incomplete and partial collections, i.e., collections where the pattern is not detected for all the sensors, and collections where the pattern is only partially visible, respectively. This section is divided in two parts: the evaluation of ATOM's performance against

Table 1: Description of the datasets used for train and evaluation. Two datasets were used for training (calibration) and one for testing (evaluation). The datasets contain incomplete collections, i.e., collections were the pattern is not detected by at least one sensor, and partial collections, i.e., collections were the pattern is partially detected by at least one sensor.

Dataset	**Nr. Collections**			**Observations**
	Total	Incomplete	Partial	
train-1	42	5	38	Dataset contains incomplete collections.
train-2	56	0	24	The dataset does not contain the point cloud generated by the ZED camera.
test-3	15	0	8	Dataset with low number of collections, only used for test.

Table 2: Summary of the methods used and evaluated in the experiments.

Method	Calibration	Properties
OpenCV (Bradski, 2000)	pairwise	reprojection error, intrinsics calibration
Stereo camera factory calibration	pairwise	reprojection error, intrinsics calibrations
ICP average (Besl & McKay, 1992)	pairwise	reprojection error, average of result of all collections
ICP best (Besl & McKay, 1992)	pairwise	reprojection error, best result of all collections
ATOM pairwise [this paper]	pairwise	reprojection error, angle-axis, intrinsics calibration
ATOM full [this paper]	full calibration	reprojection error, angle-axis, intrinsics calibration

state-of-the-art approaches, such as OpenCV stereo calibration (Bradski, 2000), and ICP point cloud alignment (Besl & McKay, 1992); the characterization of ATOM w.r.t. several characteristics of the datasets, such as the number of incomplete/partial collections, and the accuracy of the initial guess. Table 2 makes a summary of the calibration experiments that were carried out.

OpenCV and the stereo camera factory calibration were used to get the camera-to-camera extrinsic calibration, and ICP was explored to get the camera-to-LiDAR calibration. This last calibration was obtained by the alignment of the 3D LiDAR point cloud, and the 3D point cloud provided by the stereo camera software development kit. Two versions of ICP were used as camera-to-LiDAR extrinsic calibration: the one corresponding to the collection where the fitting of point clouds was more accurate, and the average of the transformations obtained in all collections. Finally, as referenced before, ATOM was calibrated both in pairwise and full modes, and both approaches are evaluated.

4.1. Methodology

One of the key characteristics of our evaluation procedure is the use of separate datasets to perform the calibration and generate the results. As discussed in Sec. 3, ATOM provides a data collection procedure, where datasets are generated and visualized on RVIZ . Datasets are composed of several collections, each one containing data of all the sensors, initial atomic transformations, pattern labelled points, and other information. To evaluate our calibration framework, two datasets where initially collected - *train-1* and *train-2*. Then, three calibrations were executed over each one of the datasets, two pairwise and one using all the sensors to be calibrated in AgRob V16 system. These calibrations generate a `json` file similar to the one generated at the end of the data collection procedure, but with the calibrated atomic transformations. After obtaining all these calibration configurations, a third dataset was recorded - *test-3*. It was used to evaluate the accuracy of the calibrations obtained. Using the metrics that will be described later on, the chain of transformations of each calibration was loaded and used to compute errors using the labelled data of the test dataset. In this way, possible influence of using the same data to calibrate and test is eliminated, and a more rigorous evaluation is achieved.

Unlike ATOM, both OpenCV and ICP perform sensor-to-sensor calibration, instead of calibrating atomic transformations without changing the topology of the chain of transformations. Thus, to evaluate these methods in the same way ATOM is evaluated, the atomic transformations set to be calibrated had to be recovered from the sensor-to-sensor calibrations. This problem is formulated in Fig. 8.

Let ${}^{link_2}\boldsymbol{T}_{base}$ be the entire chain of transformations from the base link to the data link of the anchored sensor, $\hat{\boldsymbol{T}}$ be the sensor to sensor calibration obtained, ${}^{parent_1}\boldsymbol{T}_{base}$ be the chain of

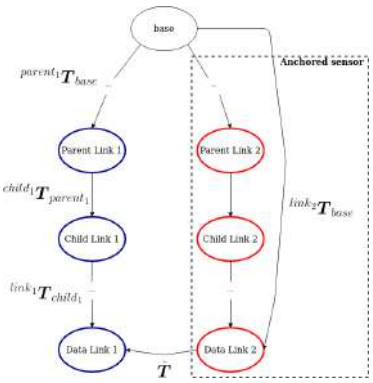


Figure 8: Chain of transformations representing the generic configuration to extract an atomic transformation from a sensor to sensor calibration. Preserving the chain of transformation of the anchored sensor, and taking into account the sensor to sensor calibration $\hat{\boldsymbol{T}}$, we recover the atomic transformation $^{child_1}\boldsymbol{T}_{parent_1}$.

transformations from the base link to the parent link of the atomic transformation to be calibrated $^{child_1}\boldsymbol{T}_{parent_1}$, and $^{link_1}\boldsymbol{T}_{child_1}$ the chain of transformation from the atomic transformation child link, to the non-anchored sensor data link. The entire chain of transformations relationships can be formulated as follows:

$$\hat{\boldsymbol{T}} \cdot {}^{link_2}\boldsymbol{T}_{base} = {}^{link_1}\boldsymbol{T}_{child_1} \cdot {}^{child_1}\boldsymbol{T}_{parent_1} \cdot {}^{parent_1}\boldsymbol{T}_{base}. \tag{15}$$

So, from (15), we can extract the atomic transformation of the non-anchored sensor as follows:

$$^{child_1}\boldsymbol{T}_{parent_1} = ({}^{link_1}\boldsymbol{T}_{child_1})^{-1} \cdot \hat{\boldsymbol{T}} \cdot {}^{link_2}\boldsymbol{T}_{base} \cdot ({}^{parent_1}\boldsymbol{T}_{base})^{-1}. \tag{16}$$

The advantage of this approach is that it is generic. For example, from OpenCV, a camera to camera metric is obtained. Thus, the procedure consists in anchoring one of the cameras, and use the obtained transformation to recover the atomic transformation marked for calibration in the original ATOM configuration. In the same way, ICP provides a camera to LiDAR calibration. Once again, we anchor one of these sensors, and apply the exact same routine to extract the non-anchored sensor atomic transformation to be calibrated. In this way, we are able to obtain a calibrated system without changing the initial chain topology, which allows the direct comparison of these state-of-the-art approaches with ATOM, using exactly the same metrics. These metrics are described in the next section.

4.2. Metrics

To evaluate the camera to camera calibration performance, the methodology used is based on three different metrics: the mean rotation error (rad), the mean translation error (m), and the reprojection error (px). To compute the reprojection error, the idea is to use the calibration result to project the detected pattern corners of one camera image into the image of the second calibrated camera image and compare the projected pixel coordinates with the ground truth pattern

corners coordinates in the image of the anchored camera. To transform pixels from one camera into another, we start from projecting the 3D world coordinates of the pattern corners into the image of a camera, using (6)-(7). Since the 3D pattern corners are defined in the local pattern reference frame, they all lie in the plane $z = 0$. Thus, (6)-(7) can be simplified to the following:

$$\boldsymbol{p}^{cam} = \boldsymbol{K} \cdot {}^{cam}\boldsymbol{T}'_p \cdot \boldsymbol{p}^{p'}, \tag{17}$$

where ${}^{cam}\boldsymbol{T}'_p$ is a portion of the matrix ${}^{cam}\boldsymbol{T}_w \cdot {}^{w}\boldsymbol{T}_p$, without the component z of the rotation, as follows:

$${}^{cam}\boldsymbol{T}'_p = \begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ r_{31} & r_{32} & t_z \end{bmatrix}, \tag{18}$$

and $\boldsymbol{p}^{p'}$ is the pattern corner, represented as a vector in its homogeneous form, without the z component, i.e., $\boldsymbol{p}^{p'} = \begin{bmatrix} x\ y\ 1 \end{bmatrix}^T$. Using the fact that the 3D coordinates of the pattern's corners are the same for both cameras, (17) can be applied to the two of them, so that we can find a relation between both expressions. This resulted in the following formulation:

$$\boldsymbol{p}^{cam2} = \boldsymbol{K}^{cam2} \cdot {}^{cam2}\boldsymbol{T}'_p \cdot ({}^{cam1}\boldsymbol{T}'_p)^{-1} \cdot (\boldsymbol{K}^{cam1})^{-1} \cdot \boldsymbol{p}^{cam1}, \tag{19}$$

where *cam*1 and *cam*2 refer to the cameras that were calibrated. This formulation provides the relationship between pixel coordinates of the pattern corners in both camera images. However, (19) requires the camera to pattern transformation matrix for both cameras. This can be a problem since, some approaches, unlike ATOM, do not estimate the camera to pattern transformation while performing the camera to camera calibration. In addition to this, ATOM estimates these transformations for a training dataset. So, the usage of a test dataset to evaluate all the frameworks, denies the use of the estimated pattern pose from ATOM. To overcome this, the pattern pose w.r.t. one of the cameras ${}^{cam1}T_p$ is computed using the PnP algorithm. Then, using the output json file from each calibration, we recover the camera to camera transformation ${}^{cam1}\boldsymbol{T}_{cam2}$, through the chain of transformations. In this manner, it is possible to determine the transformation of the other camera to the pattern, as follows:

$${}^{cam2}\boldsymbol{T}_p = ({}^{cam1}\boldsymbol{T}_{cam2})^{-1} \cdot {}^{cam1}\boldsymbol{T}_p. \tag{20}$$

From this expression, we can derive ${}^{cam2}\boldsymbol{T}'_p$ and ${}^{cam1}\boldsymbol{T}'_p$, and successfully project pixels from one image into the other. With this information, the reprojection error is computed as follows:

$$\boldsymbol{e}_{xy} = \boldsymbol{p}^{projected} - \boldsymbol{p}^{expected}. \tag{21}$$

From (21), the error can be decomposed in its x and y components. Also, considering the reprojection error for all the N collections, the root mean square error is calculted as follows:

$$e_{rms} = \sqrt{\frac{1}{N} \sum \boldsymbol{e}_{xy}^2}. \tag{22}$$

Figure 9 illustrates a resulting corner reprojection from one camera into the other using the ATOM full calibration.

For the calculation of the mean rotation and translation errors to evaluate the camera to camera calibration, we consider the following observation: the chain of transformations from the

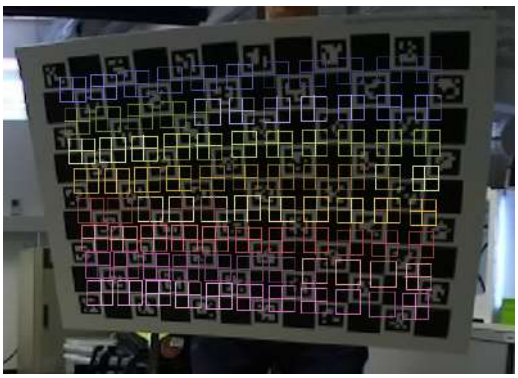


Figure 9: Camera reprojection error from one camera image into the other. Squares represent the expected corner pixel coordinates, and crosses the projected result.

base link to the pattern reference frame that passes from each one of the calibrated cameras should be equal. This happens since the calibration pattern pose in reference with the base link is fixed, and any chain of transformations that link these two referentials should represent the same spatial relationship. Thus, the difference in rotation and translation can be quantified, assessing the inequality between the two chains of transformation. Once again, this formulation requires the pattern pose w.r.t. each one of the cameras, that is again extracted solving the PnP problem. With this information, we can state that

$$\begin{bmatrix} {}^{base}\boldsymbol{R}_{cam1} & {}^{base}\boldsymbol{t}_{cam1} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{cam1}\boldsymbol{R}_{p} & {}^{cam1}\boldsymbol{t}_{p} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} {}^{base}\boldsymbol{R}_{cam2} & {}^{base}\boldsymbol{t}_{cam2} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{cam2}\boldsymbol{R}_{p} & {}^{cam2}\boldsymbol{t}_{p} \\ 0 & 1 \end{bmatrix}. \tag{23}$$

Now, we can define the rotation and translation difference as

$$\Delta\boldsymbol{R} = ({}^{base}\boldsymbol{R}_{cam1} \cdot {}^{cam1}\boldsymbol{R}_{p})^{-1} \cdot {}^{base}\boldsymbol{R}_{cam2} \cdot {}^{cam2}\boldsymbol{R}_{p} \tag{24}$$

$$\Delta\boldsymbol{t} = {}^{base}\boldsymbol{R}_{cam1} \cdot {}^{cam1}\boldsymbol{t}_{p} + {}^{base}\boldsymbol{t}_{cam1} - {}^{base}\boldsymbol{R}_{cam2} \cdot {}^{cam2}\boldsymbol{t}_{p} - {}^{base}\boldsymbol{t}_{cam2} \tag{25}$$

Finally, we can define the mean rotation error as

$$e_R = \frac{1}{N} \sum_i \|\texttt{angle}(\Delta\boldsymbol{R}_i)\|, \tag{26}$$

where `angle` is the angle-axis representation of the rotation, and the mean translation error as

$$e_t = \frac{1}{N} \sum_i \|\Delta\boldsymbol{t}_i\|. \tag{27}$$

(a) (b) (c)

Figure 10: 3D LiDAR to camera reprojection error metric calculation. **(a)** represents the annotation procedure, where four classes are annotated, each one representing a single side of the pattern; **(b)** red curves represent the approximation of each one of the classes by a polynomial function, in order to account for the distortion in the image; **(c)** reprojection error calculation - blue dots represent the 3D LiDAR pattern boundary reprojected points and yellow lines the difference of each one of the points with the labelled ground truth.

To evaluate the 3D LiDAR to camera calibration, we used the reprojection error (px) and its corresponding root mean square error (px) considering all the test collections N. In this case, the mean rotation and translation errors were not used due to the difficulty of estimating the pattern pose w.r.t. the LiDAR sensor with precision. The process of calculating the reprojection error to evaluate the camera to LiDAR calibration consists in three main steps:

1. Label the pixels that belong to the boundaries of the pattern in the image.
2. Reproject the pattern boundary points in the LiDAR's referential frame (detailed in Sec. 2.1) $\boldsymbol{p}^{boardlimit}$ into the image.
3. Compute root mean square between labeled and projected points.

Figure 10 shows these steps.

An annotation tool was developed to perform the labelling. This tool allows the user to manually annotate individual points corresponding to four classes, each one representing one side of the pattern in each image. Then, in order to account for the image distortion, we approximate each one of the pattern sides by a polynomial, fitting the labelled points. In this step, a simple linear regression would not suffice because images have distortion which transforms straight ines into curves. So, a polynomial is more suitable for modeling this phenomenon. Figures 10a and 10b show these two steps. After having the annotations for all the images of the camera to be calibrated in the test dataset (test-3), the reprojection error is calculated. To do so, the 3D LiDAR labelled points that belong to the pattern boundaries are reprojected into the image using (6)-(9). So, for each collection, the error between each projected point and the closest ground truth point belonging to one of the polynomial curves is calculated as in (21). Figure 10c shows an example of the reprojection result and the corresponding error for each point. Considering the reprojection errors calculated for each one of the N collections, the root mean square error is also calculated using (22).

It should be emphasised that, all of these metrics are publicly available, and are integrated in the ATOM software framework. A specific package called *ATOM evaluation* was created and can be easily used by the user to evaluate the calibrations performed.

4.3. Evaluation

The evaluation procedure applies the previously described metrics to compare ATOM with state-of-the-art calibration methods. Note that these state-of-art methods are pairwise and as

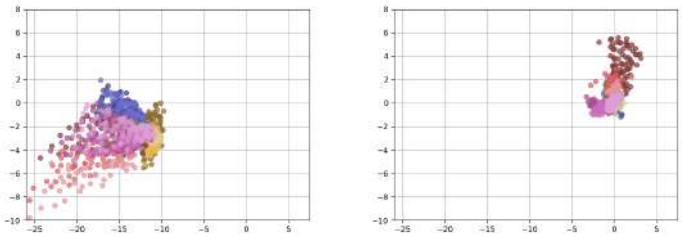
Table 3: Performance comparison of methods for camera to camera calibration.

Method	**Train dataset**	$e_{\boldsymbol{R}}$ (rad)	$e_{\boldsymbol{t}}$ (m)	e_x (px)	e_y (px)	e_{rms} (px)
OpenCV (Bradski, 2000)	train-1	Not able to calibrate due to partial pattern detections.				
ATOM pairwise		0.009	0.003	0.551 ± 0.800	0.780 ± 1.090	1.049
ATOM full		0.008	0.003	0.547 ± 0.759	0.638 ± 1.034	**0.974**
OpenCV (Bradski, 2000)	train-2	0.006	0.003	0.582 ± 0.648	0.622 ± 0.966	**0.863**
ATOM pairwise		0.010	0.006	0.655 ± 1.055	0.677 ± 0.982	1.020
ATOM full		0.008	0.005	0.594 ± 0.912	0.696 ± 1.033	0.974
ZED's factory calibration	-	0.007	0.006	2.220 ± 0.765	0.486 ± 0.823	1.757

such not able to calibrate the entire system simultaneously. The comparison with ATOM, which is a general, full calibration approach, against specialized pairwise methods is not entirely fair. However, since the nature of all the metrics used in the evaluation is also pairwise, it is ATOM that is at a disadvantage in comparison with the other methods. For the camera-to-camera calibration scenario, two versions of ATOM were calibrated in two diffeernt training datasets, and evaluated in the same test dataset. The first is a pairwise calibration between both cameras, and the second a full calibration of the entire AgRob V16 system, with the same two cameras and a 3D LiDAR. It is worth noting that, the calibrations performed consider each camera intrinsic parameters, as well as the pairwise extrinsic calibration between them. To compare ATOM with the state of the art, the OpenCV stereo calibration toolbox (Bradski, 2000) was used to calibrate exactly the same configuration. Additionally, the factory's intrinsic and extrinsic calibrations were evaluated. Table 3 summarizes all these experiments.

Starting by the analysis of ATOM pairwise and ATOM full, we can see that both versions present similar performances, with marginal differences with respect to all the metrics calculated. For example, for the train-1 dataset, we can see a root mean square error difference of 0.075 px and for train-2 0.046 px. Thus, we can conclude that, ATOM allows to optimize an entire robotic system without a significant loss of performance, when comparing with a specific pairwise calibration between the two sensors of interest. In what concerns OpenCV, Tab. 3 shows that, for the train-1 dataset, it is not able to calibrate. This happens since this framework requires collections where all the pattern corners are detected, i.e., non partial detections. So, since the majority of the collections present in this dataset are partial, OpenCV is not able to calibrate. This is a limitation since, to accomplish a dataset without partial collections, its variety can be limited due to the impossibility of collecting, e.g., collections with the pattern far away from the cameras. On the other hand, for the train-2 dataset, OpenCV achieves the smaller reprojection root mean square error. In this, the number of partial collections is low, and OpenCV, a specialized pairwise calibrator for cameras, performs an accurate intrinsic and extrinsic calibration. Even so, ATOM full shows errors only slightly higher than OpenCV for this dataset, showing that it is capable of achieving a state-of-the-art performance, even considering a non pairwise approach. Concerning the camera factory calibration, it is clear that it presents the less accurate calibration. This can be explained since the calibration the same for all the equipments. Finally, to analyse the impact of the ATOM's calibration, Fig. 11 shows the dispersion of the reprejection error per collection, before and after calibrating with ATOM full.

As expected, the dispersion of the error before calibrating is higher in almost all collections. On the contrary, after calibrating the cameras with ATOM full, the dispersion drastically reduces, with the exception of one collection (represented in brown). This collection can represent a



(a) (b)

Figure 11: Reprojection error dispersion for the camera to camera calibration using ATOM full configuration. Each color represents the error associated with one individual collection. **(a)** is the representation of this error before calibrating (using the initial guess), and **(b)** after the calibration.

Table 4: Performance comparison of methods for camera to 3D LiDAR calibration.

Method	**Type**	**Train dataset**	e_x (px)	e_y (px)	e_{rms} (px)
ICP average (Besl & McKay, 1992)	left camera - 3D LiDAR	train-1	47.210 ± 31.374	19.058 ± 28.233	44.307
ICP best (Besl & McKay, 1992)	left camera - 3D LiDAR		9.111 ± 11.950	2.625 ± 7.967	10.492
ATOM pairwise	right camera - 3D LiDAR		3.054 ± 4.727	1.031 ± 2.689	3.869
ATOM pairwise	left camera - 3D LiDAR		3.648 ± 4.846	1.260 ± 2.869	4.101
ATOM full	right camera - 3D LiDAR		3.351 ± 4.874	0.950 ± 2.279	**3.811**
ATOM full	left camera - 3D LiDAR		3.398 ± 4.923	1.100 ± 2.602	3.942
ICP average (Besl & McKay, 1992)	left camera - 3D LiDAR	train-2	-	-	-
ICP best (Besl & McKay, 1992)	left camera - 3D LiDAR		-	-	-
ATOM pairwise	right camera - 3D LiDAR		7.574 ± 6.393	1.776 ± 3.181	6.715
ATOM pairwise	left camera - 3D LiDAR		7.560 ± 5.535	1.619 ± 2.795	**6.432**
ATOM full	right camera - 3D LiDAR		7.702 ± 5.441	1.648 ± 2.781	6.537
ATOM full	left camera - 3D LiDAR		8.117 ± 5.692	1.687 ± 2.838	6.765

degenerate of data collection, due to, for example, de-synchronization of the data from from the sensors.

In order to evaluate the camera to LiDAR calibration, ATOM was used to calibrate both modalities in three different manners: two ATOM pairwise versions, one between the LiDAR and each camera, and the ATOM full version that comprises all three sensors. To compare our approach with the state-of-the-art, ICP (Besl & McKay, 1992) was used to calibrate the left camera and the LiDAR in two different ways: one considering the average of the calibration obtained in all the collections, and other considering only the collection that presents the best alignment between the two point clouds. Note that, ICP only calibrates the left camera w.r.t. the LiDAR since the stereo camera point cloud extracted directly using the manufacture's API is defined in this camera referential. Table 4 summarizes all this information.

Similarly to the camera-to-camera case, here we can verify that ATOM pairwise and ATOM full result in a similar calibration performance, with marginal reprojection error differences. So, once again, this leads to the conclusion that ATOM full can be used to calibrate all the robotic system without any significant loss of performance while evaluating calibrations between pairs of sensors. In this set of tests, a consistent decrease of performance of all the calibration con-

Figure 12: Point cloud projection into the left camera image using the 3D LiDAR to camera calibration. The color represents the points depth. It is worth noting that, this procedure was done using the ATOM full calibration result, with a collection from the test dataset, just like in all the evaluation pipeline. If the system is accuratly calibrated, changes in point's color, which denotes a variation of the measured range, should coincide with transitions between far and near objects in the image.

figurations from train-1 to train-2 dataset is observed. This consistency can be caused by synchronization errors between sensors while collecting the calibration data. Looking for the ICP performance on train-1 dataset, firstly, we can conclude that the ICP average is highly affected by outliers, i.e., collections where the calibration fails. This can be inferred by the high standard deviations present in the x and y reprojection error compoenents. ICP best, despite being significantly less accurate than ATOM, presents a better performance than ICP average. The overall bad performance of ICP can be explained by the difficulty of aligning a dense point cloud (provided by the stereo camera), and a sparse one (provided by the laser). It is worth noting that, the train-2 dataset does not contain the stereo camera point cloud, so here ICP can not be used for calibration. To have a visual perception of the ATOM full performance on the calibration of the left camera and the LiDAR, Fig. 12 shows the reprojection of the LiDAR 3D points in the left camera image.

In this Fig., color represents the points depth. Here, the transitions of the objects can be sharply observed, which is a good indicator for the calibration performance.

4.4. Impact of the number of collections used for training

One of the major questions in general for calibration procedures is the minimum amount of data required to calibrate sensors with precision. In this section we propose an evaluation of ATOM full calibration using different numbers of training collections. The calibration of the three combinations of sensors is evaluated for five different levels of collections used. Table 5 presents the results obtained for each configuration.

Starting by the analysis of the camera-to-camera calibration, here we can see that the increase of the number of collections leads to a increase in performance. Using a single collection, as expected, results in higher reprojection, rotation and translation errors. While increasing the number of training collections, the performance increases, with the best performance being observed with the maximum number of collections. Looking at the performance of the calibration

Table 5: Impact of the number of collections in ATOM's full calibration performance. The dashed entries correspond to results not generated since, for the camera-to-LiDAR case, the mean rotation and translation errors are not available.

Type	Nr. Collections	$\mathbf{e}_R$ (rad)	$\mathbf{e}_t$ (rad)	$\mathbf{e}_x$ (px)	$\mathbf{e}_y$ (px)	$\mathbf{e}_{rms}$ (px)
camera - camera	1	0.051	0.053	8.205 ± 7.201	16.089 ± 3.346	13.534
	5	0.017	0.016	3.540 ± 4.575	1.108 ± 1.449	4.233
	10	0.009	0.004	1.377 ± 1.318	0.932 ± 0.991	1.645
	20	0.008	0.003	0.515 ± 0.719	0.658 ± 1.056	0.976
	30	0.008	0.003	0.547 ± 0.759	0.638 ± 1.034	**0.974**
right camera - 3D LiDAR	1	-	-	4.348 ± 6.314	1.936 ± 4.431	5.487
	5	-	-	4.202 ± 5.179	1.144 ± 2.342	4.466
	10	-	-	3.305 ± 4.742	0.984 ± 2.387	3.820
	20	-	-	3.355 ± 4.744	1.005 ± 2.412	**3.774**
	30	-	-	3.352 ± 4.874	0.950 ± 2.279	3.811
left camera - 3D LiDAR	1	-	-	5.126 ± 6.686	1.527 ± 3.435	5.566
	5	-	-	3.381 ± 4.447	1.058 ± 2.503	3.730
	10	-	-	2.937 ± 4.430	1.115 ± 2.791	**3.712**
	20	-	-	3.411 ± 4.887	1.258 ± 2.959	4.046
	30	-	-	3.398 ± 4.924	1.100 ± 2.602	3.942

of the LiDAR with both cameras, we can see that, as expected using a single collection also results in a higher reprojection error. However, in this case, this difference is not significant, and while increasing the number of collections, the performance saturates. Thus, we can conclude that, the increase of the number of collections has a positive impact in the final performance of all the calibration configurations. However, the camera-to-camera calibration is more sensible to the lower number of collections than the camera-to-LiDAR calibration.

4.5. Impact of the number of incomplete collections used for training

The proposed calibration framework, ATOM, supports collections where the pattern is not detected by all the sensors in the calibration system. For example, suppose that the pattern is, for a specific collection, is viewed by the right camera and the LiDAR but not by the left camera. The current section intends to evaluate the impact of this type of collections, and conclude if the the presence of incomplete collections has, or not, correlation with changes on ATOM's performance. Table 6 summarizes the results obtained for three sensor configurations with four different values of incomplete collections, maintaining the same number of training collections.

ATOM can deal with incomplete collections, as long as the pattern is detected by at least one sensor. If not, the calibration system can not compute any residual, and that collection must be discarded. On the other hand, if, a single sensor does not detect the pattern for a specific collection, this leads to a reduction of the number of residuals used on the optimization procedure. This being said, results do not show any correlation between the increase of the number of incomplete collections and the performance of ATOM. So, this leads to the conclusion that ATOM can deal with the decrease of the number of residuals (as long as sufficient number of collections are provided). This conclusion consistent with the one taken from Tab. 5, where it was shown that, ATOM can calibrate accurately for a reasonable low number of collections.

4.6. Impact of the accuracy of the initial estimate

The initial estimate (or initial guess) of the calibration parameters has an impact in the outcome of the optimization. A good initial estimate provides a sufficient approximation that allows

Table 6: Impact of the number of incomplete collections in ATOM full calibration performance. The dashed entries correspond to results not generated since, for the camera-to-LiDAR case, the mean rotation and translation errors are not available.

Type	**Nr. Collections**		$\mathbf{e}_R$ **(rad)**	$\mathbf{e}_R$ **(rad)**	$\mathbf{e}_x$ **(px)**	$\mathbf{e}_x$ **(px)**	$\mathbf{e}_{rms}$ **(px)**
	Complete	Incomplete					
camera - camera	10	0	0.011	0.011	1.159 ± 1.479	0.848 ± 1.182	1.457
	10	2	0.006	0.003	0.821 ± 0.711	0.606 ± 0.900	**1.031**
	10	4	0.009	0.006	1.680 ± 2.001	0.784 ± 0.945	2.017
	10	5	0.010	0.005	0.778 ± 0.990	0.641 ± 0.908	1.076
right camera - 3D LiDAR	10	0	-	-	5.104 ± 6.517	1.370 ± 3.085	5.496
	10	2	-	-	3.272 ± 4.800	1.068 ± 2.634	**3.920**
	10	4	-	-	3.414 ± 4.934	1.040 ± 2.515	3.964
	10	5	-	-	3.734 ± 4.956	1.111 ± 2.540	4.069
left camera - 3D LiDAR	10	0	-	-	5.113 ± 6.479	1.535 ± 3.430	5.576
	10	2	-	-	3.112 ± 4.665	1.192 ± 2.970	3.930
	10	4	-	-	2.995 ± 4.541	1.137 ± 2.886	**3.849**
	10	5	-	-	3.720 ± 5.012	1.274 ± 2.907	4.169

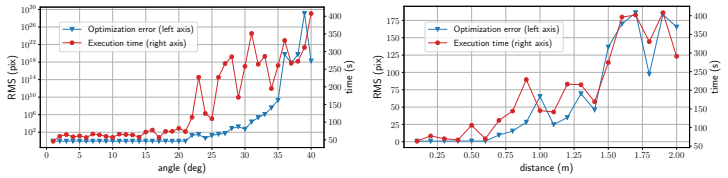
the optimization process to find the optimal solution that best represents the real calibration of the system. In turn, a inaccurate estimate may lead the optimization process to an unrecoverable state where it is not possible to achieve the optimal solution. This is known as the problem of local minima.

In this section, we are interested in assessing the robustness of ATOM to the accuracy of the initial estimate. More specifically, the angle and distance error to the optimal state that our method can handle and the additional execution times that result from said errors. To characterize the robustness to the angle error, we start with the optimal angle and then add the angle error to the the Euler components of the rotation. The sign of the error is provided by a fair binomial sampling (Girshick et al., 2006). The tolerance to the distance error is found by sampling an uniform offset from the optimal positions that sits on a sphere with a radius equal to the distance error. Because random sampling is used, we run the experiment for each error 10 times and the reported values are the mean of the runs. By increasing the error in several steps, we can pinpoint the error at which our optimization process will fail. Note that the error is applied to all pose parameters that are being estimated. The results are presented in Figure 13.

The obtained results show that our optimization process can handle an angle error of approximate 20 degrees and a distance error of 0.5 meters, for each sensor. In our opinion, these errors provide a sufficient margin of tolerance for a practical usage of our manual procedure for initial parameter estimation, described in section 3.2. The execution times are strictly related to the convergence of the optimization. Inside the margin of tolerance, the execution times are mostly constant (with some fluctuations). This means the optimization process adequately handles the imposed error with a proper convergence.

5. Conclusions

This paper solves the problem of camera-to-LiDAR calibration using the optimization of atomic transformations. To do so, this work formulates the calibration as a Bundle Adjustment problem, minimizing the reprojection error of sensors that can have different modalities. Our approach, ATOM, provides several advantages when compared with the current state-of-the-art:



(a) (b)

Figure 13: Impact of the initial angle (a) and distance (b) estimation error on the optimization error and execution time. The considered optimization error is the *root means squared* (RMS) error.

(i) it offers a framework to simultaneously calibrate any number of sensors; (ii) it improves the optimization of different sensor modalities by introducing the multi-modal normalization. (iii) it maintains the topology of the input transformation tree; (iv) it supports incomplete and partial collections of data, which makes the detection procedure more flexible and robust; (v) it uses a common calibration pattern, which generalizes the approach; (vi) it has seamless integration with ROS, setting a complete framework for camera to LIDAR calibration.

Results show that the proposed approach presents similar performance in comparison with the state-of-the-art, even calibrating the entire robotic system simultaneously. These results demonstrate that ATOM can achieve the same performance of specialized methods in pairwise calibration between specific sensors, while running a complete calibration with multiple sensors of different modalities. Furthermore, our framework proved to be robust to inaccurate initial guesses and small number of collections. Finally, the use of a generic calibration pattern constitutes a major advance since, in the current state-of-the-art, many approaches use built in-house patterns.

Future work aims to test ATOM in more advanced robotic systems, with multiple 3D LiDARs. Additionally, the problem of data synchronization while collecting data for calibration will be addressed thought the use of simple concepts such as data interpolation, and more advanced ones, such as generative adversarial networks to generate synchronized sensor data.

Acknowledgements

This Research was funded by National Funds through the FCT—Foundation for Science and Technology, in the context of the project UIDB/00127/2020. André Silva Pinto de Aguiar thanks the FCT—Foundation for Science and Technology, Portugal for the Ph.D. Grant DFA/BD/5318/2020.

References

Agarwal, S., Snavely, N., Seitz, S. M., & Szeliski, R. (2010). Bundle adjustment in the large. In K. Daniilidis, P. Maragos, & N. Paragios (Eds.), *Computer Vision – ECCV 2010* (pp. 29–42). Berlin, Heidelberg: Springer Berlin Heidelberg.

de Aguiar, A. S. P., dos Santos, F. B. N., dos Santos, L. C. F., de Jesus Filipe, V. M., & de Sousa, A. J. M. (2020). Vineyard trunk detection using deep learning – an experimental device benchmark. *Computers and Electronics in Agriculture*, *175*, 105535. URL: `https://doi.org/10.1016/j.compag.2020.105535`. doi:`10.1016/j.compag.2020.105535`.

Álvarez, S., Llorca, D., & Sotelo, M. (2014). Hierarchical camera auto-calibration for traffic surveillance systems. *Expert Systems with Applications*, *41*, 1532–1542. URL: https://doi.org/10.1016/j.eswa.2013.08.050. doi:10.1016/j.eswa.2013.08.050.

Badue, C., Guidolini, R., Carneiro, R. V., Azevedo, P., Cardoso, V. B., Forechi, A., Jesus, L., Berriel, R., Paixão, T. M., Mutz, F., de Paula Veronese, L., Oliveira-Santos, T., & De Souza, A. F. (2021). Self-driving cars: A survey. *Expert Systems with Applications*, *165*, 113816. URL: http://www.sciencedirect.com/science/article/pii/S095741742030628X. doi:https://doi.org/10.1016/j.eswa.2020.113816.

Besl, P., & McKay, N. D. (1992). A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *14*, 239–256. URL: https://doi.org/10.1109/34.121791. doi:10.1109/34.121791.

Bradski, G. (2000). The opencv library. *Dr Dobb's J. Software Tools*, *25*, 120–125. URL: https://ci.nii.ac.jp/naid/10028167478/en/.

Czyzewski, M. A. (2017). An extremely efficient chess-board detection for non-trivial photos. *ArXiv*, *abs/1708.03898*.

Dhall, A., Chelani, K., Radhakrishnan, V., & Krishna, K. M. (2017). Lidar-camera calibration using 3d-3d point correspondences. arXiv:arXiv:1705.09785.

Durrant-Whyte, H., & Bailey, T. (2006). Simultaneous localization and mapping: part i. *IEEE Robotics & Automation Magazine*, *13*, 99–110. URL: https://doi.org/10.1109/mra.2006.1638022. doi:10.1109/mra.2006.1638022.

Fabbri, R., Giblin, P., & Kimia, B. (2020). Camera pose estimation using first-order curve differential geometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (pp. 1–1). URL: https://doi.org/10.1109/tpami.2020.2985310. doi:10.1109/tpami.2020.2985310.

Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, *24*, 381–395. URL: https://doi.org/10.1145/358669.358692. doi:10.1145/358669.358692.

Foote, T. (2013). tf: The transform library. In *2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA)*. IEEE. URL: https://doi.org/10.1109/tepra.2013.6556373. doi:10.1109/tepra.2013.6556373.

Fremont, V., & Bonnifait, P. (2008). Extrinsic calibration between a multi-layer lidar and a camera. In *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*. IEEE. URL: https://doi.org/10.1109/mfi.2008.4648067. doi:10.1109/mfi.2008.4648067.

Furgale, P., Rehder, J., & Siegwart, R. (2013). Unified temporal and spatial calibration for multi-sensor systems. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. URL: https://doi.org/10.1109/iros.2013.6696514. doi:10.1109/iros.2013.6696514.

Gao, X.-S., Hou, X.-R., Tang, J., & Cheng, H.-F. (2003). Complete solution classification for the perspective-three-point problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25*, 930–943. URL: https://doi.org/10.1109/tpami.2003.1217599. doi:10.1109/tpami.2003.1217599.

Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F., & Medina-Carnicer, R. (2016). Generation of fiducial marker dictionaries using mixed integer linear programming. *Pattern Recognition*, *51*, 481–491. URL: https://doi.org/10.1016/j.patcog.2015.09.023. doi:10.1016/j.patcog.2015.09.023.

Girshick, M. A., Mosteller, F., & Savage, L. J. (2006). Unbiased estimates for certain binomial sampling problems with applications. In S. E. Fienberg, & D. C. Hoaglin (Eds.), *Selected Papers of Frederick Mosteller* (pp. 57–68). New York, NY: Springer New York. URL: https://doi.org/10.1007/978-0-387-44956-2_3. doi:10.1007/978-0-387-44956-2_3.

Guindel, C., Beltran, J., Martin, D., & Garcia, F. (2017a). Automatic extrinsic calibration for lidar-stereo vehicle sensor setups. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. URL: https://doi.org/10.1109/itsc.2017.8317829. doi:10.1109/itsc.2017.8317829.

Guindel, C., Beltran, J., Martin, D., & Garcia, F. (2017b). Automatic extrinsic calibration for lidar-stereo vehicle sensor setups. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. URL: https://doi.org/10.1109/itsc.2017.8317829. doi:10.1109/itsc.2017.8317829.

Hornegger, J., & Tomasi, C. (1999). Representation issues in the ML estimation of camera motion. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*. IEEE. URL: https://doi.org/10.1109/iccv.1999.791285. doi:10.1109/iccv.1999.791285.

Hu, D., DeTone, D., & Malisiewicz, T. (2019). Deep ChArUco: Dark ChArUco marker pose estimation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. URL: https://doi.org/10.1109/cvpr.2019.00863. doi:10.1109/cvpr.2019.00863.

Huang, J.-K., & Grizzle, J. W. (2020). Improvements to target-based 3d LiDAR to camera calibration. *IEEE Access*, *8*, 134101–134110. URL: https://doi.org/10.1109/access.2020.3010734. doi:10.1109/access.2020.3010734.

Huang, L., & Barth, M. (2009). A novel multi-planar LIDAR and computer vision calibration procedure using 2d patterns for automated navigation. In *2009 IEEE Intelligent Vehicles Symposium*. IEEE. URL: https://doi.org/

10.1109/ivs.2009.5164263. doi:10.1109/ivs.2009.5164263.

su Kim, E., & Park, S.-Y. (2019). Extrinsic calibration between camera and LiDAR sensors by matching multiple 3d planes. *Sensors*, *20*, 52. URL: https://doi.org/10.3390/s20010052. doi:10.3390/s20010052.

Liao, Y., Li, G., Ju, Z., Liu, H., & Jiang, D. (2017). Joint kinect and multiple external cameras simultaneous calibration. In *2017 2nd International Conference on Advanced Robotics and Mechatronics (ICARM)*. IEEE. URL: https://doi.org/10.1109/icarm.2017.8273179. doi:10.1109/icarm.2017.8273179.

Majumder, S., & Pratihar, D. K. (2018). Multi-sensors data fusion through fuzzy clustering and predictive tools. *Expert Systems with Applications*, *107*, 165 – 172. URL: http://www.sciencedirect.com/science/article/pii/S0957417418302677. doi:https://doi.org/10.1016/j.eswa.2018.04.026.

Melendez-Pastor, C., Ruiz-Gonzalez, R., & Gomez-Gil, J. (2017). A data fusion system of gnss data and on-vehicle sensors data for improving car positioning precision in urban environments. *Expert Systems with Applications*, *80*, 28 – 38. URL: http://www.sciencedirect.com/science/article/pii/S0957417417301641. doi:https://doi.org/10.1016/j.eswa.2017.03.018.

Mirzaei, F. M., Kottas, D. G., & Roumeliotis, S. I. (2012). 3d LIDAR–camera intrinsic and extrinsic calibration: Identifiability and analytical least-squares-based initialization. *The International Journal of Robotics Research*, *31*, 452–467. URL: https://doi.org/10.1177/0278364911435689. doi:10.1177/0278364911435689.

Oliveira, M., Castro, A., Madeira, T., Dias, P., & Santos, V. (2020a). A general approach to the extrinsic calibration of intelligent vehicles using ros. In M. F. Silva, J. Luís Lima, L. P. Reis, A. Sanfeliu, & D. Tardioli (Eds.), *Robot 2019: Fourth Iberian Robotics Conference* (pp. 203–215). Cham: Springer International Publishing.

Oliveira, M., Castro, A., Madeira, T., Pedrosa, E., Dias, P., & Santos, V. (2020b). A ROS framework for the extrinsic calibration of intelligent vehicles: A multi-sensor, multi-modal approach. *Robotics and Autonomous Systems*, *131*, 103558. URL: https://doi.org/10.1016/j.robot.2020.103558. doi:10.1016/j.robot.2020.103558.

Pandey, G., McBride, J., Savarese, S., & Eustice, R. (2010). Extrinsic calibration of a 3d laser scanner and an omnidirectional camera. *IFAC Proceedings Volumes*, *43*, 336–341. URL: https://doi.org/10.3182/20100906-3-it-2019.00059. doi:10.3182/20100906-3-it-2019.00059.

de Paula, M., Jung, C., & da Silveira, L. (2014). Automatic on-the-fly extrinsic camera calibration of onboard vehicular cameras. *Expert Systems with Applications*, *41*, 1997–2007. URL: https://doi.org/10.1016/j.eswa.2013.08.096. doi:10.1016/j.eswa.2013.08.096.

Penate-Sanchez, A., Andrade-Cetto, J., & Moreno-Noguer, F. (2013). Exhaustive linearization for robust camera pose and focal length estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*, 2387–2400. URL: https://doi.org/10.1109/tpami.2013.36. doi:10.1109/tpami.2013.36.

Pradeep, V., Konolige, K., & Berger, E. (2014). Calibrating a multi-arm multi-sensor robot: A bundle adjustment approach. In *Experimental Robotics* (pp. 211–225). Springer Berlin Heidelberg. URL: https://doi.org/10.1007/978-3-642-28572-1_15. doi:10.1007/978-3-642-28572-1_15.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., & Ng, A. Y. (2009). Ros: an open-source robot operating system. In *ICRA workshop on open source software* (p. 5). Kobe, Japan volume 3.

Rehder, J., Siegwart, R., & Furgale, P. (2016). A general approach to spatiotemporal calibration in multisensor systems. *IEEE Transactions on Robotics*, *32*, 383–398. URL: https://doi.org/10.1109/tro.2016.2529645. doi:10.1109/tro.2016.2529645.

Romero-Ramirez, F. J., Muñoz-Salinas, R., & Medina-Carnicer, R. (2018). Speeded up detection of squared fiducial markers. *Image and Vision Computing*, *76*, 38–47. URL: https://doi.org/10.1016/j.imavis.2018.05.004. doi:10.1016/j.imavis.2018.05.004.

dos Santos, F. N., Sobreira, H., Campos, D., Morais, R., Moreira, A. P., & Contente, O. (2016). Towards a reliable robot for steep slope vineyards monitoring. *Journal of Intelligent & Robotic Systems*, *83*, 429–444. URL: https://doi.org/10.1007/s10846-016-0340-5. doi:10.1007/s10846-016-0340-5.

Santos, L., Santos, F., Mendes, J., Costa, P., Lima, J., Reis, R., & Shinde, P. (2019). Path planning aware of robot's center of mass for steep slope vineyards. *Robotica*, *38*, 684–698. URL: https://doi.org/10.1017/s0263574719000961. doi:10.1017/s0263574719000961.

Sariff, N., & Buniyamin, N. (2006). An overview of autonomous mobile robot path planning algorithms. In *2006 4th Student Conference on Research and Development*. IEEE. URL: https://doi.org/10.1109/scored.2006.4339335. doi:10.1109/scored.2006.4339335.

Sturm, P. (2014). Pinhole camera model. In K. Ikeuchi (Ed.), *Computer Vision: A Reference Guide* (pp. 610–613). Boston, MA: Springer US. URL: https://doi.org/10.1007/978-0-387-31439-6_472. doi:10.1007/978-0-387-31439-6_472.

Verma, S., Berrio, J. S., Worrall, S., & Nebot, E. (2019). Automatic extrinsic calibration between a camera and a 3D lidar using 3D point and plane correspondences. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)* (pp. 3906–3912). Auckland, New Zealand: IEEE. doi:10.1109/ITSC.2019.8917108.

Wang, W., Sakurada, K., & Kawaguchi, N. (2017). Reflectance intensity assisted automatic and accurate extrinsic calibration of 3d LiDAR and panoramic camera using a printed chessboard. *Remote Sensing*, *9*, 851. URL: https:

//doi.org/10.3390/rs9080851. doi:10.3390/rs9080851.

Zhou, L., Li, Z., & Kaess, M. (2018). Automatic extrinsic calibration of a camera and a 3d LiDAR using line and plane correspondences. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. URL: https://doi.org/10.1109/iros.2018.8593660. doi:10.1109/iros.2018.8593660.

Zuniga-Noel, D., Ruiz-Sarmiento, J.-R., Gomez-Ojeda, R., & Gonzalez-Jimenez, J. (2019). Automatic multi-sensor extrinsic calibration for mobile robots. *IEEE Robotics and Automation Letters*, *4*, 2862–2869. URL: https://doi.org/10.1109/lra.2019.2922618. doi:10.1109/lra.2019.2922618.

3.2 Final remarks

The proposed camera-to-LiDAR calibration approach contributed with several novelties to the state-of-the-art. The main one is the fact that the formulated algorithm can simultaneously calibrate any number of sensors, which means that the entire sensing system present on a robot can be calibrated at once. Another key addition is the developed visualization framework, where the user can see the real-time calibration status, and also position sensors to set the initial guess for the optimization process. On the other side, this work presents two main limitations. The first is that the output of the calibration does not provide a sensor-to-robot transformation, which means that the transformation from the base of the robot to at least one sensor have to be inferred in some other way. Also, the calibration approach does not provide data synchronization which can be important when calibrating sensors with different frame rates.

4 Deep Learning-based semantic vineyard perception

One of the main goals of this thesis is to explore the concept of a semantic approach that can create maps of the crops with meaningful information in different growth stages. For this reason, it is essential to have a semantic perception system that can capture these natural features. In this work, it is proposed the detection of vine trunks and grape bunches using DL concepts. Since the perception system works on-board of robots and feeds the localization and mapping algorithms, lightweight DL models were used in an Edge-AI manner. In this way, the object detection is executed in dedicated hardware at high frequency.

This research line produced four different articles presented in this chapter that demonstrate the evolution of the semantic perception approach. Each article presents significant innovations in relation with the previous ones. The first three approach the vine trunk detection problem and the last one the grape bunch detection at different growth stages. As will be detailed, the work started with a small dataset and a low amount of models running in a Tensor Processing Unit (TPU). Then, it evolved to the exploration of a new device, with a benchmark between them both, and a higher amount of trained and deployed models. Finally, it resulted in a vineyard perception system supported by a novel dataset with more than 9000 images with annotated vine trunks and 1900 with annotated grape bunches. The main idea of this solution is to have a technology that runs in dedicated devices without using resources of the main robot's computer. Since the goal is to have high frequency semantic detections, the models used are lightweight and are

built to run in mobile or embedded devices.

4.1 Visual trunk detection using transfer learning and a Deep Learning-based coprocessor

The main goal for the semantic perception algorithm is to have a system running in an embedded device that can output the detections at high frequency. Such detections are calculated with Convolutional Neural Networks (CNN)s of adequate complexity, the so-called lightweight DL models. One of the most popular family of models that fulfill these requirements are the MobileNets (Howard et al., 2017). These models can run in dedicated hardware such as TPUs and Graphical Processing Units (GPU)s and present two hyperparameters that can be tuned to adjust the tradeoff between latency and accuracy. The framework used, Tensorflow[1], already provides a pretrained version of these models in general datasets such as Coco[2]. In this way, models can be retrained for specific datasets using a technique called Transfer Learning (TL). This thesis uses these concepts to create a vine trunk detector using two versions of the MobileNets. This work produced a publicly available dataset and an article published in the IEEE Access Journal entitled *Visual Trunk Detection Using Transfer Learning and a Deep Learning-Based Coprocessor* (Aguiar et al., 2020b). The dataset is composed of 336 different vineyard images with approximately 1600 annotated vine trunks. Results demonstrated that this approach can achieve vine trunk detection with a frame rate of approximately 50Hz and with an overall Average Precision of 52.98%.

[1] https://www.tensorflow.org/
[2] https://cocodataset.org/

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 19, 2020, accepted April 16, 2020, date of publication April 20, 2020, date of current version May 7, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2989052

Visual Trunk Detection Using Transfer Learning and a Deep Learning-Based Coprocessor

ANDRÉ SILVA AGUIAR[1,2], FILIPE NEVES DOS SANTOS[1], ARMANDO JORGE MIRANDA DE SOUSA[1,3], PAULO MOURA OLIVEIRA[2], AND LUIS CARLOS SANTOS[1,2]

[1]INESC Technology and Science (INESC TEC), 4200-465 Porto, Portugal
[2]Department of Engineering, University of Trás-os-Montes e Alto Douro (UTAD), 5000-801 Vila Real, Portugal
[3]Faculty of Engineering, University of Porto (FEUP), 4200-465 Porto, Portugal

Corresponding author: André Silva Aguiar (andre.s.aguiar@inesctec.pt)

This work was supported in part by the National Funds through the Portuguese funding agency, Fundação para a Ciência e a Tecnologia (FCT), within project under Grant UIDB/50014/2020, in part by the ERDF European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation—COMPETE 2020 under the PORTUGAL 2020 Partnership Agreement—and through the Portuguese National Innovation Agency (ANI) as a part of project "ROMOVI: POCI-01-0247-FEDER-017945."

ABSTRACT Agricultural robotics is nowadays a complex, challenging, and exciting research topic. Some agricultural environments present harsh conditions to robotics operability. In the case of steep slope vineyards, there are several challenges: terrain irregularities, characteristics of illumination, and inaccuracy/unavailability of signals emitted by the Global Navigation Satellite System (GNSS). Under these conditions, robotics navigation becomes a challenging task. To perform these tasks safely and accurately, the extraction of reliable features or landmarks from the surrounding environment is crucial. This work intends to solve this issue, performing accurate, cheap, and fast landmark extraction in steep slope vineyard context. To do so, we used a single camera and an Edge Tensor Processing Unit (TPU) provided by Google's USB Accelerator as a small, high-performance, and low power unit suitable for image classification, object detection, and semantic segmentation. The proposed approach performs object detection using Deep Learning (DL)-based Neural Network (NN) models on this device to detect vine trunks. To train the models, Transfer Learning (TL) is used on several pre-trained versions of MobileNet V1 and MobileNet V2. A benchmark between the two models and the different pre-trained versions is performed. The models are pre-trained in a built in-house dataset, that is publicly available containing 336 different images with approximately 1,600 annotated vine trunks. There are considered two vineyards, one using camera images with the conventional infrared filter and others with an infrablue filter. Results show that this configuration allows a fast vine trunk detection, with MobileNet V2 being the most accurate retrained detector, achieving an overall Average Precision of 52.98%. We briefly compare the proposed approach with the state-of-the-art Tiny YOLO-V3 running on Jetson TX2, showing the outperformance of the adopted system in this work. Additionally, it is also shown that the proposed detectors are suitable for the Localization and Mapping problems.

INDEX TERMS Deep learning, transfer learning, convolutional neural networks, tensor processing unit.

I. INTRODUCTION

The research and development of robotic solutions for the agriculture sector have been growing [1], [2]. The need for automatic machines in this area is increasing since farmers increasingly recognize its impact in agriculture [3]. Robots are now used for a variety of tasks such as planting, harvesting, environmental monitoring, supply of water and nutrients, and others [4]. In this context, developing solutions that allow robots to navigate safely in these environments is essential. To do so, localizing the robotic platform in real-time is required. In vineyards built in steep slope hills, the use of the GNSS is, in most cases, unavailable due to signal blockage and multi-reflection. Thus, several solutions redundant to GNSS have been developed. In particular, Simultaneous Localization and Mapping (SLAM) and Visual Odometry

The associate editor coordinating the review of this manuscript and approving it for publication was Gianluigi Ciocca.

(VO) approaches are in many cases adopted [5]–[8]. In these cases, to give the robot knowledge about the vineyard patterns is a smart solution. The vine trunks can be used as landmarks for the SLAM, and to build a vineyard map. There are solutions to perform these tasks using range sensors [5], [9] or camera systems [10], [11], based on traditional methods such as Kalman Filters (KF), image processing, and others. However, to the best of our knowledge, the use of DL [12] to detect vine trunks is still nonexistent in the literature. The use of this approach is interesting since it provides artificial intelligence to the robot while being, in many cases, an accurate solution. Convolutional Neural Networks (CNN) shown the greatest performance in several contests in machine learning and pattern recognition [13], [14]. This procedure, however, assumes that the training and test data must be in the same feature space, and have the same distribution [15]. However, in some real-world scenarios, data collection can be challenging, as well as time expensive. So, learners that can be trained with data easily collected from different domains are, in some cases, required [16]. In other words, the learning procedure is performed, transferring knowledge from a given task that was already learned. This methodology is called TL [17].

While computing SLAM using a Central Processing Unit (CPU) of a given machine, it is wiser to minimize the CPU resources consumption. For example, the landmark detection task can be executed in a second processing unit. In the case of trunk detection using CNNs with DL or TL, several devices can be used, such as Graphical Processing Units (GPU), TPUs, Vision Processing Units (VPU), and others [18]. This configuration allocates a dedicated device for object detection, maximizing the frame rate of the robot navigation. To do so, the CNNs can be trained and executed using several frameworks. One of the most popular is Tensorflow [19]. This tool allows to create, train, and execute models that can be transferred to heterogeneous devices. Also, this framework supports deployment in embedded and mobile devices with Tensorflow Mobile and Tensorflow Lite. It is possible to convert Tensorflow models to the Lite or Mobile versions using the framework. There are also CNNs optimized for mobile and embedded systems such as the MobileNets [20], and SqueezeNet [21].

This work aims to perform DL-based object detection to:

- Detect high-level visual features in vineyards (vine trunks), in a low-power and high-performance manner;
- Present a reliable visual landmark input to SLAM systems in the vineyard context.

To do so, this work proposes an accurate, cheap, and fast trunk detection in steep slope vineyard context. To achieve these specifications, a single camera and an Edge TPU are used. The Edge TPU is provided by Google's USB Accelerator [22]. It is a small, high-performance, and low power unit suitable for image classification, object detection, and semantic segmentation. This device provides high-performance ML inferencing for TensorFlow Lite models. The proposed approach performs object detection on this device to detect vineyard trunks, using TL. This is done using a few pre-trained versions of MobileNet V1 and MobileNet V2. These are CNNs developed for mobile and embedded vision applications. A benchmark between the two models and the different pre-trained versions is performed, both in terms of processing time and detection precision. The models are pre-trained in a built in-house dataset. Results show that this configuration allows accurate and fast trunk detection, without spending the CPU resources. When compared to Tiny YOLO-V3 [23], the architecture proposed in this work outperforms it both in terms of inference accuracy and runtime performance.

The rest of the paper is described as follows. In the next section, the related work is reviewed. Section III contains the materials used in this work. In particular, the CNN models used and their architecture, and the description of the Edge TPU used. Section IV contains the approach adopted in this work, such as the data collection method, and the training procedure adopted. Section V presents the proposed system results using the built in-house dataset, and the respective analysis and discussion. Finally, the work is summarized in Section VI.

II. RELATED WORKS

At the best of our knowledge, DL has not yet been applied to trunk detection. Even so, image classification and object detection based on DL techniques are widely present in the agriculture sector. Intensive and time expensive tasks are being replaced by automatic machines, endowed with artificial intelligence. These machines are performing operations in the agriculture context such as plant disease detection, weed identification, seed identification, fruit detection and counting, obstacle detection, and others [24]–[26].

To detect tomato plant diseases and pests, Fuentes *et al.* [27] reported a performance comparison between several families of detectors combining them with different CNN models. This work focuses on identifying the infection status, the symptom location, patterns of the leaf, type of fungus, and color and shape of the leaf. The results are generated and compared with and without data augmentation. Similarly, to detect and identify apple leaf diseases Liu *et al.* [28] created a novel CNN architecture based on AlexNet. The network was trained using 13,689 images and is used to detect four common apple diseases. The overall accuracy of the network is 97.62%, which consists of an improvement of 10.83% compared with AlexNet. Barré *et al.* [29] propose a CNN-based plant identification system called LeafNet. This work aims to have a system that learns features from leaf images capable of identifying plant species using them. The method was tested in several datasets such as LeafSnap, Flavia, and Foliage, outperforming hand-crafted-based systems. Potena *et al.* [30] used two CNNs to perform crop and weed identification. The first, a lightweight CNN, is used to segment images in order to extract 3D pixel projections of points that belong to green vegetation. The second, a deeper CNN, is used to classify these pixels to classify the crop and weed.

This configuration allows real-time crop and weed detection on top of an unmanned ground vehicle (UGV). Also, for weed detection, in [31] an unsupervised data labeling approach is proposed. This work uses unmanned aerial vehicle (UAV) images to identify inter-row weeds that constitute the training dataset for a CNN. The network is used to detect the crop and weeds in the images. The performance obtained is comparable to the traditional approaches with supervised data labeling. Ashqar *et al.* [32] use a CNN to classify plant seedlings. In this work, a dataset with approximately 5,000 images with 12 plant species is used. This approach achieved an accuracy of 99.48%. To detect different apple growth stages in orchards, Tian *et al.* created an improved version of YOLO-V3 [23]. Their architecture is prepared for variations in illumination, complex backgrounds, and overlapping apples. The dataset uses augmented images to increase the amount of training data. Results show that accurate and real-time performance is achieved using high resolution images. In this context, many works use CNNs to count fruit. For example, in [33], two CNNs are used to count both apples and oranges. The first extracts the candidate regions of the image, and the second implements a counting algorithm for each region. The performance of the approach is analyzed using both images recorded during the day and the night. Results show that this pipeline presents well behavior using a limited dataset size. Similarly, Deep Count [34] proposes a fruit counting approach. In this work, a modified version of Inception-Resnet [35] is used. The network is trained on synthetic data and evaluated on real data. Fruits are counted even under shadow, occluded by branches, and foliage, or if there is overlap between fruits. The method presents and accuracy of 91% on real data, and 93% on synthetic data. CNNs can also be applied to image segmentation, and this can be used in agriculture. For example, in [36], roots in soil are segmented using U-Net [37]. In this case, the labeling procedure is time expensive. All the images pixels considered to belong to a root, have to be manually and individually annotated. Each image annotation takes, on average, 30 minutes. So, this work uses 50 training images and is evaluated in 867 images. Results show that the system produces segmentations with higher quality than the manual annotations. In [38], DL is used to perform obstacle detection in agricultural fields. The obstacle is standardized and it is detected with a precision of 99.9% in row crops, and 90.8% in grass mowing.

TL applications are far more rare in the agricultural sector. Despite this, few works in this area are reported. For instance, in [39], a CNN is pre-trained with a large and general dataset, with approximately 1000 classes, to initialize the weights. Then, the network is retrained in order to detect 9 diseases on tomatoes. A dataset with 14,828 images of tomato leaves is used. Similarly, a TL technique is also used by Mohanty *et al.* [40] to detect plant diseases. Here, a public dataset with 54,306 images is used to retrain two CNNs, in order to identify 26 diseases. To detect plant species, Ghazi *el al.* [41] use TL on pretrained popular CNN architectures. To increase the training dataset size and reduce the chance of overfitting, the original data was augmented with operations such as rotation, translation, scaling, and reflection. The system presents an accuracy of 80%. In [42], DL and TL are used to extract land information from UAV imagery. Firstly, a CNN is used to exclude linear features, such as roads and bridges. Secondly, the feature extraction procedure is used to extract the desired information using TL. TL can also be used to segment images. For example, in [43], semantic segmentation is applied using a TL technique to identify different crop types. In this work, three datasets are used to compare the classification performance using different retraining efforts. Training data is fully and partially labeled at the pixel level. Results show that TL, even with partially labeled data, presents high accuracy. Douarre *el at.* [44], use TL to segment soil roots in X-Ray tomography data. To retrain the network, simulated training data is used. Results show that soil and root are well segmented, even with shallow contrast between them.

Despite DL being widely used in agriculture, as described in this section, vine trunk detection using CNNs is still a gap of the state-of-the-art. This work proposes to fill this gap, with a low-power and high-performance DL-based trunk detection, suitable for real-time applications in robotics.

III. MATERIALS

In order to achieve high runtime performance in robotics navigation, edge inference was chosen to perform visual detection of vine trunks. Edge inference is the use of a particular Application Specific Integrated Circuit (ASIC) accelerator to deploy a Neural Network (NN) based on a given training dataset. Using this approach, the extraction of landmarks for a SLAM problem is computed using a dedicated device and can achieve high levels of performance with low power costs. In this work, Google's Edge TPU was used. A TPU is a coprocessor designed by Google that is usually connected to a host CPU. In the ideal case, the TPU device implements all the inference operations. Otherwise, the host CPU can perform some of them, but this will slow down the process. Using this configuration, to perform the detection, MobileNets [20] with Single Shot MultiBox Detector (SSD) [45] were used.

A. GOOGLE EDGE TPU

Google's Coral USB Accelerator (Fig. 1), provides an Edge TPU machine learning accelerator coprocessor. It is connected via USB to a host computer, allowing high-speed inference. This device is compatible with Tensorflow Lite, a lightweight version of TensorFlow designed for mobile and embedded devices, and can perform image classification, object detection, and semantic segmentation. To perform such tasks, the Edge TPU uses 8-bit quantized models. So, when training a 32-bit float model from scratch, it has to be either quantized using either *quantization aware training* or *post training quantization*. The first approach simulates the effect of 8-bit values during the training process using quantization nodes in the NN graph. The second does not modify the NN structure and is applied after training. However, it is

FIGURE 1. Google Coral USB Accelerator [22].

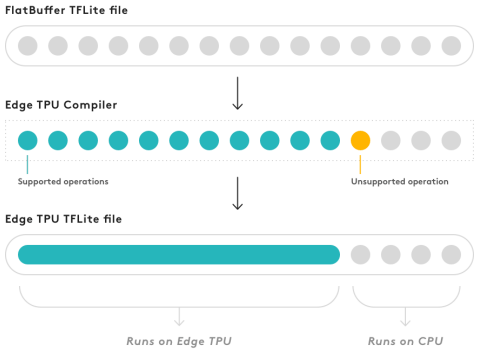


FIGURE 2. Edge TPU model compilation scheme [46].

less accurate than the first method. Alternatively, pre-trained models that are already quantized can be used if compatible with the Edge TPU. The device supports a range of operations and is most likely compatible with models designed for mobile devices, using the SSD architecture. After training the model, the *Edge TPU compiler* is used to assign inference operations to the device and the host CPU, as represented in Fig. 2. The compiled model differs from the original Tensorflow Lite model in the first operation of the graph. CPU processes the operations from the first non-supported operation of the TPU, until the end of the graph. The inference will be as faster as higher it is the number of operations assigned to the Edge TPU. Table 1 the MobileNet V1 compilation results after retraining, on this work. The table shows the supported operations performed by the model on the Edge TPU. When an unsupported operation is found, all the following ones are deployed by the host CPU, represented as *Custom* in Tab. 1.

B. SINGLE SHOT MULTIBOX DETECTOR

To perform the vine trunk detection using the Edge TPU coprocessor, we chose the SSD [45] architecture since it is fully compatible with the device. To perform object detection, this architecture uses a feed-forward CNN producing a fixed-size collection of bounding boxes and attributing a score for each one of them. The CNN contains convolutional feature layers to the end of the truncated base network. These layers allow to detect objects at multiple scales, i.e., objects of different sizes in images with different resolutions.

TABLE 1. Output of the MobileNet V1 model compiler for Edge TPU.

Operator	Status
Logistic	Mapped to Edge TPU
Concatenation	Mapped to Edge TPU
2D Convolution	Mapped to Edge TPU
Depthwise 2D Convolution	Mapped to Edge TPU
Reshape	Mapped to Edge TPU
Custom	Uunsupported data type

C. MOBILENETS

Since in this work, a coprocessor is used to perform machine learning inference, using models suitable for mobile and embedded devices is a logical solution. Thus, MobileNets [20] were chosen. This set of models provide lightweight deep NN using depthwise separable convolutions. In other words, the model factorizes convolutions into depthwise, and 1×1 convolutions called pointwise convolutions. The first applies a single filter to the input channel, and the second applies a 1×1 convolution, combining the outputs of the first. The input of a CNN is a tensor with shape $D_f \times D_f \times M$, where D_f represents the input channel spatial width and height, and M is the input depth. After the convolution, a feature map of shape $D_f \times D_f \times N$ is obtained, where N is the output depth. In this context, MobileNets propose two hyper-parameters that allow the user to resize the model so that it meets the system specifications. There hyper-parameters are: *width multiplier* and *resolution multiplier*. The *width multiplier* α is used to thin the CNN uniformly at each layer. For a given value of $\alpha \in (0, 1]$, the number of inputs channels M becomes αM, as well as the number of output channels N becomes αN. *Width multiplier* reduces the computational cost and the number of parameters by α^2. The second hyper-parameter, *resolution multiplier* ρ, is also used to reduce the computational cost. This one is applied directly to the input image, setting its resolution. The values of $\rho \in (0, 1]$ are chosen in order to obtain typical input image resolutions. *resolution multiplier* also reduces the computational cost and the number of parameters by ρ^2.

In this work, two MobileNet versions provided by TensorFlow [19] are considered, MobileNet V1 and V2. Both models, already trained using the COCO dataset [47], were retrained to detect vine trunks. To analyze the impact of the *width multiplier* hyper-parameter, a version of MobileNet V1 that was pre-trained with a non-default value for this parameter was also retrained. Table 2 indicates the models considered. The *resolution multiplier* is set to its default value in all the experiments, so that the input resolution is 300×300.

TABLE 2. Pretrained models to perform vine trunk detection on the Edge TPU.

Model	Version	*Width Multiplier*	Input Resolution
MobileNet	1	1	300 × 300
MobileNet	1	0.75	300 × 300
MobileNet	2	1	300 × 300

In this work, other variations of the hyper-parameters were not tested since a TL procedure was adopted. This means that the NNs were already trained with specific values for the hyper-parameters, and to experiment other values with the desired impact, they have to be trained from scratch.

IV. METHODS

In this work, TL is addressed to perform DL-based object detection. TL main shortcoming is *negative transfer*, which happens when the pre-training data contributes to negative learning on the target application. To detect vine trunks, several models pre-trained with the COCO dataset [47] were retrained, and was verified that using such a vast dataset, the target detectors retrained do not suffer from *negative transfer*. In order to achieve a high-performance detector, that visually recognizes trunks in real-time, a training procedure over the CNNs was performed. To do so, a training dataset was created, using our robotic platform AgRob V16 [48], represented in Fig. 3. The dataset is publicly available at our repository (`http://vcriis01.inesctec.pt/`) with the `DS_AG_39` id. It contains camera images with both an infrared, and an infrablue filter, in two different vineyards. After collecting the data, the trunks were manually annotated on the images. Then, the training procedure was performed. The main steps of this work are represented in Fig. 4, and are detailed next.

FIGURE 3. AgRob V16 robotic platform.

A. DATA COLLECTION

To build the training dataset, two onboard cameras of our robotic platform, AgRob V16, collected data in two different vineyards, represented if Fig. 5. One of them is a Raspberry

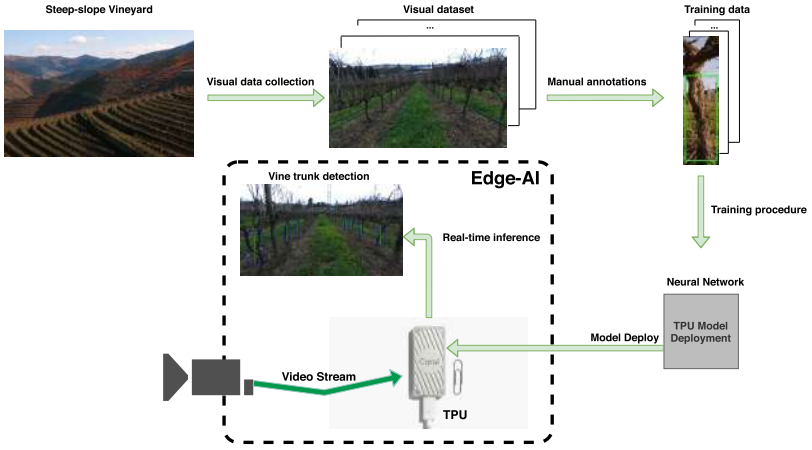


FIGURE 4. High-level design of the vine trunk detection framework.

(a)

(b)

FIGURE 5. Training data on two different vineyards, (a) and (b).

Pi camera with 640 × 480 resolution, with an infrared filter (Fig. 5(b)). The other is a Mako G-125C camera, with a resolution of 1292 × 964 resolution, and an infrablue filter (Fig. 5(a)). The dataset considers 336 different images and approximately 1,600 annotated trunks. It contains:

- Images with different resolutions.
- Two types of vine trunks, with and without foliage.
- Trunks covered by shadows.

These are challenging conditions that confer variety and robustness to the training procedure.

B. DATA ANNOTATION

Given the training dataset, vine trunks were manually annotated. For maintain consistency, on the trunks that are less than approximately 3 meters from the robot and that belong to the corridor where the robot is located, were manually annotated. Figure 6 shows an example of this procedure. The annotation procedure output is a set of bounding boxes of different sizes, for each image. These are represented in a `.xml` file with the Pascal VOC annotation syntax, containing the label class considered, and the four corners location of each bounding box. The annotations are also publicly available (`http://vcriis01.inesctec.pt/`) with the training images. This data is the input for the training procedure, described below.

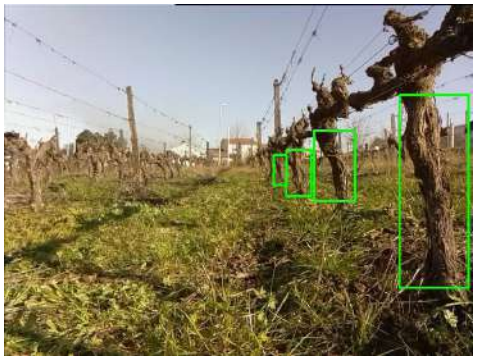


FIGURE 6. Annotation example referent to the training procedure.

C. RETRAINING PROCEDURE

The training procedure aims to create an Edge TPU compatible model, capable of detecting the vine trunks in real-time. To do so, Tensorflow and Tensorflow Lite were used, as represented in Fig. 7. The first step is to serialize the *ground truth* bounding boxes, so that Tensorflow can interpret it efficiently. The data serialization is performed using the *TFRecord* data type, which stores the data as a sequence of binary strings. This constitutes the input for the CNN retraining. This step uses the configurations present in Tab. 2, providing the ability to recognize a trunk to the CNN. After that, a Tensorflow model is generated, and, in order to save the model for posterior TL or retraining, the model is exported into a *frozen graph*. Since the Edge TPU only supports TensorFlow Lite models that are fully 8-bit quantized, the *frozen graph* is then converted into such a model. Finally, in order to assign operations to the Edge TPU device, the Tensorflow Lite model is compiled using the procedure described in Sec. III.

D. EVALUATION METRICS

In order to evaluate the CNN performance detecting vine trunks, the PASCAL Visual Object Classes (VOC) Challenge [49] was used. This method is used by many DL-based works to evaluate CNN performance, offering a fair comparison between the set of works present in the literature. To do so, given an annotated *ground truth* bounding box B_g, and a detected bounding box B_d, the IoU is firstly calculated as follows

$$IoU = \frac{m(B_g \cap B_d)}{m(B_g \cup B_d)} \tag{1}$$

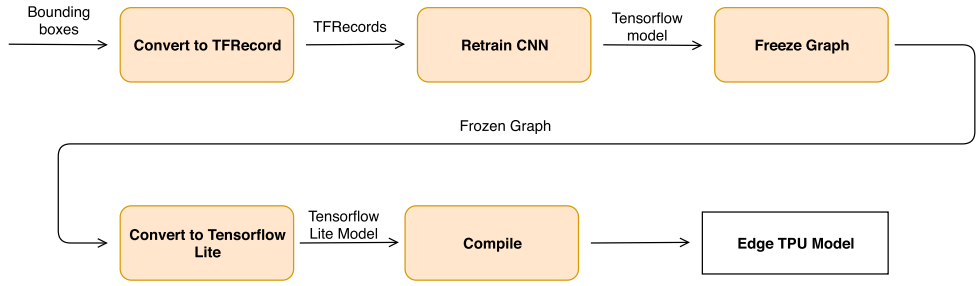


FIGURE 7. Training procedure flow.

where $m(x)$ denotes the area of x. This can be also understood, analysing Fig. 8. So, *IoU* represents the quotient between the area of overlap and the area of union between the *ground truth*, and the detection bounding boxes. Using this definition, three main concepts can be defined. For a given threshold value t, let us define:

- **True Positive** (TP): IoU $\geq t$, i.e., a correct detection.
- **False Positive** (FP): IoU $\leq t$, i.e., an incorrect detection.
- **False Negative** (FN): a *ground truth* is not detected.

It is worth noting that if more than one detection for a single *ground truth* is computed, only the one with the highest IoU is considered as TP, and all the others are FPs. This being said, with these three qualifiers it is possible to define two fundamental concepts. The first, *precision* p, is defined as the total number of **TPs** over all the detections. The second, *recall* r, is the total number of **TPs** over all the *ground truths*. Using these, is possible to plot a curve of the *recall* in function of the *precision*, $p(r)$. The evaluation considers that a suitable detector is the one that maintains the precision high for an increase in recall. With this consideration, the detector is evaluated computing the Average Precision (AP), interpolating the obtained curve, and calculating the area below the curve. Mathematically, this is expressed as follows

$$\sum_{r=0}^{1}(r_{n+1} - r_n)p_{interp}(r_{n+1}) \tag{2}$$

with

$$p_{interp}(r_{n+1}) = \max_{\widetilde{r};\widetilde{r}\geq r_{n+1}} p(\widetilde{r}) \tag{3}$$

where $p(\widetilde{r})$ is the measured precision at recall $\widetilde{r}$.

V. RESULTS

In order to evaluate the trained NNs on top of Google's Edge TPU, a subset of the dataset previously described was extracted for testing. From the total of 336 images on the dataset, 45 images were used for the test procedure, with approximately 180 vine trunks. These images were randomly extracted from the dataset before training in order to be only used for validation. The performance of the three model configurations is compared using all the evaluation data, and the images of each vineyard individually. Thus, the global performance of the detector is evaluated, as well as its isolated performance in the images with both types of filters. Tiny YOLO-V3 was trained and evaluated using exactly the same training and testing data, respectively. So, a brief comparison is performed between this model running on Jetson TX2 and the proposed system on this work. Additionally, this Sec. shows an application of the proposed detectors to a Localization and Mapping system.

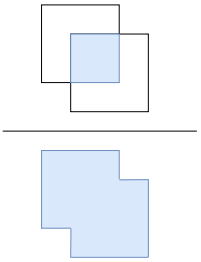


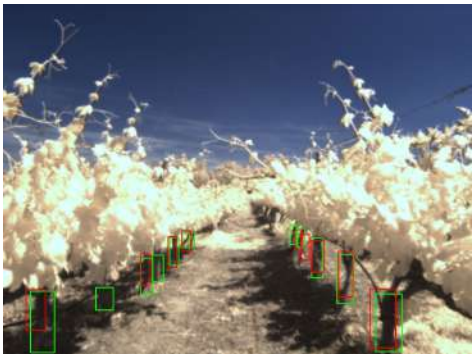
FIGURE 8. Interception over union representation.

A. DETECTION PERFORMANCE

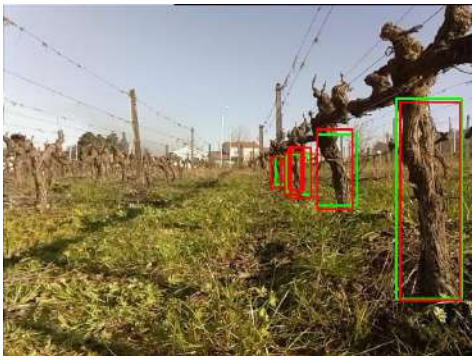
Using the metrics described on Sec. IV, and the test set of images, the detectors were evaluated. Figure 9 shows an example of a detection, provided by MobileNet V1, with $\alpha = 1.0$, for both vineyards. The green bounding boxes represent the *ground truth*, and the red ones, the detections.

Figures [10-12] represent the *precision* $\times$ *recall* curves $p(r)$, for all the considered configurations: the three retrained models, either with the *IoU* threshold t equals to 0.5 and 0.65, under the three evaluation sets. Table 3 summarizes the AP for all the configurations.

To evaluate the runtime performance of each detector, they were profiled using the *high resolution clock* from `chrono`

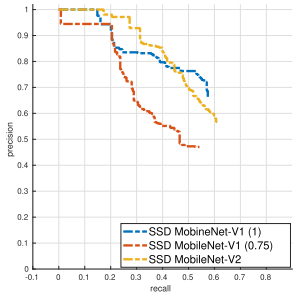


(a)

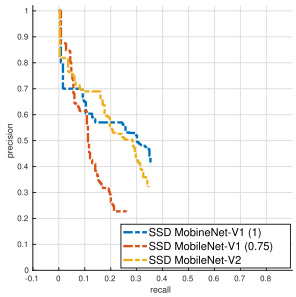


(b)

FIGURE 9. A result example for the two different vineyards, (a) and (b). The red boxes represent the detections, and the green ones the *ground truth*.



(a)



(b)

FIGURE 10. Interpolated AP p_{interp} results using all the training data and a IoU threshold of (a) 0.5 and (b) 0.65.

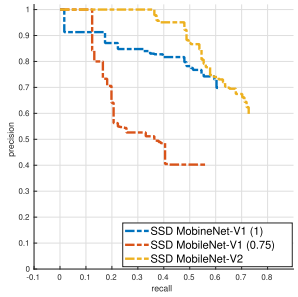
present in the *std* library. This measure consists of the time that the detector takes to return the resultant bounding boxes, since it receives the input image. Table 4 shows the obtained results for each detector.

To compare the proposed system with a state-of-the-art model, Tiny YOLO-V3 was training with the built in-house dataset and evaluated over the same 45 images using an IoU equals to 0.5. Table 5 summarizes the results obtained with this configuration, both in terms of AP and runtime performance.

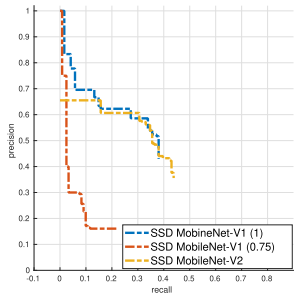
B. DISCUSSION

By analysis of Tab. 3, several conclusion can be extracted. Both for the global set of images, and the infrared filtered ones, MobileNet V2 is the best detector. For $t = 0.5$, the difference to the other detectors is significant. With $t = 0.65$, this margin tends to attenuate. On the infrablue filtered set of images, MobileNetV1 ($\alpha = 1.0$) is, not by far, the best detector. Despite this, the three detectors behave very similarly on this set of images. Comparing the performance of the detectors on the two filtered set of images, the conclusion is that the infrablue is, in general, more challenging. Except for MobileNet V1 ($\alpha = 0.75$), the other detectors behave better on the set with vineyard images using the conventional infrared filter. Obviously, increasing the IoU threshold t leads to a decrease in the average precision, as verified in all the studied cases. Globally, MobileNet V2 is the detector that presents the best performance, providing an AP of 52.98% on the global set of images, 62.95% on the infrared filtered images, and finally, 41.33% on the infrablue filtered ones, for a *width multiplier* $t = 0.5$. Comparing these results with the ones present in Tab. 5, it is visible that Tiny YOLO-V3 presents a lower AP than all the proposed configurations for the same value of t, showing an overall AP of 31.32%.

In terms of inference runtime performance, Tab. 4 shows the average inference per image for all the detectors, in milliseconds. These results were generated, taking into account all the evaluation images considered, computing the average inference time of each detector in all of them. They show that MobileNet V1 ($\alpha = 0.75$) is the fastest detector with

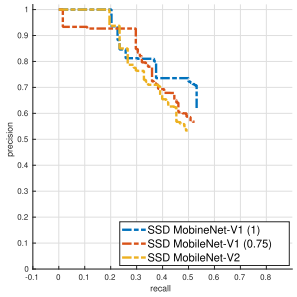


(a)

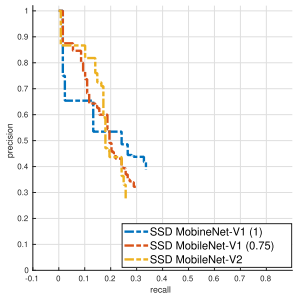


(b)

FIGURE 11. Interpolated AP p_{interp} results using the infrared training images and a IoU threshold of (a) 0.5 and (b) 0.65.



(a)



(b)

FIGURE 12. Interpolated AP p_{interp} results using the infrablue training images and a IoU threshold of (a) 0.5 and (b) 0.65.

20.5326 ms average inference per image, which corresponds to 48.7030 *frames per second*. MobileNet V1 ($\alpha = 1.0$), is slower than the previous one, but faster than MobileNet V2. The first achieves an average inference time of 21.1853 ms, which corresponds to 47.2025 *frames per second*. The second, MobileNet V2, presents 23.8238 ms of average inference time, and, consequently, 41.9748 *frames per second*. Thus, MobileNet V1 ($\alpha = 0.75$) can process 5 more images per second than MobileNet V2. Despite this, it is notorious that all the detectors being executed on top of the Edge TPU present high performance, being suitable for real-time usage. The architecture of this TPU, dedicated explicitly to processing CNNs, revealed to process DL-based object detectors with time performances that can be used in any visual SLAM system. The loop frequency of SLAM is, in most cases, not higher than 20 *frames per second*. Our detectors achieved time performances that can process more than twice this number of *frames per second*. Table 5 presents the average runtime performance of Tiny YOLO-V3 on top of Jetson TX2, resulting in an average inference time per image of 54.20 milliseconds. This result corresponds to a frame rate of 18.45 *frames per second*, which leads to the conclusion that the MobileNets run more than twice as fast over the Edge TPU than YOLO on top of the Jetson TX2.

TABLE 3. Summary of the detector performance.

Model	Overall (%)		Infrared (%)		Infrablue (%)	
	t		t		t	
	0.5	0.65	0.5	0.65	0.5	0.65
MobileNet V1 ($\alpha = 1.0$)	49.74	21.19	51.66	25.07	46.11	19.36
MobileNet V1 ($\alpha = 0.75$)	39.78	12.97	33.73	3.89	43.37	18.81
MobileNet V2 ($\alpha = 1.0$)	52.98	21.09	62.95	28.15	41.33	18.16

Globally, the main conclusions that can be taken from this analisys are:

- MobileNet V2 is the most accurate detector, from the set of three detectors analyzed.
- MobileNet V1 ($\alpha = 0.75$) is the fastest detector, but globally the least accurate one.
- Tiny YOLO-V3 is shows lower AP and lower runtime performance running on top of Jetson TX2 than the MobileNets on the Edge TPU.

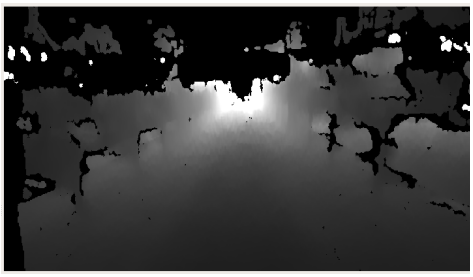
These conclusions confirm the *a-priori* knowledge about the detectors and their hyper-parameters. MobileNet V2 is an improvement to the first version of the MobileNets,

TABLE 4. Runtime performance (ms) of the different retraining configurations performed.

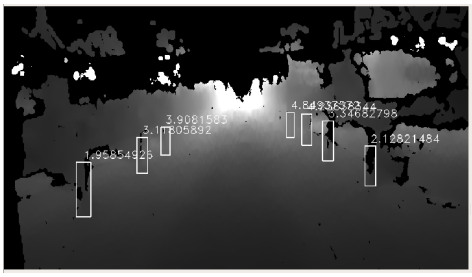
Model	Average inference time per image (ms)	Inference time standard deviation (ms)
MobileNet V1 (α = 1.0)	21.1853	0.536463
MobileNet V1 (α = 0.75)	20.5326	0.518247
MobileNet V2 (α = 1.0)	23.8238	0.662605

TABLE 5. Tiny YOLO-V3 AP and runtime performance results for an IoU equals to 0.5.

Model	Inference average precision			Average inference time per image (ms)
	Overall (%)	Infrared (%)	Infrablue (%)	
Tiny YOLO-V3	31.32	36.40	28.25	54.20



(a)



(b)

FIGURE 13. (a) Disparity map constructed using a stereo camera system and (b) vine trunk detections projected on it with the respective depth information.

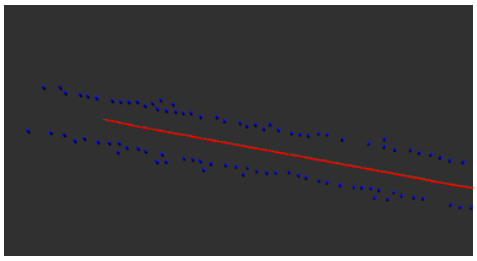


FIGURE 14. 2D vine trunk mapping using the proposed detectors.

introducing new features to the original architecture. Thus, the improvement in comparison to the first version was expected. Also, the behavior of the MobileNets with the variation of the *width multiplier* hyper-parameter, was verified. With $\alpha = 0.75$, MobileNet V1 is, in general, less accurate but faster than using a higher value for α. Additionally, Tiny YOLO-V3, a lightweight version of YOLO-V3, showed an overall lower performance than the proposed configurations on this work, even being executed in a much *high-cost* device. The optimized architecture of the Edge TPU for CNN models suitable for embedded devices leads to a much higher frame rate and a more effective inference performance.

The state-of-the-art does not currently provide any work that detects vine trunks on images using NNs. This work presents a solution to this problem using a device that is yet not popular in the literature. This solution has strong and weak points, depending on the application that uses its final results. It can be concluded that:

- The AP results obtained were not as high as many DL works present in the literature. However, in this work was used a *low-cost* and *low-power* device, that uses *lightweight* and 8-bit *quantized* models - MobileNets. This leads to least accurate inference results but, at the same time, inference is performed with low power consumption, at high *frame rate*, with much less costs than works that use, for example, powerful GPUs.

As referenced before, the desired application for the proposed detector of this work is SLAM. In steep slope vineyards, GNSS-based signals such as the Global Positioning System (GPS) are not always available. So, redundant solutions to GPS have to be developed. The solution proposed in this work can be suitable for such an application. The navigation stack where the detector will be included imposes a minimum *frame rate* of 10 *frames per second*. This condition is ensured, as demonstrated before. Additionally, the detections can be used both on mapping and localization tasks, considering, for example, a stereo camera system. Figure 13 represents the application of the proposed detector on such a system. Using both cameras it is possible to compute a disparity map, as represented in Fig. 13(a), that provides depth information. Thus, the detections on the original stereo images can be

projected on this map. The depth of each trunk is calculated computing the median of the depth of all points inside each bounding box. Figure 13(b) represents the bounding boxes projection and the depth calculation for each one. Using the computed depths, and the implicit bearing information, it is possible to calculate the position of each trunk on the world, with a given standard deviation. This information can be further used both for the mapping and localization procedures. Figure 14 shows an example of a vineyard corridor map build using this information and real data from the robotic platform. The robot trajectory is represented in red. It is worth noting that this procedure does not consider yet a procedure to remove outliers.

VI. CONCLUSION

The SLAM problem is still an intensive research topic. The primary step of any SLAM method is to extract reliable features from the surrounding environment. In the context of steep slope vineyards, the vine trunks can constitute these features. In this work, a real-time DL-based approach to compute the visual detection of vine trunks is proposed. The Edge TPU provided by Google's USB Accelerator is used, performing TL to develop reliable trunk detectors. Two versions of the MobileNets were retrained, taking into consideration their hyper-parameters. The retraining process was performed using a built in-house dataset, that is publicly available. It contains 336 different images with approximately 1,600 annotated trunks and images belonging to two different vineyards. One of them presents camera images with the conventional infrared filter, and the images with an infrablue filter. Results show that our system achieves real-time vine trunk detection. Compared with the state-of-the-art model Tiny YOLO-V3 running on Jetson TX2, the configurations proposed in this work achieve higher inference accuracy and runtime performance. MobileNet V2 revealed to be the most accurate model.

In future work, the vine dataset will be increased both in size and in variability. To do so, the objectives are: to collect data relative to different vines, to add thermal camera images to the dataset, and to augment all the images, performing operations such as rotation, translation, rescaling, etc. Also, it is planned to retrain another kind of models to perform vine trunk detection, such as VggNet, ResNet, InceptionNet, etc. Similarly, it is intended to train NNs from scratch in order to be able to vary the hyper-parameters. Finally, DL-based semantic segmentation will be considered in order to extract the exact shape of each trunk and eliminate the background pixels that are present on the bounding boxes of our detectors.

ACKNOWLEDGMENT

The opinions included in this article shall be the sole responsibility of their authors. The European Commission and the Authorities of the Programme are not responsible for the use of information contained therein.

REFERENCES

[1] F. A. Auat Cheein and R. Carelli, "Agricultural robotics: Unmanned robotic service units in agricultural tasks," *IEEE Ind. Electron. Mag.*, vol. 7, no. 3, pp. 48–58, Sep. 2013.
[2] E. A. Skvortsov, E. G. Skvortsova, I. S. Sandu, and G. A. Iovlev, "Transition of agriculture to digital, intellectual and robotics technologies," *Economy Region*, vol. 14, no. 3, pp. 1014–1028, Sep. 2018.
[3] J. Billingsley, A. Visala, and M. Dunn, *Robotics in Agriculture and Forestry*. Berlin, Germany: Springer, 2008, pp. 1065–1077, doi: 10.1007/978-3-540-30301-5_47.
[4] J. J. Roldán, J. del Cerro, D. Garzón-Ramos, P. Garcia-Aunon, M. Garzón, J. de León, and A. Barrientos, *Robots Agriculture: State Art Practical Experiences*. London, U.K.: IntechOpen, 2018. [Online]. Available: https://app.dimensions.ai/details/publication/pub.1100266943 and [Online]. Available: https://www.intechopen.com/citation-pdf-url/56199
[5] F. B. N. D. Santos, H. M. P. Sobreira, D. F. B. Campos, R. M. P. M. D. Santos, A. P. G. M. Moreira, and O. M. S. Contente, "Towards a reliable monitoring robot for mountain vineyards," in *Proc. IEEE Int. Conf. Auto. Robot Syst. Competitions*, Apr. 2015, pp. 37–43.
[6] F. Auat Cheein, G. Steiner, G. Perez Paina, and R. Carelli, "Optimized EIF-SLAM algorithm for precision agriculture mapping based on stems detection," *Comput. Electron. Agricult.*, vol. 78, no. 2, pp. 195–207, Sep. 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168169911001542
[7] M. Duarte, F. N. dos Santos, A. Sousa, and R. Morais, "Agricultural wireless sensor mapping for robot localization," in *Proc. 2nnd Iberian Robot. Conf.*, L. P. Reis, A. P. Moreira, P. U. Lima, L. Montano, and V. Muñoz-Martinez, Eds. Cham, Switzerland: Springer, 2016, pp. 359–370.
[8] A. Aguiar, F. Santos, L. Santos, and A. Sousa, "Monocular visual odometry using fisheye lens cameras," in *Progress in Artificial Intelligence*, P. Moura Oliveira, P. Novais, and L. P. Reis, Eds. Cham, Switzerland: Springer, 2019, pp. 319–330.
[9] S. Marden and M. Whitty, "GPS-free localisation and navigation of an unmanned ground vehicle for yield forecasting in a vineyard," in *Proc. Int. Workshop Collocated 13th Int. Conf. Intell. Auton. Syst. (IAS)*, 2014, pp. 1–10.
[10] J. Mendes, F. N. D. Santos, N. Ferraz, P. Couto, and R. Morais, "Vine trunk detector for a reliable robot localization system," in *Proc. Int. Conf. Auto. Robot Syst. Competitions (ICARSC)*, May 2016, pp. 1–6.
[11] J. M. Mendes, F. N. dos Santos, N. A. Ferraz, P. M. do Couto, and R. M. dos Santos, "Localization based on natural features detector for steep slope vineyards," *J. Intell. Robotic Syst.*, vol. 93, nos. 3–4, pp. 433–446, Mar. 2019.
[12] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.
[13] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608014002135
[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Red Hook, NY, USA: Curran Associates, 2012, pp. 1097–1105 [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-conv olutional-neural-networks.pdf
[15] S. Jialin Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.
[16] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *J. Big Data*, vol. 3, no. 1, Dec. 2016, doi: 10.1186/s40537-016-0043-6.
[17] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of Research on Machine Learning Applications*. Hershey, PA, USA: IGI Global, Jan. 2009.
[18] Y. E. Wang, G.-Y. Wei, and D. Brooks, "Benchmarking TPU, GPU, and CPU platforms for deep learning," arXiv, Cornell Univ., Ithaca, NY, USA, Tech. Rep. 1907.10701, 2019.
[19] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, and M. Kudlur, "Tensorflow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Operating Syst. Design Implement. (OSDI)*. Savannah, GA, USA, Nov. 2016, pp. 265–283. [Online]. Available: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi
[20] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," arXiv, Cornell Univ., Ithaca, NY, USA, Tech. Rep. 1704.04861, 2017.

[21] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size," arXiv, Cornell Univ., Ithaca, NY, USA, Tech. Rep. 1602.07360, 2016.
[22] Google. (2019). *USB Accelerator*. Accessed: Jan. 3, 2020. [Online]. Available: https://coral.ai/products/accelerator
[23] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," arXiv, Cornell Univ., Ithaca, NY, USA, Tech. Rep. 1804.02767, 2018.
[24] L. Santos, F. N. Santos, P. M. Oliveira, and P. Shinde, "Deep learning applications in agriculture: A short review," in *Proc. 4th Iberian Robot. Conf.*, M. F. Silva, J. L. Lima, L. P. Reis, A. Sanfeliu, and D. Tardioli, Eds. Cham, Switzerland: Springer International Publishing, 2020, pp. 139–151.
[25] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," *Comput. Electron. Agricult.*, vol. 147, pp. 70–90, Apr. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168169917308803
[26] A. Kamilaris and F. X. Prenafeta-Boldú, "A review of the use of convolutional neural networks in agriculture," *J. Agricult. Sci.*, vol. 156, no. 3, pp. 312–322, Apr. 2018.
[27] A. Fuentes, S. Yoon, S. Kim, and D. Park, "A robust Deep-Learning-Based detector for real-time tomato plant diseases and pests recognition," *Sensors*, vol. 17, no. 9, p. 2022, 2017. [Online]. Available: https://www.mdpi.com/1424-8220/17/9/2022
[28] B. Liu, Y. Zhang, D. He, and Y. Li, "Identification of apple leaf diseases based on deep convolutional neural networks," *Symmetry*, vol. 10, no. 1, p. 11, 2018. [Online]. Available: https://www.mdpi.com/2073-8994/10/1/11
[29] P. Barré, B. C. Stöver, K. F. Müller, and V. Steinhage, "LeafNet: A computer vision system for automatic plant species identification," *Ecological Informat.*, vol. 40, pp. 50–56, Jul. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1574954116302515
[30] C. Potena, D. Nardi, and A. Pretto, "Fast and accurate crop and weed identification with summarized train sets for precision agriculture," in *Intelligent Autonomous Systems*, W. Chen, K. Hosoda, E. Menegatti, M. Shimizu, and H. Wang, Eds. Cham, Switzerland: Springer, 2017, pp. 105–121.
[31] M. Bah, A. Hafiane, and R. Canals, "Deep learning with unsupervised data labeling for weed detection in line crops in UAV images," *Remote Sens.*, vol. 10, no. 11, p. 1690, 2018. [Online]. Available: https://www.mdpi.com/2072-4292/10/11/1690
[32] B. A. M. Ashqar, B. S. Abu-Nasser, and S. S. Abu-Naser, "Plant seedlings classification using deep learning," *Int. J. Academic Inf. Syst. Res.*, vol. 3, no. 1, pp. 7–14, 2019.
[33] S. W. Chen, S. S. Shivakumar, S. Dcunha, J. Das, E. Okon, C. Qu, C. J. Taylor, and V. Kumar, "Counting apples and oranges with deep learning: A data-driven approach," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 781–788, Apr. 2017.
[34] M. Rahnemoonfar and C. Sheppard, "Deep count: Fruit counting based on deep simulated learning," *Sensors*, vol. 17, no. 4, p. 905, 2017. [Online]. Available: https://www.mdpi.com/1424-8220/17/4/905
[35] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. (2017). *Inception-V4, Inception-Resnet and the Impact of Residual Connections on Learning*. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14806/14311
[36] A. G. Smith, J. Petersen, R. Selvan, and C. R. Rasmussen, "Segmentation of roots in soil with U-Net," arXiv, Cornell Univ., Ithaca, NY, USA, Tech. Rep. 1902.11050, 2019.
[37] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham, Switzerland: Springer, 2015, pp. 234–241.
[38] K. Steen, P. Christiansen, H. Karstoft, and R. Jørgensen, "Using deep learning to challenge safety standard for highly autonomous machines in agriculture," *J. Imag.*, vol. 2, no. 1, p. 6, 2016. [Online]. Available: https://www.mdpi.com/2313-433X/2/1/6
[39] M. Brahimi, K. Boukhalfa, and A. Moussaoui, "Deep learning for tomato diseases: Classification and symptoms visualization," *Appl. Artif. Intell.*, vol. 31, no. 4, pp. 299–315, Apr. 2017, doi: 10.1080/08839514.2017.1315516.
[40] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," *Frontiers Plant Sci.*, vol. 7, p. 1419, Sep. 2016. [Online]. Available: https://www.frontiersin.org/article/10.3389/fpls.2016.01419
[41] M. Mehdipour Ghazi, B. Yanikoglu, and E. Aptoula, "Plant identification using deep neural networks via optimization of transfer learning parameters," *Neurocomputing*, vol. 235, pp. 228–235, Apr. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231217300498
[42] H. Lu, X. Fu, C. Liu, L.-G. Li, Y.-X. He, and N.-W. Li, "Cultivated land information extraction in UAV imagery based on deep convolutional neural network and transfer learning," *J. Mountain Sci.*, vol. 14, no. 4, pp. 731–741, Apr. 2017.
[43] P. Bosilj, E. Aptoula, T. Duckett, and G. Cielniak, "Transfer learning between crop types for semantic segmentation of crops versus weeds in precision agriculture," *J. Field Robot.*, vol. 37, no. 1, pp. 7–19, Jan. 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21869
[44] C. Douarre, R. Schielein, C. Frindel, S. Gerth, and D. Rousseau, "Transfer learning from synthetic data applied to soil–root segmentation in X-ray tomography images," *J. Imag.*, vol. 4, no. 5, p. 65, 2018. [Online]. Available: https://www.mdpi.com/2313-433X/4/5/65
[45] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot MultiBox detector," 2015, *arXiv:1512.02325*. [Online]. Available: http://arxiv.org/abs/1512.02325
[46] Google. (2019). *Tensorflow Models on the Edge TPU*. Accessed: Jan. 5, 2020. [Online]. Available: https://coral.ai/docs/edgetpu/models-intro/
[47] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Computer Vision–ECCV*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham, Switzerland: Springer, 2014, pp. 740–755.
[48] L. Santos, F. Santos, J. Mendes, P. Costa, J. Lima, R. Reis, and P. Shinde, "Path planning aware of robot's center of mass for steep slope vineyards," *Robotica*, vol. 38, no. 4, pp. 684–698, 2020.
[49] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

ANDRÉ SILVA AGUIAR received the M.Sc. degree in electrical engineering (specialized in robotics) from the Faculty of Engineering, University of Porto, Portugal, in 2019. He is currently pursuing the Ph.D. degree in electrical and computers engineering with UTAD University. He is also a Researcher with the Centre for Robotics in Industry and Intelligent Systems, INESC TEC, Porto. His current research interests are focused on robotics navigation, simultaneous localization and mapping, and deep learning.

FILIPE NEVES DOS SANTOS received the degree (five years degree) in electrical and computer engineering from the Instituto Superior de Engenharia do Porto (ISEP), in 2003, the M.Sc. degree in electrical and computer engineering from the Instituto Superior Técnico (IST), Universidade Técnica de Lisboa, in 2007, and the Ph.D. degree in electrical and computer engineering from the Faculdade de Engenharia (FEUP), Universidade do Porto, Portugal, in 2014. Since 2003, he has been a formal Robotics Researcher, with a large experience on navigation systems for unnamed aerial and ground vehicles. He is currently a Senior Researcher and the Head of the Laboratory of Robotics and the IoT for Smart Precision Agriculture and Forestry, Centre for Robotics in Industry and Intelligent Systems (CRIIS), INESC TEC. He has more than 40 peer-reviewed articles in international conferences and international journals. Actually, his priority research fields are robotics for agriculture/forestry, human–robot interaction, perception, and semantic SLAM.

ARMANDO JORGE MIRANDA DE SOUSA received the Ph.D. degree in electrical and computer engineering (ECE) from the Faculty of Engineering, University of Porto (FEUP), in 2004. His thesis work was in the subarea of automation and robotics. He is currently a Professor with the ECE Department, FEUP, and also an Integrated Researcher with the Center for Intelligent and Industrial Systems, INESC TEC Interface Institute. In 2014, he received the international pedagogical certification ''ING.PAED.IGIP'' from the International Society for Engineering Education and Modern Engineering Pedagogy. His research areas include higher education side by side with more technical areas of automation and robotics. As a frequent participant in robotic contests, he has earned several national and international merits (examples: vice champion of Robotic Soccer in 2006, winner of the Autonomous Driving of Portuguese Robotics Open of 2019). He has also earned educational awards such as the University of Porto (UP) Excellence Award, in 2015, and ranked within the ten best at ECEL 2015 excellence ELearning awards. He has published more than 80 indexed peer-reviewed articles both in pedagogical issues and more technical areas. One of his most cited articles is ''Robust 3/6 DoF self-localization system with selective map update for mobile robot platforms.'' He also has an active patent entitled ''Device and method for identifying a cork stopper, and respective kit.'' He is also involved in educational and technical funded projects such as ''IntelWheels 2'' and ''EIT CPP 101.''

PAULO MOURA OLIVEIRA received the degree in electrical engineering from UTAD University, Portugal, in 1991, and the M.Sc. degree in industrial control systems and the Ph.D. degree in control engineering from Salford University, Manchester, U.K., in 1994 and 1998, respectively. He is currently an Associate Professor with Habilitation with the Engineering Department, UTAD University, and also a Senior Researcher with the Centre for Robotics in Industry and Intelligent Systems (CRIIS), INESC TEC Institute, Porto. He has authored more than 150 peer-reviewed scientific publications. His current research interests are focused on the fields of control engineering, robotics, evolutionary and nature inspired algorithms for single and multiple objective optimization problem solving, swarm optimization, intelligent control, PID control, and control engineering education.

LUIS CARLOS SANTOS received the integrated master's degree in electrical and computers engineering (specialized in robotics) from the Faculty of Engineering, University of Porto, Portugal, in 2017. He is currently pursuing the Ph.D. degree in electrical and computers engineering with UTAD University. He is also a Researcher with the Centre for Robotics in Industry and Intelligent Systems, INESC TEC, Porto. His current research interests are focused on agricultural robotic navigation, specifically in advanced path planning techniques of farming scenarios. His current research topics focus on the safety supervision of robotic software for agriculture.

4.2 Vineyard trunk detection using deep learning - An experimental device benchmark

The preliminary work presented in Section 4.1 opened an interesting research topic: the development of a fast and reliable semantic perception system that runs in dedicated hardware. This section presents the continuation of this research line with an article published in the Computers and Electronics in Agriculture Journal entitled *Vineyard trunk detection using deep learning – An experimental device benchmark* (de Aguiar et al., 2020). In this work, the device used in the previous article - Google's USB Accelerator TPU - is benchmarked against the NVIDIA's Jetson Nano GPU. With the same dataset, seven different models were retrained, deployed in both devices, and evaluated. In addition to the MobileNets already presented in the last article, this work also explored the Inception model (Szegedy et al., 2015) and a lightweight version of the YOLO model (Redmon et al., 2015). Both devices are compared considering their compatibility, i.e., the amount of models supported; the floating-point support, i.e., the ability to work with floats of 64 bits in addition to 8-bit integers; the overall average precision of the models deployed; their runtime performance; and finally, the amount of time spent to load the models to the embedded hardware. This critical analysis provides a fair benchmark and can be used by the scientific community to make a more informed choice when working with DL models in dedicated hardware.

Vineyard Trunk Detection using Deep Learning - An Experimental Device Benchmark

André Silva Pinto de Aguiar[a,b,*], Filipe Baptista Neves dos Santos[a], Luís Carlos Feliz dos Santos[a,b], Vitor Manuel de Jesus Filipe[a,b], Armando Jorge Miranda de Sousa[a,c]

[a]*INESC TEC - INESC Technology and Science; 4200-465, Porto, Portugal*
[b]*School of Science and Technology, University of Trás-os-Montes e Alto Douro; 5000-801 Vila Real, Portugal*
[c]*Faculty of Engineering, University of Porto; 4200-465, Porto, Portugal*

Abstract

Research and development in mobile robotics are continuously growing. The ability of a human-made machine to navigate safely in a given environment is a challenging task. In agricultural environments, robot navigation can achieve high levels of complexity due to the harsh conditions that they present. Thus, the presence of a reliable map where the robot can localize itself is crucial, and feature extraction becomes a vital step of the navigation process. In this work, the feature extraction issue in the vineyard context is solved using Deep Learning to detect high-level features - the vine trunks. An experimental performance benchmark between two devices is performed: NVIDIA's Jetson Nano and Google's USB Accelerator. Several models were retrained and deployed on both devices, using a Transfer Learning approach. Specifically, MobileNets, Inception, and lite version of You Only Look Once are used to detect vine trunks in real-time. The models were retrained in a built in-house dataset, that is publicly available. The training dataset contains approximately 1,600 annotated vine trunks in 336 different images. Results show that NVIDIA's Jetson Nano provides compatibility with a wider variety of Deep Learning architectures, while Google's USB Accelerator is limited to a unique family of architectures to perform object detection. On the other hand, the Google device showed an overall Average precision higher than Jetson Nano, with a better runtime performance. The best result obtained in this work was an average precision of 52.98% with a runtime performance of 23.14 milliseconds per image, for MobileNet-V2. Recent experiments showed that the detectors are suitable for the use in the Localization and Mapping context.

Keywords: Deep Learning, Object Detection, Agricultural Robots

1. Introduction

The Oporto vineyards, Fig. 1, are located in the Douro Demarched Region, the oldest controlled winemaking region in the world, a UNESCO heritage place [1]. These vineyards are built

*Corresponding author
Email addresses: andre.s.aguiar@inesctec.pt (André Silva Pinto de Aguiar), fbsantos@inectec.pt (Filipe Baptista Neves dos Santos), luis.c.santos@inesctec.pt (Luís Carlos Feliz dos Santos), vfilipe@utad.pt (Vitor Manuel de Jesus Filipe), asousa@fe.up.pt (Armando Jorge Miranda de Sousa)

Figure 1: Typical steep slope vineyard in the Douro's region.

in steep slope hills, which brings several challenges to the development of robotic solutions in this context. The characteristics of the hill cause signal blockage that decreases the accuracy of signals emitted by the Global Navigation Satellite System (GNSS), making unreliable the use of, for example, the Global Positioning System (GPS). Also, the terrain highly characterized by irregularities leads to high inaccuracy of sensors like wheel odometry and Inertial Measurement Units (IMU)s [2].

The vast extension of the vineyard and the challenging conditions that they present lead to an increasing need for the substitution of human labor by automatic and autonomous machines. These machines can be used to perform operations such as planting, harvesting, environmental monitoring, supply of water and nutrients [3] and have the potential to transform and have a significant impact in many agricultural economic sectors [4]. For mobile robots, the capability of autonomously navigating in steep slope vineyards has a mandatory requirement: real-time localization. In order for a robot to navigate safely in the vineyard, it needs to be able to localize itself. Feature-based localization is one of the most common approaches to do so [5, 6, 7]. However, the extraction of reliable and persistent features in an outdoor environment is a challenging task. In the vineyard context, it makes sense to provide the robot with the ability to recognize vine trunks as high-level features to use in the localization and mapping process. The robot can be endowed with camera systems and artificial intelligence to learn what a trunk is. This image recognition capability can also be used for management and supervision purposes when combined with data analysis [8, 9]. To performe such tasks, Deep Learning (DL)-based object detection [10] can be used. DL [11, 12] allows a machine to learn to classify, detect, and segment objects using a given training dataset. Convolutional Neural Networks (CNN) are widely used to perform such a task. They showed the highest levels of performance in several contests in machine learning and pattern recognition [13]. Despite this, training a CNN from scratch, and obtaining accurate results while deploying it on a real scenario, assumes that both training and test data must be

		Application	Images on dataset	Performance
[31]		Detect apple flowers	588	AP of **97.20%** and F1 score of **92.10%**
[32]		Detect and classify crop species	31,147	AP of **92.79%**
[33]	MangoYOLO(s)	Detect mango fruits in orchards	1,300	AP of **98.60%** and F1 score of **96.70%**
	MangoYOLO(pt)			AP of **98.30%** and F1 score of **96.80%**
[34]		Detect apples in orchards	4,800	F1 score of **81.70%**
[35]		Detect fruit in orchards	2,268	F1 score of **90.40%** for apples, **90.80%** for mangoes and **77.50%** for almonds
[36]		Detect sweet pepper and rock melon	122	F1 score of **83.80%**
[37]		Detect strawberries	150	F1 score of **74.40%**
[38]		Detect pest	177	AP of **93.10%**
[39]		Detect flying insects	12,000	Counting accuracy of **93.71%**
[40]		Detect an obstacle	437	Precision of **99.9%** and recall of **36.7%** in row crops, and precision of **90.8%** and a recall of **28.1%** in mowing grass.

Table 1: Summary of the current state-of-the-art on DL-based object detection in agriculture.

in the same feature space, and have the same distribution [14]. However, in some real-world scenarios, data collection can be challenging and time-expensive. To overcome this limitation, learners can be trained with data easily collected from different domains [15, 16, 17]. In other words, the learning procedure can be performed transferring knowledge from a given task that was already learned, and the training procedure can focus on a subset of layers of the CNN. This methodology is called Transfer Learning (TL) [18].

In this work, the feature extraction problem in the vineyard context is addressed using TL to detect vine trunks. To do so, CNNs models based in the Single Shot Multibox (SSD) [19], Pooling Pyramid Network (PPN) [20], and SSDLite [21] architectures are considered, such as several versions of the MobileNets [22], with slighly variations on the hyper-parameters. Also, a version of Inception [23] built on top of SSD was considered. Finally, a lite version of YOLO - Tiny YOLO-V3 [24, 25] - was trained and tested. Two low-cost devices are used to perform real-time inference: Google's USB Accelerator [26] and NVIDIA's Jetson Nano [27]. An experimental benchmark between the two devices is performed, using a built in-house dataset that is publicly available (`http://vcriis01.inesctec.pt` - `DS_AG_39`). The devices are compared considering the Average Precision (AP) of trunk detection resultant from the deployment of the models in each one, and the respective inference time.

The rest of the paper is described as follows. In the state-of-the-art, the related work is reviewed. Section 3 provides some basic information about the DL-based concepts and tools used in this work. Section 4 describes the two devices used in this work to perform trunk detection. Section 5 contains the methodology adopted, such as the data collection method, the training procedure, and the inference approaches. Section 6 presents the proposed system results using the built in-house dataset, and the respective analysis and discussion. Finally, the work is summarized in Section 7.

2. Related Work

Image classification and object detection based on DL techniques are widely present in the agriculture sector. Intensive and time expensive tasks are being replaced by automatic machines, endowed with artificial intelligence. These machines are performing operations in the agriculture context such as plant disease detection, weed identification, seed identification, fruit detection and counting, obstacle detection, and others [28, 29, 30]. Table 1 provides a summary of DL-based object detection works in the agricultural field.

From Tab. 1, several conclusions can be extracted. The majority of works focus on fruit detection, mainly in orchards. Relatively to these works, the most common application is fruit counting. Additionally, a minority of the state-of-the-art focuses on insect detection for pest

identification, and, also, obstacle detection. Overall, the majority of works present high performance. Even with significantly different dataset sizes, in general, the works achieve AP or F1 scores higher than 80%.

Dias et al. [31] fine-tune a pre-trained CNN to detect apple flowers. This work uses data augmentation to quadruple the original dataset size. To evaluate the proposed DL-based detector, both Precision vs Recall (PR) and the F1 score were used. This way, both the optimal performance of the method through F1 score, and the expected performance across a range of decision thresholds through Average Precision (AP) are evaluated. This work achieves a F1 score of 92.1% and a AP of 97.2%. CropDeep [32] proposes the largest dataset from all the analysed works, with 31,147 images of crop species with a total of 31 different classes. The authors train and test several state-of-the-art models for object detection. For example, for ResNet [41] they achieved a medium AP of 92.79%. In [33] several state-of-the-art DL architectures are trained to detect mango fruits in orchards. With the same purpose, the authors propose a new architecture based on YOLO, called MangoYOLO. From this architecture, two models were created, differing on the weights initialization. The first, MangoYOLO(s), was trained from scratch, while the second, MangoYOLO(pt), was pre-trained on the COCO dataset [42]. The training set is composed of 11,820 fruits. MangoYOLO(s) achieves an F1 score of 96.7% and an AP of 98.6%, while MangoYOLO(pt) results in an F1 score of 96.8% and an AP of 98.3%. To detect apples during different growth stages in orchards, Tian et al. [34] proposed an improvement to the state-of-the-art model YOLO-V3 [24]. The dataset used is composed of high-resolution images of apples in three different growth stages. The method achieved an F1 score of 81.7% for the training set containing three types of growth stages. Similarly, [35] proposes the use of state-of-the-art architecture Faster R-CNN [43] for fruit detection in orchards, including mangoes, almonds, and apples. In the first stage, the architecture was used without any modification, using TL. In the second stage, in order to improve fruit detection in images with a larger number of fruits, a modification was proposed to the raw architecture. Using all the training data, the authors obtained an F1 score of 90.4% for apples, 90.8% for mangoes and 77.5% for almonds. DeepFruits [36] proposes a DL-based fruit detector fine-tuning the state architecture Faster-RCNN. The VGG-16 [44] model, previously trained using ImageNet [45], is adapted through the use of a dataset with RGB and Near-Infrared (NIR) images and three classes named background, sweet pepper, and rock melon. As the best result, the authors obtained an F1 score of 83.8%. L*a*b*Fruits [37] combines a visual processing approach with one-stage DL networks to detect strawberries. The entire dataset is constituted by 890 ripe strawberries and 3329 unripe strawberries. While testing, the authors consider that a detection is correct when the Interception over Union (IoU) is at least 0.5. Results show that this work obtained an F1 score of 74.4% for RGB images considering both classes.

To detect and count pest in images, [38] is proposed. To test the proposed pipeline, the authors use the Interception-Over-Minimum (IOMin) concept with a threshold level of 0.5. Two different evaluation metrics were used: miss rate vs false positives per image (FPPI) and AP, in order to focus on the trade-off between reducing miss detections and reducing false positives. The best result achieved from this work is 93.1% for AP and 9.9% for miss rate vs FPPI. Also in this context, Zhong et al. [39] detect, count and classify 6 types of flying insects for pest control in the agricultural context. The detection pipeline uses a single class and is built using state-of-the-art YOLO architecture pre-trained on ImageNet. The evaluation is performed using the counting accuracy, defined as the ratio of a correctly detected number to the total number of detected flying insects. The final counting accuracy obtained by YOLO was 93.71%.

For obstacle detection, [40] proposes the fine-tune of a state-of-the-art CNN for detection

of a specific object. The detector was evaluated in row crops context, obtaining a precision of 99.9% and recall of 36.7%, and in mowing grass, obtaining a precision of 90.8% and a recall of 28.1%.

3. Deep Learning Background

This work uses two sets of models based on the SSD architecture [19] to detect vine trunks, the MobileNets [22], and Inception-V2 [23]. Tiny YOLO-V3 [24, 25] is also used to perform this task. This section presents an overview of the SSD-based architectures and the all the type of models used.

3.1. Single Shot Multibox

SSD, Fig. 2, is based on a feed-forward CNN that detects objects producing a fixed number of bounding boxes and scores. This architecture is build upon a NN based on a given standard

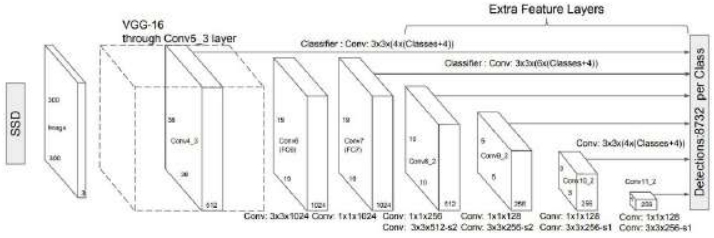


Figure 2: SSD architecture [19].

architecture. Its main modules are:

- Convolutional feature layers that decrease progressively in size, detecting objects at multiple scales.
- Convolutional filters represented on top of Fig. 2, that produce a fixed number of detection predictions.
- A set of bounding boxes associated with each feature map cell.

These characteristics allow to detect objects at multiple scales, i.e., objects of different sizes in images with different resolutions.

3.2. Single Shot Multibox Lite

SSDLite was proposed by Sandler *et al.* together with MobileNet-V2 [21], being based on the design of MobileNets. This architecture, as the name suggests, is a lighter version of the original SSD. The conventional convolutions present in the SSD architecture are replaced by separable convolutions. These ones split full convolutional operations into two separable layers.

The first, called depthwise convolution, applies a single convolution to each image channel. The second, a 1×1 convolution called pointwise convolution, builds new features computing linear combinations of the input image channels. This design revealed to be much more computationally efficient, highly reducing the CNN size. These characteristics are directly oriented to mobile and embedded devices.

3.3. Pooling Pyramid Network

PPN [20], similarly to SSDLite, is a reduced size version of SSD. The architecture was designed in order to run faster than SSD, maintaining similar detection performance. This architecture uses a backbone model as a base, as well as SSD. Also, to stabilize prediction scores, PPN uses a shared box predictor across different feature maps of different scales. Thus, this predictor takes into account all the training data, instead of independently using a portion of the training data for each predictor, as SSD does. Also, PPN uses max pooling operations to shrink the feature map down to 1×1. This operation does not have any addition and multiplication operations, and so is very fast. On the other hand, SSD uses the convolution operation to extract layers from the base network, and build the feature maps.

3.4. MobileNets

Google's USB Accelerator is fully compatible with 8-bit quantized MobileNets. Thus, in this work, several versions and variations of this DL-based model category are used. This set of models provide lightweight deep NN using depthwise separable convolutions. In other words, the model factorizes convolutions into depthwise, and 1×1 convolutions called pointwise convolutions. The first applies a single filter to the input channel, and the second applies a 1×1 convolution, combining the outputs of the first. The input of the CNN is a tensor with shape $D_f \times D_f \times M$, where D_f represents the input channel spatial width and height, and M is the input depth. After the convolution, a feature map of shape $D_f \times D_f \times N$ is obtained, where N is the output depth. In this context, these families of models use two hyper-parameters that allow the user to resize the model so that it meets the system requirements. These hyper-parameters are: *width multiplier* α, and *resolution multiplier* ρ. The first is used to reduce the size of the CNN uniformly at each layer. For a given value of $\alpha \in (0, 1]$, the number of inputs channels M becomes αM, as well as the number of output channels N becomes αN. *Width multiplier* reduces the computational cost and the number of parameters by α^2. The second hyper-parameter, ρ, is also used to reduce the computational cost. This one is applied directly to the input image, setting its resolution. The values of $\rho \in (0, 1]$ are chosen in order to obtain typical input image resolutions. Similarly to the *width multiplier*, the *resolution multiplier* also reduces the computational cost and the number of parameters by ρ^2. So, both parameters are different ways of reducing the model size and computational cost. When combined, the effects on the final model can be even more significant.

3.5. Inception

Ioffe *et al.* [46] proposed the primary version of Inception. This model design is based on the premise that the desired object to classify or detect can present several sizes on different images. This leads to the difficulty of choosing the right kernel size. To overcome this issue, Inception proposes three different convolutional filter sizes - 1×1, 3×3, and 5×5. Additionally, the NN model also computes max pooling. The output of all these operations is then concatenated, constituting the result of the respective Inception module.

Inception-V2 was developed in order to reduce the computational complexity of the original version. This is done by factorizing the convolution operations. For example, a 5×5 convolution is factorized into two 3×3 convolutions, improving runtime performance. In the same way, a $m \times m$ convolution can be factorized into a combination of $1 \times m$ and $m \times 1$ convolutions.

3.6. You Only Look Once

YOLO [47] is based on the Fully Convolution Neural Network (FCNN) architecture idea, that processes the input image once, returning its output prediction. This model divides the input image into a grid with $S \times S$ cells. Each cell that contains the center of an object is responsible for detecting it, producing a maximum of two bounding boxes and considering only one class. Each bounding box present in a given cell is associated with a score that reflects how confident the model is that the object lies inside the box, and how accurate it is predicting the object position. The model itself is composed of 24 convolutional layers followed by two fully connected layers. To improve the model, several versions of the original one were created. In particular, Tiny YOLO-V3 is a lite version of an enhanced version of YOLO, that allows increasing the inference frame rate, with a lower computational cost. In other words, this model provides a proper balance between accuracy and inference speed, overcoming the slower inference performance of YOLO-V3.

4. Experimental Devices

In this work, two types of devices are used to perform real-time trunk detection. The first, Google's USB Accelerator, uses an Edge TPU to compute inference. The second, NVIDIA's Jetson Nano, uses its own GPU to do so.

4.1. Google USB Accelerator

Google's Coral USB Accelerator (Fig. 3), provides an Edge TPU machine learning accelerator coprocessor. It is connected via USB to a host computer, allowing high-speed inference. This device is compatible with Tensorflow Lite, a lightweight version of TensorFlow designed for mobile and embedded devices, and can perform image classification, object detection, and semantic segmentation. To perform such tasks, the Edge TPU uses 8-bit quantized models. So,

Figure 3: Google Coral USB Accelerator [26].

when training a 32-bit float model, it has to be quantized using either *quantization aware training*

or *post training quantization* [48]. The first approach simulates the effect of 8-bit values during the training process using quantization nodes in the NN graph. The second does not modify the NN structure and is applied after training. However, it is less accurate than the first method. Alternatively, pre-trained models that are already quantized can be used if compatible with the Edge TPU. The device supports a range of operations and is most likely compatible with models designed for mobile devices, using the SSD architecture. After training the model, the *Edge TPU compiler* is used to assign inference operations to the device and the host CPU, as represented in Fig. 4. The compiled model differs from the original Tensorflow Lite model in the first operation

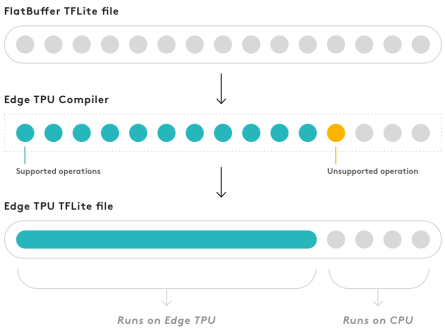


Figure 4: Edge TPU model compilation scheme [49].

of the graph. CPU processes the operations from the first non-supported operation of the TPU, until the end of the graph. The inference will be as faster as higher it is the number of operations assigned to the Edge TPU.

4.2. NVIDIA Jetson Nano

Jetson Nano Developer Kit (Fig. 5), provides a 128-core NVIDIA Maxwell @ 921MHz GPU, capable of computing image classification, object detection, semantic segmentation, and speech processing. This device is compatible with the most popular machine learning frameworks, including TensorFlow. Additionally, Jetson Nano is compatible with an accelerator library called TensorRT. This Software Development Kit (SDK) allows high-performance DL inference and is compatible with TensorFlow and other common frameworks. The main focus of this tool is to optimize already trained NNs for a given purpose efficiently. To do so, TensorRT allows compressing the network after training. Also, this SDK optimizes the kernel selection and normalizes and converts the model to a desired precision (32 or 16 floating-point (FP32/FP16), and 8 integer (INT8)) in order to improve latency, efficiency, and throughput.

5. Methods

In order to create a reliable vine trunk detector, in a first stage, a dataset was created using our robotic platform AgRob V16 [50], represented in Fig. 6. This dataset contains images captured in two different vineyards, each one with a camera with different characteristics. After data

Figure 5: Jetson Nano Development Kit [27].

Figure 6: AgRob V16 robotic platform.

collection, the vine trunks were manually annotated using a graphical tool and converted to the Pascal VOC format [51]. The dataset containing the training images and the respective annotations is publicly available at our repository (`http://vcriis01.inesctec.pt/` - `DS_AG_39`).

5.1. Data Collection

As referenced before, two onboard cameras belonging to our robotic platform collected data in two different vineyards. Representative images of both vines can be observed at Fig. 7. One of the cameras is from Raspberry Pi and possesses a 640×480 resolution, and a conventional infrared filter (Fig. 7 at the right). The other is a Mako G-125C camera, with a resolution of 1292×964 resolution, and an infrablue filter (Fig. 7 at the left). The dataset is constituted by 336 different images and approximately 1,600 annotated trunks. It contains:

- Images with two different resolutions.

Figure 7: Images of the training data correspondent to the two different vineyards, and respective annotations.

- Two types of vine trunks, one with foliage, and other without.
- Trunks covered by shadows.

These conditions confer variety to the training procedure, and consequently, robustness to the inference final result.

5.2. Data Annotation

Given the training dataset, the perceptible vine trunks were manually annotated on the images. Figure 7 shows an example of an image of each vine with the respective annotations. The output from this procedure is a set of bounding boxes with different sizes, for each image. These are represented in a `.xml` file with the Pascal VOC annotation syntax, containing the label class considered, and the four corners location of each bounding box. The annotations are also publicly available (`http://vcriis01.inesctec.pt/`) together with the training images. This data is the input for the training procedure, described below.

5.3. Training Procedure

In the training procedure, two different frameworks were used. One was Tensorflow, where all the MobileNets and the Inception models were trained. On the other side, YOLO has its own training framework, also denominated YOLO. All the models used were pre-trained using the COCO dataset [42].

Figure 8 demonstrates the step-by-step Tensorflow training procedure. This training flow converts raw bounding boxes annotations into the final models suitable for both the USB Accelerator and Jetson Nano. As referenced before, TL was used in order to retrain all the NN models used. To do so, firstly, the bounding boxes data was serialized into the *TFRecord* data type. This process stores data as a sequence of binary streams, allowing Tensorflow to interpret data in a more efficient manner. After retraining the models, in order to save them for posterior TL or retraining, they are exported into *frozen graphs*. These graphs can be exported for both devices used. For Google's USB Accelerator, the graph is converted to Tensorflow Lite and then compiled, as described in Sec. 3. If the pre-trained model is not 8-bit quantized, *post training quantization* is used to quantize it. For NVIDIA's Jetson Nano, the *frozen graph* was optimized and converted to TensorRT using 16-bit floating point precision. Table 2 shows all the

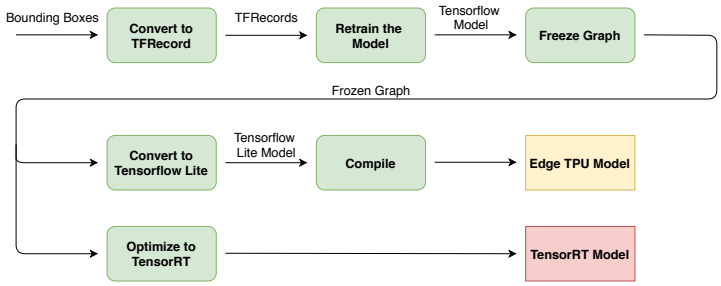


Figure 8: Tensorflow-based training procedure flow.

Model	Version	Architecture	α	Resolution
MobileNet	1	SSD	1	300×300
MobileNet	1	SSD	0.75	300×300
MobileNet	1	PPN	1	640×640
MobileNet	2	SSD	1	300×300
MobileNet	2	SSDLite	1	300×300
Inception	2	SSD	1	300×300

Table 2: Tensorflow-based trained models.

models considered, with their respective variations. It is worth noting that two equal versions of MobileNet-V1 were retrained using different α values, in order to analyse the impact of this hyper-parameter.

For Tiny YOLO-V3, the pre-trained weights were also used as well as YOLO's own training framework. This training procedure is quite more straightforward, requiring only the pre-trained model, raw images, and bounding boxes as input, and no post-processing over the trained model.

6. Results

To evaluate the considered models in both USB Accelerator and Jetson Nano, a test subset containing 45 images and approximately 180 vine trunks was extracted from the training dataset. To evaluate and compare all the detectors in both devices, a state-of-the-art metric was used. Also, the runtime performance of the models on top of each device was measured.

6.1. Evaluation Metric

The PASCAL Visual Object Classes (VOC) Challenge [51] was used to evaluate the performance of the considered models, both on Google's USB Accelerator, and NVIDIA's Jetson Nano. This method is widely used to evaluated DL-based models performing object detection, being a fair approach to compare the performance of different models resulting from different works. Pascal VOC calculates the Average Precision (AP) as follows. Given an annotated *ground truth*

bounding box B_g, and a detected bounding box B_d, the Intersection over Union (IoU) is computed using the following equation

$$IoU = \frac{m(B_g \cap B_d)}{m(B_g \cup B_d)} \tag{1}$$

where $m(x)$ denotes the area of x. Figure 9, shows a graphical representation of this concept. So,

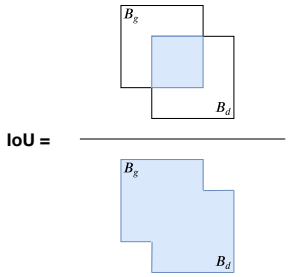


Figure 9: Interception over union representation.

IoU represents the quotient between the area of overlap and the area of union between the *ground truth*, and the detection bounding boxes. Using this definition, for a given threshold value t, three main concepts can be defined:

- **True Positive** (TP): IoU $\geq t$, i.e., a correct detection.
- **False Positive** (FP): IoU $\leq t$, i.e., an incorrect detection.
- **False Negative** (FN): a *ground truth* is not detected.

It is worth noting that if more than one detection for a single *ground truth* is computed, only the one with the highest IoU is considered as TP, and all the others are FPs. This being said, with these three qualifiers it is possible to define two fundamental concepts. The first, *precision* p, is defined as the total number of **TPs** over all the detections. The second, *recall* r, is the total number of **TPs** over all the *ground truths*. Using these, it is possible to plot a curve of the *recall* in function of the *precision*, $p(r)$. The evaluation considers that a suitable detector is the one that maintains the precision high for an increase in recall. With this consideration, the detector is evaluated computing the Average Precision (AP), interpolating the obtained curve, and calculating the area below the curve. Mathematically, this is expressed as follows

$$\sum_{r=0}^{1} (r_{n+1} - r_n) p_{interp}(r_{n+1}) \tag{2}$$

with

$$p_{interp}(r_{n+1}) = \max_{\widetilde{r}:\widetilde{r} \geq r_{n+1}} p(\widetilde{r}) \tag{3}$$

where $p(\widetilde{r})$ is the measured precision at recall $\widetilde{r}$.

For the sake of completeness, this work also evaluates the models using the F1 score. This score is the harmonic mean between the precision p and recall r, and can be calculated as follows

$$F1 = 2\,\frac{p \cdot r}{p + r} \tag{4}$$

6.2. Detection Performance

After training all the considered models, they were applied to the test dataset. Figure 10 demonstrates an example of SSD MobileNet-V2 on both vineyards. For the Localization and

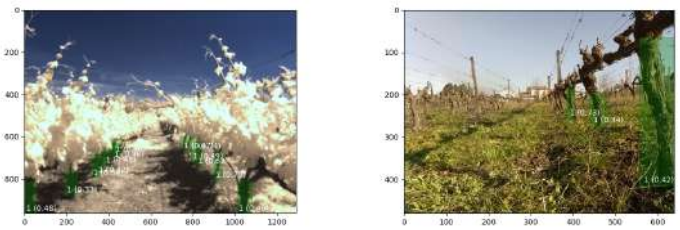


Figure 10: An example of SSD MobileNet-V2 inferece on both vineyards.

Mapping purposes, the main interest is to find the presence of vine trunks in the image so that, then, using for example depth sensors, the 3D location of each can be extracted. So, in this work, we are not only interested in highly precise detections, but also in medium precise ones. Thus, using the metric previously described, a IoU threshold t equals to 0.5 was considered. In this way, the recall × precision curves were computed, and are represented on Fig. 11. Interpolating

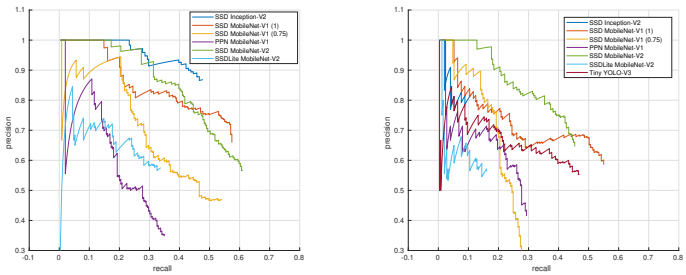


Figure 11: Recall × precision curves of the models considered on (left) Google's USB Accelerator and (right) NVIDIA's Jetson Nano.

the curves, and calculating the area bellow them, the AP was computed for each model in each

device. Additionally, using Eq. 4 the F1 score can also be computed for each model in each device. Table 3 summarizes this information, presenting also the runtime performance of each configuration. To evaluate the runtime performance of each model in each one of the devices, all

Model	α	Resolution	Google USB Accelerator			NVIDIA Jetson Nano		
			AP (%)	F1 (%)	Runtime (ms)	AP (%)	F1 (%)	Runtime (ms)
SSD MobileNet-V1	1	300 × 300	49.74	61.00	21.18	41.39	57.00	56.83
SSD MobileNet-V1	0.75	300 × 300	39.78	50.00	20.21	21.95	23.00	47.40
PPN MobileNet-V1	1	640 × 640	23.17	34.00	21.83	20.38	34.00	53.23
SSD MobileNet-V2	1	300 × 300	52.98	59.00	23.14	40.08	53.00	62.84
SSDLite MobileNet-V2	1	300 × 300	23.64	42.00	23.83	10.78	25.00	65.26
SSD Inception-V2	1	300 × 300	46.10	61.00	359.64	9.45	19.00	73.71
Tiny YOLO-V3	-	416 × 416	-	-	-	32.56	51.00	100.2

Table 3: AP (%), F1 score and average inference time per imgage (ms) obtained from the models considered using Google's USB Accelerator and NVIDIA's Jetson Nano.

the inference configurations were profiled using the *high resolution clock* from `chrono` present in the *std* library. Table 3 contains the results of the profiling.

6.3. Discussion

By analysis of Fig. 11 and Tab. 3, it is visible that Tiny YOLO-V3 is only supported on NVIDIA's Jetson Nano. In fact, besides Tiny YOLO-V3, only SSD-based models are used (PPN and SSDLite are derivations of SSD) in this work. This is due to the incompatibility of different architectures such as Faster R-CNN [43], R-FCN [52], Mask R-CNN [53], and others, with Google USB Accelerator for object detection. This is a disadvantage in comparison with Jetson Nano, that supports a wider variety of architectures, and, consequently, models. Additionally, to perform inference on NVIDIA's Jetson Nano, *quantization* is not required. Inference is supported using both *floating-point* and *integer* precision on this device. On the contrary, the USB Accelerator only supports 8-bit quantized models. Despite this, Tab. 3 shows that Google's USB Accelerator has higher AP than Jetson Nano in all the SSD-based models. Starting from the same trained model, both devices optimize and convert it to better perform on each computational environment. The USB Accelerator converts the model to Tensorflow Lite and then compiles it, while Jetson Nano uses TensorRT to do so. Results show that the optimizations performed in the context of the USB Accelerator lead to a higher inference AP anf F1 score. Table 3 shows that, for example, SSD MobileNet-V1 presents an overall AP of 49.74% and F1 score of 0.61 on the USB Accelerator and AP of 41.39% and F1 score of 57% on Jetson Nano. Similarly, SSD MobileNet-V2 has higher AP on the first device than on the second, being 52.98% on the USB Accelerator, the higher AP obtained considering all the models in both devices. For SSD Inception-V2, the difference is quite significant, leading to the conclusion that TensorRT failed to optimize the model for Jetson Nano. On the Google USB Accelerator, this model achieves an F1 score of 61%, quite higher than the 19% score obtained on the Jetson Nano.

In terms of inference time, Tab. 3 also shows that generally, Google's USB Accelerator performs trunk detection at a higher frame rate than NVIDIA's Jetson Nano. For the MobileNets, the Google device has an average inference time per image in the range of 20.21-23.83 milliseconds, while the NVIDIA device has a runtime performance in the range of 47.40-65.26 milliseconds. This means that, for the MobileNets, the USB Accelerator is more than two times faster than Jetson Nano. Only for SSD Inception-V2, Jetson Nano is faster. This is due to two main reasons:

the Inception-V2 model is much larger than the MobileNets, and the USB Accelerator is compatible with a lower number of operations of this model. Thus, when compiling it for the Edge TPU, these non-supported operations are assigned to the host CPU, which leads to an average inference time per image of 359.64 milliseconds. Another important runtime consideration is the loading model time. Our experience showed that the USB Accelerator loads all the models much faster than Jetson Nano.

In a final note about the MobileNets, results show that the decrease of the hyper-parameter *width multiplier* leads to lower inference AP and F1 score and higher frame rate, as expected. Also, the PPN architecture confirmed to be a lighter version of the SSD, allowing the use of a higher resolution with no costs on the runtime performance. However, even using a much higher input image resolution, the model that used this architecture has shown an AP much lower than models that use the SSD architecture with lower resolutions. SSDLite MobileNet-V2, a lighter version of SSD MobileNet-V2 resulted, in this specific case, in a much least precise model, not showing any advantage in terms of runtime performance.

To summarize the comparison of both devices, Tab. 4 shows the pros and cons of each device on the topics addressed in this work. The "+" denotes for a positive behavior, and "-" for a negative/worse behavior or an unsupported feature.

	Google USB Accelerator	**NVIDIA Jetson Nano**
Compatibilitty	-	+
Floating-point support	-	+
Overall AP	+	-
Runtime performance	+	-
Time spent loading the models	+	-

Table 4: Summary of the pros and cons of each device.

6.4. Detector Application

This work presents the first DL-based approach to detect vine trunks, that can be used in Localization and Mapping. Comparing with many state-of-the-art detectors for other applications, the ones proposed in this work presents lower AP and F1 score. However, in contrast with many works that use powerful GPUs, *low cost* and *low power* devices are used in this work. The models are 8-bit quantized on Google's Edge TPU and present 16-bit precision on Jetson Nano. This limits the inference performance in terms of detection AP. Even so, the detectors can be used in the Localization and Mapping procedures. Firstly, they perform inference at high frame rate. The navigation stack present on Agrob V16 (Fig. 6) imposes a minimum frame rate of ten *frames per second*. This condition is ensured, as demonstrated before. The stack uses a stereo camera system and builds a disparity map. In the context of the proposed work, recent experiments using Agrob V16 and deploying MobileNet-V1 on the vineyard in real-time shown that the proposed detectors can be used to extract vine trunks depth information using the disparity map. Figure 12 shows an example of these experiments. The detections on the stereo images are projected into the disparity map. Calculating the median of all depths inside each bounding box results in the depth of each trunk, if the trunk occupies the majority of the region inside of the box. This result can be used to construct a map of the vineyard and, consequently, to localize the robot inside this map.

Figure 12: Vine trunk depth estimation using the proposed detectors.

7. Conclusion

This work addresses the problem of feature extraction on steep slope vineyards for robotics localization and mapping. In this context, a reliable solution is required to detect vine trunks location on images. This work proposes the use of two different devices to do so: Google's USB Accelerator and NVIDIA's Jetson Nano. Seven different models were trained and deployed in these two devices. The performance of each one was analyzed both in terms of AP and runtime performance. Results showed that NVIDIA's Jetson Nano supports a wider variety of architectures and models than the USB Accelerator, that is limited to SSD-based architectures. On the other hand, for SSD-based models, Google's USB Accelerator provides higher AP and better runtime performance. In the best case, for SSD MobileNet-V2 on the USB Accelerator, an average precision of 52.98% and approximately 49 *frames per second* were obtained. Recent experiments demonstrate that, even so using these *low cost* devices, the proposed detectors can be used in Localization and Mapping.

In future work, the training dataset will be extended to use images from other vineyards, also considering thermal images. Also, a state-of-the-art DL-based model will be modified and adapted to perform better detecting vine trunks. Finally, DL-based semantic segmentation will be considered in order to segment the region occupied by the trunks.

Acknowledgments

This research was funded by the ERDF European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation — COMPETE 2020 under the PORTUGAL 2020 Partnership Agreement, and through the Portuguese National Innovation Agency (ANI) as a part of project "ROMOVI: POCI-01-0247-FEDER-017945". The opinions included in this paper shall be the sole responsibility of their authors. The European Commission and the Authorities of the Programme aren't responsible for the use of information contained therein.

References

[1] T. Andresen, F. B. de Aguiar, M. J. Curado, The alto douro wine region greenway, Landscape and Urban Planning 68 (2) (2004) 289 – 303, international Greenway Planning. doi:https://doi.org/10.1016/S0169-2046(03)00156-7.
URL http://www.sciencedirect.com/science/article/pii/S0169204603001567

[2] F. N. Dos Santos, H. Sobreira, D. Campos, R. Morais, A. P. Moreira, O. Contente, Towards a reliable robot for steep slope vineyards monitoring, Journal of Intelligent & Robotic Systems 83 (3-4) (2016) 429–444.

[3] J. J. Roldán, J. del Cerro, D. Garzón-Ramos, P. Garcia-Aunon, M. Garzón, J. de León, A. Barrientos, Robots in Agriculture: State of Art and Practical Experiences, 2018. doi:10.5772/intechopen.69874.
URL https://app.dimensions.ai/details/publication/pub.1100266943andhttps://www.intechopen.com/citation-pdf-url/56199

[4] T. Duckett, S. Pearson, S. Blackmore, B. Grieve, W.-H. Chen, G. Cielniak, J. Cleaversmith, J. Dai, S. Davis, C. Fox, P. From, I. Georgilas, R. Gill, I. Gould, M. Hanheide, A. Hunter, F. Iida, L. Mihalyova, S. Nefti-Meziani, G. Neumann, P. Paoletti, T. Pridmore, D. Ross, M. Smith, M. Stoelen, M. Swainson, S. Wane, P. Wilson, I. Wright, G.-Z. Yang, Agricultural robotics: The future of robotic agriculture (2018). arXiv:1806.06762.

[5] F. B. N. d. Santos, H. M. P. Sobreira, D. F. B. Campos, R. M. P. M. d. Santos, A. P. G. M. Moreira, O. M. S. Contente, Towards a reliable monitoring robot for mountain vineyards, in: 2015 IEEE International Conference on Autonomous Robot Systems and Competitions, 2015, pp. 37–43. doi:10.1109/ICARSC.2015.21.

[6] F. A. Cheein, G. Steiner, G. P. Paina, R. Carelli, Optimized eif-slam algorithm for precision agriculture mapping based on stems detection, Computers and Electronics in Agriculture 78 (2) (2011) 195 – 207. doi:https://doi.org/10.1016/j.compag.2011.07.007.
URL http://www.sciencedirect.com/science/article/pii/S0168169911001542

[7] A. Aguiar, F. Santos, L. Santos, A. Sousa, Monocular visual odometry using fisheye lens cameras, in: P. Moura Oliveira, P. Novais, L. P. Reis (Eds.), Progress in Artificial Intelligence, Springer International Publishing, Cham, 2019, pp. 319–330.

[8] X.-B. Jin, N.-X. Yang, X.-Y. Wang, Y.-T. Bai, T.-L. Su, J.-L. Kong, Deep hybrid model based on emd with classification by frequency characteristics for long-term air quality prediction, Mathematics 8 (2) (2020). doi:10.3390/math8020214.
URL https://www.mdpi.com/2227-7390/8/2/214

[9] Y. Bai, X. Jin, X. Wang, T. Su, J. Kong, Y. Lu, Compound autoregressive network for prediction of multivariate time series, Complexity 2019 (2019).

[10] Z. Zhao, P. Zheng, S. Xu, X. Wu, Object detection with deep learning: A review, IEEE Transactions on Neural Networks and Learning Systems 30 (11) (2019) 3212–3232. doi:10.1109/TNNLS.2018.2876865.

[11] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444. doi:10.1038/nature14539.
URL https://doi.org/10.1038/nature14539

[12] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT press, 2016.

[13] J. Schmidhuber, Deep learning in neural networks: An overview, Neural Networks 61 (2015) 85 – 117. doi:https://doi.org/10.1016/j.neunet.2014.09.003.
URL http://www.sciencedirect.com/science/article/pii/S0893608014002135

[14] S. J. Pan, Q. Yang, A survey on transfer learning, IEEE Transactions on Knowledge and Data Engineering 22 (10) (2010) 1345–1359. doi:10.1109/TKDE.2009.191.

[15] K. Weiss, T. M. Khoshgoftaar, D. Wang, A survey of transfer learning, Journal of Big Data 3 (1) (2016) 9. doi:10.1186/s40537-016-0043-6.
URL https://doi.org/10.1186/s40537-016-0043-6

[16] J. Lu, V. Behbood, P. Hao, H. Zuo, S. Xue, G. Zhang, Transfer learning using computational intelligence: A survey, Knowl.-Based Syst. 80 (2015) 14–23.

[17] L. Shao, F. Zhu, X. Li, Transfer learning for visual categorization: A survey, IEEE Transactions on Neural Networks and Learning Systems 26 (5) (2015) 1019–1034. doi:10.1109/TNNLS.2014.2330900.

[18] L. Torrey, J. Shavlik, Transfer learning, Handbook of Research on Machine Learning Applications (01 2009). doi:10.4018/978-1-60566-766-9.ch011.

[19] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, A. C. Berg, SSD: single shot multibox detector, CoRR abs/1512.02325 (2015). arXiv:1512.02325.
URL http://arxiv.org/abs/1512.02325

[20] P. Jin, V. Rathod, X. Zhu, Pooling pyramid network for object detection (2018). arXiv:1807.03284.

[21] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 4510–4520.

[22] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications (2017). arXiv:1704.04861.
[23] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision (2015). arXiv:1512.00567.
[24] J. Redmon, A. Farhadi, Yolov3: An incremental improvement (2018). arXiv:1804.02767.
[25] Joseph Chet Redmon, Yolo: Real-time object detection, https://coral.ai/products/accelerator, (accessed Jan 4, 2020) (2020).
[26] Google, Usb accelerator, https://coral.ai/products/accelerator, (accessed Jan 15, 2020) (2020).
[27] NVIDIA Corporation, Jetson nano developer kit, https://developer.nvidia.com/embedded/jetson-nano-developer-kit, (accessed Jan 17, 2020) (2020).
[28] L. Santos, F. N. Santos, P. M. Oliveira, P. Shinde, Deep learning applications in agriculture: A short review, in: M. F. Silva, J. Luís Lima, L. P. Reis, A. Sanfeliu, D. Tardioli (Eds.), Robot 2019: Fourth Iberian Robotics Conference, Springer International Publishing, Cham, 2020, pp. 139–151.
[29] A. Kamilaris, F. X. Prenafeta-Boldú, Deep learning in agriculture: A survey, Computers and Electronics in Agriculture 147 (2018) 70 – 90. doi:https://doi.org/10.1016/j.compag.2018.02.016.
URL http://www.sciencedirect.com/science/article/pii/S0168169917308803
[30] A. Kamilaris, F. X. Prenafeta-Boldú, A review of the use of convolutional neural networks in agriculture, The Journal of Agricultural Science 156 (3) (2018) 312–322. doi:10.1017/S0021859618000436.
[31] P. A. Dias, A. Tabb, H. Medeiros, Apple flower detection using deep convolutional networks, Comput. Ind. 99 (2018) 17–28.
[32] Y.-Y. Zheng, J.-L. Kong, X.-B. Jin, X.-Y. Wang, T.-L. Su, M. Zuo, Cropdeep: The crop vision dataset for deep-learning-based classification and detection in precision agriculture, Sensors 19 (5) (2019). doi:10.3390/s19051058.
URL https://www.mdpi.com/1424-8220/19/5/1058
[33] A. Koirala, K. B. Walsh, Z.-X. Wang, C. McCarthy, Deep learning for real-time fruit detection and orchard fruit load estimation: benchmarking of 'mangoyolo', Precision Agriculture (2019) 1–29.
[34] Y. Tian, G. Yang, Z. Wang, H. Wang, E. Li, Z. Liang, Apple detection during different growth stages in orchards using the improved yolo-v3 model, Computers and Electronics in Agriculture 157 (2019) 417 – 426. doi:https://doi.org/10.1016/j.compag.2019.01.012.
URL http://www.sciencedirect.com/science/article/pii/S016816991831528X
[35] S. Bargoti, J. Underwood, Deep fruit detection in orchards, in: 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 3626–3633. doi:10.1109/ICRA.2017.7989417.
[36] I. Sa, Z. Ge, F. Dayoub, B. Upcroft, T. Perez, C. McCool, Deepfruits: A fruit detection system using deep neural networks, Sensors 16 (8) (2016). doi:10.3390/s16081222.
URL https://www.mdpi.com/1424-8220/16/8/1222
[37] R. Kirk, G. Cielniak, M. Mangan, L*a*b*fruits: A rapid and robust outdoor fruit detection system combining bio-inspired features with one-stage deep learning networks, Sensors 20 (1) (2020). doi:10.3390/s20010275.
URL https://www.mdpi.com/1424-8220/20/1/275
[38] W. Ding, G. W. Taylor, Automatic moth detection from trap images for pest management, Comput. Electron. Agric. 123 (2016) 17–28.
[39] Y. Zhong, J. Gao, Q. Lei, Y. Zhou, A vision-based counting and recognition system for flying insects in intelligent agriculture, in: Sensors, 2018.
[40] K. A. Steen, P. Christiansen, H. Karstoft, R. N. Jørgensen, Using deep learning to challenge safety standard for highly autonomous machines in agriculture, Journal of Imaging 2 (1) (2016). doi:10.3390/jimaging2010006.
URL https://www.mdpi.com/2313-433X/2/1/6
[41] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
[42] T.-Y. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. L. Zitnick, Microsoft coco: Common objects in context, in: ECCV, 2014.
[43] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks (2015). arXiv:1506.01497.
[44] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition (2014). arXiv:1409.1556.
[45] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, Li Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255.
[46] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.
[47] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[48] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, D. Kalenichenko, Quantization and training of neural networks for efficient integer-arithmetic-only inference, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[49] Google, Tensorflow models on the edge tpu, https://coral.ai/docs/edgetpu/models-intro/, (accessed Jan 12, 2020) (2019).

[50] L. Santos, F. Santos, J. Mendes, P. Costa, J. Lima, R. Reis, P. Shinde, Path planning aware of robot's center of mass for steep slope vineyards, Robotica 1–15doi:10.1017/S0263574719000961.

[51] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, A. Zisserman, The pascal visual object classes (voc) challenge, International journal of computer vision 88 (2) (2010) 303–338.

[52] J. Dai, Y. Li, K. He, J. Sun, R-fcn: Object detection via region-based fully convolutional networks (2016). arXiv:1605.06409.

[53] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask r-cnn (2017). arXiv:1703.06870.

4.3 Bringing semantics to the vineyard - An approach on Deep Learning-based vine trunk detection

The two articles previously presented in Chapter 4 are a basis that sustains the research work to explore the final proposed perception solution for vine trunks. This section shows how this work was leveraged to build this solution. The article presented in this section was published in the MDPI Agriculture Journal and is entitled *Bringing Semantics to the Vineyard: An Approach on Deep Learning-Based Vine Trunk Detection* (Aguiar et al., 2021d). In comparison with the proposed previous works, this article presents a novel publicly available dataset called VineSet recognized by the ROS Agriculture community[3] as "A Large Vine Trunk Image Collection and Annotation using the Pascal VOC format". VineSet was built through an intense data collection procedure and posterior annotation, being composed of more than 900 original images. This dataset was further extended using augmentation techniques to a total of 9481 images. The new data collection procedure and the consequent dataset extension were essential to the increase of the DL models' performance. Using the same family of models, the average precision increased more than 30% with VineSet in relation with the previous works. In addition, results showed that the trained models are general enough to be used in other environments for trunk detection such as forests and orchards. Finally, this work also extends the previous works by using both TL and models trained from scratch and comparing their performances.

[3] http://wiki.ros.org/agriculture

agriculture

MDPI

Article

Bringing Semantics to the Vineyard: An Approach on Deep Learning-Based Vine Trunk Detection

André Silva Aguiar [1,2,*], Nuno Namora Monteiro [3], Filipe Neves dos Santos [1], Eduardo J. Solteiro Pires [1,2], Daniel Silva [1,2], Armando Jorge Sousa [1,3] and José Boaventura-Cunha [1,2]

[1] INESC TEC—INESC Technology and Science, 4200-465 Porto, Portugal; fbsantos@inesctec.pt (F.N.d.S.); epires@utad.pt (E.J.S.P.); daniel.q.silva@inesctec.pt (D.S.); asousa@fe.up.pt (A.J.S.); mailto:jboavent@utad.pt (J.B.-C.)
[2] School of Science and Technology, University of Trás-os-Montes e Alto Douro, 5000-801 Vila Real, Portugal
[3] Faculty of Engineering, University of Porto, 4200-465 Porto, Portugal; up201607764@fe.up.pt
* Correspondence: andre.s.aguiar@inesctec.pt

Abstract: The development of robotic solutions in unstructured environments brings several challenges, mainly in developing safe and reliable navigation solutions. Agricultural environments are particularly unstructured and, therefore, challenging to the implementation of robotics. An example of this is the mountain vineyards, built-in steep slope hills, which are characterized by satellite signal blockage, terrain irregularities, harsh ground inclinations, and others. All of these factors impose the implementation of precise and reliable navigation algorithms, so that robots can operate safely. This work proposes the detection of semantic natural landmarks that are to be used in Simultaneous Localization and Mapping algorithms. Thus, Deep Learning models were trained and deployed to detect vine trunks. As significant contributions, we made available a novel vine trunk dataset, called VineSet, which was constituted by more than 9000 images and respective annotations for each trunk. VineSet was used to train state-of-the-art Single Shot Multibox Detector models. Additionally, we deployed these models in an Edge-AI fashion and achieve high frame rate execution. Finally, an assisted annotation tool was proposed to make the process of dataset building easier and improve models incrementally. The experiments show that our trained models can detect trunks with an Average Precision up to 84.16% and our assisted annotation tool facilitates the annotation process, even in other areas of agriculture, such as orchards and forests. Additional experiments were performed, where the impact of the amount of training data and the comparison between using Transfer Learning and training from scratch were evaluated. In these cases, some theoretical assumptions were verified.

Keywords: deep learning; trunk detection; agriculture; autonomous navigation

Citation: Aguiar, A.S.; Monteiro, N.N.; Santos, F.N.d.; Solteiro Pires, E.J.; Silva, D.; Sousa, A.J.; Boaventura-Cunha, J. Bringing Semantics to the Vineyard: An Approach on Deep Learning-Based Vine Trunk Detection. *Agriculture* **2021**, *11*, 131. https://doi.org/10.3390/agriculture11020131

Academic Editor: Yanbo Huang
Received: 18 January 2021
Accepted: 1 February 2021
Published: 5 February 2021

1. Introduction

The development of robotic solutions in unstructured environments brings several challenges, mainly in developing safe and reliable navigation solutions. Agricultural environments are particularly unstructured and, therefore, challenging to the implementation of robotics. The Douro vineyards (Figure 1) are a great example of this.

These are located in the Douro Demarched Region, the oldest controlled winemaking region in the world, a UNESCO heritage place [1], and they are built in steep slope hills. The hill's characteristics cause signal blockage that decreases the accuracy of signals thtat are emitted by the Global Navigation Satellite System (GNSS), which makes the use of, for example, the standard Global Positioning System (GPS), unreliable. Additionally, the terrain that is highly characterized by irregularities leads to the high inaccuracy of sensors, like wheel odometry and Inertial Measurement Units (IMU)s [2].

Figure 1. Typical steep slope vineyard in the Douro's region. The Douro Demarched region (41º 06′ 06″ N 7º47′56″ W) extends for many Portugal cities, such as Mesão Frio, Peso da Régua, Santa Marta de Penaguião, Vila Real, and others.

The vast extension of the vineyards and their challenging conditions lead to an increasing need for human labor substitution by automatic and autonomous machines. These machines can be used to perform operations, such as planting, harvesting, monitoring, supply of water, and nutrients [3]. Moreover, they can transform and have a significant impact on many agricultural economic sectors [4]. For mobile robots, the capability of autonomously navigating in steep slope vineyards has a mandatory requirement: real-time localization. For a robot to navigate safely in the vineyard, it needs to be able to localize itself. Feature-based localization is one of the most common approaches to do so [5–7]. However, the extraction of reliable and persistent features in an outdoor environment is a challenging task. The vineyard context makes sense to provide the robot with the ability to recognize vine trunks as high-level features to use in the localization and mapping processes. The robot can be endowed with camera systems and artificial intelligence to learn what a trunk is. Moreover, the application of robotics in these tasks can have impact in the agricultural economic sector [4]. However, real-time localization is an essential requiremen in implementing mobile robotics in agriculture. Usually, in steep slope vineyards, the localization approaches should work in the absence of satellite-based systems. Thus, the implementation of these algorithms is a challenging task, due to the characteristic unstructured scenes that compose these environments [8]. In this context, natural features can be used as landmarks in the localization procedure [5–7]. In the vineyard, vine trunks can be used to this effect, allowing for localizing the robot and simultaneously creating a semantic map of the environment. Thus, the robotic platform should be capable to perceive the scene and recognize these features. In other words, the robot has a semantic perception of the environment.In order to perform such tasks, Deep Learning (DL)-based object detection [9] can be used. DL [10,11] allows for a machine to learn to classify, detect, and segment objects using a given training dataset. Convolutional Neural Networks (CNN)s are widely used to perform such a task. They showed the highest performance levels in several contests in machine learning and pattern recognition [12]. Despite this, training a CNN from scratch, and obtaining accurate results while deploying it on a real scenario, assumes that both training and test data must be in the same feature space, and they have the same distribution [13]. However, in some real-world scenarios, data collection can be challenging and time-expensive. In order to overcome this limitation, learners can be trained with data easily collected from different domains [14–16]. In other words, the learning procedure can be performed, transferring knowledge from a given task that was already learned, and the training procedure can focus on a subset of layers of the CNN. This methodology is called Transfer Learning (TL) [17]. Image classification and object detection based on DL techniques are widely present in the agriculture sector, endowing machines with the

capability to perform operations in the agriculture context, such as plant disease detection, weed identification, seed identification, fruit detection and counting, obstacle detection, and others [18–20].

Given all of the above, this work proposes using DL algorithms to detect vine trunks in a fast and precise way and while considering Edge-AI concepts. The main goal is to compute reliable semantic landmarks to use in Simultaneous Localization and Mapping (SLAM) pipelines of agricultural robots. In the current state-of-the-art, DL's use to detect tree trunks is still an area quite under developed, as described in Table 1. Badeka et al. [21] propose a DL-based approach to detect vine trunks. The authors developed a dataset with 899 vineyard images and trained two different architectures: faster regions-convolutional neural network (Faster R-CNN) [22] and You Only Look Once version (YOLO) [23]. The results show that, in the best case, this work achieved an Average Precision (AP) of 72.3% and an execution time performance of 29.6 ms. The remaining state-of-the-art approaches use conventional image processing and range-based techniques in order to detect tree trunks in agricultural contexts.

Table 1. Summary of the current state-of-the-art regarding tree trunk detection in agricultural contexts.

Reference	Approach	Performance
Badeka et al. [21]	Deep Learning-based vine trunk detection. Uses Faster R-CNN and two YOLO versions.	Average Precision of 73.2% and execution time of 29.6 ms.
Lamprecht et al. [24]	Detection based on Airbone Laser Scanning. Uses a Crown Base Height estimation and 3D clustering to isolate laser points on tree trunks.	Detection rate of 75% and overall accuracy of 84%.
Shalal et al. [25]	Orchard tree detection using a camera and a laser sensor. Based on image segmentation and data fusion techniques.	Average rate of detection confidence of 82.2%.
Xue et al. [26]	Uses a camera and a laser sensor to detect and measure the trunk width. Algorithm based on data fusion and decision with Dempster-Shafer theory.	Trunk width measurement with error rates from 6% to 16.7%.
Juman et al. [27]	Ground removal by colour space combination and segmentation and trunk detection using the Viola-Jones detector.	Detection rate of 97.8%.
Bargoti et al. [28]	Implements a Hough transformation to extract trunk candidates, and uses pixelwise classification to update their likelihood of being a tree trunk.	87–96% accuracy during the preharvest season, and 99% accuracy during the flowering season.
Colmenero-Martinez et al. [29]	Uses an infrared sensor to detect tree trunks.	Detection rate of 91%.

For example, Lamprecht et al. [24] use Airbone Laser Scanning to detect tree trunks. The authors studied their approach in an area of 109 trees and achieved an overall accuracy of 84%. Aiming to build a map of the orchard that is to be used in the mobile robotics context, Shalal et al. [25] use a camera and range sensor to detect trunks. This work uses image segmentation and data fusion techniques. Xue et al. [26] use a camera and laser sensor to detect and measure the trunk width. The experiments were conducted on 120 trees and 40 images, resulting in an error rate of 6% to 16.7%. Juman et al. [27] combine a ground removal technique with the Viola–Jones algorithm to detect trunks. This work

is proposed in order to perform autonomous navigation in oil-palm plantations, and it achieves a detection rate of 97.8%. Bargoti et al. [28] propose the detection of tree trunks in structured apple orchards. The authors implement a Hough transform to extract trunk candidates, and use pixelwise classification to update their detection likelihood.

In other agricultural contexts, DL is highly present in the detection of natural agents. Fruit detection in orchards ishe most common application. Moreover, some works focus on obstacle and insect detection, as well as pest identification. The majority of works focus on fruit detection, mainly in orchards. Relative to these works, fruit counting is the most common application. Additionally, a minority of the state-of-the-art focuses on insect detection for pest identification and obstacle detection. Overall, most of the works present high performance with Average Precision (AP) or F1 scores higher than 80%. Table 2 provides a summary of these works.

Table 2. Summary of the current state-of-the-art on Deep Learning (DL)-based object detection in agriculture.

Reference	Application	Performance
Dias et al. [30]	Detect apple flowers.	AP of 97.20% and F1 score of 92.10%.
Zheng et al. [31]	Detect and classify crop species.	AP of 92.79%.
Koirala et al. [32]	Detect mango fruit.	AP of 98.60% and F1 score of 96.70%.
Tian et al. [33]	Detect apples in orchards.	F1 score of 81.70%.
Bargoti and Underwood [34]	Detect fruit in orchards.	F1 score of 90.40% for apples, 90.80% for mangoes and 77.50% for almonds.
Sa et al. [35]	Detect sweet pepper and rock melon	F1 score of 83.80%.
Kirk et al. [36]	Detect ripe soft fruits.	F1 score of 74.40%.
Li et al. [37]	Detect and count oil palm trees from high-resolution remote sensing images.	Maximum overall detection accuracy of 99% and counting error less than 4% for each considered region.
Ding and Taylor [38]	Detect pest.	AP of 93.10%.
Zhong et al. [39]	Detect flying insects.	Counting accuracy of 93.71%.
Steen et al. [40]	Detect an obstacle.	Precision of 99.9% and recall of 36.7% in row crops, and precision of 90.8% and a recall of 28.1% in mowing grass.

Dias et al. [30] implement a technique for apple flower identification, which is robust to changes in illumination and clutter. The authors use a pre-trained CNN and Transfer Learning concepts to create the detector. Data augmentation is applied to the original collected images to increase the dataset size. The results show that this work achieves an F1 score of 92.1% and an AP of 97.2%. In the context of mango fruit detection, Koirala et al. [32] compared the performance of six state-of-the-art DL architectures. Additionally, the authors proposed MangoYOLO, a new architecture based YOLO [23], which was specifically created for mango fruit detection. As a best result, MangoYOLO performed with an AP of 98.60%. Zheng et al. [31] propose a large dataset for species classification and detection, called CropDeep. The dataset contains more than 30,000 images of 31 different classes. The au-

thors train state-of-the-art DL models to verify its validity, such as Resnet [41], where they obtained an AP of 92.79%.

Besides object detection, DL in agriculture can also be used to infer specific characteristics of the natural agents. For example, Li et al. [37] propose a DL framework to detect and count oil palm trees from high-resolution remote sensing images. The main goals of this work are to predict yield of palm oil and monitor the growth stage of palm trees. Tian et al. [33] implemented an improvement to the YOLO-V3 [42] model to estimate apples yield and their grown stages. The authors consider a variety of challenging conditions, such as overlapping apples, leaves, and branches; illumination variation; and, complex backgrounds. The experiments performed proved that, for a training dataset with three different growth stages, this approach has an F1 score of 81.7%. Bargoti and Underwood [34] use the standard Faster R-CNN architecture [22] to detect several types of fruits in orchards, such as apples, mangoes, and almonds. In this work, the authors explore the amount of data that are required to capture the variability of the agriculture environment, as well as the gain of using data augmentation techniques. Overall, this work was performed with high precision, resulting in an F1 score higher than 90%. Additionally, Sa et al. [35] propose a fruit detection system called DeepFruits while using the Faster R-CNN architecture. The proposed detectors are integrated in the software pipeline of an agricultural robot to estimate yield and automate the harvesting process. The results demonstrated that this work achieves an F1 score of 83.8% while detecting sweet pepper and rock-melon. To detect ripe soft fruits, Kirk et al. [36] propose a detector implemented as a combination of a conventional computer vision algorithm and a DL-based approach. The authors build a dataset with images captured over two months in the agricultural environment to test their implementation. The performed experiments show that this algorithm achieves an F1 score of 74.4%.

In addition to fruit detection, DL is also used in other relevant agriculture scenarios. The safety of machines and operators is essential in these environments. In this context, obstacle detection plays a major role ensuring the safety of the operations performed in agriculture. To pursue this goal, Steen et al. [40] use a CNN to detect an object type in row crops and grass mowing. The detector is able to detect the object with high precision, without detecting false positives, such as persons or other objects. Finally, insect and pest identification is also an important research area for the agriculture sector to avoid plant diseases. Zhong et al. [39] implemented a fast and accurate flying insect detection and counting. To do so, the YOLO [23] model is used in the detection stage, and an Support Vector Machine (SVM) in the counting stage. The detection pipeling supports six types of insects, and it performs with a counting accuracy of 93.71%. Ding and Taylor [38] create a CNN model to detect and count pest. The experiments show that the model is fast and precise (AP of 93.1%), and that it can be easily used to detect other kinds of pest.

Our previous works [43,44] focused on the usage and benchmark of low-power devices to deploy DL models while using a low quantity of training data. In this paper, the semantic vineyard perception problem is extended with the following main contributions and innovations:

- A novel DL-oriented dataset for vine trunk detection called VineSet, publicly available (http://vcriis01.inesctec.pt/datasets/DataSet/VineSet.zip) and recognized by the ROS Agriculture community (http://wiki.ros.org/agriculture) as "A Large Vine Trunk Image Collection and Annotation using the Pascal VOC format".
- A way of extending the dataset size using data augmentation techniques.
- The train, benchmark, and characterization of state-of-the-art Single Shot Multibox Detector (SSD) [45] models for vine trunk detection using the VineSet.
- Real-time deployment of the models using a Tensor Processing Unit (TPU).
- An automatic annotation tool for datasets of trunks in agricultural contexts.

The rest of the paper is described, as follows. Section 3 contains the methodology adopted, such as the data collection and augmentation methods, the training procedure, and the inference approaches. Section 4 presents the proposed system results while using

the VineSet dataset and the respective analysis, characterization, and discussion. Finally, Section 5 summarizes the work.

2. Background

This work uses two sets of models based on the SSD architecture [45] to detect vine trunks, the MobileNets [46], and Inception-V2 [47]. The SSD architecture and the derived models are briefly described in this section.

2.1. Single Shot Multibox

SSD, Figure 2, is based on a feed-forward CNN that detects objects producing a fixed number of bounding boxes and scores.

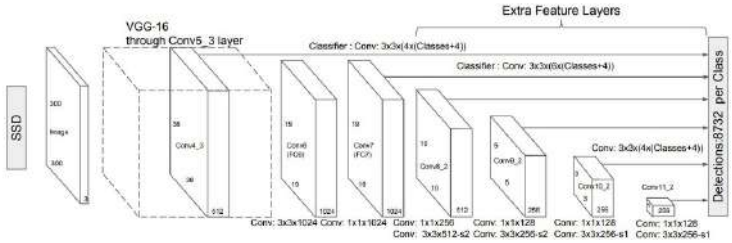


Figure 2. Single Shot Multibox architecture [45].

This architecture is built upon a Neural Network (NN) that is based on a given standard architecture. Its main modules are:

- Convolutional feature layers that decrease progressively in size, detecting objects at multiple scales.
- Convolutional filters that are represented on the top of Figure 2 produce a fixed number of detection predictions.
- A set of bounding boxes associated with each feature map cell.

These characteristics allow to detect objects at multiple scales, i.e., objects of different sizes in the images with different resolutions.

2.2. MobileNets

This set of models provide lightweight Deep Neural Networks (NNs) while using depthwise separable convolutions. In other words, the model factorizes convolutions into depthwise and 1×1 convolutions, called pointwise convolutions. The first applies a single filter to the input channel, and the second applies a 1×1 convolution, combining the outputs of the first. The CNN input is a tensor with shape $D_f \times D_f \times M$, where D_f represents the input channel spatial width and height, and M is the input depth. After the convolution, a feature map of shape $D_f \times D_f \times N$ is obtained, where N is the output depth. In this context, these model families use two hyper-parameters that allow the user to resize the model in order to meet the system requirements. These hyper-parameters are: width multiplier α and resolution multiplier ρ. The first is used to reduce the size of the CNN uniformly at each layer. For a given value of $\alpha \in (0, 1]$, the number of input channels M becomes αM, as well as the number of output channels N becomes αN. The width multiplier reduces the computational cost and number of parameters by α^2. The second hyper-parameter, ρ, is also used to reduce the computational cost. This one is applied directly to the input image, setting its resolution. The $\rho \in (0, 1]$ values are chosen to obtain typical input image resolutions. Similarly to the width multiplier, the resolution multiplier also reduces the computational cost and the number of parameters by ρ^2. Accordingly, both of the parameters are different ways of reducing the model size and computational cost. When combined, the effects on the final model can be even more significant.

2.3. Inception

Szegedy et al. [48] proposed the primary version of Inception. This model design is based on the premise that the desired object to classify or detect can present several sizes on different images. This leads to the difficulty of choosing the right kernel size. Inception proposes three different convolutional filter sizes to overcome this issue: 1×1, 3×3, and 5×5. Additionally, the NN model also computes max pooling. The output of all these operations is then concatenated, constituting the result of the respective Inception module.

Inception-V2 was developed to reduce the computational complexity of the original version. This is done by factorizing the convolution operations. For example, a 5×5 convolution is factorized into two 3×3 convolutions, improving the runtime performance. Similarly, an $m \times m$ convolution can be factorized into a combination of $1 \times m$ and $m \times 1$ convolutions.

3. Materials and Methods

The reliable semantic perception of an agricultural environment by a robot is a task that requires several development steps, as well as high amounts of learning data. In this work, a large collection of data in several vineyard contexts is proposed. This innovation created the VineSet, a dataset with RGB images of four different vineyards, and thermal images of a single one, containing the annotations for each image. The proposed dataset is available (http://vcriis01.inesctec.pt/datasets/DataSet/VineSet.zip) and it was recognized by the ROS Agriculture community (http://wiki.ros.org/agriculture) as "A Large Vine Trunk Image Collection and Annotation using the Pascal VOC format". In addition, our pipeline supports a variety of augmentation operations that allow for extending the original dataset. The augmentation procedure automatically generates the annotations for the augmented images. With this information, state-of-the-art SSD models are trained using the Tensorflow (https://www.tensorflow.org/) API and then deployed in an Edge-AI manner. Figure 3 represents the main steps performed until real-time vine trunk detection.

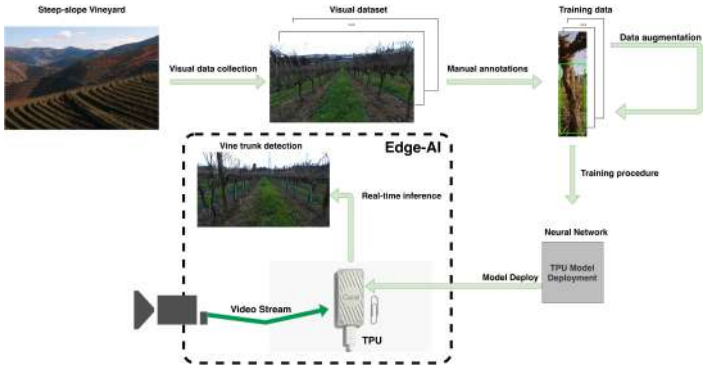


Figure 3. High-level design of the vine trunk detection framework. The procedure starts with the data acquisition in real-world vineyards, followed by the manual vine trunk annotation. The VineSet is extended using data augmentation techniques to increase the dataset size. Finally, the Neural Networks are trained and deployed in a Edge-AI manner, using dedicated hardware.

In addition to this vine trunk detection pipeline, an assisted labelling framework is also proposed. A DL model is used to automatically annotate an input dataset and provide the annotations in a standard format. The user can then load the annotations and manually annotate the remaining objects not detected by the DL model, as detailed in Section 3.5. In terms of cost, we propose a cost-effective solution that requires two main hardware components: a standard RGB camera (https://www.raspberrypi.org/products/raspberry-

pi-high-quality-camera/) (<70€), and a low-cost TPU device (https://coral.ai/products/accelerator) (<60€). The devices must be plugged to a central processing unit, such as a microprocessor or a standard computer. This being said, the proposed solution is affordable for small/medium farmers, and it can have an impact in the improvement of the semantic perception systems in vineyards.

3.1. Data Acquisition

In order to acquire images in real vineyard scenarios, we used our robotic platform AgRob V16 [49], which is represented in Figure 4.

Figure 4. The AgRob V16 robotic platform recording data in one of the vineyards that compose the VineSet.

This robot contains a frontal stereo RGB camera and a frontal thermal camera. To collect the image data, the robot travelled along the vineyard corridors of four different vineyards, and then recorded video streams saved in the ROSBag file format. In one of the vineyards, the thermal camera was activated, and also recorded video to the same file format. After all, the acquisition on the field, the ROSBag files were processed, and image frames were extracted from them at a fixed frame-rate, which resulted in a total of 952 vineyard images. Figure 5 shows an example of each type of image collected.

From this, one can see that the dataset presents considerable data variability. In fact, the VineSet contains images that were collected at different stages of the year that capture different characteristics of the vineyards imposed by the temporal offset. Additionally, it presents images of vineyards with and without foliage and with different levels of luminosity. Finally, the presence of thermal vineyard images adds the notion of temperature to the dataset, which can improve the learning procedure.

(**a**) (**b**)

(**c**) (**d**) (**e**)

Figure 5. The five categories of vineyard images that compose the VineSet. (**a**–**d**) Four of them are from different Portugal Vineyards and (**e**) the other represents the set of thermal images of the vineyard present in Figure 5a.

3.2. Data Annotation

Given the training dataset, the perceptible vine trunks were manually annotated on the images. Figure 6 shows an image example of each vine with the respective annotations.

(**a**) (**b**)

Figure 6. The result of the annotation process for trunk detection in two different vineyards represented in (**a**,**b**). The green dots represent the extremes of the annotated bounding boxes that contain the vineyard trunks.

The output from this procedure is a set of bounding boxes with different sizes for each image. These are represented in a `.xml` file with the Pascal VOC annotation format, containing the label class that is considered and the four corners location of each bounding box. It is worth noting that the annotations are a fundamental part of the VineSet dataset, since they represent trunk's location in the object detection learning procedure.

3.3. Data Augmentation

Even though DL outperforms most traditional Machine Learning (ML) methods in terms of precision and real-time application [18], one of the biggest challenges is to overcome overfitting. This frequent ML problem consists of modelling the data too well, only learning the expected output for each input instead of learning the input data's general distribution. Additionally, conditions, such as variation of sunlight illumination during the day or the outdoor environment terrain, may affect performance. In order to avoid over-

fitting and the network generalization, data augmentation is a usual method to enhance data variability for training by enlarging the dataset using label-preserving transformations. Thus, to increase the VineSet's diversity and robustness, the collected images were pre-processed with the augmentation techniques presented in Table 3, and VineSet was extended to 9481 images.

Table 3. Description of the augmentation operations used to expand the original collection of data.

Augmentation Operation	Description
Rotation	Rotates the image by 15, −15 and 45 degrees.
Translation	Translates the image by −30% to +30% on x- and y-axis.
Scale	Scales the image to a value of 50 to 150% of their original size.
Flipping	Mirrors the image horizontally.
Multiply	Multiplies all pixels in an image with a random value sampled once per image, which can be used to make images lighter or darker.
Hue and saturation	Increases or decreases hue and saturation by random values. This operation first transforms images to HSV colourspace, then adds random values to the H and S channels, and afterwards converts back to RGB.
Gaussian noise	Adds noise sampled from Gaussian distributions element-wise to images.
Random combination	Applies a random combination of three of the previous operations.

As described, the VineSet is extended by applying operations on the original images, such as rotation, translation, scaling, flipping, multiplication, saturation, and the addition of noise sampled from a Gaussian distribution. In addition, a random combination of three of the previous operations is also supported. This highly increases the number of combinations of operations possible and, consequently, increases the extended dataset variability. Figure 7 represents an example of an application of the augmentation operations to a single image.

(**a**) Original (**b**) Translation (**c**) Multiply

(**d**) Hue/Saturation (**e**) Horizontal Flip (**f**) Rotation

Figure 7. *Cont*.

(**g**) Rotation (**h**) Gaussian Noise (**i**) Combination 3

Figure 7. Set of several augmentation operations that were applied to VineSet, such as translation, multiplication, hue and saturation, flip, rotation, Gaussian noise, and, finally, a random combination of three of the previously mentioned operations.

3.4. Training Procedure

The Edge-AI-based deployment of NNs is performed while using a TPU. This hardware device is provided by Google and requires models that were trained using the Tensorflow [50] API. Tensorflow is an open-source end-to-end framework for machine learning and DL that provides tools, libraries, and models. With this tool, the implementation of DL applications can be built in a more straightforward, comprehensive, and flexible way. In the Edge-AI context, Tensorflow provides a tool, called Tensorflow Lite, which is device-oriented. Using Tensorflow Lite, the trained models can be transformed to be compatible with edge-based hardware. In this work, this tool is used for two main ends:

1. the full quantization of models to 8-bit precision; and,
2. compilation of the model to the TPU context.

Step 1. consists of converting the training models from 32-bit to 8-bit precision, since the TPU device can only deploy fully quantized models. The second step is a fundamental part of the process. In this, the model is compiled to the TPU context. In other words, the DL model operations are allocated to the device. The unsupported operations remain allocated to the host device, usually a CPU. Thus, the higher the number of allocated operations to the edge device, the faster the inference procedure will be. With this in mind, the model selection is crucial for the reliable operation of the detectors. For the object detection task, the SSD is the most appropriate, and one specific set of models was particularly implemented for edge- and embedded-based applications: the MobileNets [46]. In this work, SSD MobileNet-V1 and SSD MobileNet-V2 were both trained and deployed, as well as the SSD Inception-V2 model [47]. The three models were benchmarked and characterized by evaluating the dataset size and comparing the inference performance between training them from scratch and using Transfer Learning.

3.5. Assisted Labelling Tool

Training a DL model involves several steps, one of the most important of which is data annotation. Generally, this step is a long process, and the time that is spent depends on several factors, such as the total number of images that the dataset has the number of classes and the ease of manually identifying the bounding box that corresponds to each class. Thus, this work proposes creating an assisted labelling procedure that uses AI to help the annotation process in the detection of trunks in the vineyards. Figure 8 represents the layout of the created application.

In this way, a comprehensive and user-friendly python notebook was developed. The procedure of this new solution consists of using an online platform, Google Colaboratory (https://colab.research.google.com), so that the user can save his machine's resources. This tool provides a DL model that is trained for detecting vine trunks, and also capable of detecting trunks in other contexts such as orchards or forests. Accordingly, an essential factor for automating this process is the use of the DL model. Taking the results obtained in Section 4 into account, the SSD MobileNet-V1 trained with VineSet was the model chosen for the detection of trunks in the images introduced in this tool. The assisted labelling procedure uses this model to pre-process the user dataset, automatically annotating the

detected trunks, saving the annotations in the Pascal VOC format. The user can then load the automatic annotations and complete them manually. This tool reduces the percentage of annotations taken manually, significantly reducing the time that it takes to insert labels into relatively large datasets. It is worth noting that this procedure is iterative, in the way that the user can improve DL-based object detection models performance, by iteratively annotating objects that the model fails to recognize.

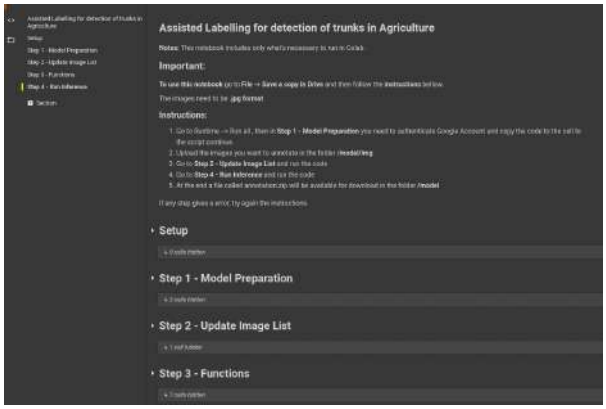


Figure 8. The assisted labelling tool interface.

4. Results

This section evaluates the semantic vineyard perception captured by on-board sensors while using Edge-AI technologies. The evaluation considers the vine trunk detection precision and inference time performance.

4.1. Methodology

A subset of the entire dataset was used for test purposes and not employed in the training procedures in order to test and evaluate the models. The test set selection is randomly generated, picking 10% of the VineSet images. In this work, two train datasets were used. This was done to evaluate the impact of the training dataset size on the detectors performance. Thus, the original VineSet dataset and a small subset of it with 336 non-augmented images were used. The evaluation approach is described in Section 4.2. In addition, the inference time per image was measured for each model deployed in the TPU, while considering the average inference time for all the images present in the test dataset. Finally, the assisted labelling procedure is evaluated when considering three experiments: the first in a vineyard not present in the VineSet dataset, other in forest images, and a final one in a hazelnut orchard. The labelling was assisted in these three scenarios, and the time that was saved in the annotation procedure was measured for each of them.

4.2. Object Detection Metrics

The PASCAL VOC Challenge [51] was used to evaluate the considered model's performance on Google's USB Accelerator. Most of DL-based works use AP to evaluate their models, as shown in Section 1. Thus, the use of this metric simplifies the comparison between state-of-the-art approaches. In order to compute the AP, Pascal VOC starts by

calculating the Intersection over Union (IoU). Given an annotated ground truth bounding box B_g, and a detected bounding box B_d, the IoU is computed, as follows:

$$IoU = \frac{m(B_g \cap B_d)}{m(B_g \cup B_d)} \tag{1}$$

where $m(x)$ denotes the area of x. Figure 9, shows a graphical representation of this concept.

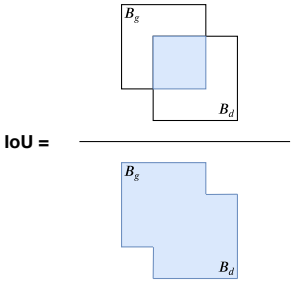


Figure 9. Interception over union representation.

Thus, IoU represents the quotient between the overlap area and the union area between the ground truth and bounding boxes' detection. Using this definition, for a given threshold value t, three main concepts can be defined:

- True Positive (TP): IoU $\geq t$, i.e., a correct detection.
- False Positive (FP): IoU $\leq t$, i.e., an incorrect detection.
- False Negative (FN): a ground truth is not detected.

In the case that multiple detections for a single annotation (or ground truth) are computed, this metric only considers as TP the one that presents the higher IoU value. All of the other detections are marked as FPs. Subsequently, to compute the model AP, two concepts are defined. The first, precision p, is defined as the total number of TPs over all the detections. The second, recall r, is the total number of TPs over all the ground truths. With these two concepts, AP is calculated as a combination of precision and recall. In other words, the AP is the average value of the precision vs recall curve $p(r)$ for $r \in [0, 1]$. The evaluation considers that a suitable detector is the one that maintains the precision high for an increase in recall. Mathematically, this is expressedm as follows

$$\sum_{r=0}^{1} (r_{n+1} - r_n) p_{interp}(r_{n+1}) \tag{2}$$

with

$$p_{interp}(r_{n+1}) = \max_{\widetilde{r}: \widetilde{r} \geq r_{n+1}} p(\widetilde{r}) \tag{3}$$

where $p(\widetilde{r})$ is the measured precision at recall $\widetilde{r}$.

This work also evaluates the models while using the F1 score. This score is the harmonic mean between the precision p and recall r, and it can be calculated as follows:

$$F1 = 2 \frac{p \cdot r}{p + r} \tag{4}$$

4.3. Detectors Performance

In this work, in order to evaluate the models performance, we consider an IoU threshold of 0.50, since we are interested in detecting trunks with high and medium precision to use as landmarks for Simultaneous Localization and Mapping in agriculture.

Additionally, trunks are non-uniform agents that present inclination and perturbations. Thus, a detection can still be valid and precise, even if it does not exactly match the annotation. Table 4 summarizes the models' performance in terms of precision and F1 score.

Table 4. AP (%) and F1 Scores of the trained models. The three models were trained while using a small subset of the VineSet, and VineSet itself. In addition, using VineSet, three experiments were performed for each model. The performance of each one was compared using Transfer Learning (by fine-tuning) and training them from scratch with two different numbers of training epochs.

M del	Train Dataset	Fine-Tuning		From Scratch		From Scratch	
		50 k		50 k		100 k	
		AP (%)	F1	AP (%)	F1	AP (%)	F1
SSD MobileNet-V1	Small subset	49.74	0.610	-	-	-	-
SSD MobileNet-V2		52.98	0.590	-	-	-	-
SSD Inception-V2		46.10	0.610	-	-	-	-
SSD MobileNet-V1	VineSet	84.16	0.841	68.44	0.685	85.93	0.834
SSD MobileNet-V2		83.01	0.808	60.44	0.639	83.70	0.812
SSD Inception-V2		75.78	0.848	58.05	0.658	76.77	0.849

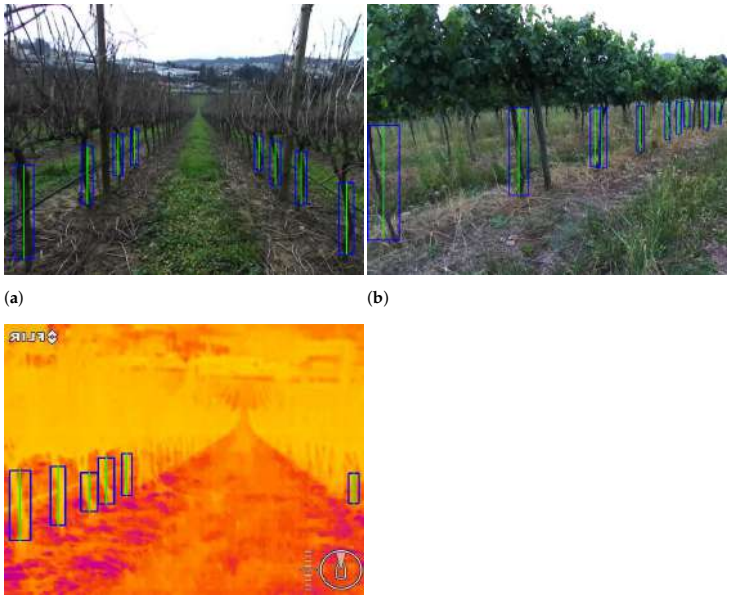
When considering the trained models with VineSet using Transfer Learning (fine-tuning), we achieved a maximum AP of 84.16%, corresponding to SSD MobileNet-V1. This proves that the VineSet dataset can be successfully used to train models to detect vine trunks, even while considering lightweight model, such as the MobileNets. SSD MobileNet-V2 achieves a similar precision (83.01%), which is expected, since both models have similar architectures. The SSD Inception-V2 presents a lower precision (75.78%), but the higher F1 score (0.848). This means that this model is the one that has the best balance between precision and recall. Figure 10 shows an example of three detections using the SSD MobileNet-V1.

In terms of inference time, from Table 5 several conclusions can be taken.

The edge TPU device is built with a specific architecture that is optimized to deploy DL models. If the models are compatible with it, then it is expected that the inference runs at high frame rate. From the experiments performed, the MobileNets achieved an average inference time of 21.18 ms and 23.14 ms. In terms of frequency, this is equivalent to approximately 50 frames per second. This means that the edge TPU can process approximately 50 images and output the desired detections in one second. For SSD Inception-V2, the processing rate is slower. The edge device has an average inference time of 359.64 ms for this device. This can be explained by two main reasons. Firstly, the MobileNets use depthwise separable convolution, while Inception uses standard convolution, which results in fewer parameters on MobileNet when compared to Inception. Secondly, the first set of models is more oriented to edge devices. Thus, in the compilation process for the TPU, a higher number of operations is allocated to it. On the opposite side, the SSD Inception-V2 allocates more operations to the host CPU, due to the non-compatibility of some of them. These two factors lead to the decrease of the inference time performance.

Table 5. Inference time per image (ms) of each trained model deployed on the edge TPU device.

Model	Inference Time per Image (ms)
SSD MobileNet-V1	21.18
SSD MobileNet-V2	23.14
SSD Inception-V2	359.64



(**a**) (**b**)

(**c**)

Figure 10. Detection results in (**a**,**b**) two RGB images from different vineyards and (**c**) one thermal image using Single Shot Multibox Detector (SSD) MobileNet-V1 trained with VineSet.

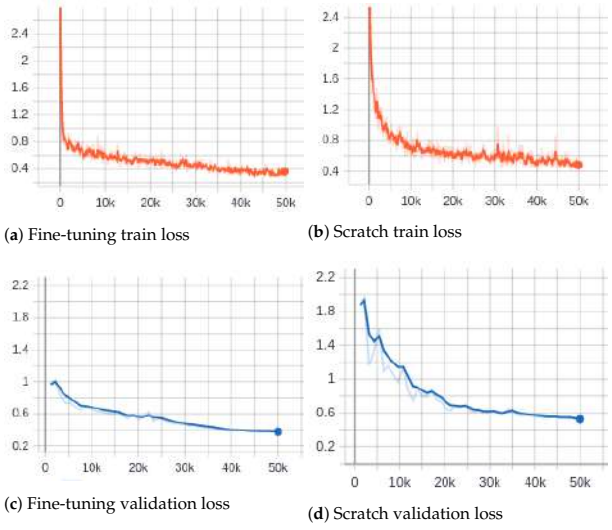
4.4. Impact of the Dataset Size on the Detection Performance

In order to evaluate the training dataset size impact in the final models' performance, we trained them using a small subset of the VineSet. Table 4 summarizes all of the obtained results in terms of AP and F1 score. As expected, the models that were trained with lower amounts of data present lower precision. The lower variability of data leads to a lower learning capability and, consequently, to a lower inference performance. In this context, we verified a decay of 34.42% of AP for SSD MobileNet-V1, 30.03 for SSD MobileNet-V2, and 29.68% for SSD Inception-V2. Thus, the decrease in the training dataset size has a significant impact on the models performance. This proves the high importance of considering a considerable amount of data with variability when dealing with DL models.

4.5. Comparison of Transfer Learning against Training from Scratch

One of the main questions of developers while training and deploying DL models is to fine-tune a pre-trained model or to train it from scratch. When using Transfer Learning, the model uses some of the pre-trained weights and restores other ones. Thus, the starting point on Transfer Learning is one step ahead when comparing training the same model from scratch. In the last case, all of the weights have to be learned, leading to a longer learning process. To test this, we train the three models from scratch using two epoch values (50,000 and 100,000). From Table 4, we can verify that, for models that are trained from scratch to achieve similar performance as compared with the ones fine-tuned, the number of training epochs has to be doubled. From Figure 11, this is also visible.

Here, it is possible to verify that the training loss for the fine-tuned models converges faster. Additionally, the validation loss has a more precise starting point for these models, as visible from Figure 11c,d. Thus, these experiments proved the theoretical assumptions that were made.



(**a**) Fine-tuning train loss

(**b**) Scratch train loss

(**c**) Fine-tuning validation loss

(**d**) Scratch validation loss

Figure 11. Train and validation loss of SSD MobileNet-V1 using 50,000 epochs and considering Transfer Learning and training from scratch.

4.6. Assisted Labelling Procedure

Several factors were analyzed in comparison with manual annotation in order to assess the performance of our assisted labelling procedure. Specifically, the average time to manually label a trunk was measured over several experiments, and it was concluded that, on average, the time spent per trunk annotation is 5 s. Thus, once this value is established, the total time that is spent on several images can also be estimated. The time spent on assisted annotation was calculated from the percentage of annotations made automatically, and the percentage of annotations made manually. In this way, the total time that is spent by the tool is calculated through the time spent by the automatic annotation plus the offset created by the missing annotations. Table 6 summarizes the results.

Table 6. Assisted labelling procedure evaluation.

Dataset	Number of Images	Number of Trunks	Automatic Annotations (%)	Average Time with Assisted Labelling (min)	Average Time without Assisted Labelling (min)
Other vineyards	11	75	72.32	1.74	6.35
Hazelnut orchard	20	139	48.34	5.99	11.58
Forest	264	1647	28.05	101.97	137.25

These experiments used the proposed assisted annotation procedure to automatically annotate the images from other vineyards, but also from an orchard, and a forest. The results estimate that the automatic annotation tool can reduce the average labelling time from 6.35 min. to 1.74 min for vineyards. In orchards, the tool annotates 48.34% of the trunks and, in forests, 28.05%. This means that only the remaining set of trunks have to be annotated by the user. The tool can be iterative improved by updating the back-end DL model with user's annotations. Figure 12 shows the result of the automatic annotation in three different contexts.

(**a**) Hazelnut orchard environment.

(**b**) Forest environment.

(**c**) Vineyard environment.

Figure 12. Automatic annotations in different areas of agriculture.

4.7. Discussion

The experiments performed revealed several takeaways. With a wide variety of data, lightweight DL models can be used for detection purposes in agricultural contexts. With these models, Edge-AI-based devices can be used to perform high-performance inference. As discussed, one of the most important factors to build successful detectors is to provide sufficient amounts of varied learning data. Additionally, using models that already have a learning history can accelerate the learning procedure, thus saving resources and time.

In comparison with the state-of-the-art, our approach outperforms the work that was proposed by Badeka et al. [21] that achieved an AP of 73.2% using DL models to detect vine trunks. Other approaches use conventional image processing techniques, or data fusion, to achieve the same goal. In particular, Lamprecht et al. [24] uses a 3D clustering procedure to isolate laser points on tree trunks, achieving an overall accuracy of 84%. Shalal et al. [25] fuse a camera with a laser sensor to detect orchard trunks with a detection confidence of 82.2%. Our approach achieves similar results using less resources, presenting extremely high inference rates. Regarding the works that use DL in other agricultural contexts, our approach presents a state-of-the-art performance (AP higher than 80%) and promotes DL concepts in vineyard contexts. We think that these concepts have extreme importance in agricultural robotics and that, shortly, they will be usually approached to the detriment of more conventional image processing techniques. In comparison with our work, some works present higher precision rates, such as Dias et al. [31], Zheng et al. [31], and Koirala et al. [32]. In relation with these, our work uses simpler DL models with less operations and being less computationally expensive. Even so, this paper can still present a state-of-the-art performance, with the advantage of running at high frame rates.

The major drawback faced while implementing the proposed techniques was the high amount of time and resources spent during the annotation process. This led to the creation

of the automatic annotation tool, so that, in the future, we can spend less time in this step. Looking to the future, one of the most important steps will be to develop models and acquire data, so that robots can also have this level of perception during the night. In most agricultural sectors, robots can be employed to autonomously perform several tasks during this period. Consequently, they should also have the ability to detect objects and natural agents at night.

5. Conclusions

In this work, DL is used to detect semantic features in vineyards. Single Shot Multibox detectors are trained while using a novel built in-house dataset, the VineSet. The models are converted to an edge TPU context and then deployed in this hardware device. Additionally, an assisted annotation tool is proposed to ease the dataset creation procedure. The results show that our detectors present an AP up to 84.16% and an F1 score up to 0.848. The MobileNets are executed in the edge TPU at a high frame rate, with an average inference time per image up to 23.14 ms. Additionally, from the characterization performed, two main conclusions can be made: the amount of training data has a significant impact on the detectors' performance; and, the number of training epochs has to be double in order for a detector trained from scratch achieve a similar performance of the one fine-tuned. Finally, the annotation tool proved to help in the annotation process, being capable of automatically annotating trunks in other agricultural contexts, such as orchards and forests.

In future work, we aim to project and implement a DL model from scratch in order to detect vine trunks. Additionally, we will integrate the proposed models in a Simultaneous Localization and Mapping stack as landmark extractors.

Author Contributions: Conceptualization, A.S.A., N.N.M. and F.N.d.S.; methodology, A.S.A., N.N.M. and F.N.d.S.; validation, E.J.S.P., D.S., A.J.S. and J.B.-C.; investigation, A.S.A., N.N.M. and F.N.d.S.; resources, F.N.d.S. and A.J.S.; writing—original draft preparation, A.S.A., N.N.M.; writing—review and editing, E.J.S.P., D.S., A.J.S. and J.B.-C.; supervision, F.N.d.S., E.J.S.P., A.J.S. and J.B.-C. All authors have read and agreed to the published version of the manuscript.

Funding: The research leading to these results has received funding from the European Union's Horizon 2020—The EU Framework Programme for Research and Innovation 2014-2020, under grant agreement No. 101000554.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Acknowledgments: André Silva Pinto de Aguiar thanks the FCT—Foundation for Science and Technology, Portugal for the Ph.D. Grant DFA/BD/5318/2020.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Andresen, T.; de Aguiar, F.B.; Curado, M.J. The Alto Douro Wine Region greenway. *Landsc. Urban Plan.* **2004**, *68*, 289–303. [CrossRef]
2. Dos Santos, F.N.; Sobreira, H.; Campos, D.; Morais, R.; Moreira, A.P.; Contente, O. Towards a reliable robot for steep slope vineyards monitoring. *J. Intell. Robot. Syst.* **2016**, *83*, 429–444. [CrossRef]
3. Roldán, J.J.; del Cerro, J.; Garzón-Ramos, D.; Garcia-Aunon, P.; Garzón, M.; de León, J.; Barrientos, A. Robots in Agriculture: State of Art and Practical Experiences. In *Service Robots*; InTech: London, UK, 2018; doi:10.5772/intechopen.69874. [CrossRef]
4. Duckett, T.; Pearson, S.; Blackmore, S.; Grieve, B.; Chen, W.H.; Cielniak, G.; Cleaversmith, J.; Dai, J.; Davis, S.; Fox, C.; et al. Agricultural Robotics: The Future of Robotic Agriculture. *arXiv* **2018**, arXiv:cs.RO/1806.06762.
5. Dos Santos, F.N.; Sobreira, H.M.P.; Campos, D.F.B.; Morais, R.; Moreira, A.P.G.M.; Contente, O.M.S. Towards a Reliable Monitoring Robot for Mountain Vineyards. In Proceedings of the 2015 IEEE International Conference on Autonomous Robot Systems and Competitions, Vila Real, Portugal, 8–10 April 2015; pp. 37–43. [CrossRef]
6. Cheein, F.A.; Steiner, G.; Paina, G.P.; Carelli, R. Optimized EIF-SLAM algorithm for precision agriculture mapping based on stems detection. *Comput. Electron. Agric.* **2011**, *78*, 195–207. [CrossRef]

7. Aguiar, A.; Santos, F.; Santos, L.; Sousa, A. Monocular Visual Odometry Using Fisheye Lens Cameras. In *Progress in Artificial Intelligence*; Moura Oliveira, P., Novais, P., Reis, L.P., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 319–330.
8. Aguiar, A.S.; dos Santos, F.N.; Cunha, J.B.; Sobreira, H.; Sousa, A.J. Localization and Mapping for Robots in Agriculture and Forestry: A Survey. *Robotics* **2020**, *9*, 97. [CrossRef]
9. Zhao, Z.; Zheng, P.; Xu, S.; Wu, X. Object Detection With Deep Learning: A Review. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 3212–3232. [CrossRef]
10. LeCun, Y.; Bengio, Y.; Hinton, G. Deep Learning. *Nature* **2015**, *521*, 436–444. [CrossRef]
11. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
12. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Netw.* **2015**, *61*, 85–117. [CrossRef]
13. Pan, S.J.; Yang, Q. A Survey on Transfer Learning. *IEEE Trans. Knowl. Data Eng.* **2010**, *22*, 1345–1359. [CrossRef]
14. Weiss, K.; Khoshgoftaar, T.M.; Wang, D. A survey of transfer learning. *J. Big Data* **2016**, *3*, 9. [CrossRef]
15. Lu, J.; Behbood, V.; Hao, P.; Zuo, H.; Xue, S.; Zhang, G. Transfer learning using computational intelligence: A survey. *Knowl. Based Syst.* **2015**, *80*, 14–23. [CrossRef]
16. Shao, L.; Zhu, F.; Li, X. Transfer Learning for Visual Categorization: A Survey. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *26*, 1019–1034. [CrossRef]
17. Torrey, L.; Shavlik, J. Transfer learning. In *Handbook of Research on Machine Learning Applications*; IGI Global: Hershey, PA, USA, 2009; doi:10.4018/978-1-60566-766-9.ch011. [CrossRef]
18. Santos, L.; Santos, F.N.; Oliveira, P.M.; Shinde, P. Deep Learning Applications in Agriculture: A Short Review. In *Robot 2019: Fourth Iberian Robotics Conference*; Silva, M.F., Luís Lima, J., Reis, L.P., Sanfeliu, A., Tardioli, D., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 139–151.
19. Kamilaris, A.; Prenafeta-Boldú, F.X. Deep learning in agriculture: A survey. *Comput. Electron. Agric.* **2018**, *147*, 70–90. [CrossRef]
20. Kamilaris, A.; Prenafeta-Boldú, F.X. A review of the use of convolutional neural networks in agriculture. *J. Agric. Sci.* **2018**, *156*, 312–322. [CrossRef]
21. Badeka, E.; Kalampokas, T.; Vrochidou, E.; Tziridis, K.; Papakostas, G.; Pachidis, T.; Kaburlasos, V. Real-time vineyard trunk detection for a grapes harvesting robot via deep learning. In Proceedings of the Thirteenth International Conference on Machine Vision, Rome, Italy, 2–6 November 2020; Osten, W., Nikolaev, D.P., Zhou, J., Eds.; International Society for Optics and Photonics, SPIE: Bellingham, WA, USA, 2021; Volume 11605, pp. 394–400. [CrossRef]
22. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv* **2015**, arXiv:cs.CV/1506.01497.
23. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.
24. Lamprecht, S.; Stoffels, J.; Dotzler, S.; Haß, E.; Udelhoven, T. aTrunk—An ALS-Based Trunk Detection Algorithm. *Remote Sens.* **2015**, *7*, 9975–9997. [CrossRef]
25. Shalal, N.; Low, T.; McCarthy, C.; Hancock, N. A preliminary evaluation of vision and laser sensing for tree trunk detection and orchard mapping. In Proceedings of the Australasian Conference on Robotics and Automation (ACRA 2013), Sydney, Australia, 2–4 December 2013; pp. 1–10.
26. Xue, J.; Fan, B.; Yan, J.; Dong, S.; Ding, Q. Trunk detection based on laser radar and vision data fusion. *Int. J. Agric. Biol. Eng.* **2018**, *11*, 20–26. [CrossRef]
27. Juman, M.A.; Wong, Y.W.; Rajkumar, R.K.; Goh, L.J. A novel tree trunk detection method for oil-palm plantation navigation. *Comput. Electron. Agric.* **2016**, *128*, 172–180. [CrossRef]
28. Bargoti, S.; Underwood, J.P.; Nieto, J.I.; Sukkarieh, S. A Pipeline for Trunk Detection in Trellis Structured Apple Orchards. *J. Field Robot.* **2015**, *32*, 1075–1094. [CrossRef]
29. Colmenero-Martinez, J.T.; Blanco-Roldán, G.L.; Bayano-Tejero, S.; Castillo-Ruiz, F.J.; Sola-Guirado, R.R.; Gil-Ribes, J.A. An automatic trunk-detection system for intensive olive harvesting with trunk shaker. *Biosyst. Eng.* **2018**, *172*, 92–101. [CrossRef]
30. Dias, P.A.; Tabb, A.; Medeiros, H. Apple flower detection using deep convolutional networks. *Comput. Ind.* **2018**, *99*, 17–28. [CrossRef]
31. Zheng, Y.Y.; Kong, J.L.; Jin, X.B.; Wang, X.Y.; Su, T.L.; Zuo, M. CropDeep: The Crop Vision Dataset for Deep-Learning-Based Classification and Detection in Precision Agriculture. *Sensors* **2019**, *19*, 1058. [CrossRef]
32. Koirala, A.; Walsh, K.B.; Wang, Z.X.; McCarthy, C. Deep learning for real-time fruit detection and orchard fruit load estimation: Benchmarking of 'MangoYOLO'. *Precis. Agric.* **2019**, *20*, 1107–1135. [CrossRef]
33. Tian, Y.; Yang, G.; Wang, Z.; Wang, H.; Li, E.; Liang, Z. Apple detection during different growth stages in orchards using the improved YOLO-V3 model. *Comput. Electron. Agric.* **2019**, *157*, 417–426. [CrossRef]
34. Bargoti, S.; Underwood, J. Deep fruit detection in orchards. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 3626–3633. [CrossRef]
35. Sa, I.; Ge, Z.; Dayoub, F.; Upcroft, B.; Perez, T.; McCool, C. DeepFruits: A Fruit Detection System Using Deep Neural Networks. *Sensors* **2016**, *16*, 1222. [CrossRef]
36. Kirk, R.; Cielniak, G.; Mangan, M. L*a*b*Fruits: A Rapid and Robust Outdoor Fruit Detection System Combining Bio-Inspired Features with One-Stage Deep Learning Networks. *Sensors* **2020**, *20*, 275. [CrossRef] [PubMed]

37. Li, W.; Fu, H.; Yu, L.; Cracknell, A. Deep Learning Based Oil Palm Tree Detection and Counting for High-Resolution Remote Sensing Images. *Remote Sens.* **2017**, *9*, 22. [CrossRef]
38. Ding, W.; Taylor, G.W. Automatic moth detection from trap images for pest management. *Comput. Electron. Agric.* **2016**, *123*, 17–28. [CrossRef]
39. Zhong, Y.; Gao, J.; Lei, Q.; Zhou, Y. A Vision-Based Counting and Recognition System for Flying Insects in Intelligent Agriculture. *Sensors* **2018**, *18*, 1489. [CrossRef]
40. Steen, K.A.; Christiansen, P.; Karstoft, H.; Jørgensen, R.N. Using Deep Learning to Challenge Safety Standard for Highly Autonomous Machines in Agriculture. *J. Imaging* **2016**, *2*, 6. [CrossRef]
41. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.
42. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:cs.CV/1804.02767.
43. Pinto de Aguiar, A.S.; Neves dos Santos, F.B.; Feliz dos Santos, L.C.; de Jesus Filipe, V.M.; Miranda de Sousa, A.J. Vineyard trunk detection using deep learning—An experimental device benchmark. *Comput. Electron. Agric.* **2020**, *175*, 105535. [CrossRef]
44. Aguiar, A.S.; Santos, F.N.D.; De Sousa, A.J.M.; Oliveira, P.M.; Santos, L.C. Visual Trunk Detection Using Transfer Learning and a Deep Learning-Based Coprocessor. *IEEE Access* **2020**, *8*, 77308–77320. [CrossRef]
45. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.E.; Fu, C.; Berg, A.C. SSD: Single Shot MultiBox Detector. In Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2015.
46. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:cs.CV/1704.04861.
47. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the Inception Architecture for Computer Vision. *arXiv* **2015**, arXiv:cs.CV/1512.00567.
48. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going Deeper With Convolutions. In Proceedings of the The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015.
49. Santos, L.; Santos, F.; Mendes, J.; Costa, P.; Lima, J.; Reis, R.; Shinde, P. Path Planning Aware of Robot's Center of Mass for Steep Slope Vineyards. *Robotica* **2020**, *38*, 684–698. [CrossRef]
50. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. TensorFlow: A system for large-scale machine learning. *arXiv* **2016**, arXiv:1605.08695.
51. Everingham, M.; Van Gool, L.; Williams, C.K.; Winn, J.; Zisserman, A. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338. [CrossRef]

4.4 Grape Bunch Detection at Different Growth Stages Using Deep Learning Quantized Models

After the research and exploration of the vine trunk detection system, and to have a more complete semantic vineyard perception, this thesis focused on grape bunch detection. This work was published in the MDPI Agronomy Journal and is entitled *Grape Bunch Detection at Different Growth Stages Using Deep Learning Quantized Models* (Aguiar et al., 2021c). In the same way as the proposed previous works, this approach uses lightweight models executed on an embedded device that is placed on-board our robotic platforms. One of the main novelties of this work is the availability of a public dataset containing 1929 images of vineyard canopies and the respective annotations. This data was collected mounting a monocular camera in a robotic arm pointing to the canopy during several robotic trajectories. The other main novelty of this work is the data acquisition and semantic perception considering different grape bunch growth stages. The data collection was performed in two different temporal samples capturing the grape bunches right after the bloom and in an intermediate growth stage. Since this perception system is intended to feed 3D mapping systems, this will allow semantic SLAM approaches (such as VineSLAM) to create maps of the vineyard considering a temporal dimension. Maps of different vineyard stages can then be analysed and compared by specialists. One straightforward application should be yield estimation. Results show that the developed detectors are robust to different illumination conditions and partial occluded grape bunches.

agronomy

MDPI

Article

Grape Bunch Detection at Different Growth Stages Using Deep Learning Quantized Models

André Silva Aguiar [1,2,*], Sandro Augusto Magalhães [1,3], Filipe Neves dos Santos [1], Luis Castro [1], Tatiana Pinho [1], João Valente [4], Rui Martins [1] and José Boaventura-Cunha [1,2]

[1] INESC TEC—INESC Technology and Science, 4200-465 Porto, Portugal; sandro.a.magalhaes@inesctec.pt (S.A.M.); fbsantos@inesctec.pt (F.N.d.S.); luis.r.castro@inesctec.pt (L.C.); tatiana.m.pinho@inesctec.pt (T.P.); rui.c.martins@inesctec.pt (R.M.)
[2] School of Science and Technology, University of Trás-os-Montes e Alto Douro, 5000-801 Vila Real, Portugal; jboavent@utad.pt
[3] Faculty of Engineering, University of Porto, 4200-465 Porto, Portugal
[4] Information Technology Group, Wageningen University and Research, 6708 WG Wageningen, The Netherlands; joao.valente@wur.nl
* Correspondence: andre.s.aguiar@inesctec.pt

Abstract: The agricultural sector plays a fundamental role in our society, where it is increasingly important to automate processes, which can generate beneficial impacts in the productivity and quality of products. Perception and computer vision approaches can be fundamental in the implementation of robotics in agriculture. In particular, deep learning can be used for image classification or object detection, endowing machines with the capability to perform operations in the agriculture context. In this work, deep learning was used for the detection of grape bunches in vineyards considering different growth stages: the early stage just after the bloom and the medium stage where the grape bunches present an intermediate development. Two state-of-the-art single-shot multibox models were trained, quantized, and deployed in a low-cost and low-power hardware device, a Tensor Processing Unit. The training input was a novel and publicly available dataset proposed in this work. This dataset contains 1929 images and respective annotations of grape bunches at two different growth stages, captured by different cameras in several illumination conditions. The models were benchmarked and characterized considering the variation of two different parameters: the confidence score and the intersection over union threshold. The results showed that the deployed models could detect grape bunches in images with a medium average precision up to 66.96%. Since this approach uses low resources, a low-cost and low-power hardware device that requires simplified models with 8 bit quantization, the obtained performance was satisfactory. Experiments also demonstrated that the models performed better in identifying grape bunches at the medium growth stage, in comparison with grape bunches present in the vineyard after the bloom, since the second class represents smaller grape bunches, with a color and texture more similar to the surrounding foliage, which complicates their detection.

Keywords: deep learning; grape bunch detection; agriculture

Citation: Aguiar, A.S.; Magalhães, S.A.; dos Santos, F.N.; Castro, L.; Pinho, T.; Valente, J.; Martins, R.; Boaventura-Cunha, J. Grape Bunch Detection at Different Growth Stages Using Deep Learning Quantized Models. *Agronomy* **2021**, *11*, 1890. https://doi.org/10.3390/agronomy11091890

Academic Editor: Roberto Marani

Received: 31 August 2021
Accepted: 17 September 2021
Published: 21 September 2021

1. Introduction

The agricultural sector plays a fundamental role in our society. Thus, research, development, and innovation should be promoted and implemented in the vast range of areas connected to agriculture. In this context, it is increasingly important to automatize processes in agricultural environments, which can generate beneficial impacts in the productivity and quality of products, minimizing the environmental impacts and production costs [1,2]. In particular, vineyards occupy large terrain extensions, which make human labor, many times, intense. Vineyards such as the ones located for example in the Douro Demarched Region, the oldest controlled wine-making region in the world, a UNESCO her-

itage place [3], are located along hills presenting harsh inclinations. In these environments, the automatization of processes becomes more challenging, as well as more necessary. Perception algorithms can be important to provide visual data processed to be further analyzed by specialists. These algorithms can be deployed onboard robots to provide detailed and large-scale information of the agricultural environments [4]. Perception applied to fruit detection can be a valuable resource. The automatic detection of fruits at early stages can be used to predict the yield estimation [5]. More advanced approaches should be able to detect fruits at different growth stages. With this, agronomists can analyze the data and collect information about, for example, the crop evolution over time.

In the past few years, Deep Learning (DL) has had a huge impact in the development of perception and computer vision algorithms [6]. This concept can be applied for object detection in images, which can be used for fruit detection in agriculture. Convolutional Neural Networks (CNNs) are widely used to perform such a task. They have shown the highest performance levels in several contests in machine learning and pattern recognition [7]. Image classification and object detection based on DL techniques are widely present in the agriculture sector, endowing machines with the capability to perform operations in the agriculture context such as plant disease detection, weed identification, seed identification, fruit detection and counting, and obstacle detection, among others [8–10]. In particular, in recent years, CNNs have been increasingly incorporated into plant phenotyping concepts. They have been very successful in modeling complicated systems, owing to their ability to distinguish patterns and extract regularities from data. Examples further extend to variety identification in seeds [11] and in intact plants by using leaves [12]. The presence of these techniques in real applications leads the state-of-the-art to develop more computationally efficient models and specific hardware to deploy such models. These low-cost and low-power hardware devices promote fast and efficient model inference and allow the deployment of DL in robotic platforms. This concept is usually known as Edge Artificial Intelligence (Edge-AI) [13].

Our previous works focused on the detection of vine trunks [14–16] and tomatoes in greenhouses [17]. This work intends to solve the problem of automatically detecting grape bunches in images considering different growth stages, so that more intelligent and advanced tasks can be performed by robots such as: harvesting, yield estimation, fruit picking, semantic mapping of cultures, and others. In particular, the motivation of this work is oriented toward several applications in the agricultural sector. The grape bunch detection can be used by Simultaneous Localization and Mapping (SLAM) systems to build precise semantic maps of the environment providing the detailed 3D location of the fruits on crops. This can be used to build prescription maps of the vineyards, which can optimize, for example, the application of fertilizers, seeds, or sprayers in different regions of the agricultural environment. In addition, considering the detection of grape bunches at different growth stages can be useful to track the evolution of the crop. Specialists in the agricultural sector can use the detection at the early growth stages for early yield estimation and then compare with the actual yield of the vineyard at more advanced growth stages.

One of the main features of this work is the use of cameras operating in the visible portion of the electromagnetic spectrum (400–700 nm). In this way, it is possible to implement an affordable solution without the requirement of trained personnel [18]. In the current state-of-the-art, however, not only a specific orientation of the object of interest in relation to the camera is required, but also defined illumination conditions, which limit the applicability to controlled-light environments [19]. The method presented in this paper is independent of the ambient light environment, making this solution cost-effective, portable (thus, in situ), and rapid. Thus, the contributions of the proposed approach are threefold:

- A publicly available dataset (https://doi.org/10.5281/zenodo.5114142) (accessed on 23 August 2021) containing 1929 images and annotations of grape bunches at different growth stages, captured by different cameras in several illumination conditions;
- A benchmark of Deep Learning (DL) quantized models for grape bunch detection at different growth stages;

- The deployment of the models in a low-cost and low-power hardware embedded device.

To sum up, this work innovates the state-of-the-art by proposing the first publicly available dataset containing images and annotations of grape bunches at different growth stages. In addition, this work proposes the benchmarking between 8 bit quantized models and their deployment in a dedicated hardware, which still is an underdeveloped area in the literature.

The rest of the paper is organized as follows. Section 2 presents the current state-of-the-art on DL-based object detection in agriculture and the current techniques for grape bunch, grape flower, and grape berry detection. Section 3 describes the proposed approach for grape bunch detection. Section 4 summarizes the obtained results. Finally, Section 5 presents the main conclusions of this work.

2. Related Work

The use of DL is present in several agricultural areas and contexts. In particular, this approach is often used for the detection of natural features in the cultures. Fruit detection and counting in orchards are the most common applications. Moreover, some works focus on obstacle and insect detection, as well as pest identification. Dias et al. [20] implemented a technique for apple flower identification, which is robust to changes in illumination and clutter. The authors used a pretrained CNN and Transfer Learning (TL) concepts to create the detector. In the context of mango fruit detection, Koirala et al. [21] compared the performance of six state-of-the-art DL techniques and proposed MangoYOLO, a new architecture based on YOLO [22]. Zeng et al. [23] proposed a large dataset for species classification and detection, called CropDeep. The dataset contains more than 30,000 images of 31 different classes. Bargoti and Underwood [24] used the standard Faster R-CNN architecture [25] to detect several types of fruits in orchards, such as apples, mangoes, and almonds. Additionally, Sa et al. [26] proposed a fruit detection system called DeepFruits while using the Faster R-CNN architecture. The proposed detectors were integrated in the software pipeline of an agricultural robot to estimate yield and automate the harvesting process. To detect ripe soft fruits, Kirk et al. [27] proposed a detector implemented as a combination of a conventional computer vision algorithm and a DL-based approach.

In vineyards, several works have tackled the problem of grape detection in images using computer vision approaches. Either DL-based or more traditional implementations are used to detect, segment, or track these natural features such as grape bunches, grape flowers, or single berries. Table 1 presents an overview of the current state-of-the-art in this area.

Table 1. Summary of the current state-of-the-art of Deep-Learning (DL)-based grape detection.

Reference	Application	Performance
Liu et al. [28] (2018)	Automated grape flower counting to determine potential yields at early stages.	Accuracy of 84.3% for flower estimation.
Diago et al. [29] (2014)	Assessment of flower number per inflorescence in grapevine.	Precision exceeding 90.0%.
Palacios et al. [30] (2020)	Estimation of the number of flowers at the bloom.	F1 score of 73.0% for individual flower detection.
Pérez-Zavala et al. [31] (2018)	Grape bunch detection for automating grapevine growth monitoring, spraying, leaf thinning, and harvesting tasks.	AP of 88.6% and Average Recall (AR) of 80.3%.
Reis et al. [32] (2012)	Support harvesting procedures by grape bunch detection.	97.0% and 91.0% correct classifications for red and white grapes.
Liu and Whitty et al. [33] (2015)	Precise yield estimation in vineyards by detecting bunches of red grapes in images.	Accuracy of 88.0% and recall of 91.6%.
Cecotti et al. [34] (2020)	Study of the best CNN architecture to detect grapes in images.	Accuracy of 99.0% for both red and white grapes.
Santos et al. [35] (2020)	Infer the crop state for yield prediction, precision agriculture, and automated harvesting.	F1 score of 91.0% for instance grape segmentation.
Xiong et al. [36] (2018)	Develop a technology for night-time fruit picking using artificial illumination.	Accuracy of 91.7% for green grape detection.
Kangune et al. [37] (2019)	Grape ripeness estimation.	Classification accuracy of 79.5% between ripened and unripened grapes.
Aquino et al. [38] (2015)	Early yield prediction and flower estimation in vineyards.	Precision and recall were 83.4% and 85.0%.

Concerning the yield estimation of grapes at early growth stages, several works approached the problem with the implementation of grape flower detectors. Liu et al. [28] proposed a detection algorithm based on the extraction of texture information from images to access the location of visible grape flowers. Diago et al. [29] aimed to assess the flower number per inflorescence in grapevine. In this work, the grape bunches were placed over uniform backgrounds and were separated from each other by the application of a threshold. Palacios et al. [30] presented a DL-based approach where the region of interest containing aggregations of flowers was extracted using a semantic segmentation architecture. For the detection of grape bunches at more advanced growth stages, Pérez-Zavala et al. [31] clustered pixels into grape bunches using shape and texture information from images. This work used conventional approaches such as local binary pattern descriptors, but also machine-learning-based such as support vector machine classifiers. More focused on DL, Cecotti et al. [34] studied the best CNN architecture to deploy in agricultural environments. In this context, the authors tested several architectures for the detection of two types of grapes in images. The results showed that Resnet [39] was the best architecture, reaching an accuracy of 99.0%.

The proposed work relates to the state-of-the-art in the way that it uses DL techniques to detect grape bunches in images. However, this paper proposes the novelty of making publicly available (https://doi.org/10.5281/zenodo.5114142) a dataset (accessed on 21 August 2021) with 1929 vineyard images. The dataset contains images of grape bunches at different growth stages, with variations of illumination and different resolutions. This dataset is more realistic than the state-of-the-art because grapes are inserted on the canopy, so not very visible. The grape bunch annotations are also provided so that the scientific community can directly use the dataset for training DL models. In addition, this work benchmarks state-of-the-art DL models for grape bunch detection at different growth stages and deploys them in a low-cost and low-power embedded device. This requirement is important since our main goal was to have this solution running on our robotic platform (Figure 1).

Figure 1. Agricultural robot used to collect visual data with onboard cameras pointing to the canopy.

Thus, power consumption was taken into consideration so that the grape detection solution required as little power as possible, and robot autonomy was not highly affected by it. With this low-power solution, the robot would operate autonomously for a longer time without needing to charge. On the other hand, high-power solutions can decrease the autonomy time of the platforms, which is essential for long-term operations. In addition, since this was intended to be a solution that runs online on the robot, runtime requirements were important, so that the detection could be performed in a time-effective manner. In this way, mobile agricultural robots could perform tasks dependent on the grape detection algorithm in an online fashion. For example, SLAM algorithms that usually run at a high frequency could use the grape detections to build prescription maps that could be used for later processing and other agricultural applications. Furthermore, harvesting procedures require the correct location of the grape bunches in relation to the robotic arm that is moving. Thus, it was essential to have a high detection frequency to have a precise location of the grapes with reference to the arm gripper.

3. Deep-Learning-Based Grape Bunch Detection

The semantic perception of agricultural environments is increasingly important for the development of intelligent and autonomous robotic solutions capable of performing agricultural tasks. Robots should be able to understand their surroundings. For example, to develop autonomous fruit picking, robots should know how to distinguish fruits from the other natural agents and calculate their position with precision. Furthermore, Simultaneous Localization and Mapping (SLAM) approaches can use semantic information to build maps with meaningful information for agricultural analysis. In this context, this work focused on the detection of grape bunches at different growth stages in images. A monocular visual setup was mounted on an agricultural robot (Figure 1) pointing to the vineyard canopy during several trials. Using this, different states of the crop were captured along different stages of the year, so that the robot could detect grape bunches at different growth stages. From the data collection until the autonomous vineyard perception, three main steps were carried out as represented in Figure 2:

- Data collection: video data recorded by cameras mounted on top of an agricultural robotic platform; image extraction and storage from videos in order to build the input dataset;
- Dataset generation: image annotation by drawing bounding boxes around grape bunches in images considering two different classes; image augmentation by the application of several operations to the images and annotations to increase the dataset size and avoid overfitting when training the DL models; image splitting of the image size, to avoid losing resolution due to the image resize operation performed by the models to their kernel size (in this case, 300 $\times$ 300 px, with three channels);
- Model training and deployment: training and quantization of the DL models to deploy them in a low-cost and low-power embedded device with the main goal of performing time-effective grape bunch detection in images.

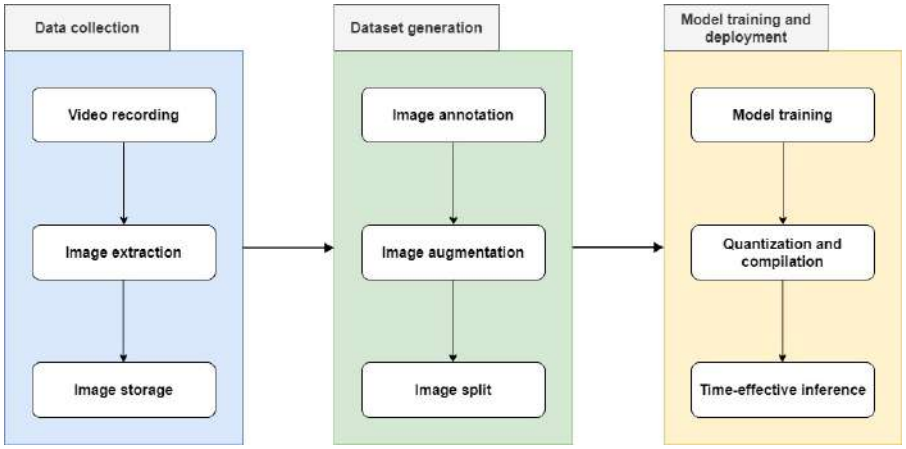


Figure 2. High-level workflow of the proposed system. The first step is data collection, where images are extracted and stored from videos recorded by onboard cameras. Then, the dataset is generated by the grape bunch annotation; data augmentation is performed to increase the dataset size; the images are split to improve the training performance. Finally, the models are trained, then quantized and compiled so that they can be deployed in a lightweight embedded device.

The following sections describe each step carefully.

3.1. Data Collection

This work proposes a novel dataset for grape bunch detection considering different growth stages. To build the dataset, several experiments were carried out considering different stages of the vineyard. To capture the data, the robot platform represented in Figure 1 was used.

This platform was equipped with two monocular RGB cameras mounted on the anthropomorphic manipulator pointing to the vineyard canopy during all the experiments. The cameras used to build the proposed dataset were the QG Raspberry Pi—Sony IMX477 and the OAK-D color camera.

To gather visual information and to be able to follow the evolution of the vineyard crop, the robot traveled the same path several times, in different stages of the vineyard. For this reason, the data collected presented variation of the illumination conditions, the visual perspective of the canopy, and the fruit growth stage. At an early stage, the robot captured the vineyard in a premature grape bunch stage, as represented in Figure 3a. At this stage, the grape bunches were captured right after the bloom. Thus, the grape berries had a light green color and a diameter of approximately 0.5 cm. In the next experiments, the grape bunches were captured at a medium growth stage, as is visible in Figure 3b. At this stage, the grape bunches were in an intermediate development stage, with a regular green color and a diameter of approximately 1.2 cm.

The data collection procedure was tackled in three different steps: video recording, image extraction, and image storage. Firstly, the robot recorded video sequences of the vineyard canopy in the ROSBag format. Then, to obtain the set of images for each experiment, the videos were sampled with a period of one second. This process had as the output a set of images per experiment that was then stored for the later processing. The data collection procedure generated 1929 original vineyard images considering different growth stages. It is worth noting that raw images were used, i.e., no calibration nor rectification were performed during the data collection procedure. Thus, it was expected that the models also received unrectified images during the inference procedure.

(**a**)

(**b**)

Figure 3. Two images of the proposed publicly available dataset considering (**a**) an early and (**b**) a medium grape bunch growth stage.

3.2. Dataset Generation

To use the data collected to train the DL models, a dataset generation procedure was carried out. Since we used a supervised learning approach, the models required the annotation of each input image. Each annotation consisted of a bounding box around each object that represented its area, position, and class. To annotate all the original 1929 collected images, the Pascal VOC format [40] was used due to its compatibility with the framework used for training (Tensorflow) and its simplicity. The annotation was carried out in a manual manner using two different software frameworks: CVAT [41], which is collaborative and thus allows the simultaneous annotation between multiple users, and LabelImg [42], which is an offline annotation tool. In the annotation procedure, two classes were considered for grape bunches, given that two different growth stages were captured during the experiments: `tiny-grape-bunch`, representing grape bunches at an early stage; and `medium-grape-bunch`, representing the same feature at a medium growth stage.

After having the entire dataset annotated, the amount of data used for training was increased using data augmentation. In past experiments, image augmentation was revealed

to be an essential step when compared to the use of only the original dataset images, due to the increase in the dataset size and variability and the reduction of overfitting during the models' training. For these reasons, this technique is widely used in the literature to improve models' performance [43]. When dealing with images, data augmentation consists of applying a set of operations to each image so that several images with slight modifications can be extracted from a single one. Thus, this approach generates synthetic data from the original data and can increase the variability of the datasets. In this work, five operations were applied to each original image:

1. Rotation;
2. Translation;
3. Scale;
4. Flipping;
5. Multiplication.

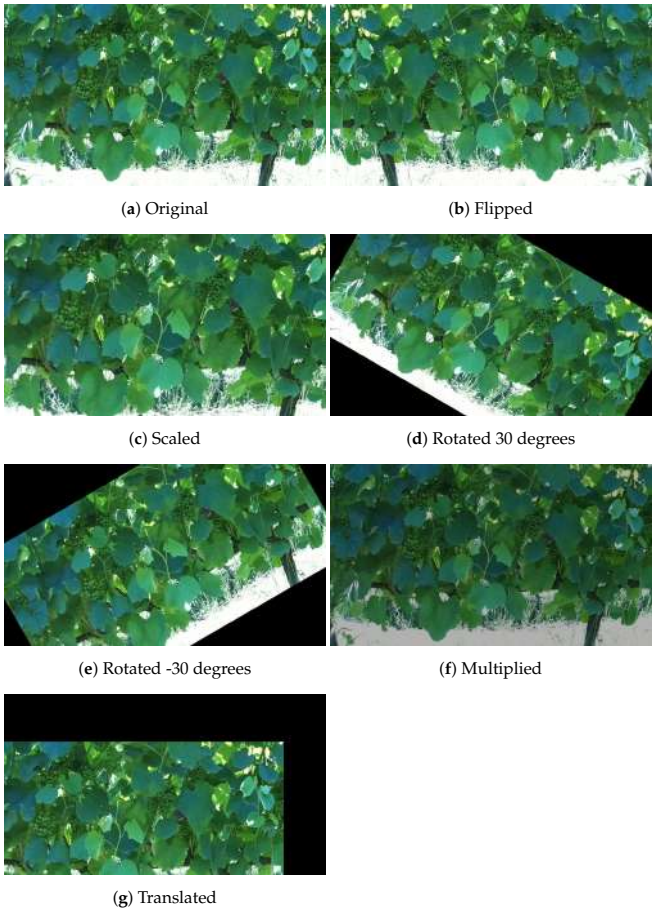
Since for the rotation operation two values were applied to each image, the dataset increased 7 times, for a total of 13,503 images. Table 2 details the augmentation operations performed.

Table 2. Description of the augmentation operations used to expand the original collection of data.

Augmentation Operation	Description
Rotation	Rotates the image by +30 and −30 degrees.
Translation	Translates the image by −30% to +30% on the *x*- and *y*-axis.
Scale	Scales the image to a value of 50 to 150% of their original size.
Flipping	Mirrors the image horizontally.
Multiply	Multiplies all pixels in an image with a random value sampled once per image, which can be used to make images lighter or darker.

In Figure 4 are represented the set of operations performed on an original image.

Finally, the last step of the dataset generation procedure was the image splitting. As referenced before, this work used lightweight models and deployed them in a low-cost and low-power embedded device, in an Edge-AI manner. Thus, the models trained can only process small images during the training procedure. In particular, the pretrained models SSD MobileNet-V1 [44] and SSD Inception-V2 [45] resized the input images to 300 × 300 px. In this case, if the dataset contained high-resolution images, many important data would be lost in this resizing process. To avoid this, in this work, the augmented dataset was extended by splitting the images into the input sizes of the trained CNN. From our past experience, this technique highly improves models' performance, especially when using high-resolution images. Without splitting these images, they would be resized to a lower resolution, and a significant amount of data would be lost in this process. On the contrary, if we split high-resolution images, no resize operation would performed by the DL model when performing image inference, and then all the data collected would be used. As represented in Figure 5, for an image with a resolution of 1920 × 1080 px, 40 other images were generated with a resolution of 300 × 300 px.



(**a**) Original (**b**) Flipped

(**c**) Scaled (**d**) Rotated 30 degrees

(**e**) Rotated -30 degrees (**f**) Multiplied

(**g**) Translated

Figure 4. Set of augmentation operations applied to a single image to extend the original dataset.

Table 3 contains the information about the number of annotated objects per class in the three different stages of the dataset: original images, augmented images, and split images.

Table 3. Number of annotated objects per class. The original dataset contains 1929 images with two different classes. To increase the dataset size, several augmentation operations were applied, increasing the number of images to 13,503. Finally, the images were split, and the final dataset was composed of 302,252 images.

Class	# of Objects	# of Objects in Augmented Images	# of Objects in Split Images
`tiny_grape_bunch`	2497	13,393	25,349
`medium_grape_bunch`	4292	25,189	51,272

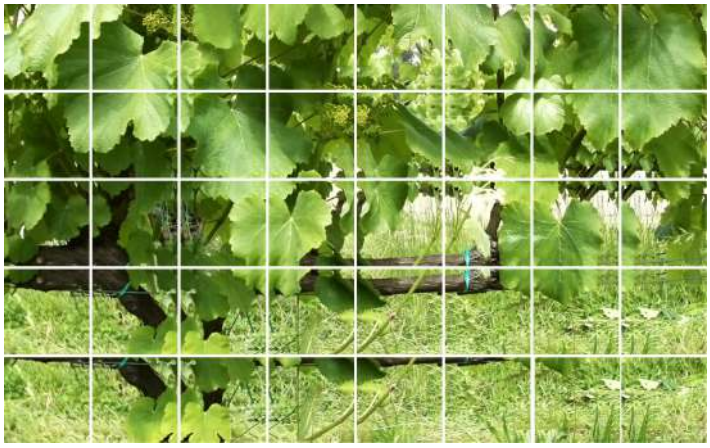


Figure 5. Split of a collected image to a 300 × 300 px resolution. The original image with a resolution of 1920 × 1080 px generated 40 split images considering an overlapping ratio of 20%.

In this process, an overlap of 20% between image patches was considered. By doing this, the models did not need to resize the input image, and no information was lost. Considering this operation, the dataset size increased to 302,252 images of 300 × 300 px.

It is worth noting that, during the two preprocessing operations where the dataset size was increased, the grape bunch annotations of both classes were automatically generated considering the original annotations. During the augmentation procedures, the operations were applied both to the images and the annotations. Similarly, the annotations were also split, together with the images.

3.3. Models' Training and Deployment

The final step to perform grape bunch detection considering different growth stages was the models' training and deployment. To achieve full compatibility with the hardware device used to deploy the models, only quantized models could be considered. Due to the higher number of compatible operations of Single-Shot Multibox (SSD) models [46] with the hardware device in comparison to other architectures, in this work, only this type of model was used. Thus, in this work, only SSD models were explored due to the constraints imposed by the hardware device used, Google's Tensor Processing Unit (TPU)—https://coral.ai/products/accelerator/ (accessed on 25 August 2021). Due to the same cause, the models were quantized to 8 bit precision. Google's Coral USB Accelerator provides an Edge TPU machine learning accelerator coprocessor. It is connected via USB to a host computer, allowing high-speed inference. This device is capable of performing four trillion operations per second (TOPS) and two TOPS per watt. It is connected to the host computed by USB requiring 5 V and 500 mA. To achieve the proposed goal—grape bunch detection considering different growth stages—two models were used and benchmarked: SSD MobileNet-V1 [44] and SSD Inception-V2 [45]. The models are briefly described bellow.

SSD MobileNet-V1:

This model is one of the most popular among the state-of-the-art models designed to run on low-power and low-cost embedded devices. One of its main novelties is the use of depthwise separable convolutions. This concept is achieved by factorization of standard convolutions into depthwise and 1 × 1 convolutions denominated pointwise convolutions. The outputs of both convolution types are then combined. The input of the CNN is a tensor with shape $D_f \times D_f \times M$, where D_f represents the input channel spatial

width and height, and M is the input depth. After the convolution, a feature map of shape $D_f \times D_f \times N$ is obtained, where N is the output depth. Thus, the model contained two hyperparameters that the user can tune in order to optimize the CNN performance. The first, width multiplier α, can be used to decrease the model size uniformly at each layer by a factor of α^2. This was performed by multiplying the number of both the input and output channels by this constant. The second hyperparameter, resolution multiplier ρ, was also used to reduce the computational cost of the model by a factor of ρ^2 by changing the input image resolution accordingly. Both parameters can be used simultaneously to achieve a balance between performance and inference time.

SSD Inception-V2:

Ioffe et al. [45] developed the original approach of Inception. The design of the model is supported by the fact that each object present in a different image can present different sizes. With this assumption, the choice if the CNN kernel size becomes difficult. To overcome this, the authors developed the model with three convolutional filter sizes of -1×1, 3×3, and 5×5. The results from the operations performed by the three filter were then concatenated, which resulted in the output of the network. SSD Inception-V2 was developed in order to reduce the computational complexity of the original version. This goal was achieved using factorization over the convolution operations. For example, a 5×5 convolution was factorized into two 3×3 convolutions, improving runtime performance. In the same way, a $m \times m$ convolution can be factorized into a combination of $1 \times m$ and $m \times 1$ convolutions.

To train and deploy these two models, they were downloaded from the Tensorflow model zoo (https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf1_detection_zoo.md, accessed on 24 August 2021). The versions considered were already pretrained on the COCO dataset [47]. To fine-tune the pretrained models, the Tensorflow [48] framework was used due to its compatibility with the TPU device used for inference. Tensorflow, a machine learning system that operates at a large scale and in heterogeneous environments, is one of the most used frameworks in the state-of-the-art. It is compatible with multiple hardware architectures such as CPUs, GPUs, and TPUs. In addition, this framework provides a version dedicated to on-device machine learning, Tensorflow Lite. This platform supports Android and iOS devices, embedded Linux, and microcontrollers. It uses hardware acceleration and model optimization to deploy high-performance models. In this work, Tensorflow Lite was used to deploy the models in the TPU embedded device.

Given all of the above, the models were trained considering the set of steps represented in Figure 6.

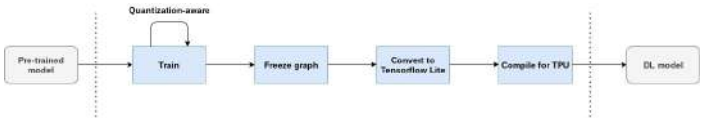
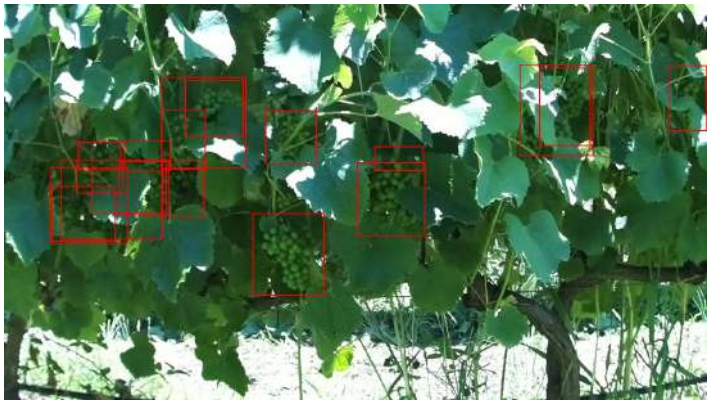


Figure 6. Steps to train and deploy a pretrained model into Google's Coral USB Accelerator (TPU).

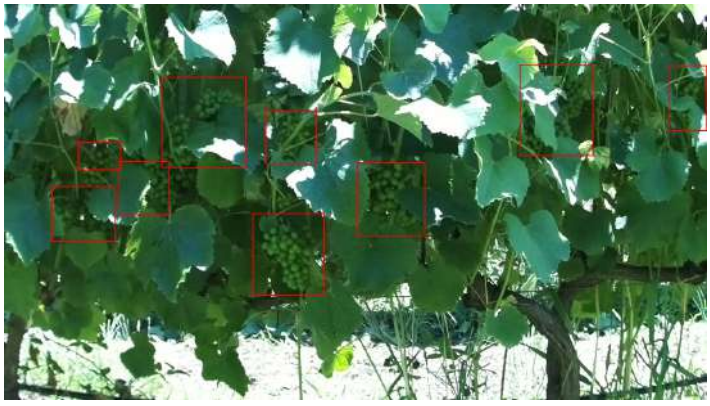
The input of the workflow was a pretrained model on the COCO dataset. The models were then fine-tuned using quantization-aware training [49] in order to convert them to 8 bit precision. This technique allowed reducing the models accuracy drop while converting from float to 8 bit precision. When the train was complete, the resultant binary files were combined in a single file containing only the useful information for inference. This procedure is called freezing, which produces a frozen graph. After this, since the hardware device used (TPU) supports only the lighter version of Tensorflow, the frozen graph was converted to Tensorflow Lite. Finally, the Tensorflow Lite model was compiled to the TPU. This compilation procedure was essential since it assigned operations either to the host CPU or to the TPU device. At the first point in the graph where an unsupported operation for the TPU occurs, the compiler separates the graph into two parts. The first is executed

on the TPU, while the second is assigned to the host CPU. It is worth noting that the higher the number of operations assigned to the TPU, the faster the inference procedure will be. Thus, it is essential that models with a high level of compatibility be used.

Finally, after having the models prepared, they were deployed on the TPU device to perform grape bunch detection. As referenced before, the original images on the dataset were split to match the models' input channels' size. Thus, to perform inference, we performed exactly the same operation to ensure that the data characteristics learned by the model matched the ones received for object detection. This means that each input image was split into a fixed number of overlapping tiles, and the model performed inference on each tile. After this, the results obtained for each tile were combined in order to compute the bounding box detections on the original image. Nonoverlapping bounding boxes were directly mapped onto the original image without any further operation. For the ones that overlapped between tiles, nonmaximum suppression [50] was used to suppress the overlapping bounding boxes for the same objects. Figure 7 shows the effects of this algorithm on the final inference result of the model SSD MobileNet-V1.



(**a**) Before nonmaximum suppression.



(**b**) After nonmaximum suppression.

Figure 7. Impact of nonmaximum suppression on the final inference result.

4. Results

This section describes the experiments performed to test the proposed approach. Firstly, the metrics used to evaluate the system are presented. Then, an evaluation is performed of the entire approach. Finally, an overall discussion of the obtained results is carried out.

4.1. Methodology

The evaluation performed used state-of-the-art metrics to evaluate the DL models deployed. In this work, seven different metrics were used: precision, recall, F1 score, precision × recall curve, AP, medium AP (mAP), and inference time. To calculate these metrics, the following set of concepts was used:

- Interception over Union (IoU): a measure based on the Jaccard index that calculates the overlap between two bounding boxes using the ground truth and the predicted bounding boxes;
- True Positive (TP): a valid detection, i.e., IoU $\geq$ *threshold*;
- False Positive (FP): an invalid detection, i.e., IoU $<$ *threshold*;
- False Negative (FN): a ground truth bounding box not detected.

Given all of the above, the metrics were calculated as follows:

- Precision: defined as the ability of a given model to detect only relevant objects, precision is calculated as the percentage of TP and is given by:

$$Precision = \frac{TP}{TP + FP}; \tag{1}$$

- Recall: defined as the ability of a given model to find all the ground truth bounding boxes, recall is calculated as the percentage of TP detected divided by all the ground truths and is given by:

$$Recall = \frac{TP}{TP + FN}; \tag{2}$$

- F1 score: defined as the harmonic mean between precision and recall, F1 score is given by:

$$2 \cdot \frac{precision \cdot recall}{precision + recall}; \tag{3}$$

- Precision × recall curve: a curve plotted for each object class that shows the tradeoff between precision and recall;
- AP: calculated as the area under the precision × recall curve. A high area represents both high precision and recall;
- mAP: calculated as the mean AP for all the object classes;
- Inference time: defined in this work as the amount of time that a model takes to process a tile or an image, on average.

In this work, the previously described metrics were used to evaluate both SSD MobileNet-V1 and SSD Inception-V2. In addition, the models were characterized by changing two parameters: the detection confidence and the IoU thresholds. Some visual results were also present to demonstrate the system robustness to occluded objects and variations in illumination conditions. To perform a fair evaluation of the DL models, the input dataset was divided into three groups: training, test, and evaluation. The larger one, the training set, was used to train the DL models. The test set was used to perform the evaluation of the models during the training by Tensorflow. The evaluation set was exclusively used to test the models by computing the metrics described above.

4.2. Evaluation

This work used quantized models to detect grape bunches at different growth stages. To evaluate these models, they were characterized by changing the confidence threshold

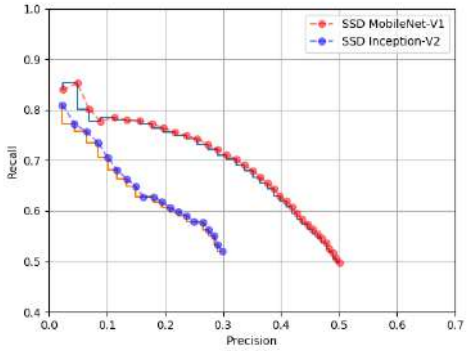
and the IoU parameter. Table 4 shows the detection performance of SSD MobileNet-V1 and SSD Inception-V2 for three values of the confidence: 30%, 50%, and 70%.

Table 4. Grape bunch detection performance considering an IoU of 50% and a variation of the confidence threshold for three different values.

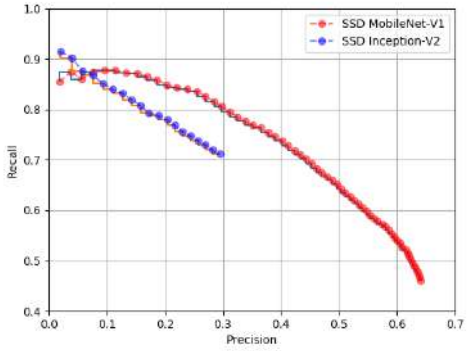
Model	Confidence (%)	Class	Precision (%)	Recall (%)	F1 Score (%)	AP (%)	mAP (%)
SSD MobileNet-V1	30	`tiny-grape-bunch`	17.38	**61.72**	27.12	**40.38**	**44.93**
		`medium-grape-bunch`	28.53	**66.44**	39.92	**49.48**	
SSD Inception-V2	30	`tiny-grape-bunch`	35.81	44.88	39.83	26.95	28.32
		`medium-grape-bunch`	64.62	37.59	47.53	29.68	
SSD MobileNet-V1	50	`tiny-grape-bunch`	49.28	50.44	**49.85**	36.29	42.47
		`medium-grape-bunch`	45.59	64.26	**53.34**	48.64	
SSD Inception-V2	50	`tiny-grape-bunch`	51.36	30.57	38.33	20.50	22.48
		`medium-grape-bunch`	70.86	29.90	42.06	24.45	
SSD MobileNet-V1	70	`tiny-grape-bunch`	**78.12**	11.85	20.58	9.86	22.45
		`medium-grape-bunch`	71.95	41.99	53.03	35.04	
SSD Inception-V2	70	`tiny-grape-bunch`	67.17	12.05	20.44	9.30	12.19
		`medium-grape-bunch`	**79.12**	17.46	28.60	15.08	

This table shows the effect of varying the confidence threshold. In particular, it is visible that when the confidence score increased, the precision also increased. This was due to the elimination of low-confidence detections. Thus, if we considered only the high-confidence detections, the model would be more suitable to detect only relevant objects, which would lead to an increase of the precision. On the contrary, when the confidence threshold increased, the number of TP decreased, which led to a decrease of the recall. Comparing both models, one can see that SSD Inception-V2 presented a higher precision than SSD MobileNet-V1 for all confidence scores, but a lower recall. This led to the conclusion that Inception presented a high rate of TP from all the detections, but a low rate of TP considering the ground truths. Overall, SSD MobileNet-V1 outperformed the Inception model, presenting a higher F1 score, AP, and mAP. This model achieved, as the best result, a mAP of 44.93% for a confidence score of 30%. Figure 8 shows the precision $\times$ recall curves for both models considering the two classes and a confidence score of 50%.

Once again, this figure shows that SSD MobileNet-V1 outperformed the Inception model. Comparing the models performance detecting objects of both classes, we verified that detecting grape bunches at an early stage (`tiny-grape-bunch`) was more challenging than at an intermediate growth stage (`medium-grape-bunch`). The first class represented smaller grape bunches, with a color and texture more similar to the surrounding foliage, which complicated their detection. SSD MobileNet-V1 presented a AP of 40.38% detecting grape bunches at an early growth stage and 49.48% at an intermediate growth stage. Finally, Figure 9 shows the impact of the confidence score on the detections for a single image.



(**a**) `tiny-grape-bunch`



(**b**) `medium-grape-bunch`

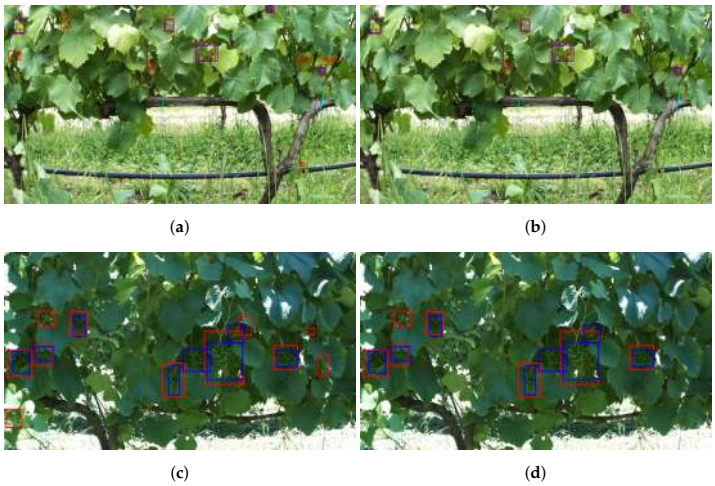
Figure 8. Precision × recall curves for both models and both classes considering a confidence score of 50% and an IoU of 50%.

One can verify that this parameter can be used to eliminate FPs that usually present low-confidence scores.

Table 5 presents the detection performance considering a variation of the IoU evaluation parameter.

This characterization was performed since different values for the overlap between detections and ground truths can give more information about the models' performance. For example, lower IoU values would consider detections that, besides not corresponding exactly to the location of the ground truths, represent annotated objects that were actually detected. To evaluate this, three values for the IoU parameter were considered: 20%, 40%, and 60%. Once again, one can verify that the SSD Inception-V2 model presented a higher precision. For an IoU value of 20%, this model had a precision of 92.57% detecting grape bunches at an intermediate growth stage. This is a satisfactory result since it means that 92.57% of the detections were TPs. On the other side, SSD MobileNet-V1 presented high recall levels. For an IoU of 20%, it achieved a recall of 87.01% for the class `medium-grape-bunch`. For higher IoU values, the performance of both models decreased.

This was expected since, for example, for an IoU of 60%, the detections that did not overlap more than 60% with the ground truths were considered as FPs, which led to a decrease in performance. Overall, the best result was achieved by SSD MobileNet-V1, which performed with a mAP of 66.96% for an IoU of 20%.



(**a**) (**b**)

(**c**) (**d**)

Figure 9. Impact of the confidence score on the final detection results. Blue bounding boxes represent the ground truth and the red ones the SSD MobileNet-V1 detections considering a confidence score of (**a**,**c**) 20% and (**b**,**d**) 50%.

Table 5. Grape bunch detection performance considering a confidence of 50% and a variation of the IoU threshold for three different values.

Model	IoU (%)	Class	Precision (%)	Recall (%)	F1 Score (%)	AP (%)	mAP (%)
SSD MobileNet-V1	20	`tiny-grape-bunch`	63.73	**65.22**	**64.47**	**56.87**	**66.96**
		`medium-grape-bunch`	61.72	**87.01**	**72.22**	**77.05**	
SSD Inception-V2	20	`tiny-grape-bunch`	**71.90**	42.80	53.66	36.42	37.22
		`medium-grape-bunch`	**92.57**	39.06	54.94	38.01	
SSD MobileNet-V1	40	`tiny-grape-bunch`	57.17	58.51	57.83	47.01	55.78
		`medium-grape-bunch`	54.96	77.47	64.30	64.55	
SSD Inception-V2	40	`tiny-grape-bunch`	64.25	38.24	47.95	30.50	31.98
		`medium-grape-bunch`	85.14	35.93	50.53	33.45	
SSD MobileNet-V1	60	`tiny-grape-bunch`	37.54	38.41	37.97	22.39	24.79
		`medium-grape-bunch`	32.17	45.34	37.64	27.19	
SSD Inception-V2	60	`tiny-grape-bunch`	33.38	19.87	24.91	8.72	9.79
		`medium-grape-bunch`	45.78	19.32	27.17	10.85	

As referenced before, the models were deployed in a low-cost and low-power embedded device, a TPU. It was intended that these models run in a time-effective manner to be integrated in more complex systems such as harvesting and spraying procedures. Thus, evaluating the runtime performance of both models was important in the context of this work. Table 6 shows the inference time results for both models.

Table 6. Runtime performance of both models.

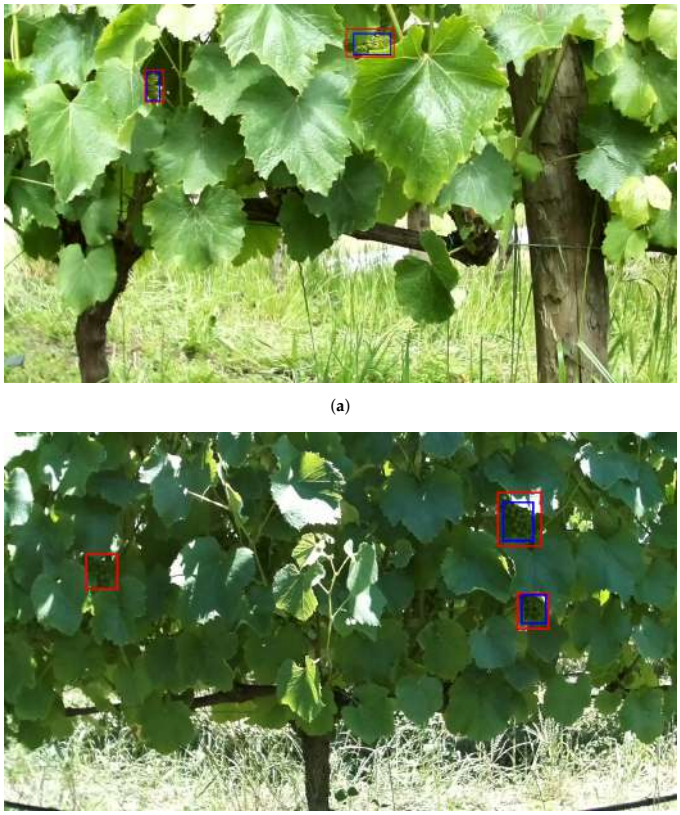
Model	Inference Time per Tile (ms)	Inference Time per Image (ms)
SSD MobileNet-V1	6.29	93.12
SSD Inception-V2	26.07	385.69

In this table, the performances per tile and per image are both described. The inference time per tile was also considered in this evaluation since it represents the time that each model would take to process an entire image if the input images were not split. The inference times were measured for each evaluation image, and the final value considered was the average for all images. The results showed that the SSD MobileNet-V1 was more than four-times faster than the Inception model. This was due to the simpler architecture of MobileNets in relation to Inception, and the higher compatibility of MobileNet compared with Inception. This model can process a single tile in 6.29 ms and an entire split image in 93.12 ms. This proved both the high performance of the model, but also that the TPU hardware device used was capable of deploying models in a very efficient way, even considering low-power costs.

This approach was intended to be robust to different light conditions since the robot would operate at different times of the day and stages of the year. Because of this reason, the built dataset considered several light conditions, and the models were trained to be robust to them. Furthermore, the dataset considered occluded grape bunches so that the models could also detect not fully visible grape bunches. To demonstrate these challenging conditions present in the proposed dataset, Figures 10 and 11 present an overview of the performance of SSD MobileNet-V1 considering occlusions in the grape bunches and variation in the illumination conditions.

For the models to be able to accommodate these conditions, the annotation procedure was crucial. In this process, the decision to consider occluded objects was made. Several times, the annotation of an occluded object was complex since there was the need to consider parts of other objects inside the bounding box corresponding to the occluded object. Figure 10 shows that occluded objects were taken into consideration during the annotation procedure and that the models were able to identify these objects in the images. Regarding the variations of the illumination conditions, one of the key steps to accomplish this goal was the capture of visual data during different days and stages of the year. To build the proposed dataset, the robot represented in Figure 1 was taken four times to the vineyard in order to capture the crop state in different conditions. The visit dates were 11 May 2021, 27 May 2021, 23 June 2021, and 26 July 2021.

On each day, images were recorded both in the morning and during the evening to account for multiple light conditions. In May, grape bunches at an early growth stage were captured, while in June and July, the intermediate growth stage was present in the vineyard. After recording all these data, the annotation process was once again essential since during the annotation, the objects were present under different light conditions. Figure 11 shows the different levels of illumination captured during the field visits performed. This figure proves that the models were able to detect grape bunches at different growth stages in these conditions.



(**a**)

(**b**)

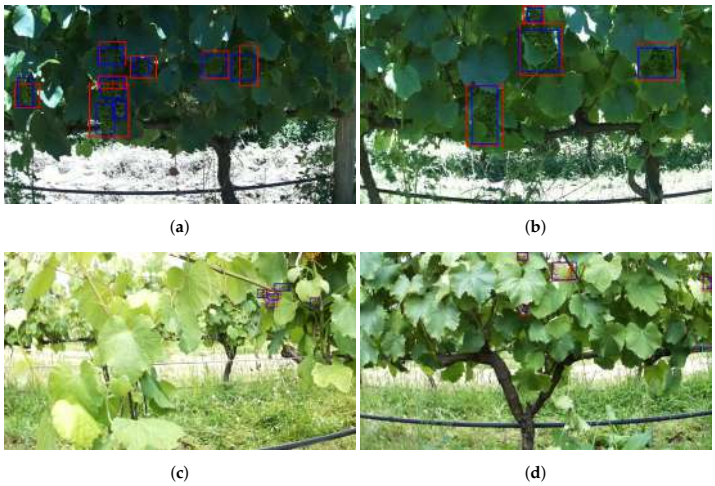
Figure 10. Detection of grape bunches (**a**) in early, and (**b**) intermediate growth stages considering occlusions caused by the dense foliage present in the vineyard. Blue bounding boxes represent the ground truth and the red ones the detections.

4.3. Discussion

This work proposed a novel dataset for grape bunch detection at different growth stages. Two state-of-the-art models were used to perform this detection. Due to the requirement of a time-effective, low-power, and low-cost detection, this work used lightweight models that were quantized to be deployed in an embedded device. Quantization was used to reduce the size of the DL models and improve runtime performance by taking advantage of high throughput integer instructions. However, quantization can reduce the detection performance of DL models. Wu et al. [51] showed that the error rates increase when the model size decreases by quantization. In this work, this decrease in detection performance was accepted due to the high gain in runtime performance. In comparison with state-of-the-art works such as the one proposed by Palacios et al. [30], which detected grape flower at the bloom with an F1 score of 73.0%, this work presented a lower detection performance. On the other hand, the tradeoff between detection performance and runtime performance was extremely satisfactory since, especially for SSD MobileNet-V1, the model could perform with a mAP up to 66.96%, performing the detection at a rate higher than

10 Hz per image. In addition, the models were able to detect grape bunches at different stages, considering occlusions and variations in illumination conditions. Since the proposed dataset is publicly available, we believe that it has potential to be used in the future by the scientific community to train more complex and nonquantized DL models in order to achieve higher detection performances for applications without runtime restrictions. Furthermore, the proposed system can be adopted in future works and applications since it is cost-effective, portable, low-power, and independent of light conditions. The solution is modular and can be placed in any robotic platform, meaning that the price of the module is completely independent of the platform where it is placed. For applications that require higher levels of detection precision and that are not dependent on a time-effective solution, more complex models can be trained with the proposed dataset. Some works may also propose new DL-based architectures or modify state-of-the-art models to better suit the application purposes. For example, Taheri-Garavand et al. [11] proposed a modification to the VGG16 model to identify chickpea varieties by using seed images in the visible spectrum, and Nasiri et al. [12] proposed a similar approach to automate grapevine cultivar identification.

One of the main goals of this work was to achieve a low-power solution. The device used operates at high inference rate with a requirement of 5 V and 500 mA. This result was aligned with the state-of-the-art works that proposed advanced solutions for object detection using accelerator devices. Kaarmukilan et al. [52] used Movidius Neural Compute Stick 2, which similar to the TPU used in this work, is connected to the host device by USB and is capable of 4 TOPS with a 1.5 W power consumption. Dinelli et al. [53] compared several field-programmable gate array families by Xilinx and Intel for object detection. From all the evaluated devices, the authors achieved a minimum power consumption of 0.969 W and a maximum power consumption of 4.010 W.



(**a**) (**b**)

(**c**) (**d**)

Figure 11. Demonstration of the differences in illumination captured by the proposed dataset and the corresponding ability of the models to deal with it. Each image (**a**–**d**) represents a different light condition. Blue bounding boxes represent the ground truth and the red ones the detections.

5. Conclusions

This work approached the problem of detecting grape bunches at different growth stages by cameras mounted onboard mobile robots. A novel dataset with 1929 images and respective annotations was proposed considering different stages of grape bunches,

constructed by visiting a vineyard four different times to record the data. To achieve time-effective, low-cost, and low-power grape bunch detection, two models were trained, quantized, and deployed in an embedded device. The results showed that the tradeoff between detection and runtime performance was satisfactory. SSD MobileNet-V1 achieved the best results, with a maximum detection performance of 66.96% and a runtime average cycle of 6.29 ms and 93.12 ms per tile and image, respectively.

In future work, we would like to extend the dataset to consider more grape bunch growth stages. Since our robotic platform was intended to run also at night, we will also consider including vineyard images captured at night using artificial illumination. Furthermore, we would like to test the proposed system in vineyards that were not considered in this work, to evaluate if the models are robust to different scenarios. Additionally, the proposed dataset could be extended to consider grape bunches of these different vineyards.

Author Contributions: Conceptualization, A.S.A., S.A.M. and F.N.d.S.; methodology, A.S.A., S.A.M., F.N.d.S., T.P., L.C. and J.V.; software, A.S.A., S.A.M. and L.C.; validation, F.N.d.S., T.P., J.V., R.M. and J.B.-C.; formal analysis, F.N.d.S., T.P., J.V., R.M. and J.B.-C.; investigation, A.S.A., S.A.M., F.N.d.S. and J.V.; resources, F.N.d.S.; data curation, A.S.A. and S.A.M.; writing—original draft preparation, A.S.A., S.A.M. and F.N.d.S.; writing—review and editing, F.N.d.S., L.C., T.P., J.V., R.M. and J.B.-C.; visualization, A.S.A., S.A.M., F.N.d.S., L.C., T.P., J.V., R.M. and J.B.-C.; supervision, F.N.d.S., J.V., R.M. and J.B.-C.; project administration, F.N.d.S.; funding acquisition, F.N.d.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by "Metbots-Metabolomic Robots with Self-learning Artificial Intelligence for Precision Agriculture" Reference PTDC/EEI-ROB/31124/2017 and FEDER funds through the COMPETE 2020 Programme under Project Number POCI-01-0145-FEDER-031124.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are openly available in https://doi.org/10.5281/zenodo.5114142 (accessed on 1 August 2021).

Acknowledgments: André Silva Pinto de Aguiar thanks the FCT—Foundation for Science and Technology, Portugal, for the Ph.D. Grant DFA/BD/5318/2020. Sandro Augusto Magalhães thanks the FCT—Foundation for Science and Technology, Portugal, for the Ph.D Grant SFRH/BD/147117/2019. Martins RC acknowledges the Fundação para a Ciência e Tecnologia (FCT) research contract grant (CEEIND/017801/2018).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Duckett, T.; Pearson, S.; Blackmore, S.; Grieve, B.; Chen, W.H.; Cielniak, G.; Cleaversmith, J.; Dai, J.; Davis, S.; Fox, C.; et al. Agricultural Robotics: The Future of Robotic Agriculture. *arXiv* **2018**, arXiv:cs.RO/1806.06762.
2. Billingsley, J.; Visala, A.; Dunn, M. Robotics in Agriculture and Forestry. In *Springer Handbook of Robotics*; Siciliano, B., Khatib, O., Eds.; Springer: Berlin/Heidelberg, Germany, 2008; pp. 1065–1077._47. [CrossRef]
3. Andresen, T.; de Aguiar, F.B.; Curado, M.J. The Alto Douro Wine Region greenway. . *Landsc. Urban Plan.* **2004**, *68*, 289–303. [CrossRef]
4. Patrício, D.I.; Rieder, R. Computer vision and artificial intelligence in precision agriculture for grain crops: A systematic review. *Comput. Electron. Agric.* **2018**, *153*, 69–81. [CrossRef]
5. Bargoti, S.; Underwood, J.P. Image segmentation for fruit detection and yield estimation in apple orchards. *J. Field Robot.* **2017**, *34*, 1039–1060. [CrossRef]
6. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef]
7. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Netw.* **2015**, *61*, 85–117. [CrossRef]
8. Santos, L.; Santos, F.N.; Oliveira, P.M.; Shinde, P. Deep Learning Applications in Agriculture: A Short Review. In Proceedings of the Robot 2019: Fourth Iberian Robotics Conference, Porto, Portugal, 20–22 November 2020; Silva, M.F., Luís Lima, J., Reis, L.P., Sanfeliu, A., Tardioli, D., Eds.; Springer: Cham, Switzerland, 2020; pp. 139–151.
9. Kamilaris, A.; Prenafeta-Boldú, F.X. Deep learning in agriculture: A survey. *Comput. Electron. Agric.* **2018**, *147*, 70–90. [CrossRef]
10. Kamilaris, A.; Prenafeta-Boldú, F.X. A review of the use of convolutional neural networks in agriculture. *J. Agric. Sci.* **2018**, *156*, 312–322. [CrossRef]

11. Taheri-Garavand, A.; Nasiri, A.; Fanourakis, D.; Fatahi, S.; Omid, M.; Nikoloudakis, N. Automated In Situ Seed Variety Identification via Deep Learning: A Case Study in Chickpea. *Plants* **2021**, *10*, 1406. [CrossRef]
12. Nasiri, A.; Taheri-Garavand, A.; Fanourakis, D.; Zhang, Y.D.; Nikoloudakis, N. Automated Grapevine Cultivar Identification via Leaf Imaging and Deep Convolutional Neural Networks: A Proof-of-Concept Study Employing Primary Iranian Varieties. *Plants* **2021**, *10*, 1628. [CrossRef] [PubMed]
13. Wang, X.; Han, Y.; Wang, C.; Zhao, Q.; Chen, X.; Chen, M. In-Edge AI: Intelligentizing Mobile Edge Computing, Caching and Communication by Federated Learning. *IEEE Netw.* **2019**, *33*, 156–165. [CrossRef]
14. Pinto de Aguiar, A.S.; Neves dos Santos, F.B.; Feliz dos Santos, L.C.; de Jesus Filipe, V.M.; Miranda de Sousa, A.J. Vineyard trunk detection using deep learning—An experimental device benchmark. *Comput. Electron. Agric.* **2020**, *175*, 105535. [CrossRef]
15. Aguiar, A.S.; Santos, F.N.D.; De Sousa, A.J.M.; Oliveira, P.M.; Santos, L.C. Visual Trunk Detection Using Transfer Learning and a Deep Learning-Based Coprocessor. *IEEE Access* **2020**, *8*, 77308–77320. [CrossRef]
16. Aguiar, A.S.; Monteiro, N.N.; Santos, F.N.d.; Solteiro Pires, E.J.; Silva, D.; Sousa, A.J.; Boaventura-Cunha, J. Bringing Semantics to the Vineyard: An Approach on Deep Learning-Based Vine Trunk Detection. *Agriculture* **2021**, *11*, 131. [CrossRef]
17. Magalhães, S.A.; Castro, L.; Moreira, G.; dos Santos, F.N.; Cunha, M.; Dias, J.; Moreira, A.P. Evaluating the Single-Shot MultiBox Detector and YOLO Deep Learning Models for the Detection of Tomatoes in a Greenhouse. *Sensors* **2021**, *21*, 3569. [CrossRef] [PubMed]
18. Fanourakis, D.; Kazakos, F.; Nektarios, P.A. Allometric Individual Leaf Area Estimation in Chrysanthemum. *Agronomy* **2021**, *11*, 795. [CrossRef]
19. Taheri-Garavand, A.; Nejad, A.R.; Fanourakis, D.; Fatahi, S.; Majd, M.A. Employment of artificial neural networks for non-invasive estimation of leaf water status using color features: A case study in *Spathiphyllum wallisii*. *Acta Physiol. Plant.* **2021**, *43*, 78. [CrossRef]
20. Dias, P.A.; Tabb, A.; Medeiros, H. Apple flower detection using deep convolutional networks. *Comput. Ind.* **2018**, *99*, 17–28. [CrossRef]
21. Koirala, A.; Walsh, K.B.; Wang, Z.X.; McCarthy, C. Deep learning for real-time fruit detection and orchard fruit load estimation: Benchmarking of 'MangoYOLO'. *Precis. Agric.* **2019**, *20*, 1–29. [CrossRef]
22. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 26 June–1 July 2016.
23. Zheng, Y.Y.; Kong, J.L.; Jin, X.B.; Wang, X.Y.; Su, T.L.; Zuo, M. CropDeep: The Crop Vision Dataset for Deep-Learning-Based Classification and Detection in Precision Agriculture. *Sensors* **2019**, *19*, 1058. [CrossRef]
24. Bargoti, S.; Underwood, J. Deep fruit detection in orchards. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 3626–3633. [CrossRef]
25. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv* **2015**, arXiv:cs.CV/1506.01497.
26. Sa, I.; Ge, Z.; Dayoub, F.; Upcroft, B.; Perez, T.; McCool, C. DeepFruits: A Fruit Detection System Using Deep Neural Networks. *Sensors* **2016**, *16*, 1222. [CrossRef]
27. Kirk, R.; Cielniak, G.; Mangan, M. L*a*b*Fruits: A Rapid and Robust Outdoor Fruit Detection System Combining Bio-Inspired Features with One-Stage Deep Learning Networks. *Sensors* **2020**, *20*, 275. [CrossRef]
28. Liu, S.; Li, X.; Wu, H.; Xin, B.; Tang, J.; Petrie, P.R.; Whitty, M. A robust automated flower estimation system for grape vines. *Biosyst. Eng.* **2018**, *172*, 110–123. [CrossRef]
29. Diago, M.P.; Sanz-Garcia, A.; Millan, B.; Blasco, J.; Tardaguila, J. Assessment of flower number per inflorescence in grapevine by image analysis under field conditions. *J. Sci. Food Agric.* **2014**, *94*, 1981–1987. [CrossRef] [PubMed]
30. Palacios, F.; Bueno, G.; Salido, J.; Diago, M.P.; Hernández, I.; Tardaguila, J. Automated grapevine flower detection and quantification method based on computer vision and deep learning from on-the-go imaging using a mobile sensing platform under field conditions. *Comput. Electron. Agric.* **2020**, *178*, 105796. [CrossRef]
31. Pérez-Zavala, R.; Torres-Torriti, M.; Cheein, F.A.; Troni, G. A pattern recognition strategy for visual grape bunch detection in vineyards. *Comput. Electron. Agric.* **2018**, *151*, 136–149. [CrossRef]
32. Reis, M.; Morais, R.; Peres, E.; Pereira, C.; Contente, O.; Soares, S.; Valente, A.; Baptista, J.; Ferreira, P.; Bulas Cruz, J. Automatic detection of bunches of grapes in natural environment from color images. *J. Appl. Log.* **2012**, *10*, 285–290. [CrossRef]
33. Liu, S.; Whitty, M. Automatic grape bunch detection in vineyards with an SVM classifier. *J. Appl. Log.* **2015**, *13*, 643–653. [CrossRef]
34. Cecotti, H.; Rivera, A.; Farhadloo, M.; Pedroza, M.A. Grape detection with convolutional neural networks. *Expert Syst. Appl.* **2020**, *159*, 113588. [CrossRef]
35. Santos, T.T.; de Souza, L.L.; dos Santos, A.A.; Avila, S. Grape detection, segmentation, and tracking using deep neural networks and three-dimensional association. *Comput. Electron. Agric.* **2020**, *170*, 105247. [CrossRef]
36. Xiong, J.; Liu, Z.; Lin, R.; Bu, R.; He, Z.; Yang, Z.; Liang, C. Green Grape Detection and Picking-Point Calculation in a Night-Time Natural Environment Using a Charge-Coupled Device (CCD) Vision Sensor with Artificial Illumination. *Sensors* **2018**, *18*, 969. [CrossRef] [PubMed]

37. Kangune, K.; Kulkarni, V.; Kosamkar, P. Grapes Ripeness Estimation using Convolutional Neural network and Support Vector Machine. In Proceedings of the 2019 Global Conference for Advancement in Technology (GCAT), Bangalore, India, 18–20 October 2019; pp. 1–5. [CrossRef]
38. Aquino, A.; Millan, B.; Gutiérrez, S.; Tardáguila, J. Grapevine flower estimation by applying artificial vision techniques on images with uncontrolled scene and multi-model analysis. *Comput. Electron. Agric.* **2015**, *119*, 92–104. [CrossRef]
39. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 26 June–1 July 2016; pp. 770–778.
40. Everingham, M.; Van Gool, L.; Williams, C.K.; Winn, J.; Zisserman, A. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338. [CrossRef]
41. Computer Vision Annotation Tool (CVAT). Available online: https://github.com/openvinotoolkit/cvat (accessed on 6 August 2021).
42. Tzutalin. LabelImg. Git Code. 2015. Available online: https://github.com/tzutalin/labelImg (accessed on 6 August 2021).
43. Shorten, C.; Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. *J. Big Data* **2019**, *6*, *1-48* . [CrossRef]
44. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
45. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the Inception Architecture for Computer Vision. *arXiv* **2015**, arXiv:cs.CV/1512.00567.
46. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 21–37.
47. Lin, T.Y.; Maire, M.; Belongie, S.J.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common Objects in Context. In Proceedings of the ECCV 2014, Zurich, Switzerland, 6–12 September 2014.
48. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. Tensorflow: A system for large-scale machine learning. In Proceedings of the 12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16), Savannah, GA, USA, 2–4 November 2016; pp. 265–283.
49. Tucker, G.; Wu, M.; Sun, M.; Panchapagesan, S.; Fu, G.; Vitaladevuni, S. Model Compression Applied to Small-Footprint Keyword Spotting. In Proceedings of the Interspeech 2016, San Francisco, CA, USA, 8–12 September 2016; pp. 1878–1882.
50. Neubeck, A.; Van Gool, L. Efficient non-maximum suppression. In Proceedings of the 18th International Conference on Pattern Recognition (ICPR'06), Hong Kong, China, 20–24 August 2006; Volume 3, pp. 850–855.
51. Wu, H.; Judd, P.; Zhang, X.; Isaev, M.; Micikevicius, P. Integer quantization for deep learning inference: Principles and empirical evaluation. *arXiv* **2020**, arXiv:2004.09602.
52. Kaarmukilan, S.; Hazarika, A.; Poddar, S.; Rahaman, H. An Accelerated Prototype with Movidius Neural Compute Stick for Real-Time Object Detection. In Proceedings of the 2020 International Symposium on Devices, Circuits and Systems (ISDCS), Howrah, India, 4–6 March 2020; pp. 1–5.
53. Dinelli, G.; Meoni, G.; Rapuano, E.; Benelli, G.; Fanucci, L. An FPGA-Based Hardware Accelerator for CNNs Using On-Chip Memories Only: Design and Benchmarking with Intel Movidius Neural Compute Stick. *Int. J. Reconfig. Comput.* **2019**, *2019*, 1–13. [CrossRef]

4.5 Final remarks

The four articles presented in this chapter deeply approached the semantic perception problem using DL models. The problem of deploying the models in dedicated hardware was analysed, and two devices were benchmarked. The dataset was gradually increased and was achieved a solution that can detect vine trunks and grape bunches in different growth stages recurring to a dedicated device. In the future, this approach can be extended to other agricultural or forest environments. This will allow the dataset to increase in size and variability, being suitable for the use of more researchers and in different applications. Also, it will allow robots to create semantic maps of different crops and environments. In addition, different growth stages can be considered. In particular, an early stage such as at the bloom so that specialists can analyse and perform, for example, yield estimation.

5 VineSLAM: localization and mapping algorithm for agricultural robots

One of the main goals of this thesis was to explore a more advanced SLAM algorithm for agricultural robots that considers the characteristics of the crops and that provides useful information for posterior analysis by agronomists. Agriculture brings several challenges to robotic-based approaches due to the characteristic high environments' extensions, terrain irregularities, harsh inclinations and elevate altitude differentials. In this context, VineSLAM[1] came out to tackle these challenges. This algorithm proposes main contributions such as: a 6-DoF localization system based on a PF that uses different sensors to estimate the robot's pose; a metric mapping procedure that extracts features from LiDAR sensors and maps them; a semantic mapping approach that uses the semantic perception system described in Chapter 4 and maps grape bunches and vine trunks; a PF pose refinement using features extracted from point clouds generated by stereo cameras; and, finally, a topological mapping algorithm that uses a graph-based structure to manage the memory resources and enable large-scale and long-term navigation.

This line of work is in the origin of four different articles. Three of them are already published, and one is submitted in peer review process. Merging all of these works enables to reach a solution with capacity to estimate the 3D robot pose in challenging agricultural environments, providing a multi-layer mapping with efficient memory management. This enables the deployment of autonomous robots in agriculture

[1] https://gitlab.inesctec.pt/agrob/vineslam_stack/vineslam

with the ability to navigate in large-scale environments and long-term periods. Moreover, with this autonomous navigation capability, robots can contribute in precision agriculture operations such as planting, harvesting, pruning, and spraying.

5.1 Localization and Mapping on Agriculture Based on Point-Feature Extraction and Semiplanes Segmentation From 3D LiDAR Data

In this thesis, the first step to accomplish a localization and mapping system for agricultural robots is made through the use of a 3D LiDAR sensor. This work was published in the Field Robotics section of the Frontiers in Robotics and AI journal, and is entitled *Localization and Mapping on Agriculture Based on Point-Feature Extraction and Semiplanes Segmentation From 3D LiDAR Data* (Aguiar et al., 2022a). The approach is designed in three main steps: feature extraction, localization, and mapping. In the first step, two types of features are extracted. Point features that are searched in sharp edges and planar surfaces and polygon-based features (or semiplanes) that are searched on the ground and in the right and left sides of the robot. The polygon features are extracted through an innovative approach using a RANSAC algorithm to find the equation of the plane and then a Convex Hull to determine the boundaries of the polygon. Each polygon is represented by its normal vector, its centroid and the set of extrema points that limit it. This three-dimensional representation brings challenges both for the localization and mapping procedures. This article describes how this representation is incorporated in the PF for localization and what is the strategy to solve the feature association problem during mapping. Also, this work shows the modularity of the PF due to its capacity to support different types of features. The theoretical polygon-based approach is validated using a simulated environment with three detectable semiplanes. The performance of the entire localization algorithm is benchmarked against a state-of-the-art algorithm using data collected in real-world agricultural environments.

frontiers in Robotics and AI

ORIGINAL RESEARCH
published: 28 January 2022
doi: 10.3389/frobt.2022.832165

Localization and Mapping on Agriculture Based on Point-Feature Extraction and Semiplanes Segmentation From 3D LiDAR Data

André Silva Aguiar[1,2]*, *Filipe Neves dos Santos*[1], *Héber Sobreira*[1], *José Boaventura-Cunha*[1,2] *and Armando Jorge Sousa*[1,3]

[1]INESC TEC—INESC Technology and Science, Porto, Portugal, [2]School of Science and Technology, University of Trás-os-Montes e Alto Douro, Vila Real, Portugal, [3]FEUP, University of Porto, Porto, Portugal

Developing ground robots for agriculture is a demanding task. Robots should be capable of performing tasks like spraying, harvesting, or monitoring. However, the absence of structure in the agricultural scenes challenges the implementation of localization and mapping algorithms. Thus, the research and development of localization techniques are essential to boost agricultural robotics. To address this issue, we propose an algorithm called VineSLAM suitable for localization and mapping in agriculture. This approach uses both point- and semiplane-features extracted from 3D LiDAR data to map the environment and localize the robot using a novel Particle Filter that considers both feature modalities. The numeric stability of the algorithm was tested using simulated data. The proposed methodology proved to be suitable to localize a robot using only three orthogonal semiplanes. Moreover, the entire VineSLAM pipeline was compared against a state-of-the-art approach considering three real-world experiments in a woody-crop vineyard. Results show that our approach can localize the robot with precision even in long and symmetric vineyard corridors outperforming the state-of-the-art algorithm in this context.

Keywords: localization, mapping, 3D LIDAR, semiplanes, agriculture

OPEN ACCESS

Edited by:
Yan Zhuang,
Dalian University of Technology, China

Reviewed by:
Hui Zhou,
Nanjing University of Science and Technology, China
Yecheng Lyu,
Volvo Car Technology USA, United States

***Correspondence:**
André Silva Aguiar
andre.s.aguiar@inesctec.pt

Specialty section:
This article was submitted to Field Robotics, a section of the journal Frontiers in Robotics and AI

Received: *09 December 2021*
Accepted: *10 January 2022*
Published: *28 January 2022*

Citation:
Aguiar AS, Neves dos Santos F, Sobreira H, Boaventura-Cunha J and Sousa AJ (2022) Localization and Mapping on Agriculture Based on Point-Feature Extraction and Semiplanes Segmentation From 3D LiDAR Data. Front. Robot. AI 9:832165. doi: 10.3389/frobt.2022.832165

1 INTRODUCTION

The development of autonomous robots in agriculture is a challenging and active research topic (Emmi et al., 2014). To implement such systems, the autonomous navigation issue must be solved, i.e., robots should be capable of driving autonomously within multiple environments Shalal et al. (2013). Consequently, autonomous robotic platforms should be endowed with robust localization systems, that allow recovering their absolute pose in the agricultural environment (Vougioukas, 2019). Simultaneous Localization and Mapping (SLAM) allows calculating the travelled trajectory while mapping the environment simultaneously (Bailey and Durrant-Whyte, 2006; Durrant-Whyte and Bailey, 2006). In agriculture, the implementation of SLAM is particularly important since it leads to creating maps that farmers can use in various tasks. When robots have this ability, they can perform several autonomous operations such as precision agriculture (application of fertilizers, nutrients and water), plant protection, harvesting, monitoring, and planting (Bergerman et al., 2016; Roldán et al., 2018; Pinto de Aguiar et al., 2020). Even so, the implementation of SLAM in outdoor agricultural environments can be challenging since the characteristics of illumination and terrain

TABLE 1 | Summary of the current state-of-the-art on plane-based localization and mapping.

References	Application	Feature extraction	Mapping
Taguchi et al. (2013)	3D reconstruction of indoor spaces using hand-held sensors.	Points: image feature detector; planes: RANSAC algorithm.	RANSAC-based registration algorithm.
Yang et al. (2016)	Localization and mapping of low-texture indoor environments.	CNN-based plane detection.	Point and semiplane registration.
Viejo and Cazorla (2007).	Egomotion estimation in indoor and outdoor semi-structured environments.	Plane extraction using a PCA technique.	Iterative Closest Point (ICP) algorithm used for plane registration.
Grant et al. (2019)	Velodyne point-plane SLAM in challenging indoor and outdoor environments.	Find groups of points that arise from planar surfaces in a scan-line basis (Grant et al., 2013).	Registration with a developed algorithm: Iterative Closest Point Plus Plane Optimization (IC3PO).
Zhang et al. (2019b)	SLAM in indoor environments.	Plane segmentation using a connected component-based approach.	Points added by triangulation and observed planes added if no correspondence is found.
Kaess (2015)	Mapping of indoor environments using hand-held sensors.	Plane segmentation from point cloud data.	Infinite plane representation and mapping.
Weingarten and Siegwart (2006)	Localization and mapping of indoor environments.	Divide and conquer approach: best-fitting planes from small regions (Weingarten et al., 2003).	3D map builds using an Extended Kalman Filter (EKF).
Gee et al. (2008)	SLAM in indoor environments using hand-held sensors.	Planar surfaces extracted using RANSAC.	Registering based on similarity test.
Elghor et al. (2015)	3D reconstruction of indoor environments.	Planes extracted using RANSAC.	Planes registered and fused using a weight function.
Lenac et al. (2017)	Planar representation of indoor and outdoor environments.	Plane segments extracted by a 2D Delaunay triangulation.	Registration using the overlapping between planes.
Ulas and Temeltas (2012)	Outdoor SLAM.	Planes extracted using RANSAC.	Planes matched and registered using: orientation, translation and closeness.
Our approach	Autonomous navigation in outdoor agricultural environments.	Point-wise and three stage semiplane-wise feature extraction.	Point registration and semiplane matching and merging algorithm for registering and mapping.

irregularities can difficult the perception stages, and therefore, compromise the SLAM systems (Aguiar et al., 2020a).

To perform accurately, one of the most important steps of SLAM is perception. In this context, the use of 3D LiDARs has become popular in outdoor environments since they allow a high-range field of view perception of the environment. The point clouds generated by these sensors contain several types of features that are important for SLAM approaches. Regarding point features, one important descriptor is *smoothness* (Zhang and Singh, 2017), that for a given point $\boldsymbol{p_i}$ and a set of continuous points S, can be calculated as

$$c = \frac{1}{|S| \cdot \|\boldsymbol{p}_i\|} \left\| \sum_{j \in S, j \neq i} \left(\boldsymbol{p}_j - \boldsymbol{p}_i \right) \right\|. \tag{1}$$

With the application of threshold levels to c, two main types of point features can be extracted. Features with small values of c are present in high-curvature locations and therefore are called edge features. On the contrary, features with high smoothness are called planar features. Shan and Englot (2018) use these concepts in their proposed SLAM pipeline, LeGO-LOAM. In this work, the raw point cloud is first segmented to remove noise, and then edge and planar features are extracted. Other approaches propose the extraction of point features in the SLAM context. Steder et al. (2010) project the point cloud onto a range image and calculate the second derivative of the depth. With this formulation, high-curvature features are extracted from the range image. Chen et al. (2015) extract curb features from a 3D point cloud with a range up to 50 meters. This approach uses a distance criteria, and a Hough transform to process the point cloud and collect the desired features.

Point features extracted from 3D point clouds can have high computational cost in localization and mapping algorithms due to their usual high density. Even if the SLAM approaches are robust to this issue, the fusion of these perception techniques with other feature types can improve performance (Grant et al., 2019). Thus, as detailed in **Table 1**, many works use plane features in the mapping and localization stages. To extract such features, one of the most common techniques is Random Sample Consensus (RANSAC) (Gee et al., 2008; Ulas and Temeltas, 2012; Taguchi et al., 2013; Elghor et al., 2015). This algorithm receives as input a set of points and calculates the best fitting plane to those points, removing the input set's outliers. Other approaches use Convolutional Neural Networks (CNN)s (Yang et al., 2016) or Principal Component Analysis (PCA) (Viejo and Cazorla, 2007) for plane extraction. In terms of representation, a plane i is usually characterized as

$$\boldsymbol{m}_{\gamma_i} = \{\boldsymbol{\pi},\ d\}, \tag{2}$$

where $\boldsymbol{\pi} = [\pi_1, \pi_2, \pi_3]^T$ represents the plane unit normal vector, and d the plane distance to the origin. Other representations are also present in the literature. For example, Elghor et al. (2015) also preserve the number of inliers found in the RANSAC procedure in the plane representation. Also, Gee et al. (2008) represent a plane by its origin and two orthogonal basis vectors. All these representations are suitable for infinite planes. For localization and mapping, this representation can be improved using semiplanes. With this type of features, the matching procedure becomes more robust since other correspondence techniques can be applied, such as semiplane overlapping (Yang et al., 2016). Ulas and Temeltas (2012) use a Convex Hull algorithm to extract semiplanes extremas and feed an

outdoor SLAM algorithm. Similarly, Weingarten and Siegwart (2006) represent semiplanes by a set of convex polygons and use them in a 3D SLAM algorithm. Lenac et al. (2017) also use this representation and incorporate the polygons extremas in each semiplane characterization vector.

Plane mapping and registration can be more challenging than feature point mapping. In the latter, the general approach is to use a nearest neighbor search, optimized by efficient data structures such as 3D voxel maps. Two main factors are usually considered to solve plane matching and mapping: the difference between the plane normals and plane-to-plane distance (Viejo and Cazorla, 2007; Grant et al., 2013). With the consideration of bounded planes, Pop-up SLAM (Yang et al., 2016) also uses the overlapping area between semiplanes. After a successful matching procedure, the plane-based mapping step either adds new features to the map or updates the existing ones in case of correspondence. In SLAM, the mapping procedure is interdependent of localization. From **Table 1**, one can verify that most works use graph-based optimization to localize the robot using planes. For example, Zhang X. et al. (2019) build a pose graph where points and planes are marked as landmark nodes, and add structural constraints between planes in the graph. Kaess (2015) formulates planar SLAM as a factor graph finding a solution for the localization and mapping through least-squares optimization. Some works also adapt the ICP scan-matching algorithm to be used considering planar features. Viejo and Cazorla (2007) propose a two-step ICP that considers separate orientation and position alignment. This approach uses information given by the normal vector orientation and the geometric plane position. In addition, some works are based on Gaussian filters (Gee et al., 2008; Ulas and Temeltas, 2012; Zhang X. et al., 2019). In these, the state vector comprises the robot pose and the plane landmarks. In comparison with point feature-based SLAM, this approach has the advantage of reducing the state's dimension.

Given all of the above, it is clear that 3D LiDARs provide rich information for localization and mapping approaches. Besides the previously mentioned LeGO-LOAM approach, and its ancestor LOAM (Zhang and Singh, 2014), other approaches use this sensor to provide reliable SLAM systems. Kuramachi et al. (2015) propose a range and inertial odometry algorithm that fuses a 3D LiDAR and a gyroscope to recover the 6-DoF robot pose and map the environment. This LiDAR odom etry technique can either be approached traditionally using iterative algorithms, or in more sophisticated manners, such as using Artificial Intelligence (AI). For example, Choy et al. (2020) propose a deep global registration algorithm that is designed for pairwise registration of 3D scans. The key innovation of this approach is the use of a 6-DoF Convolutional Neural Network (CNN) for correspondence confidence prediction. In the same context, Li and Wang (2020) propose DMLO, a deep matching LidAR odometry algorithm which presents a learning-based matching network which provides accurate correspondences between two scans. In this work, we make use of a rich feature extraction process that considers both point and semiplane features, and implement a filter-based algorithm for localization and mapping. As represented on **Table 1**, to the best of our knowledge, the Particle Filter (PF) has not been approached together with semiplanes features in the SLAM context. The most common approach is to use optimization-based localization algorithms such as Bundle Adjustment and factor graph optimization. Thus, in this work, we extend the state-of-the-art to consider the use of a 6-DoF PF that supports two modalities of features: point-wise and semiplane-wise features. This filter-based algorithm was initially applied to robotics to solve the localization and kidnapping problems (Thrun, 2002). Zhang Q.-B. et al. (2019) use the PF to develop a robust localization algorithm that works when a priori environment map is available, considering range observations. Due to its capacity to accommodate multi-dimensional problems, the PF was later used to solve the SLAM issue. FAST-SLAM (Montemerlo et al., 2002) is one of the most popular examples, being able to solve the SLAM problem with a PF considering a landmark-based feature extraction procedure. In the agricultural context, the PF was also applied in autonomous navigation algorithms. Hiremath et al. (2014) use a PF to implement a row following algorithm in a maize field. The filter state is composed of robot heading, lateral deviation, distance between the rows of plants and the end of the rows. Similarly, Blok et al. (2019) use the PF and a 2D laser sensor to localize an agricultural platform for in-row navigation in orchards.

In this work, we propose VineSLAM (dos Santos et al., 2016; Aguiar et al., 2021), a 6-DoF SLAM algorithm for agricultural environments that uses point and semiplane features extracted from 3D point clouds. Our approach presents the following main contributions:

- A three-stage algorithm for semiplane extraction;
- A semiplane matching and merging algorithm that allows efficient registering and mapping;
- A novel localization procedure based on a PF that can use both point and planar information.

We model the agricultural environment as points and semiplanes. In each time step, edge and planar point features are extracted, and three semiplanes are searched in the environment. The first semiplane is the ground, and the other are two semiplanes, one in each side of the robot, that present the higher number of inlier points. This formulation is a reaction to the context where VineSLAM is intended to solve localization and mapping: agricultural cultures mainly characterized by woody-crop topologies. Thus, besides edge and planar features, usually only three semiplanes are available in the environment. These semiplanes are essential to the estimation of the three components of the robot's orientation. The ground plane is particularly important to estimate the roll and pitch components, and the vegetation planes to estimate the yaw component. Without these features, the algorithm would rely only on point-based features which could lead to drift in the orientation components over time. Also, if we rely only on semiplanes, we would always have to extract a minimum of three non-coplanar semiplanes to compute the 6-DoF localization of the robot. Thus, it is essential to merge the point and planar features.

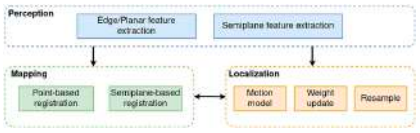


FIGURE 1 | System architecture partitioned in three main layers: perception, localization and mapping.

Our approach relates to the state-of-the-art feature extraction and matching steps but differs in the registration. We aplly a plane merging algorithm that constantly updates and grows the semiplanes present on the map. With this approach, we can capture large planar ground surfaces, as well as extensive vegetation planes. Also, we propose a novel localization algorithm that uses a PF fusing information at both point and planar levels. As reported in **Table 1**, most works in the state-of-the-art apply plane-based localization and mapping in indoor environments. Our approach is suitable for unstructured outdoor environments, and it was tested in a woody-crop vineyard. The PF approach was adopted in this work due to three main reasons. The first is relative to sensor fusion. Our aim is to implement a generic algorithm that is agnostic to the type of sensors used, i.e., an approach that can use any kind of sensor if an adequate weight function is provided for each one of them. With this, particles should be weighted by the combination of all the weight sub-functions. The second motivation for the use of this kind of filter to solve the SLAM problem is the straightforward parallelization scheme that it provides. Powerful processors such as Graphics Processing Units (GPU)s can be used considering that the calculations performed per particle can be executed in a separate processing core. Finally, the third is the support for multiple noise distributions in relation with other approaches that usually only support Gaussian noise. In this way, particles can be sampled and innovated with the distribution that best fits with the robot itself, and the scenario where it is inserted in.

The remainder of this paper is structure as follows. **Section 2** details the contributions of this work. **Section 3** contains two simulation experiences to validate the proposed approach. **Section 4** presents the test and validation of this work in real-world experiments. Finally, **Section 5** details the conclusions of this work.

2 VineSLAM: Localization and Mapping on Agriculture

This work proposes VineSLAM, a localization and mapping algorithm based on 3D points and semiplane features extracted from an input point cloud. **Figure 1** shows a high-level representation of the approach.

The system is divided in three main layers:

- Perception: 3D point cloud processing to extract edge, planar and semiplane features;
- Mapping: Multimodal registration of the types of features extracted to build a consistent 3D map of the agriculture environment;
- Localization: PF-based procedure that uses both point- and semiplane-based information to localize the robot.

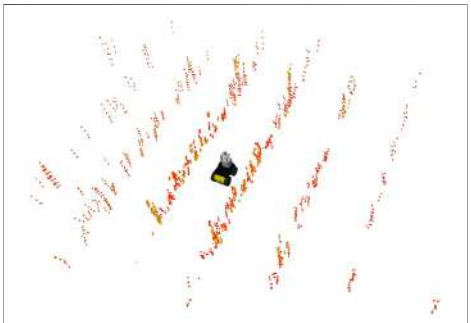


FIGURE 2 | Corner (yellow) and planar (red) feature extraction example in a woody crop vineyard.

Thus, our approach is able to efficiently extract point and semiplane features from a 3D point cloud, and use them to build a map of the crop and localize the robot within this map.

2.1 Perception

In the perception stage, three feature types of two different modalities are extracted: edge and planar features (points), and semiplanes. The point features are searched in sharp edges and planar surfaces. In the semiplane extraction case, this work searches for three semiplanes in the environment for each frame. The first is a flat ground surface, and the others are two semiplanes, one in each side of the robot. This formulation allows the extraction of large ground surfaces by recurrent plane registration. In woody-crop cultures, these semiplanes usually extract the morphology of the vegetation canopies. The extraction of point and semiplane features is described in 1) and 2) respectively.

1) Point-level feature extraction: To extract point features, our approach relates with LeGO-LOAM Shan and Englot (2018) in that it uses the smoothness descriptor c present in (**Eq. 1**). Points are projected into a range image and sorted by their value of c. The points with larger c are considered edge features, and the ones with lower c are considered planar features. **Figure 2** shows an example of this feature extraction procedure in a woody crop vineyard.
2) Semiplane-level feature extraction: Our approach can simultaneously use points and semiplanes to map the agriculture and localize the robot. The representation of semiplanes includes more dimensions than point features, and their extraction involves more complex procedures. In this work, we represent semiplanes as follows:

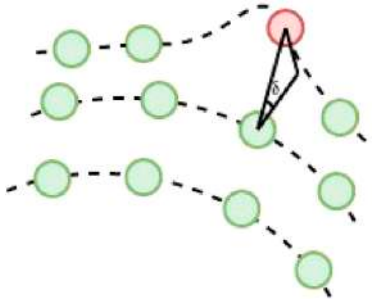


FIGURE 3 | Vertical angle definition. Green dots are estimated ground points, and red dots non-ground points. The vertical angle is estimated between two consecutive points of the same column in the range image.

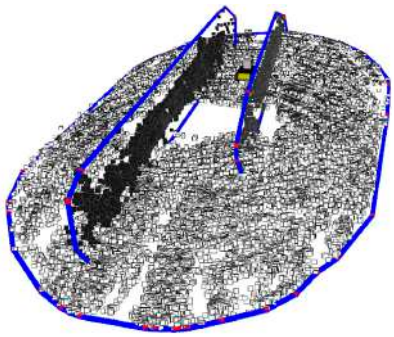


FIGURE 4 | Semiplane feature extraction example in a woody crop vineyard. The blue lines represent the polygons edges, the red dots their extremas and the dark dots the semiplane inliers points.

$$\boldsymbol{m}_{\gamma_i} = \{\boldsymbol{\pi},\ \boldsymbol{p}_0,\ e\}, \tag{3}$$

where $\boldsymbol{\pi}$ represents the unit normal vector, $\boldsymbol{p}_0$ the centroid defined by the points that compose the semiplane, and e the set of extrema points that limit the convex semiplane. The semiplane extraction is processed in two main steps: point candidate selection and plane fitting. In this step, the main goal is to extract three semiplanes from the input point cloud, the horizontal ground and two other arbitrary semiplanes.

For selecting the ground semiplane point candidates, two different parameters are used. The first is the point vertical angle Yang et al. (2021) (**Figure 3**) represented as δ. Given the point cloud projection into the range image and two points in consecutive rows, and defining the difference between these two points as $\Delta\boldsymbol{p} = [\Delta x, \Delta y, \Delta z]$, the vertical angle is compute as

$$\delta = \arctan\frac{\Delta z}{\sqrt{\Delta x^2 + \Delta y^2}}. \tag{4}$$

The second criteria is the point's height. Given the sensor position's prior knowledge in the robot's referential frame, and considering that it is mounted horizontally, only points that have a height component closer to the sensor altitude are considered. Thus, if this criteria is met and if δ is bellow a well-defined threshold, the point is considered as candidate for the ground plane. For the two remaining planes, a simpler point candidate selection is performed. In this case, the input cloud points that were not selected as ground candidates are divided into two main sets, one on each side of the robot. With this formulation, the goal is to extract two robust semiplanes both at the right and left sides of the robot.

After extracting the point candidates for the three planes, we implement a RANSAC algorithm in each set of points. This approach fits the best plane model represented by its hessian coefficients to the input set of points. In the end, the algorithm retrieves the set of inlier points that belong to the extracted plane, as well as its normal vector $\boldsymbol{\pi}$. This formulation outputs an infinite plane. To convert it to a semiplane, we extract a convex polygon that bounds all the inlier points that constitute the plane. To do so, a Convex Hull algorithm is applied to calculate the semiplane extremas e, represented in (**Eq. 3**). Since agricultural environments are highly unstructured, semiplane outliers can be extracted. In this work, the outliers are filtered based on the semiplane area. Only convex polygons with an area superior to a defined threshold configured by the user are considered and stored. **Figure 4** shows an example of the extraction of the ground plane and the vegetation canopies in a woody crop vineyard.

2.2 Mapping

In the mapping stage, two registration procedures are proposed, one for points and the other for semiplanes. A 3D voxel grid is implemented and used to store the point feature map and perform efficient search algorithms. A more complex feature matching algorithm is computed, and a map merging procedure is proposed to update the semiplanes in the global map continuously. These algorithms are described in 1) and 2) respectively.

1) Point-level mapping: Unlike LeGO-LOAM's approach that uses a standard KdTree to store feature points and map the environment, our work implements a 3D voxel grid map. This data structure, besides being less memory-efficient is more time-efficient since the searching algorithm is performed by just accessing each cell index. This 3D map is an extension of the standard 2D grid map, considering a discretization also in the z coordinate. Each cell is indexed by specific 3D discrete coordinates and can be efficiently accessed with these coordinates. Also, the map recognizes the different types of features supported. Thus, each cell can contain several types of features, and the searching procedures account for each type for faster processing. With this data structure, the point-based mapping procedure is performed using the information about the robot pose provided by the localization layer and using a local search algorithm. Given a

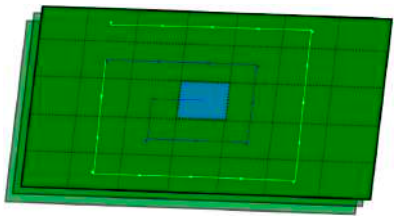


FIGURE 5 | Point feature nearest neighbor local search. In a discrete 3D space, the nearest neighbor of a point feature can be in the grid map layer where the feature is located or at the top and bottom adjacent layers. A local search in these three layers is performed to find the nearest feature as described in the figure. If a feature is found when searching in the blue path, the search ends. Otherwise, the search continues through the yellow path. The user can tune the stop criteria.

set of input point features $M_\lambda = \{\boldsymbol{m}_{\lambda_1}, \boldsymbol{m}_{\lambda_2}, \ldots, \boldsymbol{m}_{\lambda_N}\}$ in their homogeneous form and the robot's pose $\boldsymbol{T}_r$, each feature is converted to the maps' referential frame as follows:

$$\tilde{\boldsymbol{m}}_{\lambda_i} = \boldsymbol{T}_r \, \boldsymbol{m}_{\lambda_i}, \; i \in \{1, \ldots, N\}. \tag{5}$$

Then, a local search is performed for each feature in the 3D voxel grid map. The nearest neighbor of each feature is searched by the procedure represented in **Figure 5**. Each feature's nearest neighbor can be located either in its cell, in adjacent cells, or even in more distant cells. Depending on the voxel resolution, the user can specify if the local search algorithm looks for neighbors just in adjacent cells or if it continues for more remote cells in case of failure. This decision sets the stop criteria of the algorithm, that iteratively looks for the nearest feature in a region using a well-defined path as specified in **Figure 5**. In cases where no neighbor is found, the feature is registered and saved in the voxel map.

2) Semiplane-level mapping: The semiplane mapping procedure consists of two main tasks: semiplane matching and registering. The first step is to start with the matching algorithm, like in the point features case, to convert the observed semiplanes to the maps' referential frame. Thus, we apply a similar transformation as the one represented in (**Eq. 5**) to the semiplane inlier points and extremas. Then, to improve the matching procedure, we refine the normal vector estimation of the transformed semiplanes using a Principal Component Analysis (PCA) algorithm. Given the set of transformed semiplane points $\tilde{M}_y = \{\tilde{\boldsymbol{m}}_{y_1}, \tilde{\boldsymbol{m}}_{y_2}, \ldots, \tilde{\boldsymbol{m}}_{y_N}\}$, we define $\boldsymbol{S} = \boldsymbol{Y}\boldsymbol{Y}^T$, where

$$\boldsymbol{Y} = \begin{bmatrix} \tilde{\boldsymbol{m}}_{y_1} - \tilde{\boldsymbol{p}}_{\boldsymbol{0}} \\ \tilde{\boldsymbol{m}}_{y_2} - \tilde{\boldsymbol{p}}_{\boldsymbol{0}} \\ \cdots \\ \tilde{\boldsymbol{m}}_{y_N} - \tilde{\boldsymbol{p}}_{\boldsymbol{0}} \end{bmatrix} \tag{6}$$

and $\tilde{\boldsymbol{p}}_{\boldsymbol{0}}$ represents the transformed semiplane centroid. The refined normal vector corresponds to the eigenvector of $\boldsymbol{S}$ with the smallest eigenvalue. After this, the matching procedure is computed considering three different correspondence criteria:

- Overlapping area;
- Normal vector difference;
- Centroid-to-plane distance.

If an observed and a global map's semiplane meet these criteria, one can conclude that both planes overlap, have the same orientation, and are at the same position. Thus, they are considered as correspondences.

To compute the overlapping area between semiplanes, the local plane reference frame is extracted from the normal vector, and the semiplanes boundaries are projected into this frame. The z component is ignored since it is expected that, in the local plane reference frame, the boundaries lie in the X-Y plane ($z = 0$). Working in two dimensions, the interception between the two polygons is computed, and its area is calculated. If the overlapping area is higher than a threshold level, the normal vector difference is computed to check if the planes have the same orientation. Finally, the distance between the observed semiplane centroid to the semiplane in the global map is computed.

After the matching step, semiplane registration is performed to build the semiplane global map iteratively. Semiplanes that were not matched with any in the global map are directly registered in the global map. For the ones that were matched, a map merging algorithm is proposed. The semiplane representation is recomputed by merging inliers points between the two correspondences, recalculation of the normal vector using (**Eq. 6**), and boundary merging using Convex Hull. This process allows semiplanes to grow with newly observed features and to map large bounded surfaces. The process of registration can be observed in the following video: https://youtu.be/Yx8el67eTCw.

2.3 Localization

The localization procedure aims to compute the robots' 6-DoF pose using the feature extraction and mapping algorithms described. To this purpose, in contrast with the state-of-the-art, this work implements a PF with the novelty of considering point-semiplane particle weight calculation. This means that the filter can consider and balance both feature modalities and work in the absence of each one of them. The PF is standardly divided into three main steps: a prediction step where the particles are innovated through a motion model, the particles weight calculation given the observed features, and the resampling step to replace particles with lower weight by others with higher weight. These three steps are described in 1), 2) and 3) respectively.

1) Motion model: For predicting the particles likelihood distribution, they are innovated through a 6-DoF model proportional to an estimated relative motion. A LiDAR odometry algorithm based on the Iterative Closest Point (ICP) Besl and McKay (1992) method is implemented to estimate the frame-to-frame 6-DoF robot motion, where an input wheel odometry control $\boldsymbol{u}_t$ is used as first guess for the iterative

algorithm. The particles spreading is proportional to the distance measured by the scan-matching algorithm d_u. Let us define the following matrices:

- $\Delta \boldsymbol{T}$: the LiDAR odometry increment represented as an homogeneous transformation; and
- $\boldsymbol{T}_n$: an homogeneous transformation matrix computed by sampling a minimal parameter space (6-DoF) from a Gaussian distribution with standard deviation d_u.

Thus, a particle j is innovated at the instante t as follows:

$$\boldsymbol{T}_{j,t} = \boldsymbol{T}_{j,t-1}\ \Delta \boldsymbol{T}\ \boldsymbol{T}_n. \tag{7}$$

2) Weight update: The particles weight calculation is the most critical and innovative step of the proposed localization procedure. The major challenges are the consideration of multi modal inputs (points and semiplanes) and the creteria balance on the semiplane-based weight calculation, since two parameters are used to compute the weight. To account for both modalities, a two-step weight calculation is performed using two subfunctions: W_λ for the point (edge and planar) features and W_γ for the semiplane features. The particles weight is represented as a likelihood function $P(\boldsymbol{z}_t|\boldsymbol{T}_{j,t}, Z_{0:t-1})$ where $\boldsymbol{z}_t$ represent the feature observations at the instant t, $\boldsymbol{T}_{j,t}$ the particle's pose, and $Z_{0:t-1}$ the map build so far.

In the point-feature case, features are converted to maps' reference frame using each particle pose as follows:

$$\tilde{\boldsymbol{m}}_{\lambda_i} = \boldsymbol{T}_{j,t}\ \boldsymbol{m}_{\lambda_i},\ \ i \in \{1, \ldots, N\}. \tag{8}$$

Then, correspondences are found using the 3D voxel grid map search algorithms described in **Section 2.2**. Considering the set of K correspondences found $\{\tilde{\boldsymbol{m}}_{\lambda_i} \leftrightarrow \boldsymbol{m}_{\lambda,g_i} :\ i \in \{1, \ldots, K\}\}$, where the subscript g denotes for features in the *global* map, the point-feature weight subfunction is computed as

$$W_\lambda = \frac{1}{\sqrt{2\pi}\sigma_\lambda} \sum_{i=1}^{K} \exp\left(\frac{-1}{\sigma_\lambda} \cdot \|\tilde{\boldsymbol{m}}_{\lambda_i} - \tilde{\boldsymbol{m}}_{\lambda,g_i}\|\right), \tag{9}$$

where σ_λ is the standard deviation of the point-feature measurement, and $\|.\|$ represents the L2 norm. This formulation states that the weight of the particle increases exponentially with the decrease of distance between two correspondences, and that it is as higher as the number of correspondences K found.

In the semiplane-feature case, the first step is also the conversion of them to the maps' reference frame. The set of extremas of each semiplane e_i and its corresponding centroid $\boldsymbol{p}_{0_i}$ are converted to this referential using each particle pose as in (**Eq. 8**). Then, correspondences between the observed semiplanes and the ones already registered in the global map are searched using the three criterias described in **Section 2.2**: overlapping area, normal vector difference, and centroid-to-plane distance. Given the set of K correspondences found $\{\tilde{\boldsymbol{m}}_\gamma \leftrightarrow \boldsymbol{m}_{\gamma,g_i} :\ i \in \{1, \ldots, K\}\}$, where the subscript g denotes for features in the *global* map, the semi-plane weight subfunction is modelled as a multivariable function as follows:

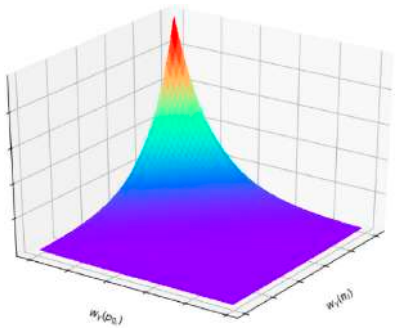


FIGURE 6 | Likelihood of the semiplane-based weight calculation represented as a multivariable function. The likelihood decreases exponentially with the increase of difference between normal vectors and centroid-to-plane distance. Their corresponding standard deviations can control the impact of each one of the variables in the final likelihood.

$$W_\gamma = \sum_{i=0}^{K} w_\gamma(\boldsymbol{\pi}_i) \cdot w_\gamma\left(\boldsymbol{p}_{0_i}\right), \tag{10}$$

where

$$w_\gamma(\boldsymbol{\pi}_i) = \frac{1}{\sqrt{2\pi}\sigma_\pi} \exp\left(\frac{-1}{\sigma_\pi} \cdot \|\tilde{\boldsymbol{\pi}}_i - \tilde{\boldsymbol{\pi}}_{g_i}\|\right)$$
$$w_\gamma\left(\boldsymbol{p}_{0_i}\right) = \frac{1}{\sqrt{2\pi}\sigma_{\boldsymbol{p}_0}} \exp\left(\frac{-1}{\sigma_{\boldsymbol{p}_0}} \cdot D\left(\boldsymbol{p}_{0_i}, \tilde{\boldsymbol{m}}_{\gamma,g_i}\right)\right),$$

$\sigma_{\boldsymbol{\pi}}$ and $\sigma_{\boldsymbol{p}_0}$ represent the standard deviations of the normal vector and centroid measurements respectivelly, $\tilde{\boldsymbol{\pi}}_{(.)}$ the semiplane's normal vector, $\tilde{\boldsymbol{p}}_{0_{(.)}}$ the semiplane's centroid, and $D(.)$ the point-to-plane distance operator. **Figure 6** represents the semiplane-based weight for a single correspondence. As represented, the importance given to the normal vector difference (to account for plane rotation) and to the centroid-to-plane distance (to account for plane displacement) can be tuned by the standard deviations of each measurement. The tunning step can be challenging since the multivariable function considers two variables in different spaces (vectors and distances). Overall, the particle's weight decreases exponentially with the increase of difference between correspondence's normal vector and centroid-to-plane distance.

Given the definitions (**Eqs 9, 10**) the particle likelihood is computed as

$$P\left(\boldsymbol{z}_t|\boldsymbol{T}_{j,t}, Z_{0:\,t-1}\right) = W_\lambda \cdot W_\gamma. \tag{11}$$

In this way, the particle with the highest weight is the one that presents the best alignment both in the point-feature and semiplane-feature spaces. The final robot pose per frame is computed by the weighted average of all particles' pose.

3) Resample: The resampling step of the PF is used to substitute low-weight by high-weight particles. In this work,

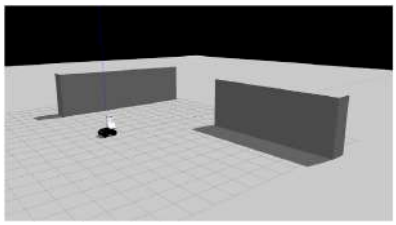


FIGURE 7 | Simulation environment containing three perpendicular planes: the ground and two walls.

the multinomial resample algorithm Douc and Cappe (2005) was implemented to accomplish this. This approach draws N samples from a uniform distribution u_i and selects the particle j for replication if

$$u_i \in \left[\sum_{p=1}^{j-1} w_p, \sum_{p=1}^{j} w_p\right), \tag{12}$$

where w_p represents the particle's p weight. To avoid the well-known problem of particle degeneracy that happens when either all the particles are in the wrong place or they are highly condensed, resampling is not executed for all iterations. This method is only employed when a significant robot motion is observed (either in translation or rotation in the six degrees of freedom). The user can set the amount of motion required to perform resampling.

3 SIMULATION EXPERIMENTS

This work's major innovation is the use of semiplanes to map the environment and localize the robot within it. As discussed in Section 2.3, one of the biggest challenges in the proposed semiplane-based localization is the weight given to the normal vector difference and the centroid-to-plane distance. This Section presents two simulation experiences to validate the numeric stability of the approach. Three orthogonal planes are used to localize the robot in the simulated environment and no point-features are considered.

3.1 Methodology

The simulation's main goal is to verify if, with three orthogonal planes is possible to localize the robot. Thus, the environment present in **Figure 7** was built containing two perpendicular walls. The third semiplane is the ground. The robotic platform used for real experiments that will be detailed in **Section 4** was modeled and inserted in the simulation. This platform is based on a Husky robotic base. To obtain the raw wheel odometry inputs, the Husky simulator[1] was used, that considers the error caused by the wheel slippage. To obtain the LiDAR data, the Velodyne simulator[2] was used with a simulated Gaussian noise with standard deviation of 0.008 m.

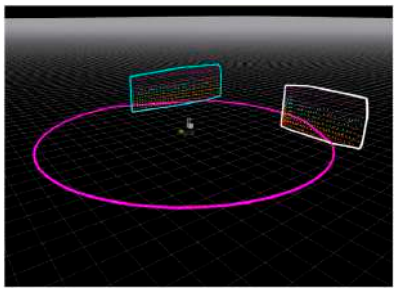


FIGURE 8 | Semiplane feature extraction of the three simulated semiplanes.

Given all of the above, the simulation experiments were carried out by two different sequences: a translation-only motion and a rotation-only motion. The idea is to validate that, using only three perpendicular semiplanes, the proposed approach can estimate translations and rotations. The PF algorithm used 500 particles to estimate the robot's motion.

3.2 Localization Performance

As referenced before, under the same simulated environment, two different experiments were performed. The semiplane feature extraction procedure describe in **Section 2.2** resulted in the successful detection of the three semiplanes (two walls and the ground) as shown in **Figure 8**. In the first experiment, the robot moves forward without rotating. **Figure 9A** shows the x and y deviation to the ground truth, as well as the absolute distance error (meters). Overall, the semiplane-based localization procedure presents an average distance error of 0.069 1 m for this sequence. This shows that the semiplane extraction algorithm is accurate in estimating the normal vectors, centroids and extremas. More importantly, using only three orthogonal semiplanes, the localization procedure can localize the robot with low error.

Regarding the rotation-only experiment, **Figure 9B** shows the estimated yaw rotation in reference with the ground truth, as well as the corresponding rotation error. For this sequence, the average rotation error obtained was 5.01 degrees. One of the well-known localization issues is the estimation of motion in rotation-only movements. These scenarios are more challenging than translation motions since the perception of the environment changes faster, challenging the matching procedures. Besides, the odometers tend to present higher errors in rotations due to wheel slippage. For all these reasons, this type of motion shows to impact the filter's performance. Even so, the localization

[1]https://github.com/husky/husky_simulator

[2]https://bitbucket.org/DataspeedInc/velodyne_simulator.git

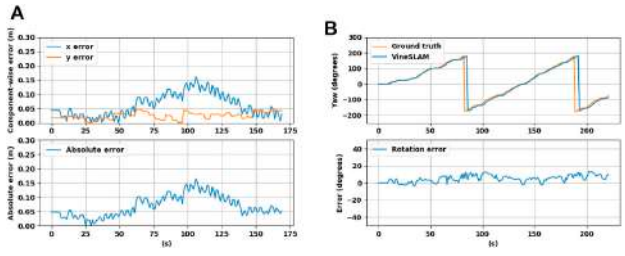


FIGURE 9 | Simulation results using three perpendicular planes in the localization and mapping procedures for a **(A)** translation-only trajectory, and for a **(B)** rotation-only trajectory.

FIGURE 10 | AgRob V16 robotic platform used to test the proposed approach placed in a woody-crop vineyard.

approach can estimate the robot rotation with acceptable performance, using only three semiplanes.

In the real world, the presence of at least three orthogonal semiplanes is not always guaranteed. For example, in vineyards, the most common scenario is detecting the ground plane and two parallel vegetation planes. With this configuration there are no constraints to estimate the forward component of the translation. Due to this, the proposed approach fuses points with semiplanes. Nevertheless, this simulation aims to show that the proposed formulation is suitable for semiplane-only localization. The simulated experiments performed show that the algorithm can perform accurately in translational and rotational movements.

4 RESULTS

To test the proposed solution, our robotic platform AgRob V16 Aguiar et al., 2020b; Santos et al. (2020) present on **Figure 10** was used. The robot is equiped with a Velodyne Puck (VLP-16) and was placed in Aveleda's vineyard, in Portugal. It travelled three different sequences that will be described later on. Section 4.1 details the experiments performed in this context, and Section 4.2 describes and analysis the results obtained.

TABLE 2 | Summary of the experiments performed in Aveleda's vineyard.

Experiment	Distance (m)	Foliage stage	Season
Sequence 1	69.73	With	Summer
Sequence 2	23.52	Without	Winter
Sequence 3	81.72	With	Summer

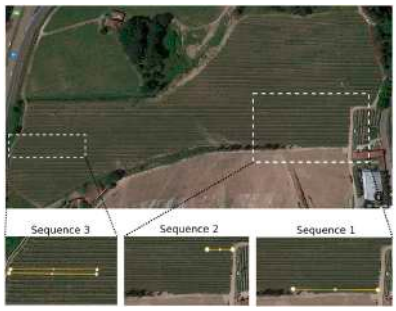


FIGURE 11 | Satellite image of Aveleda's vineyard. The sub-figures represent the sequences (1, 2 and 3) traveled by the robot.

4.1 Methodology

The proposed VineSLAM localization and mapping algorithm performance was analysed in three different sequences described in **Table 2**. The characteristics of each sequence are different not only due to the different travelled paths, but also because they were recorded in different seasons of the year. Two of them in the summer, which means that the vineyard present a high density of foliage, and other in the winter and without foliage. Sequence 1 has almost 70 meters of extension and is the less symmetric sequence since, besides being inside a corridor, it is at the border

TABLE 3 | Absolute pose error metrics for VineSLAM and LeGO-LOAM under the three test sequences.

Experiment	Method	Max (m)	Mean (m)	RMS (m)
Sequence 1	VineSLAM	2.65	1.41	1.58
	LeGO-LOAM	2.26	0.81	0.93
Sequence 2	VineSLAM	0.84	0.38	0.44
	LeGO-LOAM	20.81	10.49	11.87
Sequence 3	VineSLAM	1.17	0.64	0.69
	LeGO-LOAM	29.57	21.39	22.48

of the vineyard. Thus, the high-range LiDAR used can capture scene objects that are not present in the corridor. Sequence 2 presents the smallest path (23.52 meters). Nevertheless, it is challenging since the path is inserted in the middle of the vineyard, with a high density of corridors. This sequence was recorded during the winter, which means that the vineyard had not foliage. Finally, sequence 3 is the most extensive, with more than 80 meters. In this, the robot travels along two vineyard corridors and is also placed in the middle of the vineyard. Thus, in most cases, the environment is highly symmetric, compromising the localization and mapping algorithm. **Figure 11** shows a satellite image with the three sequences represented.

To test and our approach, the proposed VineSLAM algorithm was executed in the three sequences and compared with the state-of-the-art LeGO-LOAM (Shan and Englot, 2018) SLAM method. We consider LeGO-LOAM the state-of-the-art in outdoor 3D SLAM using LiDAR sensors. This method was tested in extensive outdoor experiments, and proved to perform better than its ancestor, LOAM. From the literature, one can see that LeGO-LOAM is aligned with robust 3D SLAM approaches, such as G-ICP (Ren et al., 2019). Thus, in this work we test LeGO-LOAM in harsh agricultural environments, and benchmark it against our approach, VineSLAM.

To validate the results, Global Navigation Satellite System (GNSS) was used as ground truth, and the Absolute Pose Error (APE) was measured using this reference. The maximum, mean and Root Mean Squared (RMS) APE errors were annotated for each experiment.

4.2 Localization Performance

The robot localization estimation is evaluated in sequences 1, 2 and 3 and compared with the state-of-the-art SLAM approach LeGO-LOAM. It is worth noting that the sequences are present in long vineyard corridors, which can be problematic for SLAM approaches. One of the key issues of SLAM is the well-known corridor problem where the forward component of motion has high uncertainty due to the symmetries of the scene. In the Aveleda vineyard, this can happen since vine trunks are equally spaced, and consequently, the agricultural environment is highly symmetric.

The APE is used to analyze the obtained results. For each timestamp, the absolute difference between the reference and the estimated poses is calculated. **Table 3** summarizes the results

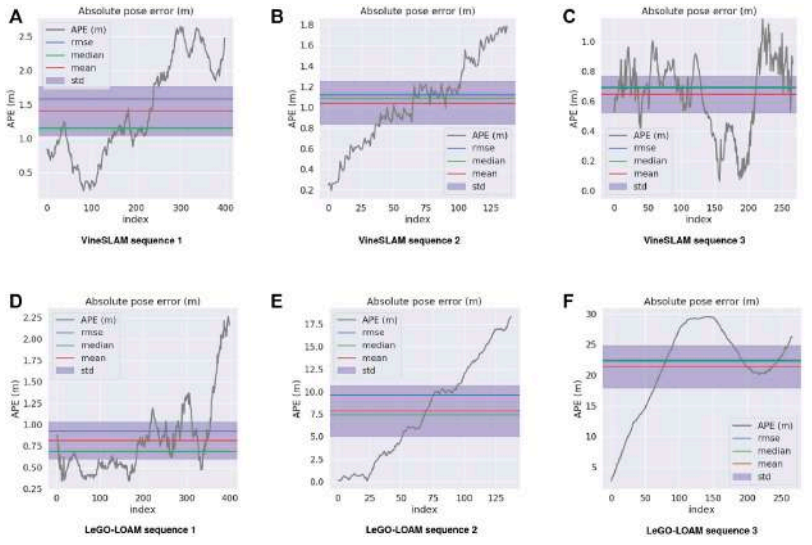


FIGURE 12 | Average pose error (m) and its corresponding Root Mean Squared Error (RMSE), median, mean and standard deviation for **(A–C)** VineSLAM and **(D–F)** LeGO-LOAM under the three experiments performed.

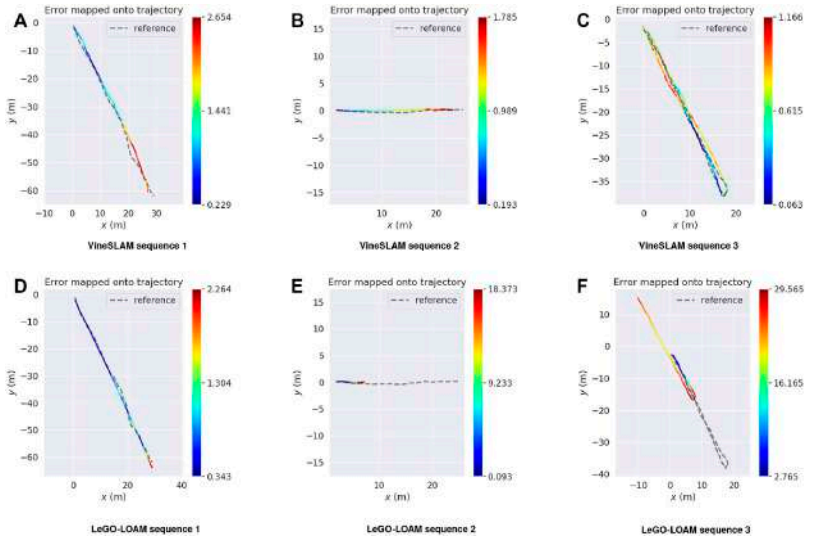


FIGURE 13 | Absolute pose error (m) mapped onto the trajectory for **(A–C)** VineSLAM and **(D–F)** LeGO-LOAM with reference to the ground truth.

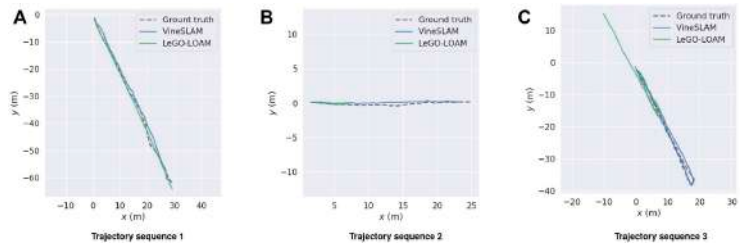


FIGURE 14 | VineSLAM's and LeGO-LOAM's localization estimation with reference to the ground truth for sequences **(A)** 1, **(B)** 2, and **(C)**3.

obtained for both methods in the three sequences. **Figure 12** plots the APE over time, as well as its corresponding RMS error, mean, median and standard deviation. To highlight the APE during the robot motion, **Figure 13** maps it onto the trajectory and represents the error through a color code. Finally, to have a clear perception of our VineSLAM approach and LeGO-LOAM in reference with the ground truth, **Figure 14** presents all the trajectories in the same graphic for each sequence.

For sequence 1 it is possible to verify that LeGO-LOAM performs better than VineSLAM. In particular, for this sequence, the state-of-the-art approach outperforms VineSLAM in approximately 0.6 meters considering the RMS error. This was the sequence where LeGO-LOAM presented better performance since, as referenced before, it could find structure outside the corridor placed in the vineyard's border. As can be observed in **Figure 13A** VineSLAM present the higher error when the robot performs a rotation inside the corridor. Even so, both methods showed acceptable performance even in this challenging environment.

For sequences 2 and 3, we verified that LeGO-LOAM localization estimation has degenerated, i.e., the state-of-the-art algorithm fails to estimate the robot pose for these two sequences.

The environment's high symmetry impacts its localization algorithm since, in many instants, LeGO-LOAM estimates that the robot is stopped when it is moving. Thus, for sequence 2 this method presents an RMS error of 11.87 meters and for sequence 3 22.48 meters. **Figure 12E** shows that LeGO-LOAM accumulates error over time for sequence 2. For the other sequence, the same happens until the robot turns around, the moment where this method starts estimating movement again. In these two sequences, the proposed VineSLAM approach's contribution is highlighted since it can maintain a precise robot localization in both of them. Especially for sequence 3, where the robot travels more than 80 meters over two vineyard corridors, VineSLAM achieved an RMS error of 0.69 meters.

Overall results show that VineSLAM is suitable for localization and mapping in agricultural environments. The PF approach considers not only point-features but also semiplanes to map and localize the robot. This approach proved to be accurate even in challenging agricultural environments, improving the state-of-the-art. In most cases, LeGO-LOAM underestimates translation due to the corridor's symmetry, while the PF approach proposed in VineSLAM can overcome this issue using a discretization of the 6-DoF state space in 500 different particles. From **Figure 14** one can verify that VineSLAM follows the GNSS reference with accuracy in the three sequences while LeGO-LOAM only does so in the first. This proved that the localization and mapping research is still open for improvement. For harsh environments such as vineyards, dedicated approaches should be proposed to tackle the more generic state-of-the-art algorithms' limitations.

5 CONCLUSION

This work proposes an extension of the state-of-the-art in localization and mapping oriented to agricultural robots. In this context, we propose VineSLAM, an algorithm that uses both points and semiplanes to map the environment and localize agricultural robots. The integration of all this information in a single pipeline is done efficiently with a 3D voxel map proposal to accelerate search algorithms and innovative semiplane-based mapping techniques. Also, a PF is used with a novel update step, where the likelihood of the particle considers both feature modalities. Results show that our formulation can localize a robot using only three orthogonal semiplanes. Under real-world experiments in a woody-crop vineyard, VineSLAM achieved RMS errors of 1.58, 0.44, and 0.69 meters for three sequences. Overall, our approach outperforms the state-of-the-art LeGO-LOAM algorithm that fails in two of the three sequences.

In future work, we would like to extend the mapping algorithms of VineSLAM. In particular, features with semantic representations will be extracted from agriculture environments, such as trunks and fruits, and used in the mapping and localization procedures. Additionally, we would like to partition the global map in a graph-like fashion considering a topological structure. A sensor fusion approach will be adopted to improve the localization redundancy and robustness. Finally, the algorithm will be tested in different agricultural scenarios such as greenhouses and orchards.

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/Supplementary Material, further inquiries can be directed to the corresponding author.

AUTHOR CONTRIBUTIONS

AA, FS, and HS contributed to conception and design of the study. AA wrote the first draft of the manuscript. JC, AS, FS and HS reviewed the entire manuscript. All authors contributed to manuscript revision, read, and approved the submitted version.

FUNDING

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101004085.

ACKNOWLEDGMENTS

AA thanks the FCT—Foundation for Science and Technology, Portugal, for the Ph.D. Grant DFA/BD/5318/2020

REFERENCES

Aguiar, A. S., dos Santos, F. N., Cunha, J. B., Sobreira, H., and Sousa, A. J. (2020a). Localization and Mapping for Robots in Agriculture and Forestry: A Survey. *Robotics* 9, 97. doi:10.3390/robotics9040097

Aguiar, A. S., dos Santos, F. N., Sobreira, H., Cunha, J. B., and Sousa, A. J. (2021). Particle Filter Refinement Based on Clustering Procedures for High-Dimensional Localization and Mapping Systems. *Robotics Autonomous Syst.* 137, 103725. doi:10.1016/j.robot.2021.103725

Aguiar, A. S., Santos, F. N. D., De Sousa, A. J. M., Oliveira, P. M., and Santos, L. C. (2020b). Visual Trunk Detection Using Transfer Learning and a Deep Learning-Based Coprocessor. *IEEE Access* 8, 77308–77320. doi:10.1109/ACCESS.2020.2989052

Bailey, T., and Durrant-Whyte, H. (2006). Simultaneous Localization and Mapping (Slam): Part Ii. *IEEE Robot. Automat. Mag.* 13, 108–117. doi:10.1109/MRA.2006.1678144

Bergerman, M., Billingsley, J., Reid, J., and van Henten, E. (2016). *Robotics in Agriculture and Forestry*. Manhattan, New York: Cham: Springer International Publishing, 1463–1492. doi:10.1007/978-3-319-32552-1_56Robotics in Agriculture and Forestry

Besl, P. J., and McKay, N. D. (1992). A Method for Registration of 3-d Shapes. *IEEE Trans. Pattern Anal. Mach. Intell.* 14, 239–256. doi:10.1109/34.121791

Blok, P. M., van Boheemen, K., van Evert, F. K., IJsselmuiden, J., and Kim, G.-H. (2019). Robot Navigation in Orchards with Localization Based on Particle Filter and Kalman Filter. *Comput. Electron. Agric.* 157, 261–269. doi:10.1016/j.compag.2018.12.046

Chen, T., Dai, B., Liu, D., Song, J., and Liu, Z. "Velodyne-based Curb Detection up to 50 Meters Away," in Proceedings of the 2015 IEEE Intelligent Vehicles Symposium (IV), Seoul, Korea (South), 28 June-1 July, 241–248. doi:10.1109/IVS.2015.7225693

Choy, C., Dong, W., and Koltun, V. "Deep Global Registration," in Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, June 2020.

dos Santos, F. N., Sobreira, H., Campos, D., Morais, R., Paulo Moreira, A., and Contente, O. (2016). Towards a Reliable Robot for Steep Slope Vineyards Monitoring. *J. Intell. Robot Syst.* 83, 429–444. doi:10.1007/s10846-016-0340-5

Douc, R., and Cappe, O. "Comparison of Resampling Schemes for Particle Filtering," in ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, Zagreb, Croatia, September 2005, 64–69. doi:10.1109/ISPA.2005.195385

Durrant-Whyte, H., and Bailey, T. (2006). Simultaneous Localization and Mapping: Part I. *IEEE Robot. Automat. Mag.* 13, 99–110. doi:10.1109/MRA.2006.1638022

Elghor, H. E., Roussel, D., Ababsa, F., and Bouyakhf, E. H. "Planes Detection for Robust Localization and Mapping in Rgb-D Slam Systems," in Proceedings of the 2015 International Conference on 3D Vision, Washington, DC, United States, October 2015, 452–459. doi:10.1109/3DV.2015.73

Emmi, L., Gonzalez-de-Soto, M., Pajares, G., and González-De-Santos, P. (2014). New Trends in Robotics for Agriculture: Integration and Assessment of a Real Fleet of Robots. *Scientific World J.* 2014, 1–21. doi:10.1155/2014/404059

Gee, A. P., Chekhlov, D., Calway, A., and Mayol-Cuevas, W. (2008). Discovering Higher Level Structure in Visual Slam. *IEEE Trans. Robot.* 24, 980–990. doi:10.1109/TRO.2008.2004641

Grant, W. S., Voorhies, R. C., and Itti, L. (2019). Efficient Velodyne Slam with point and Plane Features. *Auton. Robot* 43, 1207–1224. doi:10.1007/s10514-018-9794-6

Grant, W. S., Voorhies, R. C., and Itti, L. (2013). "Finding Planes in Lidar point Clouds for Real-Time Registration," in Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, November 2013, 4347–4354. doi:10.1109/IROS.2013.6696980

Hiremath, S. A., van der Heijden, G. W. A. M., van Evert, F. K., Stein, A., and ter Braak, C. J. F. (2014). Laser Range Finder Model for Autonomous Navigation of a Robot in a maize Field Using a Particle Filter. *Comput. Electron. Agric.* 100, 41–50. doi:10.1016/j.compag.2013.10.005

Kaess, M. "Simultaneous Localization and Mapping with Infinite Planes," in Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, Washington, USA, May 2015, 4605–4611. doi:10.1109/ICRA.2015.7139837

Kuramachi, R., Ohsato, A., Sasaki, Y., and Mizoguchi, H. "G-icp Slam: An Odometry-free 3d Mapping System with Robust 6dof Pose Estimation," in Proceedings of the 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO), Zhuhai, China, December 2015, 176–181.

Lenac, K., Kitanov, A., Cupec, R., and Petrović, I. (2017). Fast Planar Surface 3d Slam Using Lidar. *Robotics Autonomous Syst.* 92, 197–220. doi:10.1016/j.robot.2017.03.013

Li, Z., and Wang, N. (2020). "DMLO: Deep Matching Lidar Odometry," in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 6010–6017. doi:10.1109/iros45743.2020.9341206

Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B. "Fastslam: A Factored Solution to the Simultaneous Localization and Mapping Problem," in Proceedings of the AAAI National Conference on Artificial Intelligence. Aaai/iaai 593598, Edmonton, Alberta, July 28–August 1 2002.

Pinto de Aguiar, A. S., Neves dos Santos, F. B., Feliz dos Santos, L. C., de Jesus Filipe, V. M., and Miranda de Sousa, A. J. (2020). Vineyard Trunk Detection Using Deep Learning - an Experimental Device Benchmark. *Comput. Electron. Agric.* 175, 105535. doi:10.1016/j.compag.2020.105535

Ren, Z., Wang, L., and Bi, L. (2019). Robust Gicp-Based 3d Lidar Slam for Underground Mining Environment. *Sensors* 19, 2915. doi:10.3390/s19132915

Roldán, J. J., Cerro, J. d., Garzón-Ramos, D., Garcia-Aunon, P., Garzón, M., León, J. d., et al. (2018). "Robots in Agriculture: State of Art and Practical Experiences," in *Service Robots*. Editor E. Neves (London, UK: IntechOpen). doi:10.5772/intechopen.69874

Santos, L. C., Aguiar, A. S., Santos, F. N., Valente, A., and Petry, M. (2020). Occupancy Grid and Topological Maps Extraction from Satellite Images for Path Planning in Agricultural Robots. *Robotics* 9, 77. doi:10.3390/robotics9040077

Shalal, N., Low, T., McCarthy, C., and Hancock, N. "A Review of Autonomous Navigation Systems in Agricultural Environments," in Proceedings of the SEAg 2013: Innovative Agricultural Technologies for a Sustainable Future, Barton, Western Australia, September 2013.

Shan, T., and Englot, B. "Lego-loam: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain," in Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, October 2018, 4758–4765. doi:10.1109/IROS.2018.8594299

Steder, B., Grisetti, G., and Burgard, W. "Robust Place Recognition for 3d Range Data Based on point Features," in Proceedings of the 2010 IEEE International Conference on Robotics and Automation, Anchorage, Alaska, USA, May 2010, 1400–1405. doi:10.1109/ROBOT.2010.5509401

Taguchi, Y., Jian, Y.-D., Ramalingam, S., and Feng, C. "Point-plane Slam for Hand-Held 3d Sensors," in Proceedings of the 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, May 2013, 5182–5189. doi:10.1109/ICRA.2013.6631318

Thrun, S. (2002). "Particle Filters in Robotics," in Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence, Alberta, Canada (San Francisco, CA, United States: Morgan Kaufmann Publishers Inc.), 511–518. doi:10.5555/2073876.2073937

Ulas, C., and Temeltas, H. "Plane-feature Based 3d Outdoor Slam with Gaussian Filters," in Proceedings of the 2012 IEEE International Conference on Vehicular Electronics and Safety (ICVES 2012), Istanbul, Turkey, July 2012, 13–18. doi:10.1109/ICVES.2012.6294326

Viejo, D., and Cazorla, M. "3d Plane-Based Egomotion for Slam on Semi-structured Environment," in Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Diego, CA, USA, October 2007, 2761–2766. doi:10.1109/IROS.2007.4399138

Vougioukas, S. G. (2019). Agricultural Robotics. *Annu. Rev. Control. Robot. Auton. Syst.* 2, 365–392. doi:10.1146/annurev-control-053018-023617

Weingarten, J., Gruener, G., and Siegwart, R. (2003). A Fast and Robust 3d Feature Extraction Algorithm for Structured Environment Reconstruction. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.84.7823&rep=rep1&type=pdf.

Weingarten, J., and Siegwart, R. "3d Slam Using Planar Segments," in Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China, October 2006, 3062–3067. doi:10.1109/IROS.2006.282245

Yang, F., Jiang, M., and Xu, C. (2021). Real-time Ground-Plane Refined Lidar Slam. *arXiv*. Available at: https://arxiv.org/abs/2110.11517.

Yang, S., Song, Y., Kaess, M., and Scherer, S. "Pop-up Slam: Semantic Monocular Plane Slam for Low-Texture Environments," in Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Korea, October 2016, 1222–1229. doi:10.1109/IROS.2016.7759204

Zhang, J., and Singh, S. (2014). "Loam: Lidar Odometry and Mapping in Real-Time," in *Robotics: Science and Systems*. Editors N. Roy, P. Newman, and S. Srinivasa (Cambridge, Massachusetts, United States: MIT Press). doi:10.15607/rss.2014.x.007

Zhang, J., and Singh, S. (2017). Low-drift and Real-Time Lidar Odometry and Mapping. *Auton. Robot* 41, 401–416. doi:10.1007/s10514-016-9548-2

Zhang, Q.-B., Wang, P., and Chen, Z.-H. (2019a). An Improved Particle Filter for mobile Robot Localization Based on Particle Swarm Optimization. *Expert Syst. Appl.* 135, 181–193. doi:10.1016/j.eswa.2019.06.006

Zhang, X., Wang, W., Qi, X., Liao, Z., and Wei, R. (2019b). Point-plane Slam Using Supposed Planes for Indoor Environments. *Sensors* 19, 3795. doi:10.3390/s19173795

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

5.2 Semantic Mapping of Grape Bunches and Stems using Sensor Fusion and a Robust Localization Algorithm

After having a metric-based localization and mapping system, VineSLAM's localization and mapping are improved. The PF modularity is exploited to fuse other types of sensors to have a more precise robot pose estimation. Also, in this work, the concept of multi-layer mapping is introduced - the proposed semantic mapping approach is added to the metric mapping procedure described before. This work produced an article that is currently being peer reviewed in the Robotics and Autonomous Systems Journal and is entitled *Semantic Mapping of Grape Bunches and Stems using Sensor Fusion and a Robust Localization Algorithm*. Regarding the localization module, this work describes the PF-based fusion of the metric features extracted from the 3D LiDAR with an RTK-GNSS and an IMU sensor. The semantic mapping algorithm uses the perception system described in Chapter 4 and implements an EKF to create a map of the vineyard with vine trunks and grape bunches. The multi-layer mapping architecture can be enhanced since the depth of the semantic features is calculated using the metric map that VineSLAM builds while the robot operates. The system was tested in two vineyards that present different challenges to the algorithm and was benchmarked against two state-of-the-art SLAM algorithms. Results show that the proposed system can localize the robot with precision in challenging environments and create semantic maps of the vineyard. The localization approach outperforms the LeGO-LOAM state-of-the-art algorithm and present similar results compared with LOAM.

Semantic Mapping of Grape Bunches and Stems using Sensor Fusion and a Robust Localization Algorithm

André Silva Aguiar[1,2], Filipe Neves dos Santos[1], João Valente[3], and José Boaventura-Cunha[1,2]

[1]**INESC TEC - INESC Technology and Science; 4200-465, Porto, Portugal**
[2]**University of Trás-os-Montes e Alto Douro; 5000-801 Vila Real, Portugal**
[3]**Information Technology Group, Wageningen University and Research, Wageningen, The Netherlands**

ABSTRACT

Localization and mapping are fundamental aspects of mobile robotics, since they allow track the robot position and to build useful maps of its surroundings. In most cases, these algorithms build metric maps that represent the geometry of the environment. In agriculture, there is an increase need for robots to have precise and rapid information since it affords time and money, enables to trigger actions plans in case of diseases, and allows accurate yield estimation. In this context, semantic perception and mapping are essential concepts. With semantics, robots can learn how to assign meaning to data, learn to co-exist with humans, and provide useful information for further human use. For all these reasons, this work proposes a system capable of semantically mapping grape bunches and vine stems using sensor fusion through a robust Simultaneous Localization and Mapping algorithm. The contributions of the proposed approach are threefold: a robust system capable of localizing the robot in challenging agricultural environments and build 3D maps of the environment; a Deep Learning-based semantic visual perception algorithm that detects grape bunches and vine stems considering a public dataset with RGB and thermal vineyard images in different stages of the year; a semantic mapping approach that represents the vineyard as a map of grape bunches and vine stems. Results show that our system can localize the robot in several types of vineyards with high precision. Also, the Deep Learning model is able to detect vine stems and grape bunches in two different growth stages. Finally, the system is able to build rich semantic maps of the vineyard.

Keywords: SLAM, Semantic Mapping, Perception, Agriculture

1 INTRODUCTION

Environment mapping is one of the fundamental aspects of mobile robotics. Most tasks developed by autonomous robots require at least the knowledge of the environment's geometry. In these cases, robots know where the free and navigable space is and where the static obstacles that they must avoid are located Thrun (2003). For robots to perform advanced tasks besides navigation, in most cases is required more intelligent mapping techniques. In this context, semantic mapping uses robots to build maps with geometry information and assign properties to objects and places. In other words, robots learn how to assign meaning to data Wolf and Sukhatme (2008). This means that robots with this faculty can percept environments in a similar way as humans do, but also that the maps built by them can be easily read and used by humans Kostavelis and Gasteratos (2015). Thus, robots that have semantic perception can operate in human co-existent environments and even interact with them Sheng et al. (2015).

In agriculture, robots with semantic perception are fundamental to automating several tasks such as precision agriculture (application of fertilizers, nutrients and water), harvesting, monitoring, and planting. Besides, these robots can also be used for fruit and vegetable counting Das et al. (2015), which allow yield estimation and monitoring of productivity. All these tasks require that the robot perform self-localization in real time to know where it is located in the agricultural map. Consequently, autonomous robotic platforms should be endowed with robust localization systems that allow recovering their absolute pose in

the agricultural environment Vougioukas (2019). Simultaneous Localization and Mapping (SLAM) allows calculating the traveled trajectory while mapping the environment simultaneously Durrant-Whyte and Bailey (2006); Bailey and Durrant-Whyte (2006). Figure 1 shows an example of a steep slope vineyard located in the Douro's demarched region, where is visible the challenging conditions for autonomous robots. In this particular environment, and all the other agricultural contexts in general, SLAM algorithms

Figure 1. Typical steep slope vineyard in the Douro's region. Localization and mapping approaches need to be robust to the sharp slopes that characterize these environments.

face many challenges. These environments are usually highly unstructured compared to indoor scenarios, which can compromise the performance of matching algorithms that are crucial, especially in 3D SLAM. Also, the presence of terrain irregularities makes the robot trajectories less smooth, which implies that SLAM algorithms should have the ability to estimate frequent variations in the six components of motion. Finally, illumination characteristics can harm perception systems, especially vision-based feature extractors. In order to robots operate autonomously in this kind of environments, one of the critical aspects that should be present in SLAM approaches is information fusion. The algorithms that fuse information from different modalities of sensors can be more tolerable to faults and more robust Vu et al. (2011). In agriculture, this can be particularly important. For example, in the scenario present in Fig. 1, sensors that can detect slopes such as Inertial Measurement Units (IMU)s or Real-time kinematic (RTK) Global Navigation Satellite Systems (GNSS) can be fused with sensors capable of providing accurate maps of the environment such as Light Detection And Ranging (LiDAR)s or cameras. Thus, intelligent systems should be implemented to extract each sensor modality's key components of each sensor modality and incorporate all sensors simultaneously in the localization and mapping stages.

This work intends to solve the localization and mapping problems in challenging agricultural environments such as vineyards and provide a semantic representation of them by mapping and counting of grape bunches and stems in vineyards. To do so, this paper proposes the following contributions:

1. The creation of a novel annotated dataset for Deep Learning (DL) model training with RGB and thermal images of different vineyards in different stages of the year (with and without foliage) and with different grape growing stages;

2. The evaluation of a state-of-the-art trained DL model that can detect vine stems and grape bunches

using low-power resources using Edge-AI concepts;

3. The implementation and evaluation of a new SLAM framework based on a localization and 3D mapping algorithm based on a Particle Filter (PF) designed to work in challenging agricultural environments that fuses an RTK-GNSS, an IMU, a long-range 3D LiDAR, and wheel odometry;

4. The implementation and evaluation of a semantic mapping algorithm that fuses a monocular camera with the sensors used in the localization and 3D mapping stages to provide the positioning of grape bunches and vine stems in the agricultural environment.

The output of our system is a 3D map that can be visualized in several ways, such as 3D point clouds, octomaps or meshes, with the vine stems and grape bunches location mapped inside these 3D representations.

The rest of the paper is organized as follows. Section 2 provides an analysis of the state-of-the-art. Section 3 details the implementation of the proposed system. Section 4 shows the results obtained under the experiments performed. Finally, section 5 provides the main conclusions of this work.

2 RELATED WORK

The mapping procedure is crucial for localization performance. The accuracy of the onboard sensors and the quality of the data post-processing algorithms dictate the quality of the perception of the surrounding environment by the robotic platform. In agriculture, this is especially true, since in most cases, the environment present harsh conditions for robotics localization and mapping Aguiar et al. (2020a). When the localization and mapping algorithms succeed, robots can create several types of maps of the environment that can be used by humans or even by other robots to operate. The first and most common type of maps are geometry or metric maps as represented in Fig. 2. The representation is a structure

Figure 2. Metric map of a steep slope vineyard. White cells represent free space and black cells represent occupied space.

that encodes the geometry of the environments Cadena et al. (2016) and can be represented in several ways depending on the sensor input data and the dimension of the mapping area. Another popular representation is topological mapping.These algorithms represent the global map as a set of connected local maps stored in nodes Yi et al. (2012). As stated by Lowry et al. Lowry et al. (2016), a topological map is conceptually a biological, cognitive map, where nodes represent possible places in the world and edges the possible paths that connect these nodes. These maps can have several benefits. In particular, more intelligent memory management can be done since the global map is partitioned and only sub-maps are loaded at each instant. Finally, robots can conjugate the previously mentioned types of maps with more complex and informative types such as semantic maps as represented in Fig. 3. Bringing semantics to the robotic mapping procedures can improve localization systems Walter et al. (2013) and provide essential information about the agents present in the environment. For these reasons, this work fuses a monocular camera for semantic mapping with an RTK GNSS, IMU sensor, a 3D LiDAR, and wheel odometry for localization and 3D mapping to compute reliable robot self-localization and diverse map representations

with meaningful information. This approach is divided into three main stages: the localization and 3D mapping algorithm, the vineyard's semantic perception, and finally, the visual semantic mapping approach.

2.1 Localization and Mapping Algorithms

Localization and mapping, and more specifically SLAM, is one of the most common approaches used to perform robot self-localization with the advantage that a map of the environment where the robot is inserted is also generated. The main goal of SLAM is to estimate the following posterior distribution:

$$p(X_k, m \mid Z_k, U_k, x_0), \tag{1}$$

where X_k can either be the robot trajectory or the robot pose at the instant k, m represents the map, Z_k the observations, U_k the control inputs and x_0 the initial robot pose. To estimate (1) many approaches have been proposed. The Extended Kalman Filter (EKF) Paz et al. (2008); Bailey et al. (2006) is one of the most popular. This method is suitable for non-linear state and observation models and considers that a Gaussian distribution can approximate the state uncertainty. Without this restriction, Particle Filters (PF)s are also popular. These algorithms are based on Monte Carlo sampling, and they discretize the state space in a well-defined number of particles to calculate the posterior. Each particle encodes information about the robot pose and contains the map data. Besides these filter-based techniques, SLAM can also be approached using optimization. Graph-SLAM Lu and Milios (1997) is a well-known optimization approach that solves the SLAM problem by representing the robot and the map as a set of nodes and edges in a graph-way procedure.

All the mentioned approaches can fuse different types and modalities of data. In this context, Chiang et al. Chiang et al. (2019) propose a grid-based SLAM approach that fuses a GNSS sensor with an inertial sensor and a LiDAR. The authors use an EKF to fuse the three types of sensors and compute a grid-based SLAM using the LiDAR sensor. Canedo-Rodríguez et al. Canedo-Rodríguez et al. (2016) propose a multi-sensor fusion algorithm based on a PF for robot localization in crowded environments. In this work, the authors use a 2D laser range finder, a WiFi positioning system, a magnetic compass and a network of USB webcams. The main novelty of this work is the ability to fuse onboard and off-board sensors

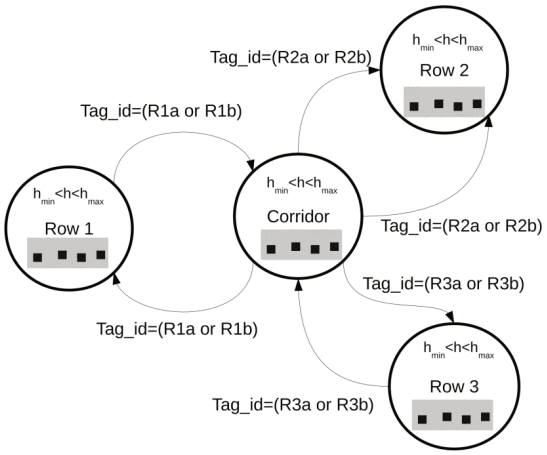


Figure 3. Semantic and topological map of a vineyard Neves Dos Santos et al. (2015). In this map, nodes represent well-defined places and contain semantic information: they either represent a vineyard row or corridor. Local maps of artificial landmarks are stored in each node. Edges encode the geometric path between nodes.

to compute the robot localization. Zureiki and Devy Zureiki and Devy (2008) fuse a 3D LiDAR sensor with a monocular camera in order to perform SLAM and build a map of planar features and lines. This algorithm uses region growing and an EKF to estimate the parameters of the 3D planar features and compute line features with the fusion of both sensors.

In agriculture, SLAM is still a research area under development due to the challenging characteristics of the environment. In this regard, Auat Cheein et al. Auat Cheein et al. (2011) propose a method based on an Extended Information Filter (EIF) for olive groves. This approach fuses a monocular vision system with a range sensor to detect and map olive stems. The authors extend this work in order to estimate the treetop volume applying a convex hull to the 3D LiDAR data Auat Cheein and Guivant (2014). In order to automate the mapping procedure in agriculture, Zhao et al. Zhao et al. (2020) propose a SLAM algorithm called Mesh-SLAM that uses a mesh-based algorithm to reconstruct the environment frame by frame. Shu et al. Shu et al. (2021) propose a monocular visual SLAM approach in a challenging dynamic agricultural environment. The authors propose an unsupervised depth estimation approach and show that their method is suitable for mapping the crop. Libby and Kantor Libby and Kantor (2011) fuse a laser sensor with wheel odometry using an EKF to estimate the robot pose in an orchard. This work innovates the state-of-the-art through a filter with two update steps, one using point features and the other using linear features extracted from three rows.

Our approach brings SLAM to challenging vineyards, operating in long planar vineyards and environments characterized by harsh steep slopes. This paper presents a novel PF-based approach capable of fusing information from various sensors such as RTK GNSS, IMU, 3D LiDAR and wheel odometry that is relevant when dealing with challenging outdoor environments. The novelty of this approach consists of introducting of the PF in SLAM approaches designed for agriculture and in the flexible and vast fusion of data that the filter supports.

2.2 Semantic Perception

The semantic characterization of the robot working environment is important in the interpretation of the state of the scene Kostavelis and Gasteratos (2013) and can also be useful to provide information for further human use. In particular, there has been a growth in the use of robots associated with processes that require thinking Crespo et al. (2020). For this reason, robots should be equipped with reliable sensors that endow them with perception systems capable of understanding their surroundings. One of the most common approaches is to use camera sensors to classify, detect or segment objects and agents semantically. In addition to traditional computer vision algorithms, DL is nowadays a widely used technique for this purpose Hao et al. (2020), In agriculture, DL is being used in the detection of natural features for multiple navigation and exploration purposes Santos et al. (2020a).

In vineyards, semantic perception is focused on detecting the presence and location of vine stems and grape bunches. This topic has been approached in the state-of-the-art in multiple manners and with several purposes. For example, Perez-Zavala et al. Pérez-Zavala et al. (2018) cluster image pixels into grape bunches using shape descriptors and texture information. This work aims essentially to automate grapevine monitoring, spraying, lead thinning and harvesting. Liu et al. Liu et al. (2018) detect grapevine flowers for determining potential early yields. To estimate the number of grape flowers per inflorescence automatically, Diago et al. Diago et al. (2014) implement an image segmentation algorithm in the CIELAB color space. Similarly, Palacios et al. Palacios et al. (2020) perform DL- and computer vision-based inflorescence segmentation and flower detection. In addition, this work presents the novelty of performing the perception at night using artificial illumination.

Several works tackle the problem in different ways concerning vine stem detection. With the purpose of grape harvesting, Badeka et al. (2021) use DL for vine trunk detection. In this work, three different DL models were trained to achieve a fast and accurate detection. Our previous works Aguiar et al. (2021); Pinto de Aguiar et al. (2020); Aguiar et al. (2020b) also approach the problem of vine stem detection. In these, we used low-power and high-performance devices for real-time inference in an Edge-AI fashion. Other works also implement the detection of trunks in other fields such as orchards Shalal et al. (2013); Colmenero-Martinez et al. (2018), oil-palm plantations Juman et al. (2016), and forests Lamprecht et al. (2015).

In this work, the semantic perception pipeline can detect vine stems and grape bunches using a state-of-the-art DL model for object detection. To do so, we propose a novel public dataset containing 2281 original images of several vineyards, extended to 22270 with augmentation. The dataset comprehends

images of the vineyard in different stages of the year (winter and summer), with different grape bunch growth stages (two different classes: tiny and medium). We also provide the fruit and trunk annotations, enabling fast-forward DL model training to the community. The dataset is publicly available in the following link: `https://doi.org/10.5281/zenodo.5114142`.

2.3 Semantic Mapping

Fruit detection, localization, and mapping are key components of the new generation of agricultural robots. They allow accurate fruit counting and yield estimation. Additionally, they can be important in the detection of fruit diseases Arnal Barbedo (2019), and they constitute the preliminary step to the automation of processes such as harvesting, spraying, crop monitoring, and others Rahnemoonfar and Sheppard (2017).

In the current state-of-the-art, many works focus on visual-only fruit counting. For example, Kestur et al. Kestur et al. (2019) present a detection and counting pipeline for mango fruits in images for further yield estimation. In this work, the authors use MangoNet, a CNN for mango detection using semantic segmentation. For the detection and counting of small passion fruits, Tu et al. Tu et al. (2020) propose a multi-scale, faster region-based convolutional neural network approach using both color and depth images. A detector for each image type is trained separately and then fused. The authors show that the system can perform in different lightning and occlusion conditions. Some works propose the detection and counting of grapes in vineyards and orchards. Santos et al. Santos et al. (2020b) detect, segment, and track grape bunches using state-of-the-art CNNs. This work makes available a public dataset with 300 RGB vineyard images showing 4,432 grape bunches from five different varieties. Also, using DL, Zabawa et al. Zabawa et al. (2020) deploy a Convolutional Neural Network (CNN) to detect and count single grape berries. This work uses a semantic segmentation technique to perform pixel-wise classification.

In addition to detecting and counting, some works already focus on creating maps of fruits and other agricultural agents, with the 2D or 3D location of them. Other works focus on mapping the agricultural environment to recover its 3D structure. In this context, Liu et al. Liu et al. (2019) propose a cost-effective and lightweight fruit and tree trunk counting and mapping pipeline. This work estimates the image location of mangoes and tree trunks on images using a CNN. To track and map their location in 3D, a KF is used to fuse the CNN output with a Structure for Motion (SfM) algorithm that tracks the camera pose. Dong et al. Dong et al. (2019) build 3D maps of orchard rows with semantic information. In this work, the two row sides are reconstructed and fitted using global features and semantic information. Here, semantics are used in the mapping procedure to find the common information between the two row sides. The output of this system is a 3D model of the orchard, where information such as canopy volume, trunk diameter, tree height and fruit counting can be extracted.

In this work, we propose a semantic mapping pipeline for grape bunches and vine stems. Our approach uses the semantic visual perception pipeline that retrieves the image location of the natural features, fusing this information with a 3D LiDAR to compute depth information. The mapping pipeline is based on a single EKF algorithm for each landmark, as proposed in FAST-SLAM Montemerlo et al. (2002). Trunks are mapped in two dimensions, using the exact model proposed in FAST-SLAM. To map grape bunches, we propose an extension of the state-of-the-art EKF model to work in three dimensions.

3 SEMANTIC VINEYARD NAVIGATION AND MAPPING

In this work we propose a novel pipeline for robotic navigation and semantic mapping in vineyards. Figure 4 represents the high-level architecture of the system. The proposed approach is segmented into three main stages:

1. A 6-DoF SLAM approach for unstructured agricultural environments that fuses four different sensor modalities: an RTK-GNSS, an IMU, wheel odometry, and a long-range 3D LiDAR;
2. A semantic vineyard perception approach that uses a state-of-the-art DL object detection model to detect grape bunches in different growth stages and vine stems;
3. A semantic mapping algorithm that computes the location of the grape bunches and vine stems in the real world.

The outputs of this online approach are an estimation of the robot path and a set of maps (semantic, 3D feature map). Since the SLAM output 3D map does not use all the 3D LiDAR information (it uses only

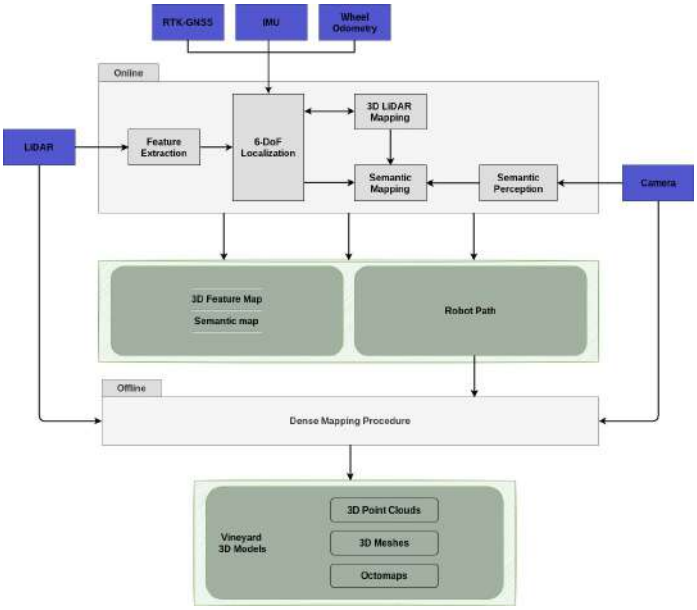


Figure 4. Localization and Mapping pipeline architecture. Our system fuses four different sensors to accomplish the SLAM algorithm: an RTK-GNSS, an IMU, wheel odometry, and a 3D LiDAR. Also, a monocular camera is integrated and fused with the LiDAR 3D map to build the semantic vineyard map of grape bunches and vine stems. The online SLAM and semantic mapping procedures output is the robot travelled path and the built maps. These outputs, along with the LiDAR raw observations, are used in an offline dense mapping procedure where 3D models of the vineyard are constructed.

only local features extracted from the input cloud), a dense mapping procedure was created to build precise maps of the environments considering all the data provided by the 3D LiDAR. Thus, our system can create 3D models of the vineyard in different formats such as point clouds, 3D meshes, and octomaps. This proposed approach uses the output robot path of SLAM and the raw 3D LiDAR observations to create the 3D models of the vineyard.

The robotic platform presented in Fig. 5 was used to test the developed approach. This hardware platform is equipped with a 3D LiDAR, several RGB and RGB-D cameras, a thermal camera, an RTK-GNSS, an IMU and a robotic arm.

The following sections describe the proposed approach in detail.

3.1 A SLAM Approach for Agricultural Environments

In order to compute robust and reliable localization and mapping procedures in unstructured environments such as agriculture, this paper proposes a SLAM approach based on a sensor fusion technique. This method is based on a PF algorithm capable of capturing information from different sensor modalities and using it to compute the robot pose. In the mapping procedure, a 3D voxel grid map was created to store the features extracted from the 3D LiDAR data. This data structure implements a fast and efficient search method that is crucial for the effective performance of the PF. This being said, the SLAM algorithm can be described in three different phases: feature extraction, mapping, and localization.

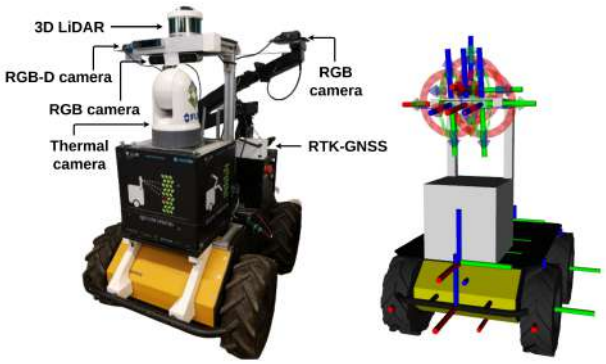


Figure 5. (left) Robotic platform used to test the developed approach and (right) the correspondent 3D computational model of the robot. The hardware platform is equipped with a 3D LiDAR (Velodyne Puck 16), several RGB (OAK-D and ZED cameras) and RGB-D (Intel RealSense Depth Camera D415) cameras, a thermal camera (FLIR M232), an RTK-GNSS (simpleRTK2B), an IMU (CH Robotics UM7) and a robotic arm.

3.1.1 Feature Extraction

In the feature extraction phase, two different types of features are extracted from the raw 3D LiDAR sensor data: corner and planar features. The first are characterized by being located in sharp edges, and the second in planar surfaces. To extract this information from the agricultural environment, a state-of-the-art approach is used, where the *smoothness* descriptor Shan and Englot (2018) is calculated as described in the follow equation

$$c = \frac{1}{|S| \cdot ||p_i||} \Big|\Big| \sum_{j \in S, j \neq i} (p_j - p_i) \Big|\Big|, \tag{2}$$

where p_i is the ith point in the set of continuous points S. After computing the descriptor, points are projected into a range image and sorted by their value of c. The points with larger c are considered edge features, and the ones with lower c are considered planar features. Figure 6 shows an example of this feature extraction procedure in a woody crop vineyard.

3.1.2 3D LiDAR Mapping

In the mapping stage, a registration procedure is proposed to store the point features in a voxel grid data structure. This map structure is implemented and used to store the point feature map and perform efficient search algorithms. This map is an extension of the standard 2D grid map, which also considers discretization in the z component. From the implementation point of view, the 3D voxel grid map is a set of interconnected 2D grid maps, each one corresponding to a certain z discrete value. Each cell is indexed by specific 3D discrete coordinates and can be efficiently accessed with them. Also, the map recognizes the different types of features supported. Thus, each cell can contain several types of features, and the searching procedures account for each type for faster processing. With this data structure, the point-based mapping procedure is performed using the information about the robot pose provided by the localization layer and using a local search algorithm. Given a set of input point features $M_\lambda = \{m_{\lambda_1}, m_{\lambda_2}, \ldots, m_{\lambda_N}\}$ in their homogeneous form and the robot's pose T_r, each feature is converted to the maps' referential frame as follows:

$$\tilde{m}_{\lambda_i} = m_{\lambda_i} \cdot T_r, \; i \in \{1, \ldots, N\}. \tag{3}$$

Then, a local search is performed for each feature in the 3D voxel grid map. The nearest neighbor of each feature is searched by the procedure represented in Fig. 7. Each feature's nearest neighbor can be located

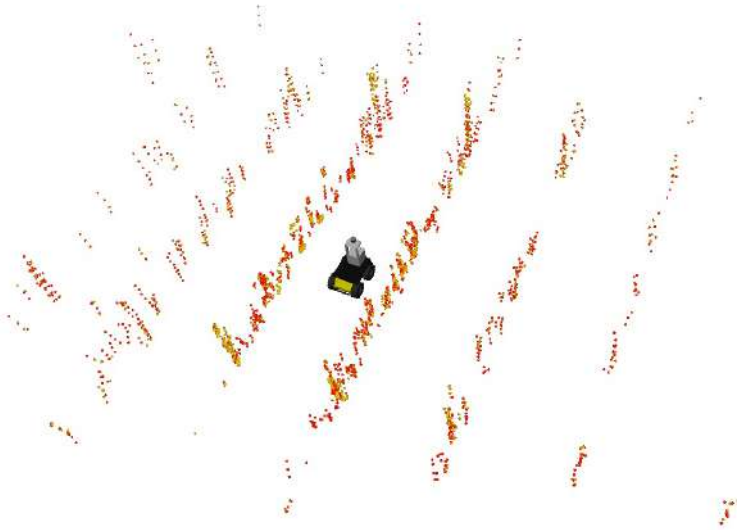


Figure 6. Corner (yellow) and planar (red) feature extraction example in a woody crop vineyard.

either in its cell, in adjacent cells, or even in more distant cells. Since we are performing 3D SLAM, the search is performed in three map layers: the top and bottom adjacent layers and the layer where the feature is located. Depending on the voxel resolution, the user can specify if the local search algorithm looks for neighbors just in adjacent cells or if it continues for more remote cells in case of failure. This decision sets the stop criteria of the algorithm that iteratively looks for the nearest feature in a region using a well-defined path as specified in Figure 7. In cases where no neighbor is found, the feature is registered and initialized in the voxel map.

3.1.3 Localization

The localization procedure uses the LiDAR features computed at each instant, the map built so far, and a set of sensors to compute the 6-DoF robot pose. This process is implemented employing a PF that fuses all the information and retrieves a robust localization estimation. The PF is standardly divided into three main steps: 1) the prediction step, where the particles are innovated using a motion model; 2) the update step where the weights of the particles are calculated using the sensors' observations, and; 3) the resampling step where low-weight particles are substituted by particles with high weight.

In the motion model, the wheel encoders and a gyroscope are used to innovate the particles state. Let $X_k = [x, y, z, \phi, \psi, \theta]$ represent the robot pose state at the instant k, and let us define the following matrices:

- ΔT: the homogeneous transformation that constitute the increment computed by the odometry and gyroscope measures; and
- T_n: an homogeneous transformation sampled from a Gaussian distribution with standard deviation d_u.

Thus, a particle j is innovated at the instante k as follows:

$$X_{j,k} = \Gamma\Big(\Gamma^{-1}\big(X_{j,k-1}\big) \cdot \Delta T \cdot T_n\Big), \tag{4}$$

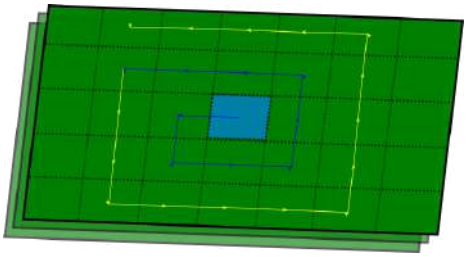


Figure 7. Point feature nearest neighbor local search. In a discrete 3D space, the nearest neighbor of a point feature can be in the grid map layer where the feature is located or at the top and bottom adjacent layers. A local search in these three layers is performed to find the nearest feature as described in the figure. If a feature is found when searching in the black path, the search ends. Otherwise, the search continues through the yellow path. The user can tune the stop criteria.

where $\Gamma(.)$ is the operator that transforms a homogeneous transformation in the 6-DoF cartesian pose state with Euler representation. The innovation increment ΔT is computed fusing the gyroscope and wheel odometry. For the components observed by both sensors, the average of both measures is computed.

After innovating the particles in step 1), their corresponding weights are calculated in step 2). This step represents the main novelty of the filter since it is capable of fusing information from multiple sensors considering multiple data modalities to compute each particle weight. To account for all sensor and data modalities, a sub-function W_i is computed for each observation type (in this case for the ith observation type). Then, all the sub-functions are combined in a single function W to calculate the final particle weight. Table 1 summarizes the information used in the filter to compute the robot pose.

Table 1. Summary of the sensors used in the update step of the PF. Each sensor has one (or more, in the case of the 3D LiDAR where two types of features are extracted) weight function. It is important to note that each modality provides information for different components of the robot pose, and that the filter must be capable of accounting all the information.

Sensor	Sub-function	Information
3D LiDAR	W_c W_p	*Corner and planar feature aligment.* *6-DoF pose calculation.*
RTK-GNSS	W_r	*Absolute position correction with altitude information.*
IMU	W_u	*Absolute orientation correction with three axes information.*

To compute the weight functions related to the 3D LiDAR, the feature extraction procedure described in section 3.1.1 is used. This algorithm extracts corner and planar features are extracted from a raw 3D point cloud, constituting two local feature maps in the robot's reference frame. As described in section 3.1.2, both types of features build a global 3D map and are stored in a voxel grid map structure. Thus, this data modality can be considered to calculate the particles' weight by computing the alignment error between the two local maps and the global map, considering each particle pose. To do so, firstly, the N local features are transformed into the global map reference frame as follows:

$$\tilde{m}_{\mathscr{C}_i} = m_{\mathscr{C}_i} \cdot \Gamma^{-1}\left(X_{j,k}\right),\ i \in \{1,\ldots,N\}, \tag{5}$$

where $X_{j,k}$ represents the jth particle pose at the instant k, and $\mathscr{C} \in \{c,p\}$ represents the corner and planar features subscript (see Table 1). Then, a searching procedure takes place to find correspondences between the projected local features and the features stored in the global map, as described in the 3D

mapping procedure detailed before. Considering the set of K correspondences found $\{\tilde{m}_{\mathscr{C}_i} \leftrightarrow m_{\mathscr{C},g_i} : i \in \{1,\ldots,K\}\}$, where the subscript g denotes for features in the *global* map, the point-feature weight sub-function is computed by the following equation:

$$W_{\mathscr{C}} = \frac{1}{\sqrt{2\pi}\sigma_{\mathscr{C}}} \sum_{i=1}^{K} \exp\left(\frac{-1}{\sigma_{\mathscr{C}}} \cdot ||\tilde{m}_{\mathscr{C}_i} - \tilde{m}_{\mathscr{C},g_i}||\right), \tag{6}$$

where $\sigma_{\mathscr{C}}$ is the standard deviation of the point-feature measurement, and $||.||$ represents the L2 norm. Equation 6 states that the particles weight increases exponentially with the decrease of distance between feature correspondences, and with the number of correspondences found.

For the RTK and IMU sensors, the weight function is designed in the same way, i.e., a negative exponential function for each sensor. In this case, the particle weight calculation can be directly extracted without transformations since both sensors provide absolute observations. This being said, considering the RTK absolute robot position observation $X_{r,xyz}$ and the jth particle position $X_{j,xyz}$, the particle weight is calculated as follows:

$$W_r = \frac{1}{\sqrt{2\pi}\sigma_r} \exp\left(\frac{-1}{\sigma_r} \cdot ||X_{r,xyz} - X_{j,xyz}||\right), \tag{7}$$

where σ_r represents the standard deviation of the RTK position observation. It is worth noting that during the experiments detailed in section 4 the GNSS-RTK was not used to perform a fair comparison of the proposed algorithm's localization precision with state-of-the-art approaches. For the IMU sensor, the weight function is adapted to work with the Euler angle representation. Let $\mathscr{L} = \{\phi, \psi, \theta\}$ represent the three orientation degrees of freedom and $X_{\mathscr{L}}$ represent a vector with the three orientations. The particle weight is calculated applying the following equation:

$$W_u = \frac{1}{\sqrt{2\pi}\sigma_u} \prod_{\eta \in \mathscr{L}} \exp\left|X_{u,\eta} - X_{j,\eta}\right|, \tag{8}$$

where σ_u represents the standard deviation of the IMU observation, $X_{.,\eta}$ represents the extraction of a single orientation component from the pose vectors, and $|.|$ represents the absolute value. Thus, in this case, the particle weight is computed as the product of differences between the IMU absolute orientations and the particle's orientations.

To fuse all the weight subfunctions in a single function and compute the final particle weight, we multiply all the sub-function results as follows:

$$W = W_c \cdot W_p \cdot W_r \cdot W_u. \tag{9}$$

The balance between the multiple modalities is controlled by the standard deviation of each observation.

To substitute low-weight by high-weight particles, the filter loop ends with the resampling step. In this work, the multinomial resample algorithm Douc and Cappe (2005) was implemented to accomplish this. This approach draws N samples from a uniform distribution u_i and selects the particle j for replication if

$$u_i \in \Big[\sum_{p=1}^{j-1} W_p, \sum_{p=1}^{j} W_p\Big), \tag{10}$$

where W_p represents the particle's p weight. To avoid the well-known problem of particle degeneracy that happens when either all the particles are in the wrong place or they are highly condensed, resampling is not executed for all iterations. This method is only employed when a significant robot motion is observed (either in translation or rotation in the six degrees of freedom). The user can set the amount of motion required to perform resampling.

The proposed SLAM approach is designed taking in consideration several well-known issues inherent to the PF approach. In woody-crop vineyards characterized by long and symmetric corridors, one of the major challenges is the aliasing in the environment, such as estimating high likelihoods for the robot pose in consecutive parallel rows. In this work, we tackle this issue in two ways. First, for the particle innovation step, six different parameters are tuned to decide the particles' innovation in each of the 6-DoF

components, and consequently, the particles' dispersion. In long and symmetric corridors, the particles' dispersion is modeled so that the filter has more uncertainty in the forward component of motion. This means that particles will not be spread to different corridors unless some error occurs during the weight calculation procedure. The second factor that addresses the aliasing issue is the fusion of sensors that retrieve absolute measures for the robot pose. If, for some reason, the LiDAR sub-function of the PF estimates several peaks for parallel rows, the GNSS sensor will retrieve a higher likelihood for one of the rows. The multimodal fusion of weights present in (9) enables the PF to know what is the most likely row where the robot is located.

Another common problem related to PFs is particle depletion, i.e., the lack of particles corresponding to the true robot pose solution. In this work, this is avoided through the resample algorithm and sensor fusion. As referenced before, the resampling step is only executed when the robot motion changes a certain amount, either in the translation or the rotation components. Thus, low-weight particles are only substituted by high-weight particles in these critical frames. If, for every frame, particles are substituted, this would lead to a high reduction in their dispersion if we maintain the same number of particles. Also, the fusion of data from many sensors leads to the requirement of fewer particles since the filter will be more confident of the robot pose in each iteration. This means that particles' dispersion will be closer to the true robot pose in each instant, and fewer particles are necessary.

The SLAM pipeline operation in a real vineyard can be visualized in the following video: `https://youtu.be/xm6VQVrkceE`.

3.2 Semantic Perception of the Vineyard

After having a reliable and robust robot localization in the agricultural environment and to create a semantic map of the vineyard, there is an intermediate step where the natural features are extracted. In this work, we propose a DL-based approach to detect vine stems and grape bunches in two different growth stages in RGB images. To feed the DL model with substantial amount of data, a dense data collection procedure was carried out. A dataset with 2281 images was created by collecting vineyard visual data with different cameras, in multiple vineyards, in different year stages, and with different grape bunch growth stages. Our publicly available dataset (at `https://doi.org/10.5281/zenodo.5114142`) provides not only the vineyard images but also the natural feature annotations so that it can be used in the scientific community train to directly DL models. It is worth noting that, to build the proposed dataset, we went to the multiple vineyards many times, during the winter and the summer. Figure 8 presents an overview of the dataset, where one can see that two different grape growth stages were captured with a camera pointing directly to the canopy, but also that it contains frontal images of different vineyards, including a thermal camera. Thus, the dataset contains three different classes: `tiny_grape_bunch`, `medium_grape_bunch`, and `stem`. The first class represents grape bunches in an early growth stage, and the second in an intermediate growth stage. To have these two classes in the dataset, the data collection procedure was performed with an interval of approximately one and a half months. For detecting and mapping vine stems (the third class), the dataset contains forward- and side-view RGB and thermal images of multiple vineyards in different years. It is worth noting that the entire dataset contains 101 thermal images and 2180 RGB images.

To increase the dataset size and data variability, a set of augmentation techniques were applied to the original images. In this process, the images were rotated by several angles, translated, flipped, and multiplied to vary their contrast. Table 2 presents an analysis of the original and augmented dataset. The

Table 2. Number of annotated objects per class. The original dataset contains **2281** images with three different classes. To increase the dataset size, several augmentation operations were applied, increasing the number of images to **22270**.

Class	# objects	# augmented objects
`tiny_grape_bunch`	2497	13393
`medium_grape_bunch`	4292	25189
`stem`	6375	51130

last step to build the dataset is image cropping. As will be mentioned, the model used has a hyperparameter

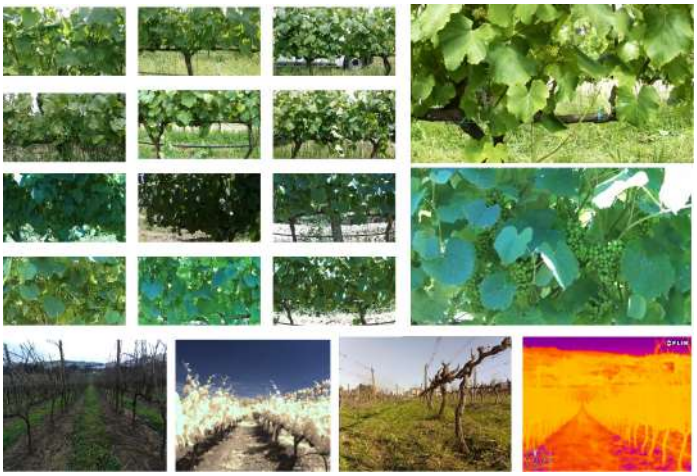


Figure 8. Public available dataset containing images of several vineyards in different stages of the year. Our dataset contains images of grape bunches in two different growth stages with different luminosity conditions. Additionally, it provides images with two different perspectives - one point directly to the canopy and another point in the direction of the robot path. Finally, besides RGB images, the dataset also contains a set of thermal images.

that tunes the input resolution of the images. Thus, if the original images used for training have a different resolution, the training pipeline will rescale them to match the resolution hyperparameter. Due to this, the dataset is extended by cropping all the images so that they match the resolution set for training (in this case, 300x300). Without splitting these images, they would be resized to a lower resolution, and a significant amount of data would be lost in this process. On the contrary, if we split high-resolution images, no resize operation would be performed by the DL model when performing image inference. Then all the data collected would be used. For an image with a 1920 × 1080 px resolution, 40 other images are generated with a of 300 × 300 px resolution. This results in a total of 302,252 images present in the dataset.

Finally, after having the dataset collected and annotated, a solution for fast and efficient image inference was implemented. The state-of-the-art lightweight DL model SSD MobileNet-V1 Howard et al. (2017) was used for this purpose. This model is one of the most popular among the state-of-the-art models designed to run on low-power and low-cost embedded devices. The input of the CNN is a tensor with shape $D_f \times D_f \times M$, where D_f represents the input channel spatial width and height, and M is the input depth. After the convolution, a feature map of shape $D_f \times D_f \times N$ is obtained, where N is the output depth. Thus, the model contains two hyperparameters that the user can tune to optimize the CNN performance. The first, width multiplier α, can be used to decrease the model size uniformly at each layer by a factor of α^2. The second hyperparameter, resolution multiplier, ρ, can be used to reduce the computational cost of the model by a factor of ρ^2 by changing the input image resolution accordingly. In this work, we only tuned the resolution multiplier parameter, using input images of 300x300 pixels. The depth multiplier was set to one to not affect the model performance. The model, pretrained with the Coco dataset, was fine-tuned to learn to detect the three classes existent in the proposed dataset. The training considered 50000 iterations and a batch size of 24. After this, the model was deployed in a low-cost embedded hardware Tensor Processing Unit (TPU)[1]. The following video demonstrates the time-effective inference

[1] https://coral.ai/products/accelerator/

procedure running on the TPU device: `https://youtu.be/B8581lfpcgI`.

3.3 Mapping Grape Bunches and Vine Stems

With the ability to localize the robot precisely and detect vine stems and grape bunches on images, all the conditions are met to perform a semantic mapping of the vineyard. In this work, we approach this problem in the same way FAST-SLAM Montemerlo et al. (2002) maps landmarks, i.e., considering a unique EKF for each landmark. For the vine stems, since their origin is always at the ground, the mapping is performed in two dimensions. Thus, we discard their z component and compute only their position in the XoY plane. Due to this, the exact FAST-SLAM landmark mapping model is applied for this type of landmarks. On the contrary, to map grape bunches, a 3D model is required to extract their precise location. This paper proposes the extension of the FAST-SLAM EKF model to map 3D landmarks. Figure 9 demonstrates the two configurations for both the 2D and 3D cases from the point-of-view of the robot.

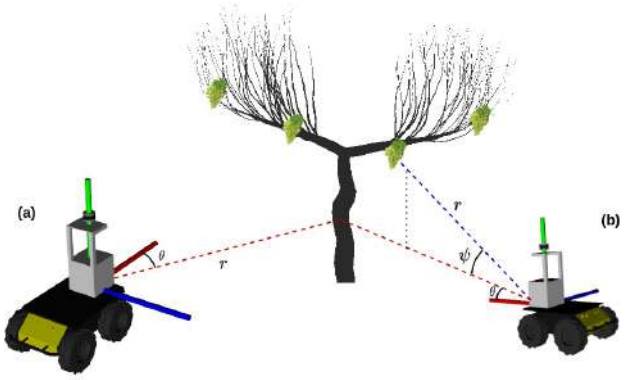


Figure 9. Semantic feature observation. (a) In the 2D case the robot observes the depth and bearing in relation with the feature; (b) in 3D, the robot also observes the pitch angle between the robot and the feature in order to calculate its 3D location.

The observation of the jth semantic feature at the k time instant $y_{j,k} = [r_{j,k}, \psi_{j,k}, \theta_{j,k}]^T$ is represented by the landmark distance to the robot $r_{j,k}$, the yaw angle between the robot and the landmark $\theta_{j,k}$, and in the case of the grape bunches (the 3D case), the landmark pitch in relation with the robot $\psi_{j,k}$. To extract these three values for each natural feature, a monocular RGB camera is fused with a 3D LiDAR using the SLAM approach. The two angles θ and ψ can be directly extracted from the image with the information of the camera intrinsic matrix. However, the feature depth is not observable with a single camera. Thus, as represented in Fig. 10, to calculate the depth observation, we compute the intersection between the ray that goes from the robot to the feature with the 3D map built so far using the LiDAR sensor in the SLAM procedure.

After having the semantic features observation, an EKF is applied to each one to compute their location in the map. Let $X_k = [x_{r,k}, y_{r,k}, z_{r,k}, \phi_{r,k}, \psi_{r,k}, \theta_{r,k}]^T$ be the robot pose calculated by the SLAM approach and $M_{\gamma_k} = \{m_{\gamma_1}, m_{\gamma_2}, ..., m_{\gamma_K}\}_k$ the vector of K landmarks on the map at the instant k. Since the semantic features are considered static, i.e., they do not change their location between observations, the state model of the EKF declares that the position of each landmark j with state vector $m_{\gamma,j,k} = [x_{\gamma,j,k}, y_{\gamma,j,k}, z_{\gamma,j,k}]^T$ is calculated as follows:

$$m_{\gamma,j,k} = m_{\gamma,j,k-1}. \tag{11}$$

With the configuration present in Fig. 9, the observation parameters are used to compute the observation model and adjust the feature location in each iteration. The observation vector $\hat{y}_{j,k} = [\hat{r}_{j,k}, \hat{\psi}_{j,k}, \hat{\theta}_{j,k}]^T$ is

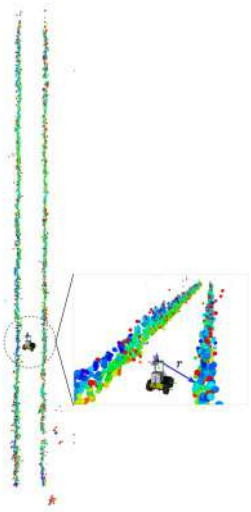


Figure 10. 3D vineyard map built in the SLAM process with the 3D LiDAR, and the semantic feature depth calculation by intersection of the ray that points to the feature and the map.

updated as follows:

$$\hat{y}_{j,k} = \begin{bmatrix} \sqrt{\Delta x_k^2 + \Delta y_k^2 + \Delta z_k^2} \\ \arctan\left(\frac{\Delta z_k}{\sqrt{\Delta x_k^2 + \Delta y_k^2}}\right) - \psi_{r,k} \\ \arctan\left(\frac{\Delta y_k}{\Delta x_k}\right) - \theta_{r,k} \end{bmatrix}, \tag{12}$$

where $[\Delta x_k, \Delta y_k, \Delta z_k]$ represent the different between the landmark and robot position at the instant k, component-wise. Since the observation model present in (12) is nonlinear, the EKF linearizes both the state and observation models about the current state estimate and covariance. The Jacobian of the state model is the identity matrix because the model is linear and static. On the contrary, the Jacobian of the observation model is computed as follows:

$$G_{j,k} = \begin{bmatrix} \frac{\Delta x_k}{r_{\gamma,j,k}} & \frac{\Delta y_k}{r_{\gamma,j,k}} & \frac{\Delta z_k}{r_{\gamma,j,k}} \\ -\frac{\Delta z_k \cdot \Delta x_k}{r_{\gamma,j,k}^2 \cdot \sqrt{\Delta x_k^2 + \Delta y_k^2}} & -\frac{\Delta z_k \cdot \Delta y_k}{r_{\gamma,j,k}^2 \cdot \sqrt{\Delta x_k^2 + \Delta y_k^2}} & \frac{\sqrt{\Delta x_k^2 + \Delta y_k^2}}{r_{\gamma,j,k}^2} \\ -\frac{\Delta y_k}{r_{\gamma,j,k}^2} & \frac{\Delta x_k}{r_{\gamma,j,k}^2} & 0 \end{bmatrix}. \tag{13}$$

Given all of the above, each landmark state is updated using the standard EKF formulation. The state estimate is updated as follows:

$$\hat{m}_{\gamma,j,k} = \hat{m}_{\gamma,j,k-1} + K_{j,k} \cdot \left(y_{j,k} - \hat{y}_{j,k}\right), \tag{14}$$

where $K_{j,k}$ is the Kalman gain computed in the EKF loop. Equation (14) calculates the semantic feature position at the instant k. This information is then stored in the voxel grid map represented in Fig. 7 like all

(a) Woody-crop Aveleda vineyard. **(b)** Seixo mountain vineyard.

Figure 11. Our robotic platform navigating in the two vineyards used as use-cases to evaluate the proposed approach.

the other features. The voxel map supports any kind of feature, as long as it has a proper template defining each type. In the case of semantic features, they should be identified by the type of natural object that they represent.

The following video demonstrates the semantic mapping procedure: `https://youtu.be/IGZ88HFt7Q0`.

4 RESULTS

To test and evaluate the proposed system, experiments were carried out in two different vineyards with different characteristics. The first, Aveleda vineyard (41°12'19.8"N 8°18'26.6"W), is a woody crop vineyard as represented in Fig. 11a with light inclinations but with high symmetry over long corridors. The second, placed in Seixo farm (41°09'59.6"N 7°33'19.0"W), is a mountain vineyard characterized by harsh steep slopes and high hills, as represented in Fig. 11b.

The SLAM approach was tested in both vinyards for two different sequences, one in each vineyard, as described in Table 3. In *seq-1* the robot traveled 282.7 meters in a long vineyard corridor and in *seq-2* the 337.1 meters along two vineyard corridors. It is important to note that, in the last sequence, the robot was exposed to harsh inclinations and went down and up more than 7 meters. Both sequences are publicly available at `https://doi.org/10.5281/zenodo.5142159` and `https://doi.org/10.5281/zenodo.5142003`

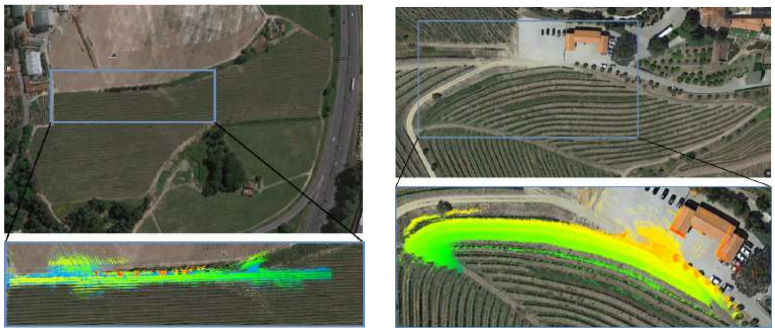
Table 3. Description of the experiments performed to evaluate the proposed SLAM approach.

Sequence	**Vineyard**	**Distance travelled** (m)	**Information**
seq-1	Aveleda	282.7	*Sequence in a long vineyard corridor*
seq-2	Seixo	337.1	*Sequence in a mountain vineyard*

The semantic perception and mapping algorithms evaluation was performed in two stages. Firstly, state-of-the-art metrics were employed to evaluate the DL object detection model performance. Then, the semantic mapping algorithm was evaluated, considering the persistency on the mapping. To do so, the robot traveled two different paths in Aveleda vineyard on two different days in order to build two maps of the same corridor. Then, an analysis of the mapping performance was carried out by comparing the two maps.

4.1 Localization and Mapping Performance

As referenced before, the SLAM approach was tested in two sequences on different vineyards. The robot was equipped with an RTK-GNSS, an IMU, a LiDAR, and an RGB camera. Wheel odometry was also available. As represented in Fig. 4, the algorithm fuses these sensors to provide reliable and robust localization and mapping outputs. In these experiments, the filter used 300 particles to discretize the 6-DoF space and calculate the robot pose.



(a) Aveleda woody-crop vineyard. **(b)** Seixo mountain vineyard.

Figure 12. 3D maps overlay over satellite images. The alignment between both representations validate the performance and robustness of the SLAM algorithm.

To evaluate and validate the performance of the SLAM system, the maps built during the process were overlapped over satellite images (Fig. 12). These images are used as ground truth for the mapping and localization procedures since they are co-dependent. In other words, if either the localization or the mapping fails, the other process will also fail. Thus, if the maps match the ground truth, we can conclude that the path traveled by the robot was well estimated, and the mapping procedure was successful. This experimental approach provides a two dimensional qualitative evaluation of the SLAM algorithm, through the bird's eye image overlapping. Figure 12 shows the overlapping between the maps built and satellite images of the vineyards where the experiments took place. For both sequences, it is possible to observe that the maps match the ground truth satellite images. In Aveleda's vineyard, the robot traveled a long path inside a highly symmetric corridor. The sequence was carried out in a stage of the year where the vineyard vegetation was extremely dense. This is challenging for SLAM algorithms since there are few places where range sensors can capture features outside the corridor to overcome its symmetry. The proposed SLAM approach can overcome the difficulty of navigating in these vineyard corridors. Figure 12a shows that the map and the satellite images overlap. In particular, the longitudinal motion component could present error since SLAM systems are often affected by corridor symmetries. In this case, the proposed algorithm shows high accuracy also in this component, as can be seen by the alignment of the trees and other agents outside the corridor. In Seixo's vineyard, the robot also traveled a long path but with different characteristics. In this case, the sequence was carried out in a stage of the year where the vineyard had no foliage. However, this vineyard presents high hills with harsh slopes. In these experiments, the robot traveled over two corridors performing an entire loop, i.e., starting and ending at the same point. The main challenge in this scenario is the estimation of the robot altitude, as well as the roll and pitch rotational components, since the robot goes down more than seven meters, and then ascends again. Figure 12b shows the overlap between the map of Seixo's vineyard created by the SLAM algorithm and the corresponding satellite image. Once again, the performance of the proposed system is validated since there is a clear overlap visible in the house, trees, and vineyard limits.

To have a three-dimensional validation of the mapping algorithm, we propose an innovative evaluation procedure. In this, a drone equipped with an RTK-GNSS and several cameras flew over the vineyard corresponding to *seq-2* and generated an RGB 3D reconstruction of the entire crop represented in Fig. 13. With this ground truth reconstruction, the map built by our SLAM algorithm was overlapped with the drone reconstruction to validate the precision of the mapping pipeline proposed. It is worth to emphasize that this is a novel evaluation approach, since we are merging together maps built by two different types of vehicles, showing that they can by used together. Figure **??** shows that there is a clear overlap between both maps, which validates the robustness of the proposed algorithm.

In Fig. 14 it is represented the path traveled by the robot in both sequences as well as the 3D vineyard models built using the dense mapping approach. In Aveleda's vineyard, the robot traveled a rectilinear

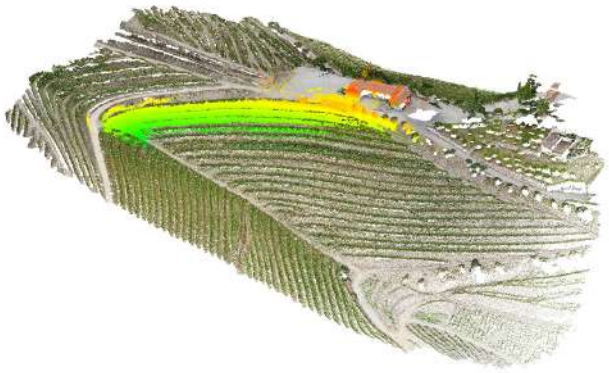


Figure 13. Comparison of the 3D reconstruction of the vineyard of *seq-2* performed by a drone (represented by the RGB pointcloud) and the 3D map generated by the proposed SLAM algorithm (represented by the yellow and green colormap pointcloud).

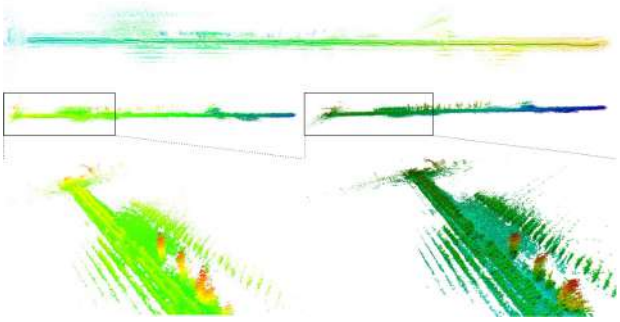
path over the corridor, as can be observed. On the contrary, in Seixo, the robot completes an entire loop in two different corridors. Figure 14b highlights the challenges of this sequence where the robot went up and down over steep slopes. In addition, this figure also shows the variety of maps supported by the proposed pipeline. The proposed approach cam export maps in their conventional format (point clouds) but also as meshes and octomaps. We highlight this feature since each representation can present advantages in the post-analysis of the built maps. The maps can be used by agronomists to extract key information of the vineyards such as the foliage density. The support for different map types leads to a more complete representation of the environment, which can be helpful in the understanding of its main characteristics.

To have a more quantifiable evaluation of the localization algorithm, the proposed approach was compared against two state-of-the-art SLAM algorithms: LOAM Zhang and Singh (2014) and LeGO-LOAM Shan and Englot (2018). For this purpose, *seq-2* was used since it is more challenging and allows to evaluate all the 6-DoF of the robot's pose estimation. To have a fair comparison with the state-of-the-art approaches, the RTK-GNSS is not used in the localization process, and is instead used as ground truth. The localization performance is quantified in the three dimensions component-wise, and considering the mean absolute error for the entire trajectory. This error is computed as follows:

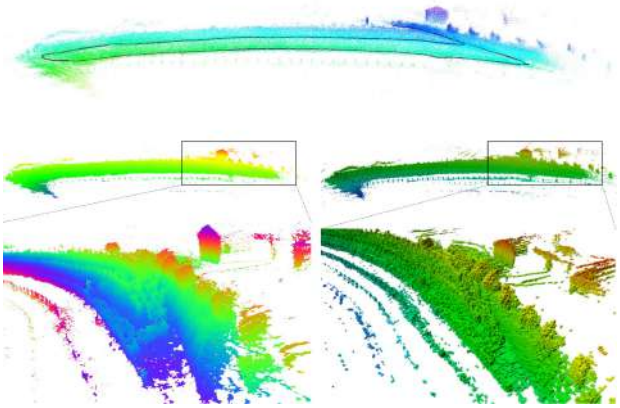
$$\omega = \frac{1}{N}\sum_{i}^{N} e\left(X_i^G - X_i\right), \tag{15}$$

where e represents the euclidean distance operator, N represents the number of localization samples estimated by the algorithms along the trajectory, X_i^G represents the ground truth 3D position at the instant i and X_i represents the 3D position estimated by the localization system at the instant i. Table 4 shows the error of the three algorithms in comparison with the GNSS-RTK ground truth. Figure 15 represents the trajectory calculated by the three algorithms in this sequence. From these experiments, one can see that LeGO-LOAM lost track of the robot pose at a given instant, presenting a high error. On the other hand, our approach and LOAM perform well, presenting low errors for each component, and an overall low absolute error. LOAM is the algorithm that presents the lower error, which can be explained by the presence of a loop closure algorithm that corrected the robot pose between the start an the end points. This evaluation validates the performance of the proposed algorithm under harsh outdoor conditions, showing that even without loop closure it presents low drift accumulation, with an absolute error of 0.902 meters during the entire trajectory.

As a final remark, the proposed SLAM approach was successfully integrated into a navigation stack and was used for autonomously driving the robot in the Aveleda vineyard. The next video shows a



(a) **(top)** Top view of the map and robot trajectory on Aveleda vineyard; **(bottom-left)** mesh version of the original map; **(bottom-rigth)** octomap version of the original map.



(b) **(top)** Side view of the map and robot trajectory on Seixo vineyard; **(bottom-left)** mesh version of the original map; **(bottom-rigth)** octomap version of the original map.

Figure 14. 3D maps built using the dense mapping approach. The algorithm is able to export maps in three different formats: a standard 3D point cloud, a 3D mesh, and a 3D octomap.

percentage of the autonomous path traveled by the robot: `https://youtu.be/OMzXx1fLsnM`.

4.2 Semantic Perception and Mapping Performance

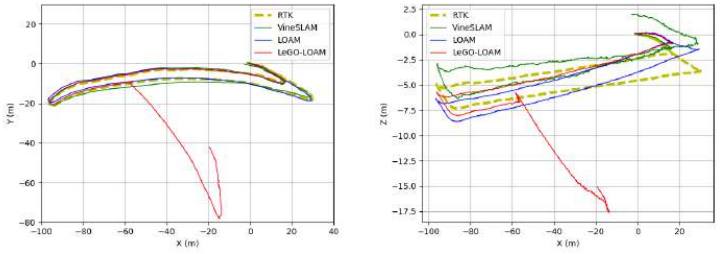
As referenced before, the semantic perception and mapping processes were evaluated in two different phases. Firstly, the DL model was evaluated using a set of metrics, and then the mapping procedure was analized by mapping the same vineyard corridor in two different days and two growth stages. To perform a fair evaluation of the DL models, the input dataset was divided into three groups: training (80%), test (10%), and evaluation (10%). The larger one, the training set, was used to train the DL models. The test set was used to perform the evaluation of the models during the training by Tensorflow. The evaluation set was exclusively used to test the models by computing the metrics described above.

In this work, five different parameters were used to evaluate the DL model: precision p, recall r, F1 score, Average Precision (AP), and medium AP (mAP). The first, precision p, is defined as the total

Table 4. Localization performance of the three algorithms against the GNSS-RTK ground truth.

Algorithm	X error (m)	Y error (m)	Z error (m)	Absolute error (m)
VineSLAM	0.400	0.710	1.496	0.902 ± 0.619
LOAM	0.202	0.586	0.929	0.638 ± 0.429
LeGO-LOAM*	1.402	17.756	3.953	16.844 ± 23.189

*LeGO-LOAM lost track of the robot pose in this sequence.



(a) Top view - xOy plane. **(b)** Side view - xOz plane.

Figure 15. Evaluation of the trajectory estimated by the three algorithms against the GNSS-RTK ground truth. The figure shows (a) a top view perspective where a 2D trajectory can be observed and (b) a side view perspective where the altitude estimation can be evaluated.

number of true positives over all the detections. The second, recall r, is the total number of true positives over all the ground truths. The F1 score is the harmonic mean between the precision p and recall r, and it can be calculated as follows:

$$F1 = 2\,\frac{p \cdot r}{p + r}. \tag{16}$$

AP is also calculated as a combination of precision and recall. In other words, the AP is the average value of the precision vs recall curve $p(r)$ for $r \in [0,1]$, for each class. Finally, to have a global parameter that evaluates the model for all the classes, the mAP is calculated by averaging the AP of all classes. Table 5 summarizes the results obtained considering all the mentioned metrics.

Table 5. DL model performance for grape bunch and vine stem detection. The model performs with a mAP of 62.43%.

Class	**Precision** (%)	**Recall** (%)	**F1 Score** (%)	**AP** (%)
`stem`	34.85	88.51	49.60	67.80
`tiny-grape-bunch`	54.21	62.11	57.89	50.87
`medium-grape-bunch`	53.26	82.13	64.62	68.63

For the vine stem detection, one can see that the model is able to detect a high percentage of true positives over all the ground truths due to the high recall. However, the lower precision indicates that the model has trouble identifying only relevant objects in the entire set of detections performed. Detecting the grape bunches in an early growth stage is a challenge since the bunches are small and many times occluded. For this reason, the model presents the lower AP for the `tiny-grape-bunch` class. For the

grape bunches in a medium growth stage, the model performs better since they are bigger and more visible. Summarizing, the model achieves an mAP of 62.43%. This value is aligned with the state-of-the-art, considering that we use a lightweight model built specifically to run on embedded devices. The use of more complex models could improve the detection performance but impact the runtime. From the point of view of this application, recall is the most important metric since having a high recall means that the model is able to detect a large percentage of grapes and stems. Even so, precision represents the presence of false detections and can lead to the presence of unreal natural features in the semantic map. The presence of these false positives can be filtered by the mapping procedure to reduce the impact of a lower precision. Each feature is mapped through the use of an EKF. In these filters, the covariance of the observations decreases with the number of times a feature in the map is associated with a new observation. Thus, the filters that present high covariance correspond to features that were not matched between iterations, and that most likely represent false positives. This can be used to remove them from the map, either during the online operation or during the post-processing of the semantic map. Figure 16 shows the detections performed on three images, each one with a different camera and in a different stage of the year.

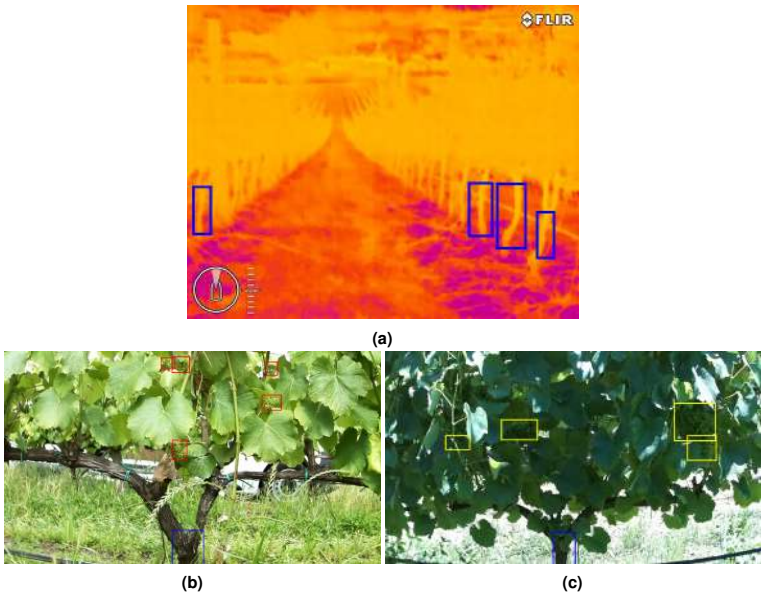


Figure 16. DL-based detection of vine stems and grape bunches in two different growth stages. Each color represent one class.

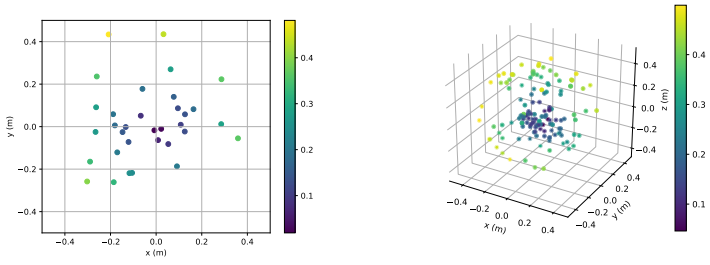
In order to evaluate the semantic mapping approach, two semantic maps were built in different stages of the year. The first, with grape bunches in an early growth stage (corresponding to the class `tiny-grape-bunch`), and the second in an intermediate stage (corresponding to the class `medium-grape-bunch`). To analyse the performance of this semantic mapping approach, both maps were compared with the premise that it is expected that for stems, the maps are identical, and for grape bunches, only slight changes might be noticed. These changes are related to the change of the center of mass of the grape bunches due to the difference in size between the two growth stages present in the dataset. The robot traveled approximately 30 meters in both sequences using two different camera setups pointing at the same vineyard corridor. In the first experiment, an OAK-D camera was used, and in the

second experiment, we used a standard Raspberry Pi camera. To compare the maps, the error between each semantic feature and its correspondence in the other map was calculated in two different ways. For stems, since they are mapped in two dimensions, the x and y distances between correspondences are computed. For grape bunches, the z distance is also considered. Finally, to have a global metric for the mapping of each semantic feature type, the root mean squared (RMS) error is compute as follows:

$$e_{rms} = \sqrt{\frac{1}{N}\sum_{i}\left(m_{\gamma,a,i} - m_{\gamma,b,i}\right)^2}, \tag{17}$$

where $m_{\gamma,a,i}$ and $m_{\gamma,b,i}$ represent the ith correspondence between semantic features in the maps a and b.

Figure 17 shows the dispersion of errors between correspondences both for vine stems and grape bunches. In this figure, each dot represents the error of each correspondence characterized by the



(a) **(b)**

Figure 17. Semantic mapping error dispersion for (a) vine stems and (b) grape bunches. The colorbar represents the distance between correspondences.

difference of the position of the features in both maps. The color of each dot is encoded by a colorbar and represents the euclidean distance between each correspondence. For vine stems this figure shows that most correspondences have a distance lower than 0.20 meters. In fact, for this semantic feature, our approach achieved an RMS error of 0.174 meters. Similarly, for grape bunches the majority of correspondences are not seperated more then 0.20 meters. In this case, the RMS error of correspondence between maps was of 0.180 meters. Considering the challenging characteristics of the environment, these correspondence errors are considered low since the maps captured the vineyard in two different stages, with significant amount of scenario changes (illumination, ground vegetation, foliage density, grape growth stage). In addition, these errors can be also related with aspects of the hardware setup and the approach itself. Firstly, two different camera setups were used. Thus, the camera intrinsic and extrinsic parameters changed between experiences, which can have impact on the mapping since there are always marginal errors associated with the intrinsic and extrinsic calibration of the cameras. Additionally, the fusion of information between the camera and LiDAR sensors during the semantic mapping can also have some impact. As referenced before, the depth of each semantic feature is calculated by intersecting the rays that point to the features and the vegetation plane computed in the 3D LiDAR mapping procedure. This procedure, besides providing a robust depth estimation, is an approximation, i.e., it is not as accurate as a direct pixel depth measurement such as RGB-D cameras do. Finally, slight imperfections on the robot pose estimation also directly impact the mapping performance. Given all of the above, and considering that the semantic mapping is dependent on so many factors, we consider that the system is robust and that it can provide accurate semantic maps of the agricultural environments. Figure 18 shows a semantic map built in Aveleda vineyard, together with the 3D LiDAR map built during SLAM.

4.3 Discussion

The main challeges of this work consisted in the resolution of two main problems: the fusion of data of multiple sensors and the integration of multiple systems to achieve the goal of semantic mapping. In

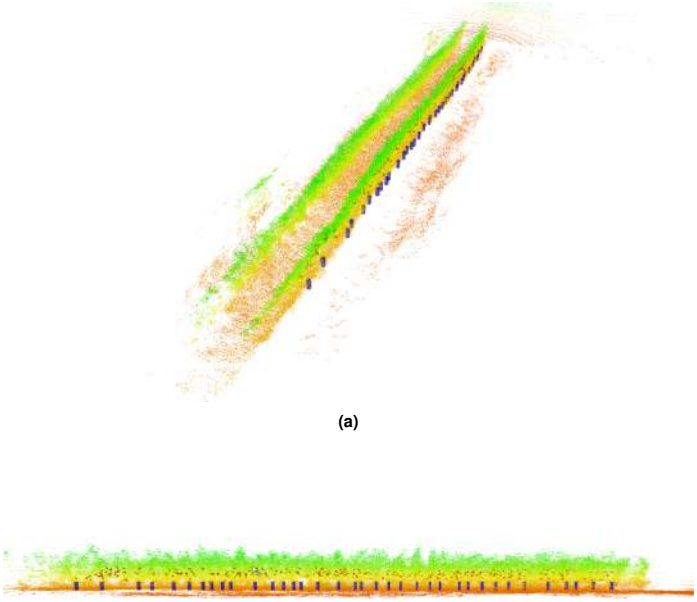


Figure 18. Two perspectives of the semantic map built in Aveleda vineyard, together with the 3D LiDAR map built during SLAM. In (b) the map is presented in a side-view. black bounding boxes represent stems and the red ones represent grape bunches.

particular, to create these semantic maps, two previous systems were developed: a robust SLAM pipeline and a semantic perception algorithm. In addition, this work comprehended an intense and long-term field work that was carried out by visiting the vineyard multiple times in different stages of the year.

Compared with the state-of-the-art, this work fills the gap of robust semantic mapping in vineyards. The proposed approach provides an expeditious way of creating maps with useful information that agronomists can later use. Of course, there is still space to improve in the future. For example, we highlight the work of Xiong et al. Xiong et al. (2018) that uses artificial illumination to develop a technology for night-time fruit-picking by detecting green grapes. This state-of-the-art approach can be used as a basis to enable the creation of semantic maps of the vineyard during the night. Also, Aquino et al. Aquino et al. (2015) propose the grapevine flower estimation, which can be essential to perform an early yield estimation. This concept can also be used in the future to create a new class in the proposed publicly available dataset, which will enable the creation of semantic maps of the vineyard in a very early stage.

5 CONCLUSIONS

This work proposes a system capable of building semantic maps of agricultural environments considering a robust SLAM approach. The contributions of this work are: a robust SLAM algorithm that fuses four different sensors (wheel odometry, RTK-GNSS, IMU, and 3D LiDAR) to compute the robot pose and simultaneously build a 3D map of the culture; a DL-based semantic perception approach that uses a built

in-house public dataset to train a lightweight model to detect vine stems and grape bunches on images; and finally a semantic mapping procedure that uses the two other systems to build a map of the culture with semantic information. Results show that the proposed approach can localize the robot with precision in challenging scenarios and build 3D maps of them. Additionally, the experiments also show that the system can build semantic maps in different growth stages of the culture.

For future work, we would like to extend the semantic dataset to consider more growth stages of grape bunches and include images at the night with artificial illumination. In this context, we would like to test all the proposed approach (SLAM, perception, and semantic mapping) in real experiments at night. Also, since our robots will operate autonomously over larger periods of time, and higher extensions, we would like to validate our approach for continuous operation over several hours, and kilometers. Finally, we would like to develop a different semantic mapping validation procedure by labeling each grape bunch with a unique identifier that will serve as ground truth.

REFERENCES

Aguiar, A. S., dos Santos, F. N., Cunha, J. B., Sobreira, H., and Sousa, A. J. (2020a). Localization and mapping for robots in agriculture and forestry: A survey. *Robotics*, 9(4).

Aguiar, A. S., Monteiro, N. N., Santos, F. N. d., Solteiro Pires, E. J., Silva, D., Sousa, A. J., and Boaventura-Cunha, J. (2021). Bringing semantics to the vineyard: An approach on deep learning-based vine trunk detection. *Agriculture*, 11(2).

Aguiar, A. S., Santos, F. N. D., De Sousa, A. J. M., Oliveira, P. M., and Santos, L. C. (2020b). Visual trunk detection using transfer learning and a deep learning-based coprocessor. *IEEE Access*, 8:77308–77320.

Aquino, A., Millan, B., Gutiérrez, S., and Tardáguila, J. (2015). Grapevine flower estimation by applying artificial vision techniques on images with uncontrolled scene and multi-model analysis. *Computers and Electronics in Agriculture*, 119:92–104.

Arnal Barbedo, J. G. (2019). Plant disease identification from individual lesions and spots using deep learning. *Biosystems Engineering*, 180:96–107.

Auat Cheein, F., Steiner, G., Perez Paina, G., and Carelli, R. (2011). Optimized eif-slam algorithm for precision agriculture mapping based on stems detection. *Computers and Electronics in Agriculture*, 78(2):195–207.

Auat Cheein, F. A. and Guivant, J. (2014). Slam-based incremental convex hull processing approach for treetop volume estimation. *Computers and Electronics in Agriculture*, 102:19–30.

Badeka, E., Kalampokas, T., Vrochidou, E., Tziridis, K., Papakostas, G., Pachidis, T., and Kaburlasos, V. (2021). Real-time vineyard trunk detection for a grapes harvesting robot via deep learning. In Osten, W., Nikolaev, D. P., and Zhou, J., editors, *Thirteenth International Conference on Machine Vision*, volume 11605, pages 394 – 400. International Society for Optics and Photonics, SPIE.

Bailey, T. and Durrant-Whyte, H. (2006). Simultaneous localization and mapping (slam): part ii. *IEEE Robotics Automation Magazine*, 13(3):108–117.

Bailey, T., Nieto, J., Guivant, J., Stevens, M., and Nebot, E. (2006). Consistency of the ekf-slam algorithm. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3562–3568.

Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., and Leonard, J. J. (2016). Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332.

Canedo-Rodríguez, A., Álvarez Santos, V., Regueiro, C., Iglesias, R., Barro, S., and Presedo, J. (2016). Particle filter robot localisation through robust fusion of laser, wifi, compass, and a network of external cameras. *Information Fusion*, 27:170–188.

Chiang, K., Tsai, G., Chang, H., Joly, C., and EI-Sheimy, N. (2019). Seamless navigation and mapping using an INS/GNSS/grid-based SLAM semi-tightly coupled integration scheme. *Information Fusion*, 50:181–196.

Colmenero-Martinez, J. T., Blanco-Roldán, G. L., Bayano-Tejero, S., Castillo-Ruiz, F. J., Sola-Guirado, R. R., and Gil-Ribes, J. A. (2018). An automatic trunk-detection system for intensive olive harvesting with trunk shaker. *Biosystems Engineering*, 172:92 – 101.

Crespo, J., Castillo, J. C., Mozos, O. M., and Barber, R. (2020). Semantic information for robot navigation: A survey. *Applied Sciences*, 10(2).

Das, J., Cross, G., Qu, C., Makineni, A., Tokekar, P., Mulgaonkar, Y., and Kumar, V. (2015). Devices, sys-

tems, and methods for automated monitoring enabling precision agriculture. In *2015 IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE.

Diago, M. P., Sanz-Garcia, A., Millan, B., Blasco, J., and Tardaguila, J. (2014). Assessment of flower number per inflorescence in grapevine by image analysis under field conditions. *Journal of the Science of Food and Agriculture*, 94(10):1981–1987.

Dong, W., Roy, P., and Isler, V. (2019). Semantic mapping for orchard environments by merging two-sides reconstructions of tree rows. *Journal of Field Robotics*, 37(1):97–121.

Douc, R. and Cappe, O. (2005). Comparison of resampling schemes for particle filtering. In *ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005.*, pages 64–69.

Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, 13(2):99–110.

Hao, S., Zhou, Y., and Guo, Y. (2020). A brief survey on semantic segmentation with deep learning. *Neurocomputing*, 406:302–321.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications.

Juman, M. A., Wong, Y. W., Rajkumar, R. K., and Goh, L. J. (2016). A novel tree trunk detection method for oil-palm plantation navigation. *Computers and Electronics in Agriculture*, 128:172 – 180.

Kestur, R., Meduri, A., and Narasipura, O. (2019). Mangonet: A deep semantic segmentation architecture for a method to detect and count mangoes in an open orchard. *Engineering Applications of Artificial Intelligence*, 77:59–69.

Kostavelis, I. and Gasteratos, A. (2013). Learning spatially semantic representations for cognitive robot navigation. *Robotics and Autonomous Systems*, 61(12):1460–1475.

Kostavelis, I. and Gasteratos, A. (2015). Semantic mapping for mobile robotics tasks: A survey. *Robotics and Autonomous Systems*, 66:86–103.

Lamprecht, S., Stoffels, J., Dotzler, S., Haß, E., and Udelhoven, T. (2015). aTrunk—an ALS-based trunk detection algorithm. *Remote Sensing*, 7(8):9975–9997.

Libby, J. and Kantor, G. (2011). Deployment of a point and line feature localization system for an outdoor agriculture vehicle. In *2011 IEEE International Conference on Robotics and Automation*, pages 1565–1570.

Liu, S., Li, X., Wu, H., Xin, B., Tang, J., Petrie, P. R., and Whitty, M. (2018). A robust automated flower estimation system for grape vines. *Biosystems Engineering*, 172:110–123.

Liu, X., Chen, S. W., Liu, C., Shivakumar, S. S., Das, J., Taylor, C. J., Underwood, J., and Kumar, V. (2019). Monocular camera based fruit counting and mapping with semantic data association. *IEEE Robotics and Automation Letters*, 4(3):2296–2303.

Lowry, S., Sünderhauf, N., Newman, P., Leonard, J. J., Cox, D., Corke, P., and Milford, M. J. (2016). Visual place recognition: A survey. *IEEE Transactions on Robotics*, 32(1):1–19.

Lu, F. and Milios, E. (1997). Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4(4):333–349.

Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B., et al. (2002). Fastslam: A factored solution to the simultaneous localization and mapping problem. *Aaai/iaai*, 593598.

Neves Dos Santos, F., Sobreira, H. M. P., Campos, D. F. B., Morais, R., Moreira, A. P. G. M., and Contente, O. M. S. (2015). Towards a reliable monitoring robot for mountain vineyards. In *2015 IEEE International Conference on Autonomous Robot Systems and Competitions*, pages 37–43.

Palacios, F., Bueno, G., Salido, J., Diago, M. P., Hernández, I., and Tardaguila, J. (2020). Automated grapevine flower detection and quantification method based on computer vision and deep learning from on-the-go imaging using a mobile sensing platform under field conditions. *Computers and Electronics in Agriculture*, 178:105796.

Paz, L. M., TardÓs, J. D., and Neira, J. (2008). Divide and conquer: Ekf slam in $o(n)$. *IEEE Transactions on Robotics*, 24(5):1107–1120.

Pinto de Aguiar, A. S., Neves dos Santos, F. B., Feliz dos Santos, L. C., de Jesus Filipe, V. M., and Miranda de Sousa, A. J. (2020). Vineyard trunk detection using deep learning – an experimental device benchmark. *Computers and Electronics in Agriculture*, 175:105535.

Pérez-Zavala, R., Torres-Torriti, M., Cheein, F. A., and Troni, G. (2018). A pattern recognition strategy for visual grape bunch detection in vineyards. *Computers and Electronics in Agriculture*, 151:136–149.

Rahnemoonfar, M. and Sheppard, C. (2017). Deep count: Fruit counting based on deep simulated learning. *Sensors*, 17(4).

Santos, L., Santos, F. N., Oliveira, P. M., and Shinde, P. (2020a). Deep learning applications in agriculture: A short review. In Silva, M. F., Luís Lima, J., Reis, L. P., Sanfeliu, A., and Tardioli, D., editors, *Robot 2019: Fourth Iberian Robotics Conference*, pages 139–151, Cham. Springer International Publishing.

Santos, T. T., de Souza, L. L., dos Santos, A. A., and Avila, S. (2020b). Grape detection, segmentation, and tracking using deep neural networks and three-dimensional association. *Computers and Electronics in Agriculture*, 170:105247.

Shalal, N., Low, T., McCarthy, C., and Hancock, N. (2013). A preliminary evaluation of vision and laser sensing for tree trunk detection and orchard mapping. In *Proceedings of the Australasian Conference on Robotics and Automation (ACRA 2013)*, pages 1–10. Australasian Robotics and Automation Association.

Shan, T. and Englot, B. (2018). Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765.

Sheng, W., Du, J., Cheng, Q., Li, G., Zhu, C., Liu, M., and Xu, G. (2015). Robot semantic mapping through human activity recognition: A wearable sensing and computing approach. *Robotics and Autonomous Systems*, 68:47–58.

Shu, F., Lesur, P., Xie, Y., Pagani, A., and Stricker, D. (2021). Slam in the field: An evaluation of monocular mapping and localization on challenging dynamic agricultural environment. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1761–1771.

Thrun, S. (2003). *Autonomous Robots*, 15(2):111–127.

Tu, S., Pang, J., Liu, H., Zhuang, N., Chen, Y., Zheng, C., Wan, H., and Xue, Y. (2020). Passion fruit detection and counting based on multiple scale faster r-CNN using RGB-d images. *Precision Agriculture*, 21(5):1072–1091.

Vougioukas, S. G. (2019). Agricultural robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 2(1):365–392.

Vu, T.-D., Burlet, J., and Aycard, O. (2011). Grid-based localization and local mapping with moving object detection and tracking. *Information Fusion*, 12(1):58–69. Special Issue on Intelligent Transportation Systems.

Walter, M. R., Hemachandra, S., Homberg, B., Tellex, S., and Teller, S. (2013). Learning semantic maps from natural language descriptions. *The International Journal of Robotics Research*.

Wolf, D. and Sukhatme, G. (2008). Semantic mapping using mobile robots. *IEEE Transactions on Robotics*, 24(2):245–258.

Xiong, J., Liu, Z., Lin, R., Bu, R., He, Z., Yang, Z., and Liang, C. (2018). Green Grape Detection and Picking-Point Calculation in a Night-Time Natural Environment Using a Charge-Coupled Device (CCD) Vision Sensor with Artificial Illumination. 18(4):969.

Yi, C., Jeong, S., and Cho, J. (2012). Map representation for robots. *Smart Computing Review*, 2:18–27.

Zabawa, L., Kicherer, A., Klingbeil, L., Töpfer, R., Kuhlmann, H., and Roscher, R. (2020). Counting of grapevine berries in images via semantic segmentation using convolutional neural networks. *ISPRS Journal of Photogrammetry and Remote Sensing*, 164:73–83.

Zhang, J. and Singh, S. (2014). Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*.

Zhao, W., Wang, X., Qi, B., and Runge, T. (2020). Ground-level mapping and navigating for agriculture based on iot and computer vision. *IEEE Access*, 8:221975–221985.

Zureiki, A. and Devy, M. (2008). SLAM and data fusion from visual landmarks and 3d planes. *IFAC Proceedings Volumes*, 41(2):14651–14656.

5.3 Particle filter refinement based on clustering procedures for high-dimensional localization and mapping systems

The proposed PF presents a discretization of the six-dimensional robot's pose representation. In challenging environments such as agriculture, the number of required discrete samples (or particles) can increase to improve the localization precision. This can be computationally expensive and can harm the online autonomous navigation of the robot. To overcome this issue, this thesis proposes an algorithm that enables the refinement of the PF performance without the need to increse the number of particles. This work was published in the Robotics and Autonomous Systems Journal and is entitled *Particle filter refinement based on clustering procedures for high-dimensional localization and mapping systems* (Aguiar et al., 2021b). This work uses a stereo camera and a feature descriptor to build a point cloud with color and signature information. Besides producing a new map and contributing to the multi-layer mapping architecture of VineSLAM, these features are used to refine the particles' pose. A clustering procedure is applied to the set of particles to group them in a fixed-size number of clusters. Each cluster executes a scan-matching algorithm and refines its particles through the calculation of the matching likelihood.

It is worth noting that to build the multi-layer mapping, i.e., to provide a representation of the different types of maps in the same reference frame, the work described in Chapter 3 is essential. The metric map is build using a 3D LiDAR, while this visual map (and also the semantic map) is built using cameras onboard of agricultural robotic platforms. The extrinsic calibration of these modalities of sensors enables the representation and correlation of all these types of maps.

Particle filter refinement based on clustering procedures for high-dimensional localization and mapping systems

André Silva Aguiar[a,b], Filipe Neves dos Santos[a], Héber Sobreira[a], José Boaventura Cunha[b], Armando Jorge Sousa[a,c]

[a]INESC TEC - INESC Technology and Science; 4200-465, Porto, Portugal
[b]School of Science and Technology, University of Trás-os-Montes e Alto Douro; 5000-801 Vila Real, Portugal
[c]FEUP, University Of Porto, Porto, Portugal

Abstract

Developing safe autonomous robotic applications for outdoor agricultural environments is a research field that still presents many challenges. Simultaneous Localization and Mapping can be crucial to endow the robot to localize itself with accuracy and, consequently, perform tasks such as crop monitoring and harvesting autonomously. In these environments, the robotic localization and mapping systems usually benefit from the high density of visual features. When using filter-based solutions to localize the robot, such an environment usually uses a high number of particles to perform accurately. These two facts can lead to computationally expensive localization algorithms that are intended to perform in real-time. This work proposes a refinement step to a standard high-dimensional filter-based localization solution through the novelty of downsampling the filter using an online clustering algorithm and applying a scan-match procedure to each cluster. Thus, this approach allows scan-matchers without high computational cost, even in high dimensional filters. Experiments using real data in an agricultural environment show that this approach improves the Particle Filter performance estimating the robot pose. Additionally, results show that this approach can build a precise 3D reconstruction of agricultural environments using visual scans, i.e., 3D scans with RGB information.

Keywords: SLAM, Clustering, Agricultural robotics

1. Introduction

The Oporto vineyards, Fig. 1, are located in the Douro Demarched Region, the oldest controlled winemaking region in the world, a UNESCO heritage place [1]. These vineyards are built in steep slope hills, which brings several challenges to the development of robotic solutions in this context. The hill's characteristics cause a signal blockage that decreases the accuracy of signals emitted by the Global Navigation Satellite System (GNSS), making unreliable the use of, for example, the Global Positioning System (GPS). Also, the terrain highly characterized by irregularities leads to high inaccuracy of sensors like wheel odometry and Inertial Measurement Units (IMU)s. Due to all these factors, there is a significant need to have a robotic localization system redundant to satellite-based sensors and robust to the environment main challenges. Simultaneous Localization

Figure 1: Typical steep slope vineyard in the Douro's region.

and Mapping (SLAM) [2, 3] comes up as the most reliable and reasoning solution to solve this. SLAM consists of estimating the location of a mobile robot in an unknown environment while simultaneously mapping it [4]. Theoretically, SLAM was formulated as a probabilistic mathematical problem. This method considers a set of controls $\boldsymbol{U_{0:k}}$ and observations $\boldsymbol{Z_{0:k}}$ until the current time instant k to formulate a motion model in the form

$$P(x_k|x_{k-1}, u_k) \tag{1}$$

i and an observation model in the form

$$P(z_k|x_k, m) \tag{2}$$

where x_k is the robot position at the instant k, u_k is the control vector, z_k is the set of observation taken at the instant k, and m is the global map built so far. To estimate the so-called joint posterior $\boldsymbol{P(x_k, m|Z_{0:k}, U_{0:k}, x_0)}$, i.e., the robot pose and the map, this pipeline uses a recursive two-step approach: prediction and update. The prediction step uses the control inputs to estimate the robot pose, and the update step uses the observations to correct the first estimation. The map creation can also be addressed as a mapping with known poses problem, where the map is built after computing the robot pose, solving $\boldsymbol{P(m|X_{0:k}, Z_{0:k}, U_{0:k})}$.

With this formulation, several solutions for the SLAM problem emerged. The main problem is to find a suitable representation for the motion and observation models.

EKF-SLAM [5, 6, 7] was the first influential SLAM algorithm proposed and is based on the extended Kalman Filter (EKF). This method represents the robot estimate as a multivariable Gaussian distribution that contains the robot location and the location of all the mapped landmarks. Another way of solving the SLAM problem is the graph-based optimization techniques [8]. These build a graph composed by nodes - the robot locations over time - and arcs - connections between nodes or between nodes and landmarks - using the controls and observations. The graph is sparse in that each node is connected to a small number of other nodes. So, this procedure computes a nonlinear sparse optimization technique to solve the SLAM problem. Finally, as proposed in this work, exist the particle filter- (PF)-based SLAM solutions [9]. In this context, FastSLAM [10] was a huge mark in probabilistic SLAM research. This algorithm considers $\boldsymbol{N}$ particles where each one contains the robot trajectory $\boldsymbol{X_{0:k}}$ and a set of 2-dimensional Gaussians representing each landmark on the map. After this version, FastSLAM2 [11] emerged with an improvement in the proposal distribution. Both methods apply a technique called Rao-Blackwellization by sampling the path posterior $\boldsymbol{P(x_k^i|u_k, z_k)}$ and representing the map $\boldsymbol{P(m|x_k^i, u_k, z_k)}$ in a Gaussian form (here, i represents the *ith* particle). Based on this Rao-Blackwellization concept, Grisetti et al. [12] proposed a grid-based SLAM approach that intends to reduce the number of particle samples improving the filter time performance. The great innovation of this work is an improved proposal distribution that considers a scan-matching procedure. The scan-matching algorithm finds the rigid body transformation that best aligns, in this case, a scan and a map. So, here particles are drawn not only by the odometry controls but also by the scan matching result. This highly improves filter performance, decreasing the required number of samples. Other works came up with the implementation of Rao-Blackwellized PFs, such as, for example the one proposed by Grisetti and Tipaldi [13].

As in [12], scan-matching is a popular technique used in SLAM algorithms in several different ways, with many other purposes. In particular, Iterative Closest Point (ICP) [14] is one of the most popular scan-matching techniques and is widely applied in SLAM. For example, Nüchter el at. [15] uses ICP performing scan-to-scan alignment, i.e., aligning consecutive laser scans, and compute the robot 6-DoF pose and a heuristic-based closed-loop detection to refine the estimation. Tiar and Lakrouf [16] implement the ICP algorithm in a local ICP-SLAM manner. In this method, the global map is partitioned in several local maps over time. The scan-matcher is applied in each one of them, reducing the computational cost of the algorithm. In [17], an RGB-D camera system is used to extract dense 3D data, and a local ICP alignment is applied on subsampled range images. In this work, different correspondence techniques are implemented in a visual SLAM procedure. Results show that the point-to-plane approach has the best tradeoff between efficiency and low error. Masahiro Tomono [18] proposes a robust 3D visual SLAM approach using edge features and a two-stage ICP algorithm. Here, the scan-matcher is used to align successive frames. As this approach is susceptible to cumulative error, ICP is also used to perform keyframe adjustment, i.e., a global alignment on selected frames. Similarly, Holz et al. [19] propose an ICP-based incremental registration technique in a SLAM pipeline. This method also implements a scan-to-scan alignment. To correct the cumulative error that characterizes this approach and improve registration results, the authors apply a heuristic based on the number of correspondences to reject wrong alignment estimates or large odometry drifts. Differently, [20], just like in [21], uses an EKF

with an augmented state representation to perform visual scan-matching. So, instead of using laser range data, the visual scan-matcher uses camera information to generate 3D visual range data with color specification. The 3D reconstruction can be denoted as a visual scan. The authors propose the use of the Scale Invariant Feature Transform (SIFT) [22] to extract visual features that are then projected into the 3D space to build the visual scans. Then, a scan-to-scan alignment is applied, and the map is estimated using the augmented EKF. In the same context, ICP and scan-matchers, in general, can also be used to detect loop closure in SLAM frameworks. In particular, [23, 24] use ICP to do so. The first, add new constraints to the robot path if a match is found between the current feature scan and a previous one, eliminating drift. In the second, ICP is used to detect loop closures in the robot path using the Euclidean distance between the current and all the previous poses.

In this work, we propose a novel PF refinement step for 6-DoF robot pose estimation that uses a clustering procedure to downsample the filter and applies an ICP algorithm to refine the robot pose represented by each cluster. This refinement step uses a stereo camera system to extract Speeded Up Robust Features (SURF) [25] and build a 3D visual scan that is used in the ICP alignment for each cluster. This algorithm is applied to a modified version of VineSLAM [26] that uses a standard PF to localize a robot in mountain vineyards. At the best of our knowledge, this is the first work that applies an online clustering procedure and a scan-matching approach to each cluster as a refinement step to a PF. Our approach relates to [27, 28] in that it uses clusters to represent sets of individual particles. However, the proposed method differs from both approaches in several aspects: the first maintains the identity of the same cluster during all the filter operation, the second merges overlapping clusters and splits diffuse ones, while our method computes new clusters in each iteration; both approaches use the clustered PF to represent the posterior while our approach works just as a refinement step of an ordinary PF.

The remainder of this paper is organized as follows: Sec. 2.1 provides an overview of the proposed system as well as the improvements in relation with the original VineSLAM approach; Sec. 2.2 includes information about the extraction and generation of the feature observations used in this work; Sec. 2.3 details the implementation of the online particles clustering procedure; Sec. 2.4 explains the refinement step applied to the original PF; Sec. 3 compares the performance of the proposed algorithm in relation with the original VineSLAM version; and finally, Sec. 4 summarizes the conclusions of the work.

2. Particle Filter clustering for SLAM systems

This work proposes the addition of a feature-based 3D map to the multi-layer mapping procedure proposed in VineSLAM [26] and a robot localization estimation refinement based on a PF clustering procedure. The main contribution of this work is the refinement step to standard high dimensional PFs, i.e., filters with a high number of particles, or with a dense presence of input data. It is worth noting that this technique can be applied in any other PF-based context than VineSLAM.

2.1. System overview

The original version of VineSLAM uses a standard PF with the controls given by wheel odometry and observations by a set of two feature types: semantic features and features extracted directly from a 3D point cloud. Each particle represents the 6-DoF robot pose $[x, y, z, \phi, \psi, \theta]_i$. With this, for each particle, the observation probability is computed by means of a standard normal distribution for each feature observation type j with mean $\boldsymbol{\mu_j}$ and covariance $\boldsymbol{\Sigma_j}$ as follows:

$$\omega_k^i := \prod_j \mathcal{N}(z_{k,j}|x_k^i, \mu_j, \Sigma_j) \tag{3}$$

where $\boldsymbol{\omega_k^i}$ represents the *ith* particle weight at the time instant k. As VineSLAM works in Douro's mountain vineyards, a challenging outdoor environment, the PF requires high-quality observations and as many particles as possible to have a proper performance. This leads to a high dimensional filter that can be computationally expensive. To overcome this issue and add a new layer of features and more robustness to the robot localization estimation on VineSLAM, this work refines the output of the PF using a scan-matching procedure, applying an ICP to a set of clusters extracted from the global set of particles. The work is segmented into three main stages: the extraction and generation of 3D visual scans, the online clustering procedure, and the PF refinement step.

2.2. Visual observations extraction

To extract visual observations and build 3D visual scans was used the ZED stereo camera[1]. In this process, using the reference RGB image of the stereo camera system, the SURF [25] feature detector is applied, and 2D image features are extracted. Then, a disparity image captures the depth information of the vineyard, as represented in Fig. 2. So, for each extracted 2D feature, the disparity image has its corresponding depth information, allowing the conversion of the feature into a 3D point. So, for a feature with coordinates $[x^c, y^c]$ and depth d in the disparity image, the corresponding 3D point in the local robot's referential frame is computed as follows:

$$\begin{bmatrix} x^r \\ y^r \\ z^r \end{bmatrix} = {}^c\boldsymbol{T_r} \begin{bmatrix} (x^c - c_x) \cdot \frac{d}{f_x} \\ (y^c - c_y) \cdot \frac{d}{f_y} \\ d \end{bmatrix} \tag{4}$$

where ${}^c\boldsymbol{T_r}$ is the homogeneous transformation that converts the camera referential frame into the robot referential frame, $[c_x, c_y]$ is the image principal point, and $[f_x, f_y]$ is the respective focal length. It is worth noting that all the obtained 3D points have RGB information since they were directly projected from the image. This allows the creation of a visual scan that considers color information of the observed environment.

2.3. Online clustering procedure

To add the proposed visual scan map layer to VineSLAM, an online clustering procedure is applied to the PF. In this process, particles are clustered based on their 3D

[1] https://www.stereolabs.com/zed/

Figure 2: Disparity image of a vineyard captured during tests.

spatial distribution, i.e., using their respective $[x, y, z]$ states. To do so, this work proposes a variation of the K-means++ [29], which we call equal-sized K-means++. One of the limitations of K-means [30] and K-means++ itself is that they do not ensure that the clusters' size is the same. In this work, the number of clusters computed from the entire set of particles at each instant is an input parameter of the system. This is due to the direct influence of the number of clusters in the algorithm's computational cost since one scan-matching procedure is executed per cluster. To guarantee that the particles are always clustered in equal-sized N different sets (as chosen by the user), it is proposed an improvement to the original K-means++ algorithm. As said before, we named our algorithm as equal-sized K-means++.

In this work, giving k particles in $\mathcal{X}$ and N desired number of clusters in $\mathcal{C}$, the equal-sized K-means++ is applied aiming to minimize the following function

$$\gamma = \sum_{x \in \mathcal{X}} \min_{c \in \mathcal{C}} \|x - c\|^2. \tag{5}$$

To initialize the clustering algorithm, the cluster centroids and particles are computed as follows.

Initialization:.

1. Choose the first centroid c_1 randomly from $\mathcal{X}$.
2. Choose a new center c_i as $x \in \mathcal{X}$ with probability $\frac{D(x)^2}{\sum_{x \in \mathcal{X}} D(x)^2}$ proportional to the distance to the nearest centroid $D(x)$ already chosen.
3. Repeat 2. until we have chosen N centroids.
4. For each particle, compute the distance to all the centroids and build a heap with this information.

5. Draw particles from the heap, assigning them to the nearest cluster. If the cluster is already full, assign to the next non-full nearest cluster centroid.

The innovation of equal-sized K-means++ is based on the guarantee that all the clusters have the same number of particles. If, the number of particles and the desired number of clusters are not multiples, some clusters will have an extra particle. The key idea is that the size of the clusters is balanced as much as possible. To ensure that, steps 4. and 5. of the initialization algorithm are proposed. In these, a heap data structure is built to accelerate the clustering assignment procedure. The heap contains the ordered distances of each particle to each cluster. With this information, each particle is assigned to the closer non-full cluster in a faster way, comparing to traditional search approaches. After initializing the clustering algorithm, Eq. 5 is minimized. Once again, equal-sized K-means++ presents innovations so that the clusters size remain constant. To ensure that, the algorithm requires a data structure to save swap proposals, i.e., particles that are candidates to move to another cluster. This feature is added to the standard K-means method, and the algorithm is proposed as follows.

Algorithm:.

1. Compute each cluster centroid c_i as the center of mass of all the particles in the respective cluster C_i as $c_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$.
2. For each particle, compute the nearest cluster centroid. If the found cluster centroid is different from the one currently associated with the particle:
   (a) If the target cluster size is smaller than the current cluster size, add the particle to the target cluster.
   (b) Otherwise, search for a swap proposal incoming from the target cluster. If there is one, switch the particles and delete the swap proposal. If not, maintain the particle on the same cluster and create a swap proposal for a future transition.

The algorithm ends if, for a given iteration, no swap proposals have been satisfied or if a maximum number of iterations is reached. By this stage, each cluster is represented as a Gaussian $\mathcal{N}(\mu_i, \Sigma_i)$ where the mean μ_i and the covariance Σ_i for the *ith* cluster are computed as a weighted mean and covariance given each particle weight. With this, the cluster centroid is adjusted, taking into account the particles' weight, i.e., this subsampling procedure considers k clusters affected by the weight of the particles.

2.4. Particle filter refinement step

The PF refinement's last and primary step consists of applying a scan-matching procedure considering the computed clusters. The 3D visual scan global map is initialized with the first observation of the stereo camera system. After that, the scan-matching procedure is computed, aligning the local visual scan observation with the global map. In this, the goal is to find the homogeneous transformation $\boldsymbol{T}_{c_i} = [\boldsymbol{R}|\boldsymbol{t}]_{c_i}$ that, given a set of associations between the observed 3D visual feature points $\boldsymbol{L} = \{\boldsymbol{l}_{p,1}, ..., \boldsymbol{l}_{p,n}\}$ and the global map $\boldsymbol{m} = \{\boldsymbol{l}_{g,1}, ..., \boldsymbol{l}_{g,n}\}$ computed using the SURF descriptors, minimizes the sum of the squared error

$$E(R, t) = \frac{1}{n} \sum_{i=1}^{n} \|l_{p,i} - R l_{g,i} - t\|^2. \quad (6)$$

After performing the clustering procedure, the 3D pose of each of the N clusters is used as the initial guess to an ICP scan-matching algorithm. By minimizing the cost function 6, an homogeneous transformation $\boldsymbol{T}_{c_i}$ is obtained per cluster. A refinement procedure is then applied in each cluster, as represented in Fig. 3, where the particles present in each one are affected by the same homogeneous transformation. This procedure proves

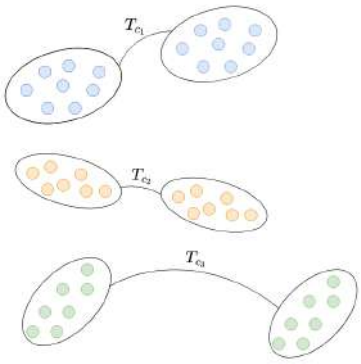


Figure 3: Propagation of particles belonging to each cluster. The scan-matching procedure is computed using the centroid of each cluster as initial guess. Then, all the particles inside each cluster are affected by the same homogeneous transformation.

to refine the particles pose, as discussed in Sec. 3.

After refining each particle pose by the homogeneous transformation resultant in each cluster, each scan-matching likelihood is computed to update the particles' weight. Clusters that do not represent the real robot pose will input an erroneous first guess to ICP, leading to local minimum solutions on the scan-matching procedure. Thus, by modeling the likelihood of the alignment resultant from ICP, the particles that belong to clusters that do not represent the real robot pose can be penalized. To do so, based on the algorithm present in [31] for computing the likelihood of a landmark measurement, the likelihood of the ICP algorithm for each cluster is calculated. So, after computing $\boldsymbol{T}_{c_i}$ for the *ith* cluster, the local 3D visual scan map is projected into the global map referential frame applying the resultant homogeneous transformation. With this, a set of associations is computed between the features on the projected map and the ones present on the global map. Each visual feature is represented as $[l, s]$ where l states for its 3D location and s refers to its visual descriptor extracted using SURF detector. So, considering n associations $\{[l, s]_{p,1}, ..., [l, s]_{p,n}\} \longleftrightarrow \{[l, s]_{g,1}, ..., [l, s]_{g,n}\}$ found, the likelihood of each ICP alignment is computed as follows

$$P(z_k|x_k^j, Z_{0:k-1}) = \sum_{j=1}^{n} \Delta_{l,j} \cdot \Delta_{s,j} \tag{7}$$

where

$$\Delta_{l,j} = \frac{1}{\sqrt{2\pi}\sigma_l} \exp\left(\frac{-1}{\sigma_l} \cdot \|l_{p,j} - l_{g,j}\|\right)$$
$$\Delta_{s,j} = \frac{1}{\sqrt{2\pi}\sigma_s} \exp\left(\frac{-1}{\sigma_s} \cdot \|s_{p,j} - s_{g,j}\|\right).$$

Figure 4 shows the representation of $f = \Delta_{l,j} \cdot \Delta_{s,j}$. Here we can see that, the likelihood

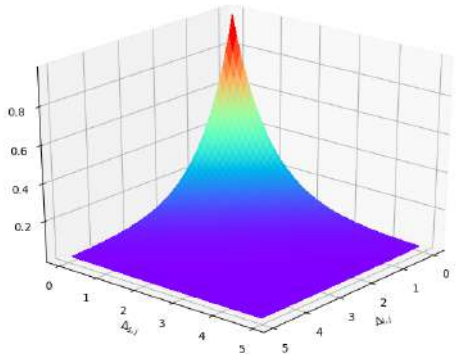


Figure 4: Likelihood of the ICP alignment algorithm represented as a multivariable function. The likelihood decreases exponentially with the increase of absolute difference between correspondences descriptors l and spatial position s. The impact of each one of the variables in the final likelihood can be controlled by their corresponding standard deviations.

of the ICP alignement decreases exponentially with the increase of absolute difference between correspondences descriptors l and spatial position s. It is worth noting that, each feature descriptor is a represented as a float-array of 64 positions, so $\|s_{p,j} - s_{g,j}\|$ is a 64 float-array position-wise difference norm between correspondences. Also, Eq. 7 shows that for this specific observation type, the proposed SLAM framework computes the likelihood of the observations similarly to FastSLAM1.0 [10], but instead of considering the entire robot path $X_{0:k}$, we use only the current robot pose x_k that in this case is represented by the homogeneous transformation obtained from ICP. Finally, the likelihood computed $P(z_k|x_k^i, Z_{0:k-1})$ for each cluster is applied to each particle i of each cluster at instant k in the following way

$$\omega_{k,refined}^i = \omega_k^i P(z_k|x_k^i, Z_{0:k-1}). \tag{8}$$

After this, the resampling procedure takes place and the final robot pose is computed by the average of all particle poses. With the robot pose, the local 3D visual scan map is registered in the global map.

The following algorithm summarizes the entire procedure.

Initialization.

1. Draw particles' initial pose from a standard Gaussian distribution.
2. Initialize the global map by registering the first visual scan observation.

Algorithm.

1. For each cluster:
   (a) Apply the ICP algorithm minimizing Eq. 6.
   (b) Compute the likelihood of the ICP alignment using Eq. 7.
   (c) Refine all the particles weights in each cluster with the computed likelihood.
2. Compute final robot pose as the mean and covariance of all particles.
3. Register visual scan observation using the respective pose.

3. Results

To evaluate the validity of the ICP-based proposed approach, two experiments were conducted using our robotic platform, AgRob V16 [32, 33] (Fig. 5), where GPS was used as ground-truth. These sequences consist of two travelled paths by the robot in Aveleda

Figure 5: AgRoB V16 robotic platform.

vineyard. Figure 6 shows a satellite image with a top view of the entire vineyard and two smaller images focusing on each sequence - 1 and 2. Table 1 summarizes the main characteristics of each sequence. As can be seen, sequence 1 is more challenging than sequence 2. In the first, the robot travels a longer path, performing three pure rotations - points B, C and D of Fig. 7a. On the contrary, sequence 2 is much smaller and approximately a straight path, as represented in Fig. 7b. It is worth noting that, both

Table 1: Characteristics of the travelled sequences.

	Travelled distance (m)	Pure rotations	Foliage stage
Sequence 1	~75	3	with
Sequence 2	~27	0	without

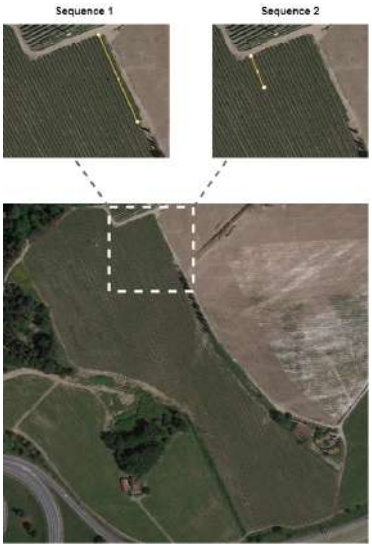


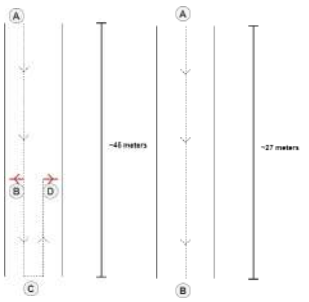
Figure 6: Satellite image of Aveleda's vineyard. The sub-figures represent the sequences (1 and 2) travelled by the robot.

sequences were recorded in different stages of the year, and, consequently, in sequence 1 the vines present high density of foliage, and in sequence 2 they do not have any.

To evaluate the performance of the proposed approach, results are divided into three sections. Section 3.1 presents a brief overview of the clustering results, Sec. 3.2 analyses the execution time of the proposed approach, Sec. 3.3 analyses the ultimate goal of this work - the robot pose estimation performance, and finally, Sec. 3.4 shows some examples of the generated vineyard maps using the proposed approach.

3.1. Particles clustering

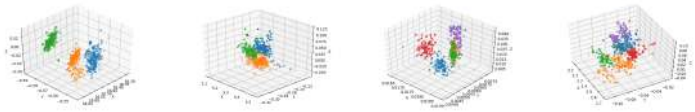
The performance of the clustering procedure is crucial for the proper behavior of the system. The application of the same homogeneous transformation to all the particles in each cluster requires that both inter- and intra-dispersion of the clusters to be as



(a) (b)

Figure 7: Schema of the approximated trajectory of the robot in both sequences.

precise as possible so that the localization system can perform accurately. Figure 8 presents four results of the clustering procedure applied to different distributions of sets of particles. By analysis of this figure, it is possible to sense different levels of difficulty



(a) (b) (c) (d)

Figure 8: Spatial layout (in meters) of the particles clusters using (a), (b) three clusters and (c), (d) five clusters.

on the clustering procedure, depending on the particles distribution. For example, in Fig. 8a is represented the easy case, where three clusters are computed from three sets of particles spatially separated. On the other hand, Fig. 8d represents a case where the particles are clustered in five sets and there is no clear visual way of separating them. Even so, the clustering procedure presents a good performance in this case, separating the total set in sets composed of spatially related particles. As a final note, it is worth noting that the restriction imposed in the proposed clustering method - equal-sized K-means++ - of considering N equal-sized cluster, can lead to non optimal solutions. Even so, the clustering algorithm presents a good performance, as represented in Fig. 8, in the way that it always finds solutions that split the probability density function that represents the particles distribution in well-defined groups.

In order to test the impact of the values of clusters in the final system performance, the SLAM pipeline was tested in sequence 1 for 8 different numbers of clusters. The number of particles was fixed to 500 in all the experiments. Figure 9 shows the RMS error (m) for all the different number of clusters. Here, we can conclude that the number

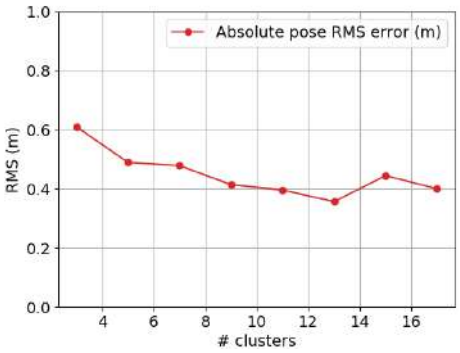
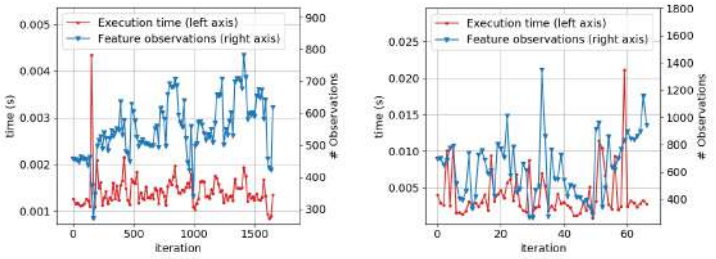


Figure 9: Absolute pose RMS error (m) of the VineSLAM-ICP approach in sequence 1 for different numbers of clusters.

of clusters can impact the performance of the SLAM system. This is because a specific number of clusters can be more suitable to split the particles probability density function. This can vary with the input data of the system or with the particles innovation criteria. For this reason, we consider that this hyperparameter should be tuned by the user for any specific use case. The most important thing is that the clustering algorithm improves the original SLAM approach performance, given the number of clusters chosen.

3.2. Execution time

The main goal of the clustering procedure is to allow the SLAM approach to use a scan-matching procedure and still perform online. Here, the online concept is defined by the AgRob V16 working loop frequency. In other words, the SLAM approach must have an execution frequency lower than the robots' loop frequency. In the use case present in this paper, AgRob V16 has a working loop frequency of 10Hz. The scan-matching procedure, and, specifically, ICP, can be computationally expensive, especially when working with dense inputs. Thus, the proposed version of VineSLAM should improve the original one, integrating ICP in the SLAM pipeline and considering the time restrictions. To evaluate if the scan-matcher satisfies these restrictions, the execution time per cluster was measured in the two test cases (sequences 1 and 2). Additionally, the number of input observations in each SLAM iteration is also presented. From the graphics present in Fig. 10, the extreme importance of the clustering procedure in the new VineSLAM pipeline is outlined. From these, we can conclude that the scan-matching algorithm presents an average execution time per cluster of approximately 4 milliseconds and 1.4 milliseconds for, on average, 632 and 554 input observations in each iteration, for sequences 1 and 2, respectively. If the SLAM pipeline executes one scan-matching per particle, the number of particles has to be very limited so that the time restrictions are not violated. For example, for sequence 1, with 25 particles the PF would waste, on



(a) Sequence 1

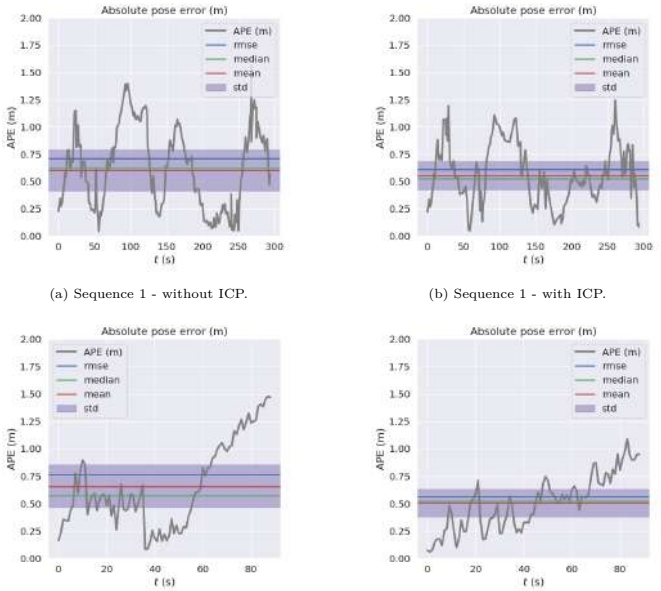
(b) Sequence 2

Figure 10: Execution time per cluster (s), and the respective number of feature observations considered in each scan-matching procedure.

average, the maximum cycle time imposed by the robot working rate (100 milliseconds), only computing ICP. On the positive side, the clustering procedure allows integrating the scan-matcher in the PF, considering the time restrictions. In this way, the filter can consider a relatively dense number of input features, perform the ICP, and still save time for other tasks such as the remaining routines of the PF and map registration.

3.3. Localization performance

The robot localization estimation is evaluated both in sequence 1 and 2 without the ICP-based approach and with it, using three clusters and 500 particles. The proposed approach is compared with the state-of-the-art SLAM algorithm, LeGo-LOAM [23]. This method extracts features from 3D LiDAR scan data, and is based on a LiDAR odometry algorithm, where consecutive 3D scans are matched to compute the relative 6-DoF pose iteratively. It is worth noting that all the SLAM pipelines are evaluated in sequences placed in vineyard long corridors, which constitutes the well known corridor translation estimation problem. This happens due to the fact that data incoming from sensors can be very similar along sequential robot positions over the corridor. So, this can lead to translation underestimation or, in the worst case scenario, estimating that the robot is stopped. In Aveleda's vineyard corridors, this can be a real issue, since the vine trunks are equally spaced and the corridor width is always constant. To evaluate the performance of the proposed approach in both sequences and compare it with the original VineSLAM algorithm, the robot localization estimation is described and represented in several different ways. Firstly, the absolute pose error (APE) is plotted over time in each sequence (Fig. 11), as well as its mean, minimum, maximum, median, root mean square (RMS) and standard deviation values. Table 2 summarizes these results. Additionally, the APE is mapped onto the robot trajectory in Fig. 12 to represent the displacement of the localization estimation at each location, in relation with the ground-truth. As a ground-truth, the GPS was used since it is the most accurate sensor present in AgRob V16. In sequence 1 there is another reference that might help to evaluate the performance of the methods, since the final robot position - point D of Fig. 7a - should approximately



(a) Sequence 1 - without ICP.

(b) Sequence 1 - with ICP.

(c) Sequence 2 - without ICP.

(d) Sequence 1 - with ICP.

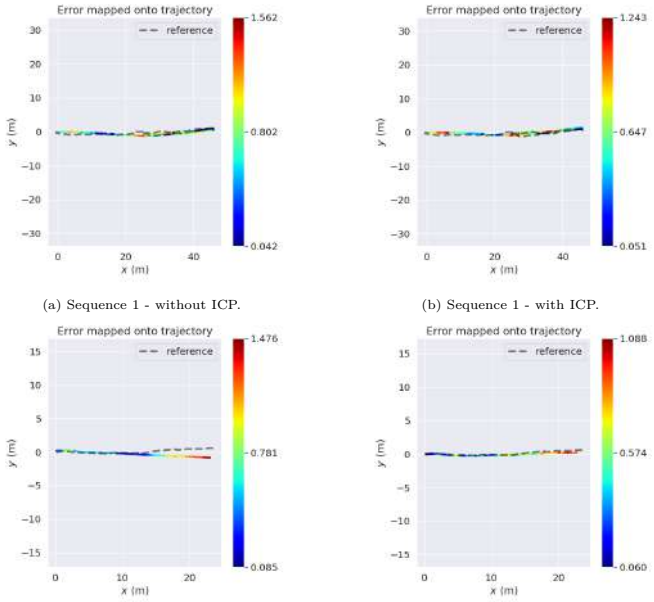
Figure 11: Representation of the APE obtained in both sequences with and without the ICP-based approach. The graphics also show the values for the RMS error, the error median, mean and APE standard deviation.

coincide with the first pure rotation where the robot faces the vineyard side - point B of Fig. 7a. Finally, in Fig. 13 are represented all the trajectories in the same graphic for each sequence including wheel odometry. Here, this last sensor is included to represent the difficulties of computing autonomous driving in these agricultural places, also due to the high wheel slippage.

For sequence 1, from Figs. 11a and 11b is possible to see that the ICP-based PF refinement leads to a lower mean and RMS APE error than using the standalone VineSLAM approach. From Table 2 is possible to infer that the proposed approach reduces the robot localization mean APE in 5 cm and RMS APE in 10 cm for this sequence. Looking for the error mapped onto trajectory represented in Fig. 12 and, more specifically, in Figs. 12a and 12b for sequence 1, it is possible to observe the improvement of the localization performance with ICP (Fig. 12b) since the APE is lower for the same color in comparison with Fig. 12a that represents the standalone VineSLAM solution. For sequence 2, the difference between the two configurations' performance is more significant.

Table 2: Absolute pose error evaluated using different metrics for the two configurations in both sequences.

	APE (m)	Maximum	Minimum	Mean	RMS
Sequence 1	VineSLAM ICP	1.24	0.05	0.55	0.61
	VineSLAM standalone	1.56	0.04	0.60	0.71
	Lego-LOAM [23]	8.73	0.16	2.69	3.36
Sequence 2	VineSLAM ICP	1.08	0.06	0.50	0.56
	VineSLAM standalone	1.48	0.09	0.65	0.76
	Lego-LOAM [23]	10.81	0.21	5.57	6.29



(a) Sequence 1 - without ICP.

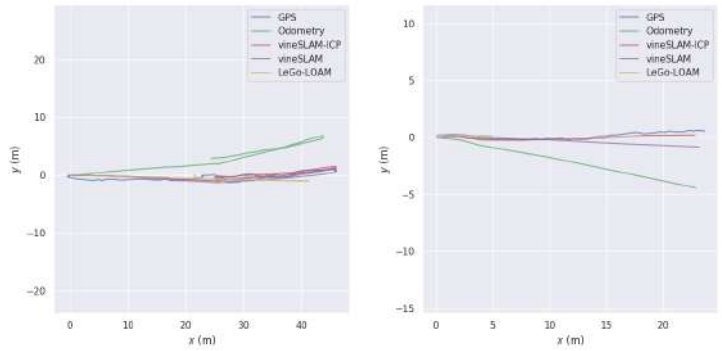
(b) Sequence 1 - with ICP.

(c) Sequence 2 - without ICP.

(d) Sequence 2 - with ICP.

Figure 12: APE mapped onto trajectory. The colorbar encodes the absolute deviation to the reference that is mapped onto the localization estimation in every instant.

From Figs. 11c and 11d, as well as in sequence 1, is also possible to infer that the ICP refinement leads to a lower mean and RMS APE error than using the standalone VineSLAM approach. Analyzing Tab. 2 leads to the conclusion that without the refinement,



(a) Sequence 1.

(b) Sequence 2.

Figure 13: Estimated robot path in both sequences versus the GPS ground-truth, wheel odometry, and LeGo-LOAM [23].

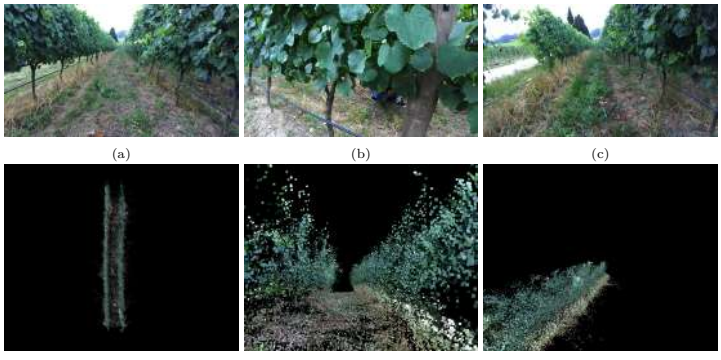
VineSLAM has a mean APE higher in 15 cm and an RMS APE higher in 20 cm. As in sequence 1, the error mapped onto the trajectory for sequence 2 shows that the RGB color bar for the localization estimation with the ICP approach leads to a smaller error during the robot path.

In what concerns to LeGo-LOAM [23], Table 2 and show that this method can not perform accurately in this vineyard context. This algorithm is highly affected by the corridor issue mentioned before. We can see that the performance in sequence 2 is highly lower compared to sequence 1. This is because sequence 1 is placed in a side corridor, with a nonsymmetric scene at the left (Fig. 6). On the other hand, sequence 2 is placed in the middle of the vineyard, which leads to highly symmetric sensor data. For this reason, in this sequence, LeGo-LOAM highly underestimates translation.

Overall, results show that the scan matching refinement procedure improves the robot localization estimation performance. In this complex challenging outdoor environment, this SLAM approach is capable of maintaining a reliable localization, outperforming LeGo-LOAM [23]. Figure 13 presents a visual way of comparing all the configurations performances. Although the theoretical higher difficulty of sequence 1 is in sequence 2, where the original version of VineSLAM deviates more from the ground-truth. In this sequence, the correction over wheel odometry by the original SLAM pipeline is lower. So, it is here where the proposed refinement has more impact. At the moment where VineSLAM misses estimating a rotation, which makes it deviate, the scan matching procedure corrects this performing an adjustment on the particle's position of each cluster. Figure 13b also shows the low accuracy of LeGo-LOAM in this particular scenario. Due to the presence of many adjacent corridors in sequence 2, this method frequently estimates that the robot is stopped when it is actually moving.

3.4. Vineyard mapping

Using 3D features with RGB information allows to perform a realistic 3D reconstruction of the vineyard corridors if the localization procedure is accurate. Figures 14 and 15 show different results of the 3D reconstruction performed in VineSLAM with the features used by the scan matcher. Also, to be aware of the input data used by the pipeline, Figs. 14a, 14b, 14c, 15a, 15b, 15c show the robot view of the vineyard in both sequences.



(a) (b) (c)

(d) (e) (f)

Figure 14: Snapshots of the on-board visual sensor images and the generated map from several views in sequence 1.

(a) (b) (c)

(d) (e) (f)

Figure 15: Snapshots of the on-board visual sensor images and the generated map from several views in sequence 2.

Here, the difference in the aspect of the vineyard with and without foliage is clear. From Figs. 14d, 14e, 14f, 15d, 15e, 15f is visible the resultant 3D reconstruction of the agricultural environment, resultant from the online map registration performed after each execution of the PF. These results allow to validate the performance of the proposed SLAM pipeline since the vineyard corridors 3D reconstruction is realistic.

4. Conclusion

Scan-matching algorithms can be computationally expensive when agents' density to align is high, making it difficult to use in online localization algorithms. This work proposes a refinement step to standard high-dimensional PFs that can incorporate a scan-matching algorithm in the SLAM pipeline using a designed online clustering procedure. The approach uses a stereo camera system to extract features and build 3D visual scans of an agricultural environment. Results show that the proposed PF refinement step improves VineSLAM performance, a SLAM approach that uses a standard high-dimensional PF to localize a robot in mountain vineyards. The experiments also prove that the proposed approach can build accurate 3D reconstructions of the agricultural environment with RGB information, the so-called 3D visual scans.

In future work, we would like to test our solution in different agricultural places such as forests and orchards, and in more challenging conditions, to test the system with different levels of illumination, speed, and wheel slippage. Also, we want to be able to deal with the well-known problem of particle degeneracy, which will help improve the final robot pose estimation. Finally, the integrated VineSLAM framework with the proposed refinement will be integrated into the AgRob V16 navigation stack, aiming to be the main localization and mapping system that allows the robot to perform missions in agriculture.

Acknowledgments

This work is financed by ERDF - European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 under the PORTUGAL 2020 Partnership Agreement, and through the Portuguese National Innovation Agency (ANI) as a part of project ROMOVI: POCI-01-0247-FEDER-017945.

References

[1] T. Andresen, F. B. de Aguiar, M. J. Curado, The alto douro wine region greenway, Landscape and Urban Planning 68 (2) (2004) 289 – 303, international Greenway Planning. doi:https://doi.org/10.1016/S0169-2046(03)00156-7.
URL http://www.sciencedirect.com/science/article/pii/S0169204603001567

[2] H. Durrant-Whyte, T. Bailey, Simultaneous localization and mapping: part i, IEEE Robotics & Automation Magazine 13 (2) (2006) 99–110. doi:10.1109/mra.2006.1638022.
URL https://doi.org/10.1109/mra.2006.1638022

[3] T. Bailey, H. Durrant-Whyte, Simultaneous localization and mapping (SLAM): part II, IEEE Robotics & Automation Magazine 13 (3) (2006) 108–117. doi:10.1109/mra.2006.1678144.
URL https://doi.org/10.1109/mra.2006.1678144

[4] S. Thrun, Simultaneous localization and mapping, in: Robotics and Cognitive Approaches to Spatial Mapping, Springer Berlin Heidelberg, pp. 13–41. doi:10.1007/978-3-540-75388-9_3.
URL https://doi.org/10.1007/978-3-540-75388-9_3

[5] R. Smith, M. Self, P. Cheeseman, Estimating uncertain spatial relationships in robotics, in: Autonomous Robot Vehicles, Springer New York, 1990, pp. 167–193. doi:10.1007/978-1-4613-8997-2_14.
URL https://doi.org/10.1007/978-1-4613-8997-2_14

[6] R. C. Smith, P. Cheeseman, On the representation and estimation of spatial uncertainty, The International Journal of Robotics Research 5 (4) (1986) 56–68. doi:10.1177/027836498600500404.
URL https://doi.org/10.1177/027836498600500404

[7] J. Castellanos, R. Martinez-Cantin, J. Tardós, J. Neira, Robocentric map joining: Improving the consistency of EKF-SLAM, Robotics and Autonomous Systems 55 (1) (2007) 21–29. doi:10.1016/j.robot.2006.06.005.
URL https://doi.org/10.1016/j.robot.2006.06.005

[8] F. Lu, E. Milios, Globally consistent range scan alignment for environment mapping, Autonomous Robots 4 (1997) 333–349.

[9] K. Murphy, S. Russell, Rao-blackwellised particle filtering for dynamic bayesian networks, in: Sequential Monte Carlo Methods in Practice, Springer New York, 2001, pp. 499–515. doi:10.1007/978-1-4757-3437-9_24.
URL https://doi.org/10.1007/978-1-4757-3437-9_24

[10] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, et al., Fastslam: A factored solution to the simultaneous localization and mapping problem, Aaai/iaai 593598 (2002).

[11] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, et al., Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges, in: IJCAI, 2003, pp. 1151–1156.

[12] G. Grisettiyz, C. Stachniss, W. Burgard, Improving grid-based SLAM with rao-blackwellized particle filters by adaptive proposals and selective resampling, in: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, IEEE. doi:10.1109/robot.2005.1570477.
URL https://doi.org/10.1109/robot.2005.1570477

[13] G. Grisetti, G. D. Tipaldi, C. Stachniss, W. Burgard, D. Nardi, Fast and accurate SLAM with rao–blackwellized particle filters, Robotics and Autonomous Systems 55 (1) (2007) 30–38. doi:10.1016/j.robot.2006.06.007.
URL https://doi.org/10.1016/j.robot.2006.06.007

[14] P. Besl, N. D. McKay, A method for registration of 3-d shapes, IEEE Transactions on Pattern Analysis and Machine Intelligence 14 (2) (1992) 239–256. doi:10.1109/34.121791.
URL https://doi.org/10.1109/34.121791

[15] A. Nüchter, K. Lingemann, J. Hertzberg, H. Surmann, 6d SLAM—3d mapping outdoor environments, Journal of Field Robotics 24 (8-9) (2007) 699–722. doi:10.1002/rob.20209.
URL https://doi.org/10.1002/rob.20209

[16] R. Tiar, M. Lakrouf, O. Azouaoui, Fast ICP-SLAM for a bi-steerable mobile robot in large environments, in: 2015 International Conference on Advanced Robotics (ICAR), IEEE, 2015. doi:10.1109/icar.2015.7251519.
URL https://doi.org/10.1109/icar.2015.7251519

[17] N. Fioraio, K. Konolige, W. Garage, Realtime visual and point cloud slam nicola fioraio willow garage, 2011.

[18] M. Tomono, Robust 3d SLAM with a stereo camera based on an edge-point ICP algorithm, in: 2009 IEEE International Conference on Robotics and Automation, IEEE, 2009. doi:10.1109/robot.2009.5152529.
URL https://doi.org/10.1109/robot.2009.5152529

[19] D. Holz, S. Behnke, Sancta simplicitas - on the efficiency and achievable results of SLAM using ICP-based incremental registration, in: 2010 IEEE International Conference on Robotics and Automation, IEEE, 2010. doi:10.1109/robot.2010.5509918.
URL https://doi.org/10.1109/robot.2010.5509918

[20] F. Bertolli, P. Jensfelt, H. Christensen, SLAM using visual scan-matching with distinguishable 3d points, in: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2006. doi:10.1109/iros.2006.281865.
URL https://doi.org/10.1109/iros.2006.281865

[21] R. Eustice, O. Pizarro, H. Singh, Visually augmented navigation in an unstructured environment using a delayed state history, in: IEEE International Conference on Robotics and Automation,

2004. Proceedings. ICRA '04. 2004, Vol. 1, 2004, pp. 25–32 Vol.1.

[22] D. Lowe, Object recognition from local scale-invariant features, Proceedings of the Seventh IEEE International Conference on Computer Vision 2 (1999) 1150–1157 vol.2.

[23] T. Shan, B. Englot, LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2018. doi:10.1109/iros.2018.8594299.
URL https://doi.org/10.1109/iros.2018.8594299

[24] J. Sprickerhof, A. Nüchter, K. Lingemann, J. Hertzberg, An explicit loop closing technique for 6d slam, in: ECMR, 2009.

[25] H. Bay, T. Tuytelaars, L. V. Gool, SURF: Speeded up robust features, in: Computer Vision – ECCV 2006, Springer Berlin Heidelberg, 2006, pp. 404–417. doi:10.1007/11744023_32.
URL https://doi.org/10.1007/11744023_32

[26] F. N. dos Santos, H. Sobreira, D. Campos, R. Morais, A. P. Moreira, O. Contente, Towards a reliable robot for steep slope vineyards monitoring, Journal of Intelligent & Robotic Systems 83 (3-4) (2016) 429–444. doi:10.1007/s10846-016-0340-5.
URL https://doi.org/10.1007/s10846-016-0340-5

[27] Z. Liu, Z. Shi, M. Zhao, W. Xu, Mobile robots global localization using adaptive dynamic clustered particle filters, in: 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2007. doi:10.1109/iros.2007.4399050.
URL https://doi.org/10.1109/iros.2007.4399050

[28] A. Milstein, J. N. Sánchez, E. T. Williamson, Robust global localization using clustered particle filtering, American Association for Artificial Intelligence, USA, 2002, p. 581–586.

[29] D. Arthur, S. Vassilvitskii, k-means++: The advantages of careful seeding, Technical Report 2006-13, Stanford InfoLab (June 2006).
URL http://ilpubs.stanford.edu:8090/778/

[30] A. Likas, N. Vlassis, J. J. Verbeek, The global k-means clustering algorithm, Pattern Recognition 36 (2) (2003) 451–461. doi:10.1016/s0031-3203(02)00060-2.
URL https://doi.org/10.1016/s0031-3203(02)00060-2

[31] S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics (Intelligent Robotics and Autonomous Agents), The MIT Press, 2005.

[32] A. S. P. de Aguiar, F. B. N. dos Santos, L. C. F. dos Santos, V. M. de Jesus Filipe, A. J. M. de Sousa, Vineyard trunk detection using deep learning – an experimental device benchmark, Computers and Electronics in Agriculture 175 (2020) 105535. doi:10.1016/j.compag.2020.105535.
URL https://doi.org/10.1016/j.compag.2020.105535

[33] L. Santos, F. Santos, J. Mendes, P. Costa, J. Lima, R. Reis, P. Shinde, Path planning aware of robot's center of mass for steep slope vineyards, Robotica 38 (4) (2019) 684–698. doi:10.1017/s0263574719000961.
URL https://doi.org/10.1017/s0263574719000961

5.4 Topological map-based approach for localization and mapping memory optimization

After VineSLAM presenting robust localization and mapping processes, there is the need to adapt the pipeline to the use in real agricultural applications. Robots that operate in such scenarios usually need to cover extensive fields. One of the major issues of SLAM algorithms is the increase need for memory resources over time due to the growth of the map size as the robot discovers new spaces. As referenced in the state-of-the-art review present in Chapter 2, localization and mapping algorithms should work under large-scale and long-term conditions. This thesis approaches this problem through the use of a topological map. This work was published in the Field Robotics Journal and is entitled *Topological map-based approach for localization and mapping memory optimization* (Aguiar et al., 2022b). The topological map extraction was proposed in a previous work (Santos et al., 2020) and uses satellite images to segment the space in a graph structure. This structure is used to perform a more efficient mapping. The mapping pipeline only loads the necessary graph nodes in each instant. The remaining nodes are stored in the processor system and can be loaded at any time and be used to load the map, update it, and localize the robot. Results show that this approach can maintain a memory allocation constant over time, which means that VineSLAM can be executed without time restrictions, and thus work in large-scale environments and long-term periods.

ORIGINAL ARTICLE

Journal Section

Topological map-based approach for localization and mapping memory optimization

André Silva Aguiar[1,2*] | Filipe Neves dos Santos[1*] | Luis Carlos Santos[1,2*] | Armando Jorge Sousa[1,3*] | José Boaventura-Cunha[1,2*]

[1]INESC TEC - INESC Technology and Science; 4200-465, Porto, Portugal

[2]School of Science and Technology, University of Trás-os-Montes e Alto Douro; 5000-801 Vila Real, Portugal

[3]FEUP, University Of Porto, Porto, Portugal

Correspondence
André Silva Aguiar, INESC TEC - INESC Technology and Science; 4200-465, Porto, Portugal
Email: andre.s.aguiar@inesctec.pt

Funding information
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101004085.

Robotics in agriculture faces several challenges, such as the unstructured characteristics of the environments, variability of luminosity conditions for perception systems and vast field extensions. To implement autonomous navigation systems in these conditions, robots should be able to operate during large periods and travel long trajectories. For this reason, it is essential that Simultaneous Localization and Mapping algorithms can perform in large-scale and long-term operating conditions. One of the main challenges for these methods is maintaining low memory resources while mapping extensive environments. This work tackles this issue, proposing a localization and mapping approach called VineSLAM that uses a topological mapping architecture to manage the memory resources required by the algorithm. This topological map is a graph-based structure where each node is agnostic to the type of data stored, enabling the creation of a multi-layer mapping procedure. Also, a localization algorithm is implemented, which interacts with the topological map to perform access and search operations. Results show that our approach is aligned with the state-of-the-art regarding localization precision, being able

*Equally contributing authors.

to compute the robot pose in long and challenging trajectories in agriculture. In addition, we prove that the topological approach innovates the state-of-the-art memory management. The proposed algorithm requires less memory than the other benchmarked algorithms, and can maintain a constant memory allocation during the entire operation. This consists of a significant innovation, since our approach opens the possibility for the deployment of complex 3D SLAM algorithms in real-world applications without scale restrictions.

KEYWORDS
SLAM, Agriculture, Topological Mapping, Memory Management, Autonomous Robots

1 | INTRODUCTION

The development of autonomous robots in agriculture is a challenging and active research topic (Emmi et al., 2014). To implement such systems, the autonomous navigation issue must be solved, i.e., robots should be capable of driving autonomously within multiple environments (Shalal et al., 2013; Abdallah et al., 2007). Consequently, autonomous robotic platforms should be endowed with robust localization systems that allow recovering their absolute pose in the agricultural environment (Vougioukas, 2019). Simultaneous Localization and Mapping (SLAM) allows calculating the travelled trajectory while mapping the environment simultaneously (Durrant-Whyte and Bailey, 2006; Bailey and Durrant-Whyte, 2006; Lowe et al., 2021). In agriculture, the implementation of SLAM is essential since it leads to the creation of maps that farmers and agriculture specialists can use in various tasks. These maps can be saved and posteriorly analysed to extract relevant information about the crops, such as canopy volume, yield state, and geo-referenced position of fruits, vegetables and trunks. Also, when robots have this mapping ability, they can perform autonomous operations such as precision agriculture (application of fertilizers, nutrients and water), plant protection, harvesting, monitoring, and planting (Roldán et al., 2018; Bergerman et al., 2016; Pinto de Aguiar et al., 2020). However, the challenging conditions of agricultural environments, such as characteristics of illumination and terrain irregularities, can difficult the perception stages and therefore compromise the SLAM algorithms' performance (Aguiar et al., 2021c). In addition, localization and mapping approaches should be prepared to deal with large-scale environments since agriculture is characterized by vast terrain extensions (Chen et al., 2021).

Large-scale localization and mapping bring several challenges, namely map memory management. When robots operate in extensive environments, approaches that are not developed in a memory-efficient manner can try to allocate more than the available memory resources of the processor that is being used (Suger et al., 2014). This challenges the scalability of SLAM systems and puts at risk the possibility for robots to perform autonomous long-term operations (Li et al., 2018). Although the scalability of SLAM approaches to accommodate large-scale agricultural tasks is still an open issue (Capua et al., 2018), there is already a state-of-the-art concept to decrease memory consumption of SLAM algorithms. Bosse et al. (2003) propose a graph representation of the environment, called a Topological Map. This concept allows SLAM approaches not to maintain a single and global coordinate frame but an interconnected set

of local maps in a graph-like fashion. Topological maps consist of a graph of multiple local maps of limited size. Thus, they allow intelligent map memory management by allocating only the strictly necessary graph vertexes. The graph nodes store the local maps, and the edges represent the spatial transformation between nodes and can be used to track the robot pose during the localization procedure (Garcia-Fidalgo and Ortiz, 2015). Figure 1 shows a possible topological map architecture for a vineyard.

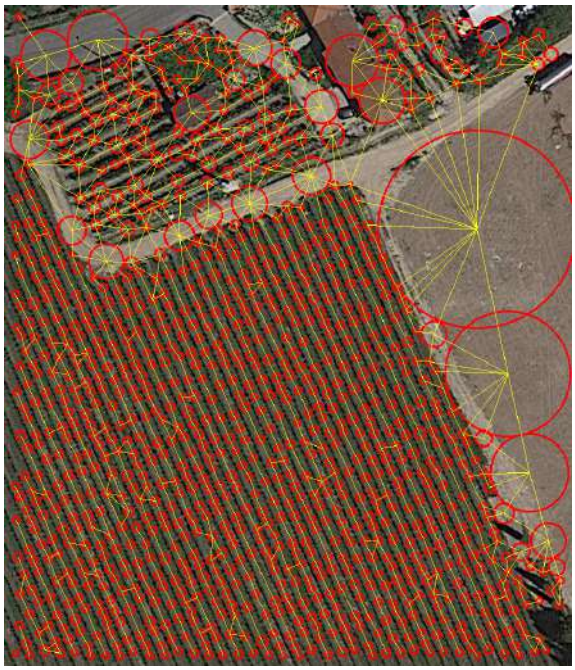


FIGURE 1 Topological representation of a vineyard characterized by an extensive area.

Topological-based localization and mapping bring several challenges into the development stage. In particular, and looking at the agricultural scenario, one of the main research questions is how are topological maps extracted given the characteristics of the environment. Also, there is the problem of minimizing the memory usage without compromising the localization and mapping procedures. And finally, how to store, load, and update local maps visited or re-visited by the robot. This work proposes a solution to these research questions with a localization and mapping algorithm called VineSLAM that uses a topological mapping approach efficiently. Our approach innovates the state-of-the-art by proposing a novel pipeline that enables SLAM algorithms to survive for unlimited amounts of time, without memory restrictions. In a previous work we proposed the extraction of topological maps of agricultural places from satellite images (Santos et al., 2020). In this work, we use this extraction procedure and propose a novel localization and mapping system based on topological concepts. The contributions of this work are threefold:

1. The creation of a topological representation of the agricultural environment that divides the space in a graph structure;

2. A multi-layer mapping procedure where the top layer, the topological, is agnostic to the bottom layer;
3. A localization approach that uses the multiple mapping layers to track the 6-DoF robot pose.

The main novelty of this approach is the support for any type of map in the topological nodes. This abstraction allowed us to create a multi-layer mapping algorithm, where the bottom layers are different representations of the environment, and the top layer serves as a memory management agent. In this way, our previous localization solutions (Aguiar et al., 2021a) is extended to the use of several types of feature maps and are adapted to deal with the submap architecture imposed by the topological map.

The remainder of this paper is structured as follows. Section 2 reviews the current state-of-the-art on topological localization and mapping. Section 3 details the contributions of this work. Section 4 presents the test and validation of the proposed methods in experiments considering real data collected by our agricultural robots. Finally, Section 6 details the conclusions of this work.

2 | RELATED WORK

The mapping procedure is crucial for the localization performance (Nüchter et al., 2007; Fairfield et al., 2010). The accuracy of the onboard sensors and the quality of the data post-processing algorithms dictate the quality of the perception of the surrounding environment by the robotic platform. In agriculture, this is especially true since, in most cases, the environment presents harsh conditions for robotics localization and mapping (Aguiar et al., 2020a). When the localization and mapping algorithms succeed, robots can create several types of maps of the environment that can be used by humans or even by other robots to operate.

Many techniques were proposed in the literature to solve the SLAM problem. Graph-based solutions (Grisetti et al., 2010) are popular, especially in 3D SLAM. Folkesson and Christensen (2007) propose a SLAM approach for outdoor applications where localization and mapping procedures are reduced to a graph problem. The algorithm was successfully tested in an outdoor environment, operating at 5Hz. In the same context, Schuster et al. (2015) proposes an online graph optimization solution to estimate the 6-DoF robot pose and 3D maps that are then used in a multi-robot context. This is done through the use of an inertial-vision system. Le et al. (2019) propose a solution based on (Zhang and Singh, 2014) and (Shan and Englot, 2018), adapted for agricultural robots. This work uses a LiDAR odometry approach and a Graph-SLAM optimization procedure to localize the robot and map the environment. The algorithm was successfully tested in agricultural scenarios. Besides optimization-based solutions, filter-based approaches are also considered in the literature, particularly the Extended Kalman Filter (EKF). For example, Cole and Newman (2006) use an EKF to localize the 3D robot position in outdoor environments. The filter state is composed of the entire robot trajectory, and LiDAR scan registrations are used to form measurements between the robot positions over time. Although the state-of-the-art already solves the SLAM problem efficiently, robotic operations are not only dependent on the localization and mapping accuracy but also on their ability to survive for large periods and extensive trajectories. In other words, these classes of algorithms should have large-scale capabilities (Dellaert et al., 2010). One of the critical aspects that can put at risk the large-scale operation of SLAM approaches is the map memory management (Labbé and Michaud, 2017). As the areas visited by the robot start to grow, the memory requirements are incrementally more significant if the algorithm considers a single map memory allocation without any management mechanisms. To address this issue, several techniques were already proposed in the state-of-the-art, as described in Table 1. One of the popular approaches is to perform map compression (Van Opdenbosch et al., 2018; Contreras and

TABLE 1 Approaches to address memory management optimization in large-scale environments.

Approach	Description	Advantages	Disadvantages
Map compression	Use standard compression algorithms (lz4[1], Zstandard[2], image compression) to decrease the map memory requirements.	Do not compromise time efficiency; can highly improve memory management.	Possible loss of important information; challenge to restore original information when decompressing data.
Hierarchical trees	Popularized by KdTrees and OcTrees - structures that recursevely divide the space into k sub-volumes.	Do not need to know the map size a-priori; supports multiple voxel sizes.	Despite efficiently storing map data, memory requirements still grow over time.
Topological maps	Partition of the map into a graph structure; local allocation of graph nodes.	Constant memory resources are required by allocating only the necessary information in each instant.	Increase mapping complexity; possible slower access and insert operations.

Mayol-Cuevas, 2017). For example, for visual information in video sequences, visual descriptors can be reused between frames and store only the differences found in the current frame (Baroffio et al., 2015). Also, data compression algorithms can be applied to reduce the memory requirements of maps. In this scope, Pedrosa et al. (2018) compress the map with state-of-the-art data compression techniques. The major innovation of this approach is a caching mechanism that avoids the compress-decompress penalty. Another possible approach is the use of memory-efficient data structures such as hierarchical trees: OctoMap (Hornung et al., 2013) or KdTrees (Greenspan and Yurick, 2003). These structures recursively divide the space into k sub-volumes until a minimum voxel resolution is achieved. One of the main advantages of these structures is that the map's extent does not have to be known a-priori. Instead, the map is dynamically expanded as needed. Even with this intelligent management, the maps can grow to a size that is not feasible to standard computer specifications for large-scale environments. Thus, the concept of topological map is essential to solve the problem of memory management for robots operating in vast extensions. This data structure organizes the map in a graph structure, i.e., it creates a set of nodes connected by edges that can store information individually. Since map data is loaded from and saved to graph nodes considering their proximity to the robot, the mapping resources can be maintained approximately constant over time. This consists of an essential breakthrough for large-scale SLAM algorithms.

Several works that use the concept of topological maps have already been proposed in the literature, as referenced in Table 2. The computation of the topological map can be either done before the robot operation, or in an online manner, i.e., during the SLAM operation. For example, Heiwolt et al. (2020) extract the topological map from high-resolution satellite images. This means that the map is pre-partitioned before the robot operation, which has two main advantages: there is no need to spend time and resources computing the map nodes and edges during the SLAM procedure; and, allows a validation of the map structure before its usage to optimize the memory management. On the other side, Bernuy and Ruiz Del Solar (2015) propose an online topological map built incrementally from

[1] `https://github.com/lz4/lz4`
[2] `https://github.com/facebook/zstd`

TABLE 2 Summary of the current state-of-the-art on outdoor topological-based localization and mapping.

Reference	Extraction	Localization and Mapping
Emmi et al. (2021)	Semantic field place partitioning. Categorization of a crop in four classes, used to build the topological map.	LiDAR- and camera-based localization and metric mapping inside the topological map.
Heiwolt et al. (2020)	Automated topological map extraction from high-resolution satellite images.	N/A.
Khan et al. (2020)	N/A.	Particle Filter (PF) localization algorithm. Topological map used to constraint the particles pose.
Bernuy and Ruiz Del Solar (2015)	Topological nodes extracted using a graph structure on the semantic representation of images and context information.	Localization reduced to the problem of finding the topological node corresponding to the observations obtained by the robot sensors.
Schleicher et al. (2009)	Online topological map creation by travelling unknown environments. Newly visited places are converted in topological nodes.	Low-level metric SLAM approach using a visual system and GPS. High-level topological procedure used to maintain the global map consistency.
Bradley et al. (2005)	Creation of topological database with feature vectors for each image, extracted in every iteration.	Topological information stored in the database used for localization-only in posterior runs of the algorithm in the same environment.
Ehlers et al. (2020)	Creation of topological nodes with semantic meaning every time the robot enters a new environment (suitable for indoor and outdoor).	Localization and mapping with any state-of-the-art SLAM algorithm - topological structure agnostic to the SLAM method.
He et al. (2006)	Topological nodes created incrementally by finding prototype image-features that can represent any image within a portion of the environment.	Visual mapping and localization using topological visual places as reference.

semantic information. In this work, only road scenarios are considered, and the map nodes are labelled as road and non-road (environmental nodes). Schleicher et al. (2009) also present an online topological map creation by travelling unknown environments. In this work, newly visited places are converted into topological nodes. The localization

(a) AgRob robot.

(b) Weta robot.

(c) Modular-E robot.

FIGURE 2 Autonomous robotic platforms used in the scope of this work.

procedure is divided into a low-level thread where SLAM is solved with a visual system and GPS and a high-level one where the topological procedure is used to maintain the global map consistency. Compared with offline topological map extraction algorithms, these approaches have the advantage of presenting a single solution that solves all the problems: localizing the robot while mapping the environment with efficient memory management.

This work is inserted in the context of robotics in large-scale agricultural environments. The proposed solution is deployed in our robot built from a commercial platform (Fig. 2a) and on our last generation and built in-house robots Weta (Fig. 2b) and Modular-E (Fig. 2c). These built in-house platforms are specially designed for harsh environments characterized by terrain irregularities, presenting a mechanical structure capable of surviving in such conditions. All robots are equipped with cameras, LiDARs, RTK GNNS, and IMU sensors.

The implemented robots operate in vineyards with long corridors and vast extensions. For this reason, a memory management procedure for the proposed SLAM system is essential. In this sense, we propose a topological map to efficiently store the information and provide intelligent resource management.

3 | TOPOLOGICAL LOCALIZATION AND MAPPING

Besides the challenging conditions that agricultural environments bring to localization, a key research question is also imposed by them: how to manage and optimize the memory consumption to map the entire environment without restrictions? To address this issue, this paper proposes a solution based on a topological map to efficiently store the map data during the Localization and Mapping operation. This map is agnostic to the type of data stored in each node, which enables the creation of a multi-layer mapping structure considering multiple modalities of map features. Finally, we propose a localization approach that interacts with the topological map efficiently. This entire pipeline is called

FIGURE 3 Satellite image from Aveleda vineyard (41°12'19.8"N 8°18'26.5"W) in Portugal.

VineSLAM, and all of these contributions are detailed below.

3.1 | Topological Map Extraction

The first stage of the proposed algorithm is the topological map extraction. In previous work, we proposed a topological map extraction procedure from satellite images for path planning algorithms (Santos et al., 2020). This work briefly describes this approach and its adaptation to the SLAM context.

The main stages from the original satellite image (Fig. 3) until the final map are represented in Fig. 4. The first step is the extraction of an occupancy map from the original RGB satellite image (Fig. 4a). For this, a Support Vector Machine (SVM) classifier is used to classify pixels according to two classes: vegetation and path lines of the vineyard. A descriptor based on the Local Binary Pattern codes (LBP), a gray-level invariant texture primitive, is used to perform this classification. This step is essential for the proper extraction of the final map since the presence of obstacles in the map will be used to compute the graph nodes. After having a reliable occupancy grid map, the next step is to extract a Voronoi diagram (Fig. 4c), which consists of dividing the space by Voronoi segments. These segments represent all the points in the plane that are closer to a given seed than to any other. Its construction originates from a beforehand distance map (Fig. 4b), which contains the Euclidean distance of every cell to the closest obstacle. The development of the algorithm was based on the work of Lau et al. (2010). The final step for the construction of the graph consists of computing the set of interconnected circles (Fig. 4d) based on the Voronoi diagram and the distance map. Each circle location corresponds to the corresponding Voronoi vertex, and its radius is computed through the distance map. The next step consists of finding the connections between the circles. For that operation, all the pixels of each circle

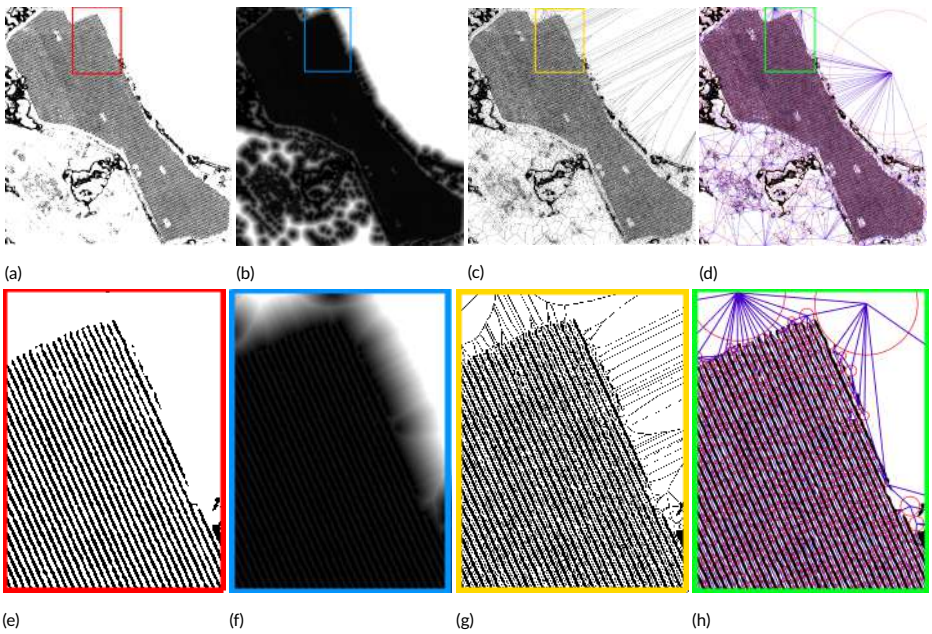


FIGURE 4 Step-by-step representation of the topological map extraction. This procedure starts by (a) the extraction of an occupancy grid map, followed by (b) the computation of a distance map, (c) a voronoi graph, and finally (d) a fully-connected topological map. Figures (e-f) are extractions of portions of figures (a-d), respectively.

will have associated a unique label. Then all the pixels containing a label are expanded until a different label is found. This operation is similar to a recursive process of a morphological operation of erosion until there are no more pixels without a label associated. The interconnection between graph nodes is then computed by finding the zones where the labels change. This results in the topological map.

For the Localization and Mapping context, one of the key information is the area occupied by each node and not only the graph morphology. For this reason, a place delimitation procedure is performed after extracting the topological map. In this step, the algorithm takes advantage of the pixel labels extracted before to define delimited places on the map. In our previous work, the place delimitation was performed by approximating each label to the nearest not rotated rectangle. In other words, each set of pixels belonging to the same label is converted into a single rectangle. Each rectangle is computed to contain all the pixels belonging to the same label without considering rotation.

These rectangles will then represent a sub-map area that can be loaded and saved during the Localization and Mapping operation. However, as seen in Fig. 5a, this approach is inefficient since it computes a lot of overlapping areas between rectangles. Not rotated rectangles can not approximate the topology of the environment efficiently. With this solution, this would mean that different graph nodes would contain the same information, which will lead to an unnecessary allocation of memory resources. To solve this, in this work is proposed an adaptation of the algorithm to consider oriented rectangles (Fig. 5b). Thus, each labelled area is approximated by the rectangle that best fits it.

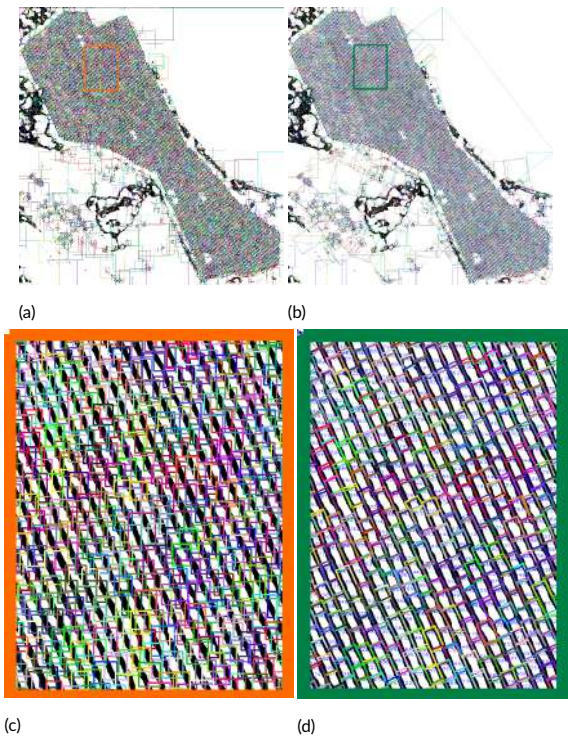


FIGURE 5 Place delimitation of the topological map. In our previous work (a,c) there was a lot of overlapping between rectangles. To adapt this approach to the SLAM context, we propose (b,d) a more efficient place delimitation where the overlapping area between rectangles is minimized. This will lead to a more efficient mapping process since the overlapping memory allocation between nodes is minimized.

To do so, we find the minimum area rectangle that contains all the points of a given label, returning also the angle of the oriented rectangle. Since this is performed at the image level, we make use of a state-of-the-art algorithm implemented in the OpenCV framework[3]. This way, it is possible to minimize the overlapping between graph nodes and have a more memory-efficient mapping procedure.

3.2 | Multi-layer Mapping

In this work, a novel multi-layer mapping architecture is proposed. This mapping structure was implemented due to two main reasons: to obtain an efficient memory management algorithm since our robots are inserted in large-scale agricultural environments; and to consider different types of features for the representation of the agricultural environment. Figure 6 represents this architecture, where one can see that the manager of the map is the topological graph. As we move down in the structure, the representation is increasingly lower-level until we access the cell data.

[3] `https://theailearner.com/tag/cv2-minarearect/`

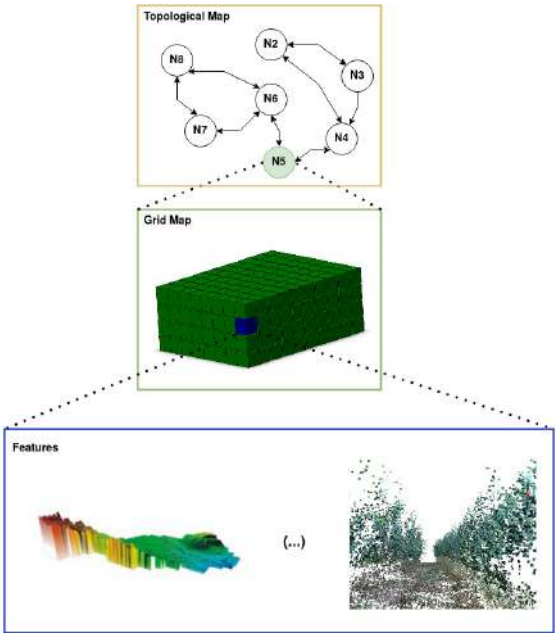


FIGURE 6 Multi-layer mapping structure. Each topological node N_i maintains a grid map consisting of a submap that can be allocated and deallocated as needed. The topological map is agnostic to the type of features present in the map since the grid map manages and defines the data structures present in each cell.

Here, the definition of cell can be of any type, which enables the support for any type of features and data structures in this multi-layer mapping. This work shows our use case, a multi-layer mapping consisting of 3D features extracted from LiDAR data and semantic features extracted from visual data.

Depending on the mapping approach, SLAM algorithms either have to pre-allocate all the memory required for the entire mapping operation, or have to use more efficient data structures that allocate more memory over time to fulfill the needs of the environment where the robot is located. The first approach is usually accomplished with grid maps. These structures discretize the space into cells and provide fast and efficient access and search algorithms. To access the data stored in each cell, a uni-dimensional operation is sufficient, i.e., each cell is accessed simply through its index. More complex data structures such as Octomaps and KdTrees provide a slower access approach since they are built using a tree structure, and to access a node is required to travel from the top of the tree through the parent nodes. On the other hand, grid maps are more inefficient in terms of memory allocation because they require the entire map memory to be allocated a-priori. In this work, we solve this issue using the topological map. Our localization and mapping approach takes advantage of the fast and efficient access algorithm provided by grid maps and optimize the memory allocation using the topological structure.

Figure 7 shows the mapping strategy implemented in this work. As seen, during the localization and mapping process, three main operations are required: (1) access, to get the data stored on a given cell; (2) insertion, to update the cell data given a new feature; and (3) deallocation, to store a given graph node into the external memory and free

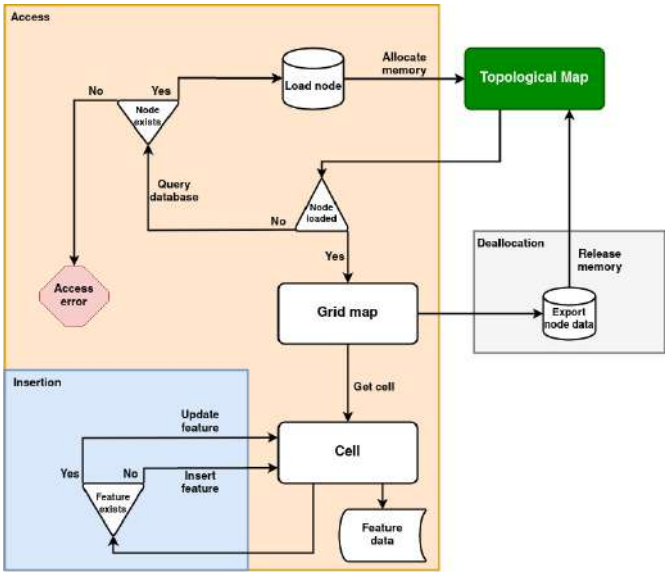


FIGURE 7 Mapping interaction flowchart.

the RAM memory used by the local map of the corresponding node. The access operation is critical since it dictates the execution time performance of both the mapping and localization processes. This operation can be used either to get a given cell data or update it by inserting new features. To access a given cell, we have to go from the higher to the lower level, starting from the topological graph until the cell itself. As example, suppose that the mapping system wants to insert a 3D feature with coordinates $[x_f, y_f, z_f]$ into the map. The first step is to find the correct topological node that will store the feature. The more straightforward solution would be to search for the node that contains the cell that will store the feature. However, this solution is inefficient, especially when dealing with a high-dimension graph, due to the high number of nodes. To overcome this, we implement a faster solution to get the topological node correspondent to any feature. This implementation is based on a lookup table represented in an image, where each pixel color is linked with a node. This image is a full representation of the topological map and is built using the place delimitation procedure described in section 3.1. Using this information, each pixel inside a given rectangle extracted in the place delimitation procedure is assigned to a numerical label corresponding to a given topological node. The access to the lookup table is performed by converting the 3D point location to the image space. The topological node corresponding to the given point is then given by only accessing the image at the transformed coordinates.

Thus, there is a function $f(x_f, y_f, z_f)$ that maps the 3D location of a given feature to its corresponding node, only by accessing the lookup table. If the node memory is already allocated, the process can continue. Otherwise, a query is done to the external memory where the nodes' information is stored. If the node is actually stored, the program allocates memory and loads it into the topological map. After that, the grid map stored inside the topological node is accessible and can be used. To insert the feature, the function $h(x_f, y_f, z_f)$ retrieves the cell where it will be stored by direct access. With this, the feature can be either inserted or updated if it already exists on the cell.

This configuration allows strategical memory management. In each iteration of the localization and mapping

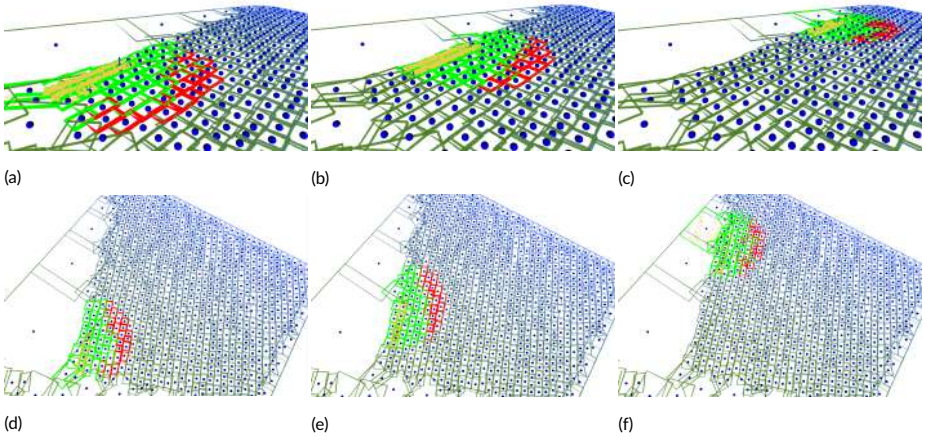


FIGURE 8 (a)-(c) Side and (d)-(f) top view of the online topological-based localization and mapping procedures. The topological map is loaded by the SLAM algorithm and represented by oriented rectangles (green rectangles) with a node location (blue spheres). This figure shows three separate iterations of the process. Active nodes are represented as red rectangles, and allocates nodes in highlighted green. All the other nodes are inactive and without allocated memory in these specific iterations.

procedures, the algorithm decides the active nodes and the allocated nodes. The active nodes are the ones that can be loaded in each iteration and are computed by applying a distance threshold t_d, i.e., all the nodes that have their center closer to the robot than t_d are considered active. The allocated nodes are active nodes that were already used either to map or, as detailed later, for localization. For example, when the mapping process wants to insert a feature in the active node i, the memory for this node is reserved, and this node passes to the set of allocated nodes. On the contrary, if a feature is to be inserted, but the lookup table returns a node that is not currently active, the insertion operation is discarded. At the end of the SLAM loop, when the robot localization is updated, the set of active nodes is updated by deallocating the memory for nodes that are now more distant than t_d from the robot. The data contained in these nodes is stored in individual files in the external memory so that it can be loaded later in the process or for post-processing after the SLAM operation. With this implementation, it is intended that the RAM memory used remains approximately constant during all the robot operations. The threshold t_d defines the range allowed for mapping and has a direct influence in the memory used by the program. Figure 8 illustrates the multi-layer mapping process. This figure shows three separate iterations of the SLAM algorithm, and one can see that only the necessary nodes are allocated in each one (highlighted green rectangles). In this case, the mapping was performed using 3D features extracted from a 3D LiDAR, and one can see that only the features stored in the allocated nodes are visible in each iteration (yellow dots on Figs. 8(a)-(c)), since the others are stored in the external memory. This entire operation is demonstrated in the following video: `https://youtu.be/cy4wGZUzB2Q`.

This architecture allows the algorithm to perform efficiently in large-scale agricultural environments. As mentioned in this work, each cell is defined by a set of semantic and 3D LiDAR features. In previous works, we proposed the extraction of natural features from vineyards (Aguiar et al., 2020b, 2021b) and the inclusion of 3D features in the

(a) (b)

FIGURE 9 (a) Semantic features extracted from an image of a vineyard canopy, and (b) corner (yellow) and planar (red) features extracted from 3D LiDAR data. The two types of features are stored in the proposed mapping architecture, and are used to provide a rich representation of the environment.

SLAM process (Aguiar et al., 2021a). In this work, we use the proposed architecture to create a multi-layer mapping procedure that considers both these semantic features and 3D LiDAR features. To do so, two different feature extractors are implemented. The first, represented in Fig. 9a, is based on a Deep Learning (DL) model trained to extract vine trunks and grape bunches from images. These features are then converted from image to world coordinates and stored in the topological structure presented before. The second, represented in Fig. 9b, are two types of features extracted directly from a raw point cloud generated by long-range 3D LiDARs: corner and planar features. Corners are features located on sharp edges, and the planar features are located along flat surfaces. With this, it is possible to achieve a rich representation of the agricultural environment. It is worth noting that the mapping architecture is agnostic to the type of data stored in the topological nodes, and thus each application can adapt the definition of grid map cell to its own context.

3.3 | Localization using Topological Concepts

The localization procedure aims to compute the robots' 6-DoF pose using the previously described feature extraction and mapping algorithms. To this purpose, this work implements a PF with the novelty of interacting efficiently with the topological architecture proposed. The PF is used due to its modularity in that it can consider different modalities of features and sensors just by adding new weight functions. Also, this filter can consider different types of noise models besides Gaussian noise, which can be an advantage for environments characterized by more complex models. The PF is standardly divided into three main steps: a prediction step where the particles are innovated through a motion model, the particles' weight calculation given the observed features and the resampling step to replace particles with lower weight by others with higher weight. The major contribution of this work is in the weight calculation procedure since it is in this step that the filter interact and use the topological structure.

To predict the particles' likelihood distribution, they are innovated through a 6-DoF model proportional to an estimated relative motion. To estimate the frame-to-frame robot motion, the control input $\boldsymbol{u}_t$ is extracted from the robot wheel odometry. Let us define the following matrices:

- $\Delta \boldsymbol{T}$: the wheel odometry increment represented as an homogeneous transformation; and
- $\boldsymbol{T}_n$: an homogeneous transformation matrix computed by sampling a minimal parameter space (6-DoF) from a Gaussian distribution with standard deviation proportional to the norm of the displacement measured by the wheel odometry.

Thus, a particle j is innovated at the instant t as follows:

$$\boldsymbol{T}_{j,t} = \boldsymbol{T}_{j,t-1}\, \Delta \boldsymbol{T}\, \boldsymbol{T}_n. \tag{1}$$

After spreading the particles in a given iteration, each hypothesis is tested using the feature extraction described before. In this work, the 3D LiDAR features are used to localize the robot. The system iteratively builds a map with corners and planar features and uses the observations in each iteration to compute the best alignment with the global map. Thus, the local N observations are converted to maps' reference frame using each particle $\boldsymbol{T}_{j,t}$ pose as follows:

$$\tilde{\boldsymbol{m}}_{f_i} = \boldsymbol{T}_{j,t}\, \boldsymbol{m}_{f_i},\ i \in \{1, \ldots, N\}. \tag{2}$$

The critical step in this calculation is to find correspondences between features observed in each iteration and the ones present on the global map. This is the major novelty of the localization algorithm since, to find correspondences, the nearest neighbor algorithm is implemented considering the topological architecture proposed. In this way, the access operation represented in Fig. 7 is executed per particle for every feature. The graph node is chosen using the lookup table method described, considering the projected feature position $\tilde{\boldsymbol{m}}_{f_i}$. After having the node, the cell is searched by indexing the grid map stored inside the node using the feature coordinates. Thus, each particle searches for the cells where the correspondences of each feature should be. In each cell, it is possible to store multiple features, and thus a search for the nearest correspondence is executed. Considering the set of K feature correspondences found $\{\tilde{\boldsymbol{m}}_{f_i} \leftrightarrow \boldsymbol{m}_{f,g_i} \ : \ i \in \{1, \ldots, K\}\}$, where the subscript g denotes for features in the *global* map, the point-feature weight sub-function is computed by the following equation:

$$W_f = \frac{1}{\sqrt{2\pi}\sigma_f} \sum_{i=1}^{K} \exp\left(\frac{-1}{\sigma_f} \cdot ||\tilde{\boldsymbol{m}}_{f_i} - \tilde{\boldsymbol{m}}_{f,g_i}||\right), \tag{3}$$

where σ_f is the standard deviation of the point-feature measurement, and $||.||$ represents the L2 norm. Equation 3 states that the particles weight increases exponentially with the decrease of distance between feature correspondences and with the number of correspondences found.

The fact that the search for the topological nodes and the grid map cells is done by simply indexing a lookup table and the grid map itself ensures the fastest access to the cell data as possible. Also, the topological architecture brings another great advantage to the localization algorithm. Since the topological graph only loads the necessary portions of the map (the ones closer to the last robot pose calculated), the number of local minimas is reduced. Consider a scenario of a woody crop vineyard with successive long and symmetric corridors. If the portions of the map corresponding to multiple vineyard corridors were loaded in the SLAM program, and if for some reason a particle is innovated to the incorrect vineyard corridor, this would be a particle with high weight and with a wrong estimate of the robot pose. By restricting the global map to the essential and necessary parts and considering a highly symmetric environment, we are reducing the number of possible local minimas.

As referenced before, one of the advantages of this PF is its modularity, i.e., the capacity of considering multiple

modalities of sensors by stacking the corresponding weight functions. Due to the harsh characteristics of the agricultural environments where our robots are inserted, the PF allows the fusion of the 3D LiDAR with an Real Time Kinematic (GNSS-RTK) and an Inertial Measurement Unit (IMU) sensor. To incorporate them, the weight function is designed considering a negative exponential function for each one. In this case, the particle weight calculation can be directly inferred without transformations since both sensors provide absolute observations. This being said, considering the RTK absolute robot position observation $\boldsymbol{X_{r,xyz}}$ and the jth particle position $\boldsymbol{X_{j,xyz}}$, the particle weight is calculated as follows:

$$W_r = \frac{1}{\sqrt{2\pi}\sigma_r} \exp\left(\frac{-1}{\sigma_r} \cdot ||\boldsymbol{X_{r,xyz}} - \boldsymbol{X_{j,xyz}}||\right), \tag{4}$$

where σ_r represents the standard deviation of the RTK position observation. It is worth noting that during the experiments detailed in section 4 the GNSS-RTK was not used to perform a fair comparison of the proposed algorithm's localization precision with state-of-the-art approaches. For the IMU sensor, the weight function is adapted to work with the Euler angle representation. Let $\mathcal{L} = \{\phi, \psi, \theta\}$ represent the three orientation degrees of freedom and $\boldsymbol{X_{\mathcal{L}}}$ represent a vector with the three orientations. The particle weight is calculated applying the following equation:

$$W_u = \frac{1}{\sqrt{2\pi}\sigma_u} \prod_{\eta \in \mathcal{L}} \exp\left|X_{u,\eta} - X_{j,\eta}\right|, \tag{5}$$

where σ_u represents the standard deviation of the IMU observation, $X_{.,\eta}$ represents the extraction of a single orientation component from the pose vectors, and $|.|$ represents the absolute value. Thus, in this case, the particle weight is computed as the product of differences between the IMU absolute orientations and the particle's orientations.

To fuse all the weight subfunctions in a single function and compute the final particle weight, we multiply all the sub-function results as follows:

$$W = W_f \cdot W_r \cdot W_u. \tag{6}$$

The balance between the multiple modalities is controlled by the standard deviation of each observation.

After computing all the particles' weights, they are resampled. The resampling step of the PF is used to substitute low-weight by high-weight particles. In this work, the multinomial resample algorithm (Douc and Cappe, 2005) was implemented to accomplish this. This approach draws N samples from a uniform distribution u_i and selects the particle j for replication if

$$u_i \in \left[\sum_{p=1}^{j-1} w_p, \sum_{p=1}^{j} w_p\right), \tag{7}$$

where w_p represents the particle's p weight. To avoid the well-known problem of particle degeneracy that happens when either all the particles are in the wrong place or they are highly condensed, resampling is not executed for all iterations. This method is only employed when a significant robot motion is observed (either in translation or rotation in the six degrees of freedom). The user can set the amount of motion required to perform resampling.

3.4 | Implementation Details

This section intends to briefly describe some implementation details that were taken into consideration to optimize the memory management of the algorithm. The algorithm is entirely implemented in the C++ programming language and is designed on top of the ROS2 platform[4]. All the structure was implemented from scratch: the multi-layer mapping, including the topological and grid map structures, and the PF-based localization algorithm. The key optimizations performed were on the definition of the topological and grid map classes. These optimizations consist of the way of declaring the members inside the classes. In this context, all the members are declared as pointers since each pointer, considering that it does not have allocated memory, only occupies 8 bytes of RAM memory, whatever type it has. A simple structure with two doubles would occupy twice the memory. Considering that the topological nodes and grid map cells are complex structures, this approach boosts the memory performance of the algorithm. With this being said, every topological node contains a pointer to a grid map. Thus, only the allocated nodes require memory allocation and occupy more than 8 bytes. In the same way, each grid map is composed of an array of cells. To optimize the memory allocation of the grid maps, each array index is a pointer to a cell. In this way, only non-empty cells are allocated. This is of extreme importance because the cells can have complex definitions, and they can be hidden when are not in use, through the use of pointers. Consider a graph node with a size of 10x10x10 meters and a resolution of 0.25 meters. This would result in a grid map with 4000 cells. If, in average, each cell occupies 14 bytes of memory, this would result in 56MB of allocated memory for the entire grid map. Now, consider that only 20% of the map is occupied and that we use pointers to hide the information of non-allocated cells. With this, only 36.8MB would be needed. This consists of a considerable memory saving and is highlighted in large-scale environments such as extensive vineyards.

4 | RESULTS

To test our approach we make use of two sequences correspondent to two different vineyards: Quinta da Aveleda (41°12'19.9"N 8°18'26.6"W), and Quinta do Seixo (41°10'00.0"N 7°33'18.3"W), both located in the north of Portugal. Both sequences are publicly available at `https://doi.org/10.5281/zenodo.5142159` and `https://doi.org/10.5281/zenodo.5142003`. Figure 10 shows our robotic platform navigating autonomously on the second vineyard. As visible, this is a mountain vineyard that presents high hills and steep slopes, which are challenging conditions for SLAM algorithms. The validation of the proposed approach is divided into three main categories:

- Memory management evaluation, where the topological architecture is compared with the usage of a standard 3D grid map;
- Execution time evaluation, where the proposed approach is extensively evaluated considering different number of particles; and,
- Localization and Mapping evaluation, where the performance of the SLAM algorithm is tested using satellite images as ground truth both for the localization and mapping stages.

Both sequences evaluate the large-scale capability of the proposed approach due to their high extension: 282.7 meters for sequence 1 and 337,1 meters for sequence 2. In addition, they present different challenges to the algorithm: on the one hand, a woody-crop vineyard characterized by extensive corridors, and on the other hand, a mountain vineyard composed of steep slope hills. Also, both sequences are composed of unstructured scenarios, that are characteristic of agricultural environments. For all these reasons, the experiments are suitable and challenging for

[4] `https://docs.ros.org/en/foxy/index.html`

FIGURE 10 Our robot navigating in Quinta do Seixo (sequence 2), a mountain vineyard characterized by steep slope hills.

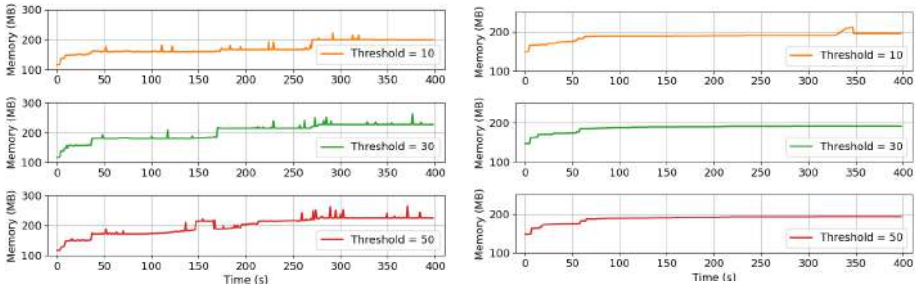
the evaluation of the proposed approach. It is worth noting that all the experiments were performed using a standard computer with a Intel(R) Core(TM) i5-1030H @ 2.5 GHz CPU with 16 GB of RAM memory.

To benchmark our solution with the state-of-the-art, we test two LiDAR-based SLAM algorithms: LOAM (Zhang and Singh, 2014) and LeGO-LOAM (Shan and Englot, 2018). We consider these two approaches key solutions regarding 3D SLAM in outdoor environments. Both have been tested in publicly available datasets, showing high accuracy on localization and mapping. We compare their performance against our own in terms of memory management, execution time and localization and mapping precision.

4.1 | Memory Management Evaluation

The biggest motivation of the topological proposal in this work is to improve the memory management procedure of our SLAM algorithm. This would enable agricultural robotic platforms to navigate autonomously over larger periods and more significant terrain extensions. This is a key requirement in the agricultural sector since the tasks developed in the area are often performed in vast fields. In this section, we analyse the proposed approach in terms of memory management, and in the next one, the execution time requirement is evaluated. To evaluate the capacity of the topological approach to manage the memory resources in an efficient manner, twelve different experiences were carried out regarding VineSLAM. Six of them consider the SLAM algorithm using the topological map for three different distance thresholds t_d (that is used to compute the active nodes of the graph). The remaining six experiments consider the SLAM approach using a single grid map for each sequence. It is expected that the comparison between the two approaches enhances the advantages of the topological map architecture. In addition, to compare VineSLAM's memory management performance with LOAM's and LeGO-LOAM's, the memory consumption of each algorithm was captured for both sequences.

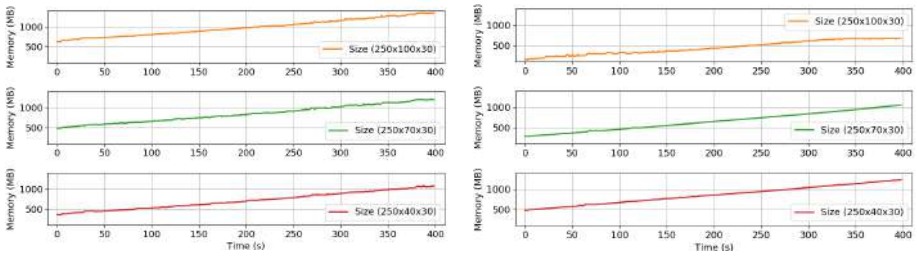
Figure 11 represents the memory consumption over time of the entire SLAM pipeline using the topological architecture. This procedure generated six different subfigures, as represented, considering the experiments for the two sequences and the three distance threshold values. Similarly, Fig. 12 represents the memory consumption of the



(a) Sequence 1 (b) Sequence 2

FIGURE 11 Memory allocation by the SLAM pipeline during the robot operation on (a) sequence 1 and (b) sequence 2 considering the topological architecture proposed. Three different values for the distance threshold to consider active nodes were used.

SLAM algorithm considering a single grid map of fixed size pre-allocated at the beginning of the operation. In this case, six experiments were also carried out considering the same two sequences and three grid map sizes.



(a) Sequence 1 (b) Sequence 2

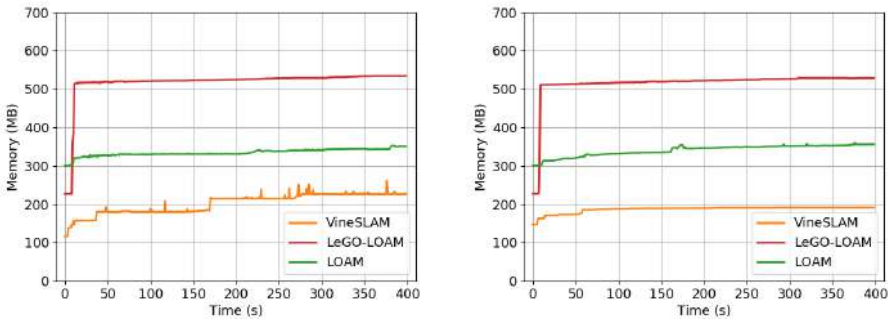
FIGURE 12 Memory allocation by the SLAM pipeline during the robot operation on (a) sequence 1 and (b) sequence 2 considering a pre-allocation of a single grid map. Three grid dimensions were used for each sequence.

As referenced in Section 3 the major aim while developing the topological architecture is that the memory consumption remains approximately constant over time. The threshold value t_d defines the active graph nodes and the deallocated ones in each iteration. In addition, only the grid map cells in use are allocated. From Fig. 11 one can see that this goal was accomplished. This figure shows that for both sequences, the memory consumption remains mainly constant during the entire operations. This procedure has an associated standard deviation due to several reasons. First, the topological map is extracted from satellite images and takes into consideration the characteristics of the environment. Thus, the topological nodes have different areas between them, which means that while the robot travels, there will be zones that require higher or lower memory allocation. Second, the number of features observed in each zone dictates the number of grid map cells allocated. This means that there will be zones where the grid maps stored on the topological nodes will require more or less memory. Despite all of this, Fig. 11 shows that the topological architecture can perform an efficient memory allocation. It is possible to observe that this approach

achieves low memory consumption standard deviations, which proves the validity of the algorithm. Comparing the results obtained for both sequences in this table, one can see that in sequence 2 we obtained a lower variation of the memory allocation over time. This evidences the different topologies of the maps used. On the one hand, for sequence 1, the higher the distance threshold, the higher the standard deviation, and on the other, for sequence 2, the distance threshold does not have an impact on the memory consumption standard deviation. This means that in sequence 1 some features are captured far away from the robot, which leads to the allocation of grid map cells of topological nodes distant from the robot's position.

The SLAM performance using a single grid map is represented in Fig. 12.As visible, this approach suffers from an increasingly need for more memory resources over time. Also, the memory requirements are much higher compared with the topological approach. In addition, as the robot moves, new cells are allocated since new features and observed and stored. This leads to an approximately linear memory increase over time, which reflects in the high standard deviation.This leads to the conclusion that this approach is not suitable for long-term operations since, in the worst case, robots are intended to operate continuously in agricultural environments without scale restrictions. This would lead instantly to an overflow of memory resources of any standard computer, considering the allocation of a fixed-size 3D grid map. Thus, this proves the validity of the proposed topological approach, that has no scale restrictions, since it is able to maintain an approximately constant memory consumption, independent of time.

Finally, Fig. 13 presents the memory management benchmark between VineSLAM, LOAM and LeGO-LOAM. These two state-of-the-art approaches use a KdTree to store the 3D map and perform access and search opera-



(a) Sequence 1 (b) Sequence 2

FIGURE 13 Memory management benchmark between VineSLAM, LeGO-LOAM and LOAM algorithms for (a) sequence 1 and (b) sequence 2.

tions during the localization process. This structure is implemented as a binary tree where data in each node is a K-dimensional point in space. One of the advantages of this method is that it iteratively generates new nodes for new and unvisited places that the robot discovers. Thus, it is more efficient than using a fixed size grid map that assumes a fixed size for the global map before the robot starts exploring the space. However, for large-scale applications, the memory issue is still present since the KdTree will increasingly grow as the robot travels in the environment. Figures 13a and 13b show that the topological architecture innovates and improves the state-of-the-art regarding the memory consumption of SLAM algorithms. For both sequences, our approach requires much less memory than LOAM and LeGO-LOAM. It is also worth noting that both state-of-the-art approaches present a positive derivative

on the memory consumption curves. VineSLAM memory requirements remain approximately constant, especially for sequence 2. It is expected that this difference in memory performance will be higher as longer are the trajectories performed by the robot.

4.2 | Execution Time Evaluation

As referenced before, besides the memory consumption, the execution time is a key aspect of the success of a SLAM algorithm. If one fails, the localization and mapping stages can be compromised. For this reason, this work evaluates the execution time performance of the proposed approach considering the topological architecture. To do so, 30 different experiments were carried out, varying the number of particles used in the localization algorithm. As explained before, the localization uses the topological map to search for nearest neighbor features in the global map. Thus, varying the number of particles N allows evaluating the performance of the topological implementation. Also, it allows understanding how many particles are supported without compromising the robot loop frequency, which can change between applications and robots. In our case, the robot requires a frequency loop equal to or higher than 10Hz. To benchmark our solution with the state-of-the-art, an analysis of the execution time performance for both sequences is carried out for VineSLAM, LOAM and LeGO-LOAM.

Figure 14 shows the loop time spent by the entire SLAM pipeline considering 30 experiments with different number of particles. As visible, the execution time increases almost linearly with the variation of the number of par-

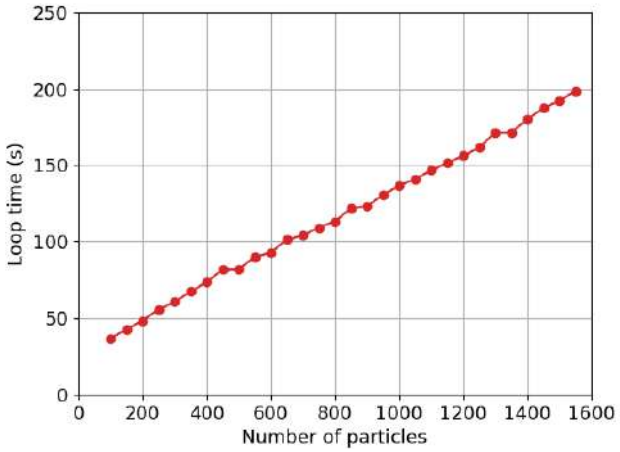


FIGURE 14 Relation between the number of particles used in the localization process and the average execution time of the entire SLAM loop.

ticles used in the localization procedure. The greater the number of particles N, the higher the number of operations executed to compute the robot localization. Since the particle filter discretizes the 6-DoF space in N hypotheses (particles), when N increases, the robustness of the filter also increases. This is true until a certain value of N where the increase in the number of hypotheses does not bring new or relevant information to the filter. Thus, the most

important aspect is to be able to choose N that optimizes the filter performance without compromising the robotic loop frequency. From Fig. 14 one can see that the SLAM algorithm can perform with a frequency higher than 10Hz for $N \leq 600$. This shows that the algorithm implementation is efficient, being able to process 600 hypotheses for the localization plus all the mapping pipeline in less than 100 milliseconds. Given all of the above, it is crucial that the SLAM algorithm is precise for values of N lower than 600.

Figure 15 shows the execution time performance of VineSLAM using 300 and 500 particles, LOAM, and LeGO-LOAM over time. The two state-of-the-art approaches use a localization algorithm based on two parallel processes. A

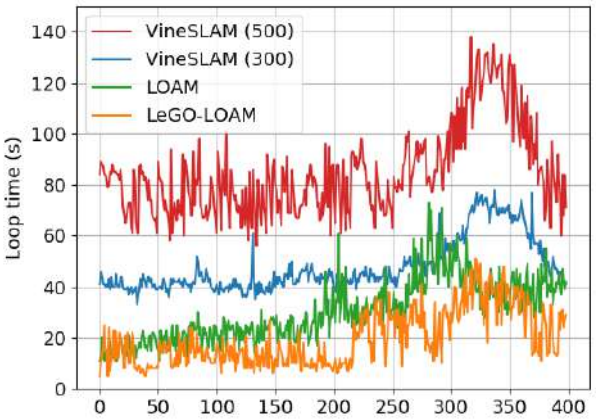


FIGURE 15 Execution time benchmark between VineSLAM using 300 and 500 particles, LOAM, and LeGO-LOAM algorithms.

high-frequency LiDAR odometry algorithm that calculates the frame-to-frame robot displacement, and a low-frequency graph-SLAM approach that optimizes the robot pose and the 3D map. Also, their implementation is divided in four different running threads related to LiDAR odometry, LiDAR mapping, the calculation of the final robot transformation, and feature extraction. For this comparison, we only measure the time required by the most expensive process, the LiDAR mapping algorithm that is based on a graph-SLAM approach. As visible, LeGO-LOAM is the fastest approach, followed by LOAM and VineSLAM. This result was expected since VineSLAM is based on a much more computationally expensive localization algorithm, a PF. Nevertheless, VineSLAM can perform with a frequency higher than 10Hz for almost all the time using 500 particles and with a frequency higher than 12.5Hz using 300 particles. This means that this approach is suitable for execution on mobile robots, which usually require a frequency loop of 10Hz.

4.3 | Localization and Mapping Evaluation

In this section, we show the localization and mapping results of our approach (VineSLAM) and LOAM's and LeGO-LOAM's for both sequences. For sequence 1, since it is a smaller trajectory without high variations on altitude or rotation, we just show the 3D maps generated by the three algorithms. Sequence 2 is extremely challenging for

SLAM algorithms. For this reason, we benchmark the three algorithms against a GNSS-RTK localization along the entire trajectory. The localization performance is quantified in the three dimensions component-wise, and considering the mean absolute error for the entire trajectory. This error is computed as follows:

$$\omega = \frac{1}{N}\sum_{i}^{N} e\Big(X_i^G - X_i\Big), \tag{8}$$

where e represents the euclidean distance operator, N represents the number of localization samples estimated by the algorithms along the trajectory, X_i^G represents the ground truth 3D position at the instant i and X_i represents the 3D position estimated by the localization system at the instant i. Finally, to have a detailed validation of VineSLAM performance, the maps generated by the algorithm are overlapped with satellite images of both vineyards. These images are used as ground truth for the generated maps. This method allows the proper validation of mapping and localization since both processes are interdependent. If we verify that mapping is successful, we can infer that we have a precise localization, i.e., if one of the processes fails, the other will automatically fail.

Figure 16 presents the maps built by the three SLAM algorithms corresponding to sequence 1. Even without a quantitative evaluation of the performance of each approach, in this case, it is visible that the three algorithms produce similar 3D maps that characterize the environment realistically.

For sequence 2, a more detailed evaluation was performed. Figure 17 shows the maps generated by VineSLAM and LOAM. Since LeGO-LOAM failed for this sequence, it could not provide a reliable 3D map of the environment. As represented in Fig. 18, both VineSLAM and LOAM are able to estimate the robot pose for the entire sequence, while LeGO-LOAM fails. It is worth noting that this is a challenging trajectory, with 337.1 meters of extension, where the robot goes down and up more than 7 meters through steep slopes. Table 3 shows the error of the three algorithms in comparison with the GNSS-RTK ground truth. As visible, LeGO-LOAM completely fails for this sequence. VineSLAM and LOAM present low error, with the state-of-the-art approach achieving the lowest absolute error, 0.638 meters. These results show that VineSLAM can achieve localization errors almost similar with one of the most robust state-

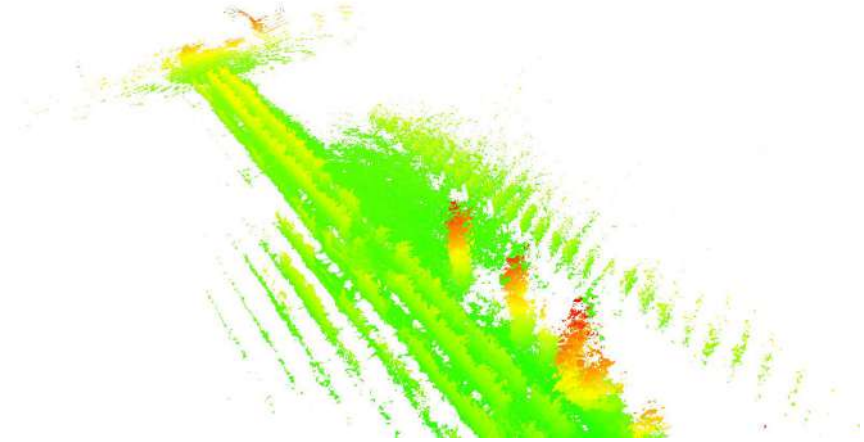
TABLE 3 Localization performance of the three algorithms against the GNSS-RTK ground truth.

Algorithm	X error (m)	Y error (m)	Z error (m)	Absolute error (m)
VineSLAM	0.400	0.710	1.496	0.902 ± 0.619
LOAM	0.202	0.586	0.929	0.638 ± 0.429
LeGO-LOAM*	1.402	17.756	3.953	16.844 ± 23.189

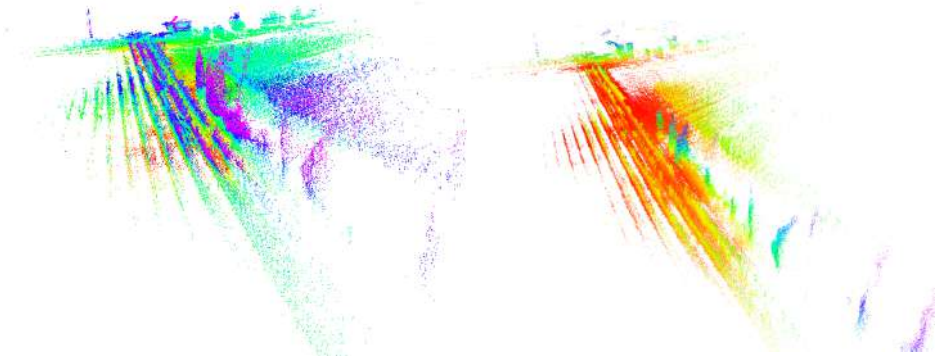
*LeGO-LOAM lost track of the robot pose in this sequence.

of-the-art SLAM algorithms, without compromising the available memory resources in a long-term and large-scale perspective. All the algorithms present higher error in the altitude component. The harsh terrain inclinations are challenging for SLAM algorithms that, without an additional source of information to avoid this, present error in this component.

Finally, to have a more clear perception of the performance of VineSLAM's localization and mapping, the 3D maps generated were overlapped with satellite images for both sequences. As visible in Fig. 19, both for sequence 1 and 2, there is a clear alignment between the 3D maps and the ground truth satellite images. This can be mainly seen by the external agents to the vineyard, such as trees in sequence 1, and the house in sequence 2 that clearly matches the satellite image. This proves the validity of VineSLAM to localize the robot and map the environment in harsh



(a) VineSLAM



(b) LOAM

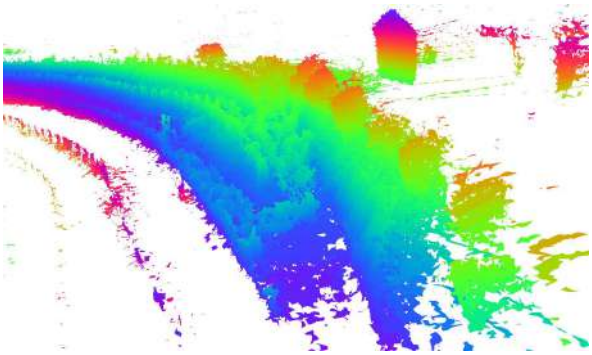
(c) LeGO-LOAM

FIGURE 16 3D maps generated by the three SLAM algorithms for sequence 1.

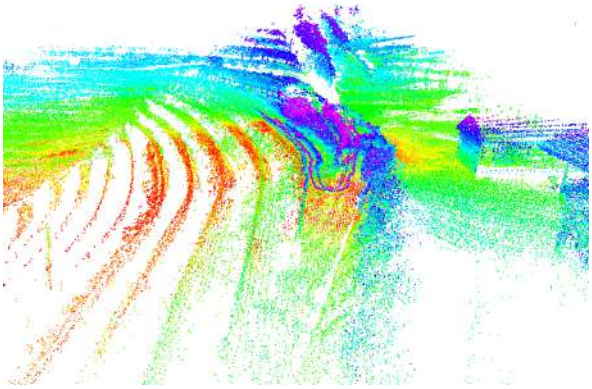
conditions such as the ones presented in this paper.

5 | DISCUSSION

As stated by Bresson et al. (2017) SLAM algorithms should be able to work continuously with constant memory usage. This scalability challenge is still an open topic in the literature. As shown in this work, even robust SLAM algorithms for 3D outdoors navigation (LOAM and LeGO-LOAM) can still not solve this issue. Besides the problem of localization and mapping being sufficiently tackled in the state-of-the-art, most approaches can not be used in real applications without time and scale restrictions. This is crucial to solve the scalability issues of SLAM without compromising



(a) VineSLAM

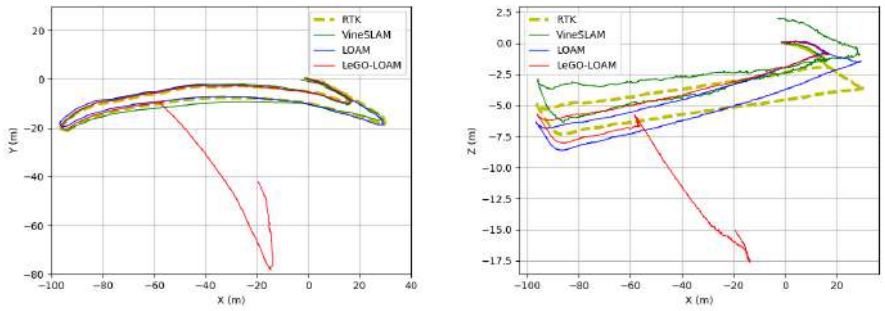


(b) LOAM

FIGURE 17 Inside view of the 3D maps generated by the VineSLAM and LOAM algorithms for sequence 2.

localization and mapping performance because it will open the possibility for the application of such algorithms in real applications. For all these reasons, we consider the proposed approach of high relevance for the evolution of autonomous mobile robots.

As shown in section 4, our approach can maintain the memory requirements constant over time. On the contrary, state-of-the-art approaches, besides using efficient data storage structures (tree-based), increasingly require more memory resources over time. This is not suitable for real-world large-scale applications. Even so, these approaches are well-known in the literature for being extremely accurate in terms of localization and mapping. For example, LOAM occupies the third place in the Kitti dataset Geiger et al. (2012) regarding SLAM evaluation, and LeGO-LOAM is an improved version of LOAM. Thus, an approach that presents similar localization and mapping results to LOAM and LeGO-LOAM, solving the SLAM scalability challenges, consists of a significant innovation of the state-of-the-art. For this reason, we also compare the three approaches in terms of execution time and localization and mapping performance. As shown in the results section, LOAM and LeGO-LOAM are faster than VineSLAM. These algorithms



(a) Top view - xOy plane.

(b) Side view - xOz plane.

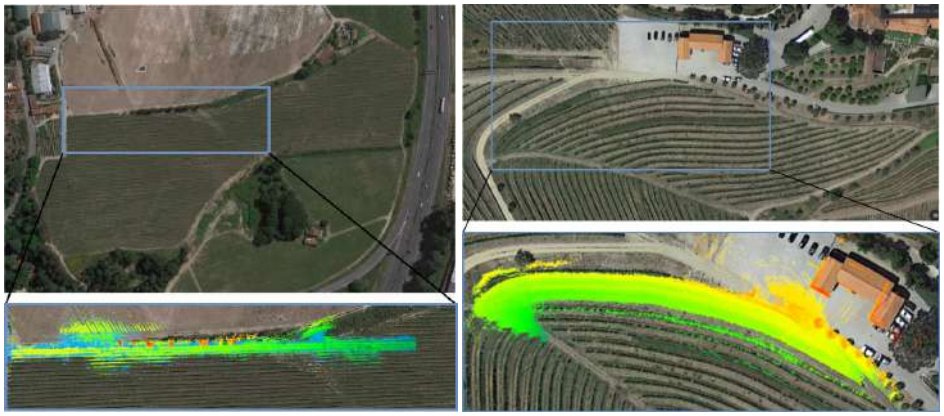
FIGURE 18 Evaluation of the trajectory estimated by the three algorithms against the GNSS-RTK ground truth. The figure shows (a) a top view perspective where a 2D trajectory can be observed and (b) a side view perspective where the altitude estimation can be evaluated.

are based on a Graph-SLAM efficient approach. This was the expected outcome since our approach is based on a PF that is natively more computationally expensive. We chose the PF for two main reasons: first, as shown before, this approach is under-explored in the literature, especially in outdoor 3D SLAM. Second, the morphology of the PF is extremely parallelizable. Our main goal in the future is to deploy VineSLAM in dedicated hardware with Graphical Processing Unit (GPU) optimizations. Without these optimizations, our main goal in this work was to develop a solution that, besides being slower than optimization-based state-of-the-art algorithms, does not compromise the robot loop frequency. As shown, our approach presents a frequency higher than 10Hz, which is suitable for deployment on our robotic platforms.

In terms of localization, we tested the three approaches in harsh conditions, especially sequence 2 since it presents steep slopes, challenging inclinations, and navigation over long vineyard corridors. These conditions are reflected in LeGO-LOAM's performance, which fails (lost track of the robot motion). As analyzed before, to evaluate the localization performance, the trajectory estimated by each algorithm was benchmarked against an GNSS-RTK localization. VineSLAM and LOAM can track the robot motion with low error during the entire trajectory, being able to deal with the challenging conditions present in this agricultural scenario. Besides LOAM presenting the lowest absolute error in relation with the RTK sensor, VineSLAM can also maintain low errors with the advantage being scalable for larger environments since its memory requirements are independent of the operation time. Results also show that the future of localization algorithms should focus on dealing with high altitude variations. Both LOAM and VineSLAM present higher errors in this component, since they accumulate drift when the robot is going up or down along the steep slope hills.

6 | CONCLUSIONS

Agriculture robotics brings the challenge of scalability. The implementation of robotics in these environments requires that robots are autonomous along the vast terrain extensions considering the limited resources of the processors used. One of the major limitations of the state-of-the art of 3D localization and mapping algorithms are the memory



(a) Sequence 1 (b) Sequence 2

FIGURE 19 Overlap between the 3D maps generated by VineSLAM and satellite images of the vineyards corresponding to the two sequences.

management procedures. The majority of the algorithms require more memory to store the maps as it grows.

This work proposes a localization and mapping algorithm suitable for operation in real robotic applications providing a memory management algorithm that is constant over time. This structure enables a memory consumption almost independent of time, and lower than the required by the current state-of-the-art algorithms. Also, results show that our approach presents a localization precision aligned with the state-of-the-art and is more efficient in terms of memory consumption.

The localization algorithm presented is based on a PF, which is a computationally expensive approach. As described in section 4, VineSLAM has a frequency loop lower than LOAM and LeGO-LOAM. For this reason, in future work, we will accelerate our SLAM algorithm in a GPU with parallelization techniques. Additionally we will deploy the algorithm on robots that are intended to perform agricultural tasks such as spraying and pruning.

acknowledgements

André Silva Pinto de Aguiar thanks the FCT—Foundation for Science and Technology, Portugal, for the Ph.D. Grant DFA/BD/5318/2020.

references

Abdallah, S. M., Asmar, D. C. and Zelek, J. S. (2007) A benchmark for outdoor vision SLAM systems. *Journal of Field Robotics*, **24**, 145–165. URL: https://doi.org/10.1002/rob.20180.

Aguiar, A. S., dos Santos, F. N., Sobreira, H., Cunha, J. B. and Sousa, A. J. (2021a) Particle filter refinement based on clustering procedures for high-dimensional localization and mapping systems. *Robotics and Autonomous Systems*, **137**, 103725. URL: https://www.sciencedirect.com/science/article/pii/S0921889021000105.

Aguiar, A. S., Magalhães, S. A., dos Santos, F. N., Castro, L., Pinho, T., Valente, J., Martins, R. and Boaventura-Cunha, J. (2021b)

Grape bunch detection at different growth stages using deep learning quantized models. *Agronomy*, **11**. URL: https://www.mdpi.com/2073-4395/11/9/1890.

Aguiar, A. S., Monteiro, N. N., Santos, F. N. d., Solteiro Pires, E. J., Silva, D., Sousa, A. J. and Boaventura-Cunha, J. (2021c) Bringing semantics to the vineyard: An approach on deep learning-based vine trunk detection. *Agriculture*, **11**. URL: https://www.mdpi.com/2077-0472/11/2/131.

Aguiar, A. S., dos Santos, F. N., Cunha, J. B., Sobreira, H. and Sousa, A. J. (2020a) Localization and mapping for robots in agriculture and forestry: A survey. *Robotics*, **9**. URL: https://www.mdpi.com/2218-6581/9/4/97.

Aguiar, A. S., Santos, F. N. D., De Sousa, A. J. M., Oliveira, P. M. and Santos, L. C. (2020b) Visual trunk detection using transfer learning and a deep learning-based coprocessor. *IEEE Access*, **8**, 77308–77320.

Bailey, T. and Durrant-Whyte, H. (2006) Simultaneous localization and mapping (slam): part ii. *IEEE Robotics Automation Magazine*, **13**, 108–117.

Baroffio, L., Canclini, A., Cesana, M., Redondi, A., Tagliasacchi, M. and Tubaro, S. (2015) Coding local and global binary visual features extracted from video sequences. *IEEE Transactions on Image Processing*, **24**, 3546–3560.

Bergerman, M., Billingsley, J., Reid, J. and van Henten, E. (2016) *Robotics in Agriculture and Forestry*, 1463–1492. Cham: Springer International Publishing. URL: https://doi.org/10.1007/978-3-319-32552-1_56.

Bernuy, F. and Ruiz Del Solar, J. (2015) Semantic mapping of large-scale outdoor scenes for autonomous off-road driving. In *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, 124–130.

Bosse, M., Newman, P., Leonard, J., Soika, M., Feiten, W. and Teller, S. (2003) An atlas framework for scalable mapping. In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 2, 1899–1906 vol.2.

Bradley, D., Patel, R., Vandapel, N. and Thayer, S. (2005) Real-time image-based topological localization in large outdoor environments. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3670–3677.

Bresson, G., Alsayed, Z., Yu, L. and Glaser, S. (2017) Simultaneous localization and mapping: A survey of current trends in autonomous driving. *IEEE Transactions on Intelligent Vehicles*, **2**, 194–220.

Capua, F. R., Sansoni, S. and Moreyra, M. L. (2018) Comparative analysis of visual-slam algorithms applied to fruit environments. In *2018 Argentine Conference on Automatic Control (AADECA)*, 1–6.

Chen, M., Tang, Y., Zou, X., Huang, Z., Zhou, H. and Chen, S. (2021) 3d global mapping of large-scale unstructured orchard integrating eye-in-hand stereo vision and slam. *Computers and Electronics in Agriculture*, **187**, 106237. URL: https://www.sciencedirect.com/science/article/pii/S0168169921002544.

Cole, D. and Newman, P. (2006) Using laser range data for 3d slam in outdoor environments. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, 1556–1563.

Contreras, L. and Mayol-Cuevas, W. (2017) O-poco: Online point cloud compression mapping for visual odometry and slam. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 4509–4514.

Dellaert, F., Carlson, J., Ila, V., Ni, K. and Thorpe, C. E. (2010) Subgraph-preconditioned conjugate gradients for large scale slam. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2566–2571.

Douc, R. and Cappe, O. (2005) Comparison of resampling schemes for particle filtering. In *ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005.*, 64–69.

Durrant-Whyte, H. and Bailey, T. (2006) Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, **13**, 99–110.

Ehlers, S. F. G., Stuede, M., Nuelle, K. and Ortmaier, T. (2020) Map management approach for slam in large-scale indoor and outdoor areas. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 9652–9658.

Emmi, L., Flécher, E. L., Cadenat, V. and Devy, M. (2021) A hybrid representation of the environment to improve autonomous navigation of mobile robots in agriculture. *Precision Agriculture*, **22**, 524–549. URL: https://doi.org/10.1007/s11119-020-09773-9.

Emmi, L., de Soto, M. G., Pajares, G. and González-De-Santos, P. (2014) New trends in robotics for agriculture: Integration and assessment of a real fleet of robots. *The Scientific World Journal*, **2014**.

Fairfield, N., Wettergreen, D. and Kantor, G. (2010) Segmented SLAM in three-dimensional environments. *Journal of Field Robotics*, **27**, 85–103. URL: https://doi.org/10.1002/rob.20320.

Folkesson, J. and Christensen, H. I. (2007) Graphical slam for outdoor applications. *Journal of Field Robotics*, **24**, 51–70.

Garcia-Fidalgo, E. and Ortiz, A. (2015) Vision-based topological mapping and localization methods: A survey. *Robotics and Autonomous Systems*, **64**, 1–20. URL: https://www.sciencedirect.com/science/article/pii/S0921889014002619.

Geiger, A., Lenz, P. and Urtasun, R. (2012) Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Greenspan, M. and Yurick, M. (2003) Approximate k-d tree search for efficient icp. In *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings.*, 442–448.

Grisetti, G., Kümmerle, R., Stachniss, C. and Burgard, W. (2010) A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, **2**, 31–43.

He, X., Zemel, R. S. and Mnih, V. (2006) Topological map learning from outdoor image sequences. *J. Field Robotics*, **23**, 1091–1104.

Heiwolt, K., , Mandil, W., Cielniak, G., Hanheide, M., and and (2020) Automated topological mapping for agricultural robots. EPSRC UK-RAS Network. URL: https://doi.org/10.31256/ze8ex1v.

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C. and Burgard, W. (2013) OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*. URL: https://octomap.github.io. Software available at https://octomap.github.io.

Khan, M. W., Das, G. P., Hanheide, M. and Cielniak, G. (2020) Incorporating spatial constraints into a bayesian tracking framework for improved localisation in agricultural environments. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2440–2445.

Labbé, M. and Michaud, F. (2017) Long-term online multi-session graph-based SPLAM with memory management. *Autonomous Robots*, **42**, 1133–1150. URL: https://doi.org/10.1007/s10514-017-9682-5.

Lau, B., Sprunk, C. and Burgard, W. (2010) Improved updating of euclidean distance maps and voronoi diagrams. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 281–286.

Le, T., Gjevestad, J. G. O. and From, P. J. (2019) Online 3d mapping and localization system for agricultural robots. *IFAC-PapersOnLine*, **52**, 167–172. URL: https://doi.org/10.1016/j.ifacol.2019.12.516.

Li, F., Yang, S., Yi, X. and Yang, X. (2018) Towards visual slam with memory management for large-scale environments. In *Advances in Multimedia Information Processing – PCM 2017* (eds. B. Zeng, Q. Huang, A. El Saddik, H. Li, S. Jiang and X. Fan), 776–786. Cham: Springer International Publishing.

Lowe, T., Moghadam, P., Edwards, E. and Williams, J. (2021) Canopy density estimation in perennial horticulture crops using 3d spinning lidar SLAM. *Journal of Field Robotics*, **38**, 598–618. URL: https://doi.org/10.1002/rob.22006.

Nüchter, A., Lingemann, K., Hertzberg, J. and Surmann, H. (2007) 6d SLAM-3d mapping outdoor environments. *Journal of Field Robotics*, **24**, 699–722. URL: https://doi.org/10.1002/rob.20209.

Pedrosa, E., Pereira, A. and Lau, N. (2018) A sparse-dense approach for efficient grid mapping. In *2018 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 136–141.

Pinto de Aguiar, A. S., Neves dos Santos, F. B., Feliz dos Santos, L. C., de Jesus Filipe, V. M. and Miranda de Sousa, A. J. (2020) Vineyard trunk detection using deep learning – an experimental device benchmark. *Computers and Electronics in Agriculture*, **175**, 105535. URL: http://www.sciencedirect.com/science/article/pii/S0168169920304555.

Roldán, J. J., del Cerro, J., Garzón-Ramos, D., Garcia-Aunon, P., Garzón, M., de León, J. and Barrientos, A. (2018) *Robots in Agriculture: State of Art and Practical Experiences*. URL: https://app.dimensions.ai/details/publication/pub.1100266943andhttps://www.intechopen.com/citation-pdf-url/56199.

Santos, L. C., Aguiar, A. S., Santos, F. N., Valente, A. and Petry, M. (2020) Occupancy grid and topological maps extraction from satellite images for path planning in agricultural robots. *Robotics*, **9**. URL: https://www.mdpi.com/2218-6581/9/4/77.

Schleicher, D., Bergasa, L. M., Ocana, M., Barea, R. and Lopez, M. E. (2009) Real-time hierarchical outdoor slam based on stereovision and gps fusion. *IEEE Transactions on Intelligent Transportation Systems*, **10**, 440–452.

Schuster, M. J., Brand, C., Hirschmüller, H., Suppa, M. and Beetz, M. (2015) Multi-robot 6d graph slam connecting decoupled local reference filters. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 5093–5100.

Shalal, N., Low, T., McCarthy, C. and Hancock, N. (2013) A review of autonomous navigation systems in agricultural environments. *Society for Engineering in Agriculture Conference: Innovative Agricultural Technologies for a Sustainable Future*.

Shan, T. and Englot, B. (2018) Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4758–4765.

Suger, B., Diego Tipaldi, G., Spinello, L. and Burgard, W. (2014) An approach to solving large-scale slam problems with a small memory footprint. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 3632–3637.

Van Opdenbosch, D., Aykut, T., Alt, N. and Steinbach, E. (2018) Efficient map compression for collaborative visual slam. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 992–1000.

Vougioukas, S. G. (2019) Agricultural robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, **2**, 365–392. URL: https://doi.org/10.1146/annurev-control-053018-023617.

Zhang, J. and Singh, S. (2014) Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*.

5.5 Final remarks

VineSLAM formulates the localization and mapping problem in a way that allows agricultural robots to: localize themselves in 3D; reconstruct the environment considering point- and polygon-based features; create semantic maps and visualize them on top of the 3D reconstruction; and operate without time or scale limitations through the use of a topological map. This approach opens the door to the implementation of autonomous precision agriculture tasks and represents a step forward regarding autonomous navigation in agriculture. It is worth noting that the algorithm was not only tested in agricultural environments such as vineyards and orchards, but also in indoor scenarios. On the other hand, forest environments were not considered, and should be approached in the future. In addition, the algorithm does not provide a loop closure approach which should be researched and implemented in the future. This will allow the correction of accumulated drift when robots navigate in large extensions.

6 Conclusions and future work

The overall thesis achievements, its limitations and its conclusions are presented in this chapter. Then, the guidelines for a continuation of the work proposed in this thesis are presented.

6.1 Research Achievements

Chapter 1 defined the main statement of this thesis: **it is possible to localize a robot and simultaneously map the environment in long-term and large-scale considering metric, semantic, and topological information by the fusion of different types of sensors.** To fulfil this statement, this work partitioned it in five main goals:

G1 - An approach for automatic extrinsic calibration method to calibrate the main sensors used in the proposed localization and mapping algorithms - cameras and 3D LiDARs
The multi-layer mapping architecture proposed in this thesis imposes the use of two main sensors: cameras and 3D LiDARs. Since it is intended that VineSLAM works in any robotic platform with any physical sensor configuration, it is important that the extrinsic calibration between the cameras and LiDARs is made automatically.

This approach is more precise and robust than finding the physical relationship between sensors manually. In this way, sensors can be positioned in any way, and the extrinsic calibration procedure is responsible for estimating and providing the proper spacial relationship between them. Thus, this thesis contributed to a general calibration toolbox called "ATOM - A Calibration Framework using the Atomic Transformations Optimization Method" by implementing an algorithm for the simultaneous calibration of multiple cameras and LiDARs. Results show that the algorithm is able to calibrate simultaneously two cameras and one 3D LiDAR, also estimating the intrinsic parameters of the cameras. The published paper proved that ATOM has a similar or better performance than conventional pairwise calibration methods even when calibrating the entire sensing system simultaneously. For camera-to-camera calibration it achieved an RMS error of 0.97 pixels, and for camera-to-LiDAR 3.81 pixels. This work made possible the use of any camera-to-LiDAR configuration, enabling sensor fusion on VineSLAM. Thus, **G1** was achieved with success.

G2 - An approach for a perception system that can recognize and detect semantic elements in agricultural environments

One of the key roles of VineSLAM is to have a semantic perception of the agricultural environments. The creation of maps of the crops with the position of natural agents and their corresponding labels is extremely important for precision agriculture tasks such as spraying, harvesting, or pruning. This thesis explores the use of lightweight DL models to detect vine trunks and grape bunches aiming to create semantic maps of the vineyard. Four papers were published in this scope. Three of them regarding vine trunk detection and one regarding grape bunche detection. These papers evidence an evolution of the approach, with the increase of the dataset size over time, the exploitation of new models and hyperparameters, and the benchmark between different devices to execute the perception system. This research resulted in a perception system that is executed at the Edge, i.e., in a dedicated hardware device, and that is capable to detect vine trunks and grape bunches in different growth stages. For this reason, the semantic system conceptualized in **G2** was achieved with success. Results demonstrated that vine trunks are detected with an average precision of 84.16% and grape bunches with 44.93%. The lower detection precision of grape bunches is a limitation of the semantic perception system, that should be overcomed in the future.

G3 - An approach for a 6-DoF localization approach based on 3D metric information considering the geometry of the environment

The multi-layer mapping architecture must be supported by a robust localization system. In addition, one of VineSLAM's main goals is to enable agricultural robots to be autonomous. They require a precise localization to be able to perform precision agriculture tasks. Agricultural environments present many challenges for localization systems such as terrain irregularities, unstructured scenes, among others. Also, crops located along mountain hills present additional challenges such as the recurrent unavailability of GNSS due to signal blockage or multi reflection. VineSLAM's localization system should be robust to all these conditions to be generalized for any kind of agricultural environment and enable autonomous processes. Chapter 5 shows that the proposed approach can localize robots using metric features such as points and polygons. The addition of semiplanes to the conventional point features consists of a major innovation since it enables to continuously estimate the ground plane and vegetation walls considering that these planes are finite due to the lack of structure in agriculture. To reduce the computational cost associated with the PF, a refinement step was also proposed using point clouds generated from stereo cameras that allows the increase of performance without needing to increase the number of particles. Results in the four published/submitted articles show that VineSLAM can localize robots with precision even under these challenging conditions without using a source of GNSS information. VineSLAM was formulated to work in agricultural contexts, but is general enough to operate in other contexts, namely indoor environments. This approach was tested in multiple robots and contexts, being proved that it works with at least three LiDAR models, and that can localize robots with precision in vineyards, orchards and indoor environments. Due to all of this, **G3** was completed successfully. It is worth noting that VineSLAM does not provide a loop closure procedure, which can be a limitation, especially when working without an absolute positioning sensor (like GNSS) since the localization error is cumulative which can lead to drift on the pose estimation.

G4 - An approach for semantic mapping to create maps of agricultural environments with meaningful information in different growth stages

The thesis statement emphasis the importance of developing a semantic mapping algorithm. After having a robust localization approach and a perception system, all the required inputs to create semantic agricultural maps are available. The

designed mapping approach uses a monocular camera for the semantic detections and fuses them with the metric map that is simultaneously being build by VineSLAM to calculate the depth of grape bunches and vine trunks. Since the dataset built considers the different vineyard states, the semantic mapping approach can map it considering different fruit growth stages. This semantic feature mapping is then presented on top of the metric map to improve the visualization quality and further analysis of the crop state. A part of the multi-layer mapping architecture is here evidenced. With this, the goal proposed in **G4** was achieved with success. It should be noted that the semantic mapping process is agnostic to the type of semantic elements detected, and thus it can be used to create semantic maps of other environments besides agriculture. The evaluation of the semantic mapping precision was performed by comparing multiple maps of the same crop in different stages. This technique is not the most suitable for quantifying the effective performance of the mapping, and should be improved in the future by collecting the real position of the semantic elements and comparing them with the generated map.

G5 - An approach for a multi-layer mapping algorithm that is suitable for long-term operations and large-scale environments without time and scale limitations

One of the main goals of this thesis is to implement VineSLAM on robots that will develop precision agriculture operations in large extensions. A state-of-the-art problem related with SLAM systems is their large-scale capability. Many algorithms degenerate over time since they consume all the memory resources available on the processors that execute them. VineSLAM solves this by adding a new layer to the mapping architecture, a topological map. This map, extracted from high-resolution satellite images, partitions agricultural environments in a graph structure. This graph enables the SLAM algorithm to efficiently load parts of the map located in each node instead of considering the entire map extension in each time instant. The non-allocated map nodes are stored into the system and can be loaded, deallocated, or updated at any time. Results show that this method can maintain a memory consumption constant over time, which enable VineSLAM to be executed without time or scale restrictions. This fulfils with success the **G5**.

Overall, all the goals were achieved with success and, consequently, the thesis statement was proven. The developed solution achieved a high level of maturity, with an innovative multi-layer mapping architecture and a robust localization system.

Most importantly, this thesis is a step forward to the implementation of autonomous robots in agriculture, solving many issues that were listed in the state-of-the-art. This work proposed several novelties and resulted in many scientific publications and publicly available datasets that can be used and improved. Moreover, the implementation of this thesis is entirely open-source[1][2][3]. The contributions made to the calibration toolbox ATOM and the VineSLAM framework can be downloaded, tested, and improved. Finally, one of the greatest achievements of this thesis was the deployment of VineSLAM in three different agricultural robots, making them autonomous in two different agricultural contexts. The robots AgRob, Modular-E, and Weta were developed in R&D European projects aiming to perform tasks such as spraying, mowing, pruning and harvesting.

6.2 Future Directions

The work developed in this thesis has several limitations and opened several research topics that were not approached. Besides the future work proposals presented in the articles, others can be considered.

Regarding the extrinsic calibration procedure, this approach is a simultaneous calibration between multiple sensors. However, the algorithm does not consider the calibration in relation with the robot reference frame. This means that after calibrating the robot's sensor topological tree, at least one sensor-to-robot transformation has to be manually measured. In the future, the calibration procedure could consider the robot reference frame to eliminate all the required manual measures.

Regarding the semantic perception system, the future work could consider using more complex DL models to achieve higher precision and recall levels. The idea of this system is to be executed on the Edge using embedded hardware devices, which limits the possible size and complexity of the models. In the future, more powerful embedded devices could be considered to achieve a more robust semantic perception system without loosing its cost-effective and time-effective characteristics.

In relation with the localization system, one of the future goals is to unify

[1] https://gitlab.inesctec.pt/agrob/vineslam_stack/vineslam
[2] https://zenodo.org/record/5362354
[3] https://github.com/lardemua/atom

the developed software solution in an embedded hardware device with GPU optimizations. This will allow VineSLAM to be a plug-n-play solution that can be easily installed in any robotic platform by simply connecting a network and a power cable. Since the embedded device will have limited CPU resources, and due to the high parallelization level of the PF algorithm, the GPU can be used to accelerate the localization process. Additionally, a loop closure algorithm should be researched and integrated to account for the correction of drift accumulating during the navigation.

Regarding the semantic mapping algorithm, its main limitation is the validation process used. In this, the maps built during different growth stages were compared and the error was calculated. Although this metric is valid, it does not provide a ground truth for the localization of the semantic objects. In the future, a methodology can be developed to geo-reference every semantic element present in the crop, so that the mapping error can be directly calculated. Additionally, the semantic mapping should be tested with different semantic elements to generate maps of different crops, or even non-agricultural environments.

Finally, in what concerns the topological mapping procedure, one key aspect could be developed in the future. The topological structure was used in this thesis only for mapping purposes. One interesting research topic would be to include this structure in the localization loop. In particular, this structure can impose constraints to the robot's localization and improve its performance. For example, in woody crops with highly symmetric corridors, localization systems can fail to determine in which corridor the robot is. The topological map can constrain the robot pose to a single corridor by not loading adjacent corridor sub-maps. Also, since the nodes are interconnected, it is expected that the robot transitions only for adjacent nodes.

References

Aguiar, A., Oliveira, M., Pedrosa, E., and Santos, F. (2021a). A camera to lidar calibration approach through the optimization of atomic transformations. *Expert Systems with Applications*, page 114894. IF 5.423 (2020). 42

Aguiar, A. S., dos Santos, F. N., Cunha, J. B., Sobreira, H., and Sousa, A. J. (2020a). Localization and mapping for robots in agriculture and forestry: A survey. *Robotics*, 9(4). 13

Aguiar, A. S., dos Santos, F. N., Sobreira, H., Boaventura-Cunha, J., and Sousa, A. J. (2022a). Localization and mapping on agriculture based on point-feature extraction and semiplanes segmentation from 3d LiDAR data. *Frontiers in Robotics and AI*, 9. 164

Aguiar, A. S., dos Santos, F. N., Sobreira, H., Cunha, J. B., and Sousa, A. J. (2021b). Particle filter refinement based on clustering procedures for high-dimensional localization and mapping systems. *Robotics and Autonomous Systems*, 137:103725. 208

Aguiar, A. S., Magalhães, S. A., dos Santos, F. N., Castro, L., Pinho, T., Valente, J., Martins, R., and Boaventura-Cunha, J. (2021c). Grape bunch detection at different growth stages using deep learning quantized models. *Agronomy*, 11(9):1890. 134

Aguiar, A. S., Monteiro, N. N., dos Santos, F. N., Pires, E. J. S., Silva, D., Sousa, A. J., and Boaventura-Cunha, J. (2021d). Bringing semantics to the vineyard: An approach on deep learning-based vine trunk detection. *Agriculture*, 11(2):131. 112

Aguiar, A. S., Santos, F. N. d., Santos, L. C., Sousa, A. J., and Boaventura-Cunha, J. (2022b). Topological map-based approach for localization and mapping memory optimization. *Journal of Field Robotics*. 232

Aguiar, A. S., Santos, F. N. D., Sousa, A. J. M. D., Oliveira, P. M., and Santos, L. C. (2020b). Visual trunk detection using transfer learning and a deep learning-based coprocessor. *IEEE Access*, 8:77308–77320. 78

Andresen, T., de Aguiar, F. B., and Curado, M. J. (2004). The alto douro wine region greenway. *Landscape and Urban Planning*, 68(2):289 – 303. International Greenway Planning. 3

Bailey, T. and Durrant-Whyte, H. (2006). Simultaneous localization and mapping (slam): part ii. *IEEE Robotics Automation Magazine*, 13(3):108–117. 5

Billingsley, J., Visala, A., and Dunn, M. (2008). *Robotics in Agriculture and Forestry*, pages 1065–1077. Springer Berlin Heidelberg, Berlin, Heidelberg. 3

de Aguiar, A. S. P., dos Santos, F. B. N., dos Santos, L. C. F., de Jesus Filipe, V. M., and de Sousa, A. J. M. (2020). Vineyard trunk detection using deep learning – an experimental device benchmark. *Computers and Electronics in Agriculture*, 175:105535. 92

de Oliveira, M. A. R., Pedrosa, E. F., de Aguiar, A. S. P., Rato, D. F. P. D., dos Santos, F. B. N., de Jesus Dias, P. M., and dos Santos, V. M. F. (2022). Atom: A general calibration framework for multi-modal, multi-sensor systems. *Expert Systems with Applications*, page 118000. 42

Duckett, T., Pearson, S., Blackmore, S., Grieve, B., Chen, W.-H., Cielniak, G., Cleaversmith, J., Dai, J., Davis, S., Fox, C., From, P., Georgilas, I., Gill, R., Gould, I., Hanheide, M., Hunter, A., Iida, F., Mihalyova, L., Nefti-Meziani, S., Neumann, G., Paoletti, P., Pridmore, T., Ross, D., Smith, M., Stoelen, M., Swainson, M., Wane, S., Wilson, P., Wright, I., and Yang, G.-Z. (2018). Agricultural robotics: The future of robotic agriculture. 3

Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, 13(2):99–110. 5

Fahimi, F. (2009). Autonomous robots. *Modeling, Path Planning and Control*. 5

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. 78

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2015). You only look once: Unified, real-time object detection. 92

Robotics, E. (2014). Strategic research agenda for robotics in europe 2014–2020. *IEEE Robot. Autom. Mag*, 24:171. 3

Santos, L. C., Aguiar, A. S., dos Santos, F. N., Valente, A., and Petry, M. R. (2020). Occupancy grid and topological maps extraction from satellite images for path planning in agricultural robots. *Robotics*, 9:77. 232

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2015). Rethinking the inception architecture for computer vision. 92