Collision Avoidance for Safe Structure Inspection with Multirotor UAV

F. Azevedo[1], A. Oliveira[1], A. Dias[1], J. Almeida[1], M. Moreira[1], T. Santos[1], A. Ferreira[1], A. Martins[1] and E. Silva[1]

***Abstract*— The multirotor UAVs are being integrated into a wide range of application scenarios due to maneuverability in 3D, versatility and reasonable payload of sensors. One of the application scenarios is the inspection of structures where the human intervention is difficult or unsafe and the UAV can provide an improvement of the collected data. At the same time introduce challenges due to low altitude missions and also the fact of being manually operated without line of sight. In order to overcome these issues, this paper presents a LiDAR-based real-time collision avoidance algorithm, denoted by Escape Elliptical Search Point with the ability to be integrated into autonomous and manned modes of operation. The algorithm was validated in a simulation environment developed in Gazebo and also in a mixed environment composed by a real robot in an outdoor scenario and simulated obstacle and LiDAR.**

I. INTRODUCTION

In recent years, there has been an increasing research effort with multirotor Unmanned Aerial Vehicle (UAV) in a wider range of scenarios, such as search rescue missions, surveillance and inspection tasks. One of the reasons is the fact that this type of vehicles provides the required maneuverability to navigate through complex three-dimensional scenarios with a reasonable payload of sensors. Considering the application scenarios of inspection, the multirotor UAV provides the ability to collect data from different positions, angles and distances, and at the same time reduce the cost and the human risk. Most of this operations are performed through an operator or more recently in fully autonomous missions. In both cases, and due to low altitude operation and the existence of structures like buildings, power lines or even natural obstacles like trees, the risk of crashing, damage structures and injury surrounding people has been increased.

Therefore, this paper proposes to address the research area of real-time obstacle avoidance for manned and autonomous multirotor UAVs. Based on the work developed by [1] for rotorcraft UAVs and the evaluation performed of the advantages of LiDAR solutions [2] for obstacle avoidance, we propose to extend the method with a reactive obstacle avoidance algorithm based on LiDAR and applied to multirotor, denoted by *Escape Elliptical Search Point - LiDAR-based Collision Avoidance (E^{2}SP-LCA)*.

The E^{2}SP-LCA algorithm bounds the map and searches for any point that lies inside a safety volume (obstacle). Due to the dynamics of the vehicle, the safety volume will be proportional to the vehicle speed, taking also into account the direction of the velocity and the position of the *waypoint*. When an obstacle is found, the algorithm evaluates a set of candidate points to avoid the collision (escape points). As the UAV is normally blind above and below its position, the algorithm proposes to avoid the obstacle performing a path as horizontal as possible. If no valid escape point is found, it will intentionally deviate from the original trajectory to try to overpass the obstacle.

The paper is outlined as follows: Section II presents the related work starting with the obstacle detection and map representation followed by real-time obstacle avoidance techniques. In Section III is detailed the E^{2}SP-LCA, followed by its implementation and the obtained results, in Section IV. In Section V are exposed some algorithm remarks and considerations, followed by Section VI that presents the conclusions and the proposed future work to improve this project.

II. RELATED WORK

This section presents the research works related to map representation and obstacle detection, as well as real-time obstacle avoidance algorithms.

A. Obstacle Detection and Map Representation

In the field of multirotor UAVs, the detection of an obstacle is mainly performed through monocular cameras[3], LiDAR[2][4], stereo cameras[2] or their combination[2], depending on the application scenario. Their advantages and drawbacks have been evaluated in [2].

Being active sensors, LiDARs are typically insensitive to the light of the environment, having more accuracy and better performance for far obstacles. The low processing power required makes them more efficient for real-time applications. However, the collected data is produced sequentially, the maximum range is limited and requires more electrical power. Approaches like [4] and [5] are some of the examples that use LiDAR-based detection systems.

On the other side, the stereo cameras provide a snapshot of the environment at one instant (global shutter cameras), producing a dense 3D range information, with color correspondence and capable of detecting objects from long distances (depending on the baseline between the cameras). However, this system highly depends on the visual environment conditions and requires a significant processing power. Besides that, its range accuracy decreases with range squared. In [6] is used a stereo vision system for obstacle detection.

[1]INESC Technology and Science, ISEP - School of Engineering, Porto, Portugal fabio.a.azevedo, alexandre.a.oliveira, andre.dias, jose.m.almeida, miguel.m.moreira, tiago.a.santos, andre.f.ferreira, alfredo.martins, eduardo.silva@inesctec.pt

In addition to the systems referred above, there are other solutions like [7] and [3] that uses a monocular technique to avoid collision with structures.

In the most basic way, the obstacles can be represented on a map by a simple point cloud with the measures given by a LiDAR or the features extracted from images. However, this is computationally costly and can compromise the real-time requirement. For reducing this cost, the data can be clustered, resulting in a sparse representation. Another disadvantage of this method is that is not possible to distinguish between free and unmapped spaces.

Using a point cloud as input, the memory space required for storing the map information can be reduced using techniques like the representation by means of *octrees*, *Octomaps* or *Voxel Grids* [8].

Other ways of representing occupancy maps are analyzed and summarized in [9].

B. Real-time Obstacle Avoidance Algorithms

In [4] is presented a solution with a helicopter to perform infrastructure inspection with collision avoidance, using a fixed 2D LiDAR and two flight modes. The *pirouette descent* mode creates a spinning LiDAR with a cylindrical field of view by rotating the helicopter around its yaw axis while descending vertically. The *waggle cruise* flight mode performs a horizontal sweep while flying forward, allowing to scan a corridor-shaped space. It provides two solutions to avoid obstacles and reach the goal, however, it does not have a global map, which implies some constraints for the avoidance maneuver to be completed successfully, as not having obstacles above the vehicle. Besides that, as it only uses a fixed 2D LiDAR, the quality of the generated map is strongly dependent on the quality of its position estimation.

Another obstacle avoidance maneuver is presented in [1] that considers the vehicle as a sphere and constructs a safety volume around it. Whenever an obstacle enters the safety volume, it constructs an ellipse around the obstacle and searches for a point that allows a free path from the current position to the *escape point* and that also ensures a collision-free path through a defined distance from the *escape point*, on the direction to the *waypoint*. If no clear path is found, it extends the ellipse radius (a certain number of times) and performs another search. If no free path is found with the maximum ellipse radius, the UAV will hover until a pilot takes control of it.

The *escape point* has the advantage of allowing an uninterrupted flight for avoiding the obstacle, otherwise, the vehicle would need to stop (hover) and recalculate the trajectory considering arbitrary avoidance points.

Although it is applied to aircraft, Sabatini et al. [10] have implemented an obstacle avoidance ellipsoid-shaped safety zone around obstacles. The planning algorithm for the obstacle avoidance takes into account the aircraft dynamics, velocity, acceleration and distance to the obstacle. In a case of high velocities and/or accelerations, the time to find an alternative path and the distance to the obstacle are the major inputs of the cost function, as they are the main parameters to be considered in critical situations (an aircraft cannot hover).

III. E^{2}SP-LCA - Escape Elliptical Search Point - LiDAR-based Collision Avoidance

The E^{2}SP-LCA can be resumed as an algorithm that follows the classical architecture of mobile robots navigation. Whenever it gets an update of the occupancy map, performs a search for potential obstacles between the UAV position and the *waypoint*, that can cause a damage on the vehicle or blocking it from reaching the desired position (algorithm 1), and tries to avoid them.

Figure 1 illustrates the algorithm behavior. On every map update, it starts to search for obstacles inside a safety zone that is created around the UAV, and propagated through a certain direction, by means of spheres of variable radius s_{rad} and centers distance d_{center}.

The direction of the safety zone is given by the weighted sum (parameters n and m) of the vector from the vehicle position to the *waypoint* ($\vec{u}$) and the UAV velocity (${}^W\mathrm{P}_{\vec{velocity}}$). For calculating the center's separation is used the equation 1, presented in [1], where V is the voxel size, however, the radius of the spheres is a defined parameter that is proportional to the vehicle's velocity norm. This approach ensures that some vehicle dynamics (at every moment) is taken into account in the search for potential obstacles, as the velocity affects the safety volume size and propagation direction.

Algorithm 1 ${}^W\mathrm{U}$ = E^{2}SP(${}^W\mathrm{O}_b$, ${}^W\mathrm{P}$, waypoint)

$obstacle \leftarrow false$
$s_{rad} \leftarrow k \cdot \left(\left\| {}^W\mathrm{P}_{\vec{velocity}} \right\| + 1) \right)$
$\vec{u} \leftarrow waypoint - {}^W\mathrm{P}_{position}$
$\vec{d} \leftarrow n \cdot \hat{u} + m \cdot {}^W\mathrm{P}_{velocity}$
$d_{center} \leftarrow 2\sqrt{s_{rad} \cdot V - \frac{V^2}{4}}$
for $i = 0 \rightarrow n\_spheres$ **do**
  $center[i] = {}^W\mathrm{P} + i \cdot d_{center} \cdot \frac{\vec{d}}{\|d\|}$
end for
for each $cell \in {}^W\mathrm{O}_b$ **do**
  for each $center$ **do**
    $distance \leftarrow \|cell - center\|$
    if $distance < s_{rad}$ **then**
      $obstacle \leftarrow true$
      if $distance < closest\_dist$ **then**
        ${}^W\mathrm{O}_c \leftarrow cell$
        $closest\_dist \leftarrow distance$
      end if
    end if
  end for
end for
if $obstacle$ **then**
  ${}^W\mathrm{U} \leftarrow sch\_esc({}^W\mathrm{O}_b, {}^W\mathrm{P}, waypoint, {}^W\mathrm{O}_c)$
end if
return ${}^W\mathrm{U}$

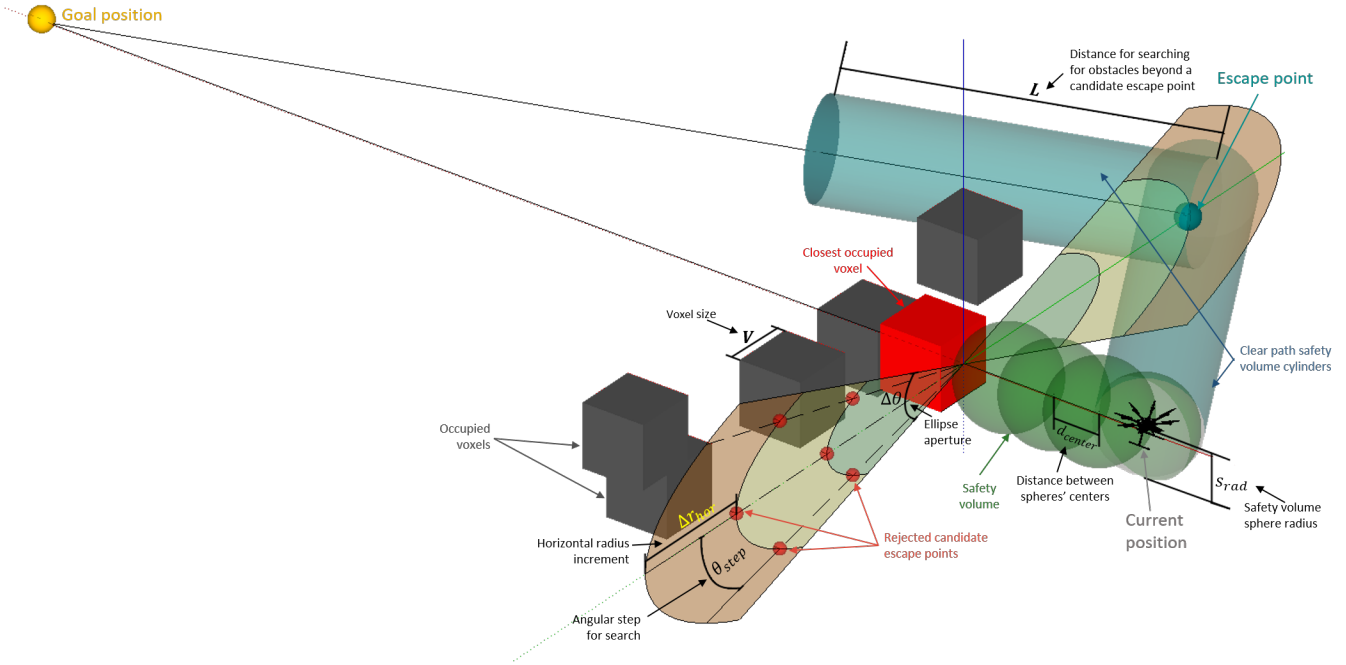

Fig. 1. E^{2}SP-LCA algorithm representation.

$$d_{center} = 2\sqrt{s_{rad}V - \frac{V^2}{4}} \tag{1}$$

If any occupied cell is inside the spheres, the path is considered obstructed and is called the function to search for an alternative path (sch_esc). If more than one obstacle is found, for searching an escape will be only considered the closest obstacle ${}^{W}\mathrm{O}_c$, as it is the one with the greatest probability of causing an UAV crash.

For search for an alternative path, it is placed an ellipse centered on the closest obstacle, with a horizontal direction $\vec{e_{hor}}$ normal to the vector $\vec{w}$, defined by the vehicle position and the closest obstacle, and a vertical direction $\vec{e_{ver}}$ parallel to the vertical axes of the world frame ($\hat{k} = (0, 0, 1)$).

With an ellipse defined by equation 2, θ can be limited to a $\Delta\theta$ value. For example, if $\Delta\theta = \pi/2$, $\theta \in [-\pi; -3\pi/4] \cup [-\pi/4; \pi/4] \cup [3\pi/4; \pi[$, will result a valid search marked in blue on figure 2.

$$\begin{aligned} x &= r_{hor} \cdot cos(\theta) \\ y &= r_{ver} \cdot sin(\theta) \end{aligned} \tag{2}$$

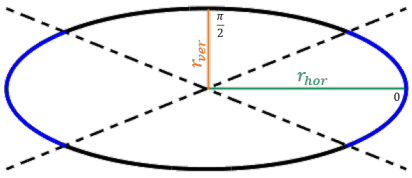

Fig. 2. Ellipse aperture. Valid zone in blue. Ellipse defined by a horizontal radius r_{hor} and a vertical radius r_{ver}.

Defining the valid angular aperture of the ellipse will constrain the avoidance path, not allowing the avoidance from above or below the obstacle. This angular aperture can be configured taking into account the application and the sensor in use. For example, if it is used a LiDAR sensor with a low vertical aperture, it is interesting to keep the vehicle as horizontal as possible.

For optimizing the avoidance path, the distance to travel should be as small as possible, so the direction of search (right to left or left to right) of a valid escape point will depend on the values of the distance from the left and right horizontal limit edges of the ellipse to the *waypoint* ($dist_l$ and $dist_r$, respectively). Those limit edges are obtained by setting θ to 0 (zero) or π on the ellipse equation 6 with the parameters represented on figure 2.

After been chosen the first escape point to be considered, the algorithm will search candidates on the edge of the ellipse (limited by an angular aperture $\Delta\theta$), by incrementing (or decrementing) an angular step θ_{step}. For each candidate escape point, the following conditions are evaluated:

- Clear path from current position to the candidate escape point;
- Clear path between the escape point and the waypoint along a predefined distance (L);

If both conditions are verified, the candidate escape point is considered valid, otherwise, the ellipse size is increased by Δr_hor and the procedure is repeated for the new ellipse.

A clear path is determined by verifying if there is any point/obstacle p_t that lies inside a cylinder between two points p_1 and p_2 with radius s_{rad}. For that, two vectors are generated, $\vec{d_{12}} = p_2 - p_1$ and $\vec{d_{1t}} = p_t - p_1$. Calculating the

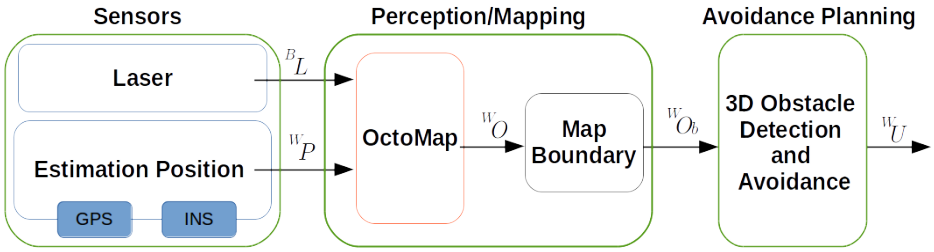

Fig. 3. Obstacle avoidance high level architecture.

dot product:

$$D = \vec{d_{12}} \cdot \vec{d_{1t}} \tag{3}$$

If $D < 0$ or $D > \left\|\vec{d_{12}}\right\|^2$, the point is outside the cylinder limits, otherwise, it has to be tested the closest distance from p_t to the line segment defined by p_1 and p_2.

Assuming α as the angular difference between vectors d_{12} and d_{1t}, and considering the fact that $\sin^2 + \cos^2 = 1$ and the dot product $D = \cos\alpha \cdot \left\|\vec{d_{12}}\right\| \cdot \left\|\vec{d_{1t}}\right\|$ which is equivalent to equation 3. Considering the distance from a point to the line segment, defined by $\sin\alpha \cdot \left\|\vec{d_{1t}}\right\|$ and the squared distance to the cylinder center given by $d_{cyl}^2 = (1 - \cos^2\alpha) \cdot \|d_{1t}\|^2$ we obtain

$$d_{cyl}^2 = \left(1 - (\vec{d_{1t}} \cdot \vec{d_{12}})^2 / \left(\left\|\vec{d_{1t}}\right\|^2 \cdot \left\|\vec{d_{12}}\right\|^2\right)\right) \cdot \left\|\vec{d_{1t}}\right\|^2 \tag{4}$$

by applying the dot product D. Therefore, considering equation 3, d_{cyl} can be rewrite as:

$$d_{cyl}^2 = \left\|\vec{d_{1t}}\right\|^2 - \left(\frac{D}{\left\|\vec{d_{12}}\right\|}\right)^2 \tag{5}$$

Having equation 5, if $d_{cyl}^2 > s_{rad}^2$, the path is clear, otherwise, there is an obstacle on the evaluated path and the candidate escape point is not valid.

Once a valid candidate escape point is found (using equation 6 with θ constrained), it is passed for the navigation through (${}^{W}\mathrm{U}$).

$$\begin{aligned} {}^{W}\mathrm{U} = {}&{}^{W}\mathrm{O}_c + r_{hor} \cdot \hat{e_{hor}} \cdot \cos\theta \\ &+ r_{ver} \cdot \hat{e_{ver}} \cdot \sin\theta \end{aligned} \tag{6}$$

If no valid escape point is found, after a predefined number of increases of the ellipse radius, the vehicle is commanded to move side-to-side through parameterized distance. If in that movement a clear path is found, the normal operation is returned, if not, the vehicle return to the point where the obstacle was detected, generates a warning message and waits for a manual control.

IV. Implementation and Results

In order to evaluate the E^{2}SP-LCA algorithm, we divided the validation into two phases: the first one was performed with the support of the simulation environment Gazebo[11][12] and the second one in a mixed environment composed by a real multirotor UAV in an outdoor scenario and a simulated obstacle and LiDAR sensor. Both approaches were implemented to validate the robustness of the E^{2}SP-LCA algorithm into different scenarios and in the particular case of the second test, also to ensure that the first tests of the algorithm were performed without risking a UAV with higher payload and more costly sensors like the LiDAR *Velodyne VLP-16*.

The algorithm was implemented in the framework ROS (Robotic Operating System)[13] in order to ensure a more straightforward integration between the simulation environment and the real multirotor UAV. The high level architecture is despited in figure 3 and is composed by three layers: **Sensors**, responsible for the sensor data acquisition, providing the LiDAR output in body frame reference ${}^{B}\mathrm{L}$ and an estimated vehicle pose, velocity and acceleration in global frame ${}^{W}\mathrm{P}$ through an Extended Kalman Filter (EKF) data fusion block of GPS and INS; **Perception/Mapping**, responsible for building a list of obstacles with the support of the Octomap toolbox that will generate unbounded voxels ${}^{W}\mathrm{O}$ with a predefine resolution of 0.4 meters. To improve the CPU performance and ensure real-time requirements we introduce a new feature to the Octomap to create a bounded voxels ${}^{W}\mathrm{O}_b$ that will be passed through a topic to the avoidance planning layer; **Avoidance Planning**, implements the E^{2}SP-LCA algorithm detail in section III based on the bounded voxels ${}^{W}\mathrm{O}_b$ and provides an output action denoted by ${}^{W}\mathrm{U}$ with a collision avoidance path planning escape point expressed in equation 6.

A. Simulation

The simulation environment chosen to benchmark E^{2}SP-LCA algorithm was Gazebo. Other simulators were considered, like MORSE[14] but the Gazebo was the one that provides a feasible integration with the autopilot PX4 project[15] through the *Software In The Loop* (SITL) and a simulated multirotor UAV model with a LiDAR payload sensor[16].

The simulation environment is depicted in figure 4, is composed by several walls and a path defined by a purple

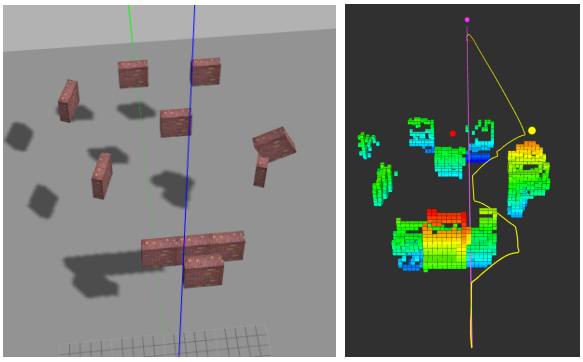

Fig. 4. Avoidance path into a complex scenario, with the purple line as ideal path, the yellow line as the UAV trajectory, red dot as the left extreme of the ellipse and the yellow dot as the chosen escape point

line in the right figure. The obstacle avoidance trajectory is represented by the yellow line and is possible to observe that the UAV was able to overcome the obstacles and reach in a safe manner the desired position.

Figure 5 presents a situation where the UAV was not able to detect an escape point based on the predefined angular constrain ($\Delta\theta = \pi/2$) in order to avoid the UAV pass the obstacle from above (figure 2). This figure also represents a situation where the vehicle is capable of finding an escape after performing a movement parallel to the wall.

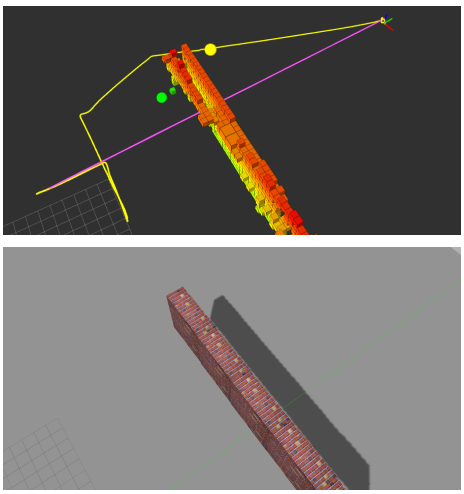

Fig. 5. Avoidance of a large obstacle, with the purple line as ideal path, the yellow line as the UAV trajectory, green dot as the right extreme of the ellipse and the yellow dot as the chosen escape point

In order to evaluate the contribution of the bounded voxels ${}^{W}\mathrm{O}_{b}$ method against the unbounded voxel map, it was created a simulated environment, depicted in figure 6. The UAV navigate through it and the processing time for the obstacle search algorithm took an average of 7 ms with a standard deviation of 4.43 ms. For a fixed volume of 20 meters around the UAV position, the processing time decrease to an average of 1 ms with a standard deviation of 0.36 ms. Once the map representation is completed, the unbounded method will stabilize in processing time while the bounded method keeps a low and constant time of processing during the UAV navigation. This allows us to conclude that this approach is more feasible in unstructured scenarios where the vehicles must navigate and keep the real-time constraints independent of the scenario.

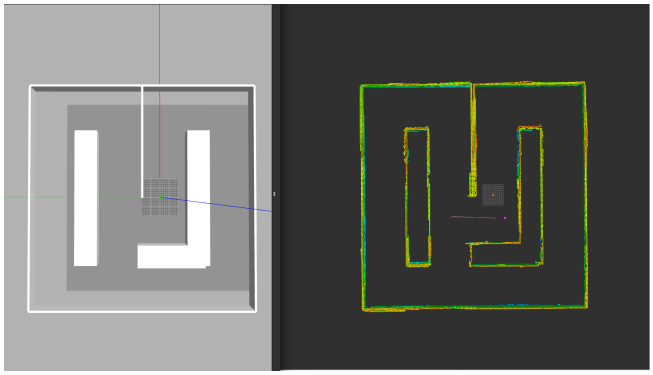

Fig. 6. Map for testing bounding method

With respect to the avoidance escape point, the required time processing in the simulation environment was 1.41 ms with a standard deviation of 0.97 ms for a scenario detailed in figure 4 composed by $\sim$ 760 voxels (bounded voxels). The simulations were performed with a CPU i7-740QM @ 1.73GHz, 8 GB of DDR3 RAM, NVIDIA GeForce GTX 460M, running the Ubuntu 14.04 LTS.

B. Field tests with a real UAV and a simulated sensor and obstacle

Based on the results obtained in the simulation environment, the second phase was the validation with a real UAV in an outdoor scenario (ISEP Campus) mixed with a simulated obstacle and payload LiDAR. The LiDAR used during field tests and the obstacle were the one that has been used for the simulation tests detail in section IV-A.

The implemented architecture is detailed in figure 7. The UAV is running internally the obstacle avoidance describe in figure 3 and receives remotely the simulated data from the LiDAR.

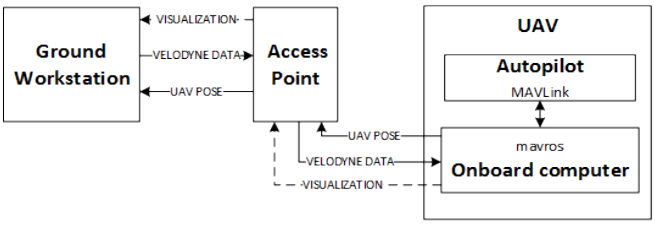

Fig. 7. Implemented architecture for real UAV in an outdoor scenario (ISEP Campus) mixed with a simulated obstacle and payload LiDAR.

The UAV is a customized hexacopter, depicted in figure 8, equipped with an open-source autopilot, Pixhawk board running PX4 Firmware and an embedded onboard computer, Odroid XU3, running Ubuntu 14.04 with Robot Operating System (ROS) Indigo.

The outdoor field test was composed by a simulated obstacle (3x3x1 meters) and the real UAV perform a trajectory

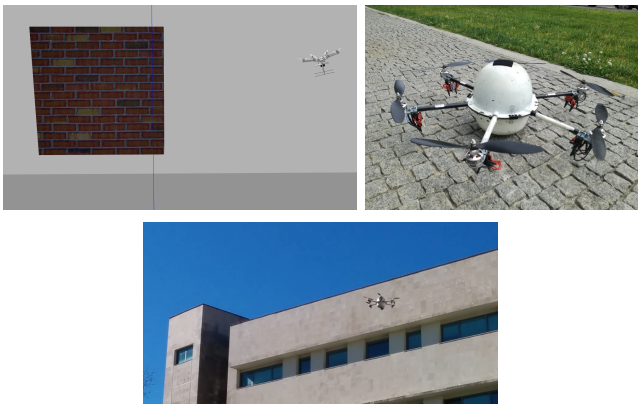

Fig. 8. Real UAV in an outdoor scenario (ISEP Campus) with virtual wall.

towards a position that requires an avoidance maneuver. The trajectory and the avoidance path is depicted in figure 8, with the yellow line being the UAV avoidance trajectory ${}^{W}\mathrm{U}$.

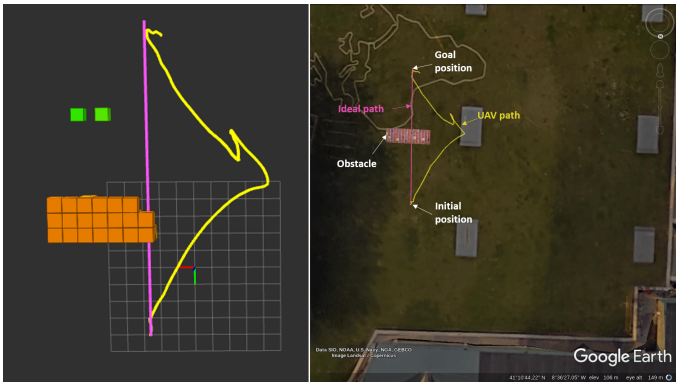

Fig. 9. UAV trajectory in RVIZ and in the Google Earth image. The yellow line represents the real UAV avoidance trajectory ${}^{W}\mathrm{U}$.

For the field test scenario, the embedded CPU average time processing for the obstacle search algorithm was 0.397 ms with a standard deviation of 0.1041 ms ($\sim$ 83 bounded voxels), with the avoidance escape point requiring 8.63 ms with a standard deviation of 0.136 ms

V. Remarks

The E^{2}SP-LCA is a collision avoidance algorithm that is capable of performing a safe inspection with low computational cost. This is obtained by considering as potential obstacles only the ones that lie inside a bounded volume, around the UAV position. The safety volume (volume inside which any occupied cell will be treated as an obstacle) has a dynamic behavior, once it is clearly dependent on the UAV's velocity, both in size (that depends on the velocity module) and direction of propagation, that depends both on the vector that connects the UAV current position and the desired one, and on the direction of its velocity vector). This dependence on the velocity vector can be tuned using the parameters n and m (algorithm 1), which makes E^{2}SP-LCA an algorithm that takes into account some vehicle dynamics and suitable for any multirotor UAV.

Another parameter that can be tuned is the aperture of the search ellipse, as well as the valid zone, meaning that it can be configured to work on a wide set of cases. For example, considering the figure 2, if a vehicle is able to detect what is above him and is operating on an environment where the obstacles are wide and have low height, the algorithm can be adapted to accept the top part of the ellipse as a valid zone and give a greater value to r_{ver} than to r_{hor} (this will set a preference to overpass the obstacles from above).

Adding to this, this approach tries to find a solution whenever a valid escape point is not found, moving parallel to the obstacle and trying to find a clear path from a different position(figure 5). However, the algorithm will perform this maneuver only a limited number of times, not ensuring that the desired point will be reached. If no valid path is found, the algorithm will ask to the pilot to take control of the vehicle, hovering on the position where it first detected the obstacle.

As all the obstacles are referenced to a global frame, the E^{2}SP-LCA relies on a good navigation and estimation of the vehicle's position. Another drawback of this approach is that the algorithm can enter on an infinite loop mode. This case might happen on an environment with many obstacles, if it keeps finding an obstacle while avoiding another (previously detected), entering on a mode of constant avoidance that might lead to a deviation from the desired point.

VI. Conclusions and Future Work

The paper presents a LiDAR-based real-time collision avoidance method for multirotor UAVs with the ability to ensure an autonomous structure inspection mission without a predefined out of bounds areas. The collision avoidance method was validated in a simulation environment developed in Gazebo and also in a mixed environment composed by a real UAV performing a mission in an outdoor scenario and a simulated obstacle and LiDAR. This approach provides a safe method to validate the vehicle behavior without the possibility of damage the sensors like LiDAR and also the ability to test in a small-scale UAV (low payload). In both scenarios, campus ISEP, and simulation environment, the vehicle was able to detect the obstacle and generate a collision avoidance safe path. For future work, we intend to validate the algorithm with an UAV with payload capability for a LiDAR Velodyne VLP-16 and perform the validation with natural obstacles like trees and also in the presence of structure obstacles, for instance, power lines, bridges and electricity poles. Another line of work will be the integration of the vision-based power line detection method denoted by PLineD[17] with the E^{2}SP - Escape Elliptical Search Point. The expected output of this future research work is the ability to combine the LiDAR information with the monocular vision system required by the PLineD algorithm and ensure an, even more, robustness UAV autonomous inspection procedure.

References

[1] S. Hrabar, "Reactive obstacle avoidance for Rotorcraft UAVs," in *2011 IEEE/RSJ International Conference on Intelligent Robots and*

Systems. IEEE, sep 2011, pp. 4967–4974. [Online]. Available: http://ieeexplore.ieee.org/document/6094629/

[2] ——, "An evaluation of stereo and laser-based range sensing for rotorcraft unmanned aerial vehicle obstacle avoidance," *Journal of Field Robotics*, vol. 29, no. 2, pp. 215–239, mar 2012. [Online]. Available: http://doi.wiley.com/10.1002/rob.21404

[3] L. Kovacs, "Visual Monocular Obstacle Avoidance for Small Unmanned Vehicles," *Conference on Computer Vision and Pattern Recognition Workshops*, pp. 59–66, 2016.

[4] T. Merz and F. Kendoul, "Beyond visual range obstacle avoidance and infrastructure inspection by an autonomous helicopter," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, sep 2011, pp. 4953–4960. [Online]. Available: http://ieeexplore.ieee.org/document/6094584/

[5] S. Ramasamy, R. Sabatini, A. Gardi, and J. Liu, "LIDAR obstacle warning and avoidance system for unmanned aerial vehicle sense-and-avoid," *Aerospace Science and Technology*, vol. 55, pp. 344–358, aug 2016. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1270963816301900

[6] S. Hrabar, "3D path planning and stereo-based obstacle avoidance for rotorcraft UAVs," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, sep 2008, pp. 807–814. [Online]. Available: http://ieeexplore.ieee.org/document/4650775/

[7] D. Magree, J. G. Mooney, and E. N. Johnson, "Monocular visual mapping for obstacle avoidance on UAVs," in *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, may 2013, pp. 471–479. [Online]. Available: http://ieeexplore.ieee.org/document/6564722/

[8] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013, software available at http://octomap.github.com. [Online]. Available: http://octomap.github.com

[9] Wolfram Burgard, Maren Bennewitz, Diego Tipaldi, and Luciano Spinello, "Techniques for 3D Mapping." [Online]. Available: http://ais.informatik.uni-freiburg.de/teaching/ss14/robotics/slides/17-3dmapping.pdf

[10] R. Sabatini, A. Gardi, and M. A. Richardson, "Lidar obstacle warning and avoidance system for unmanned aircraft," *International Journal of Mechanical, Aerospace, Industrial, Mechatronic and Manufacturing Engineering*, vol. 8, no. 4, pp. 711 – 722, 2014. [Online]. Available: http://waset.org/Publications?p=88

[11] Open Source Robotics Foundation, "Gazebo," 2014. [Online]. Available: http://gazebosim.org/

[12] M. Zhang, H. Qin, M. Lan, J. Lin, S. Wang, K. Liu, F. Lin, and B. M. Chen, "A high fidelity simulator for a quadrotor uav using ros and gazebo," in *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society*, Nov 2015, pp. 002 846–002 851.

[13] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[14] A. Dias, J. Almeida, N. Dias, E. Silva, and P. Lima, "Simulation environment for multi-robot cooperative 3d target perception," *SIMPAR 2014 4th International Conference on Simulation, Modeling , and Programming for Autonomous Robots, Springer-Verlag Lecture Notes in Computer Science*, 2014.

[15] "Gazebo Simulation · PX4 Devguide," 2016. [Online]. Available: https://dev.px4.io/simulation-gazebo.html

[16] K. Hallenbeck, "DataspeedInc / velodyne_simulator Bitbucket," 2015. [Online]. Available: https://bitbucket.org/DataspeedInc/velodyne_simulator

[17] T. Santos, M. Moreira, J. Almeida, A. Dias, A. Martins, J. Dinis, J. Formiga, and E. Silva, "PLineD: Vision-based Power Lines Detection for Unmanned Aerial Vehicles," in *17th International Conference on Autonomous Robot Systems and Competitions ICARSC*. IEEE, 2017.