Dynamic collision avoidance system for a manipulator based on RGB-D data

Thadeu Brito[1,2], Jose Lima[2,3], Pedro Costa[3,4], and Luis Piardi[1,2]

[1] Federal University of Technology - Paraná, Brazil
thadeu_brito@hotmail.com
luis_piardi@outlook.com
[2] Polytechnic Institute of Bragança, Portugal
jllima@ipb.pt
[3] Faculty of Engineering of University of Porto, Portugal
pedrogc@fe.up.pt
[4] INESC-TEC, Centre for Robotics in Industry and Intelligent Systems, Portugal

Abstract. The new paradigms of Industry 4.0 demand the collaboration between robot and humans. They could help and collaborate each other without any additional safety unlike other manipulators. The robot should have the ability of acquire the environment and plan (or re-plan) on-the-fly the movement avoiding the obstacles and people. This paper proposes a system that acquires the environment space, based on a kinect sensor, performs the path planning of a UR5 manipulator for pick and place tasks while avoiding the objects, based on the point cloud from kinect. Results allow to validate the proposed system.

Keywords: Collaborative robots, manipulator path planning, collision avoidance, RGB-D.

1 Introduction

One of the most important task in Industry 4.0 related to cooperation is the ability to estimate and avoidance of collision for a robot manipulator. Collaborative robotics is a topic addressed in Industry 4.0 where humans and robots can share and help each other in a cooperative way. Collaborative robot can be used without any additional safety unlike other manipulators. This means, the robot should have the capacity of acquire the environment and plan the movement avoiding the obstacles and people. The cooperation between human and robot requires that the robot could re-plan the path to reach the target position that avoids the collision with human parts and obstacles in real time, this means on the fly, while the arm is moving. Such process can be called dynamic collisions avoidance.

Nowadays RGB-D sensors help this environment acquisition and perception so that the system can do the path planning with constraints. The depth cameras are increasing its popularity and decreasing its prices. The well-known kinect sensor is an example of that. This paper proposes a system that acquires the

environment space, based on a kinect sensor, performs the path planning of a UR5 manipulator while avoiding the objects. Two algorithms were tested in real acquired situations with a simulated UR5 robot and the results point the advantages for this approach.

2 Related work

An important step to be considered when developing the manipulator system is the path planning. Path planning is a key area of robotics. It comprises planning algorithms, configuration space discretization strategies and related constraints. It is well known that path planning for robots with many degrees of freedom is a complex task. Barraquand and Latombe [1], in 1991, proposed a new approach to robot path planning that consisted of building and searching a graph connecting the local minima of the potential function defined over the robot's configuration space. This new approach was proposed considering robots with multiple degrees of freedom. Later Ralli and Hirzinger [2] refined that same algorithm accelerating the system, calculating solutions with a lower estimated executing time. Probabilistic methods were introduced by Kavraki et al [3] with the objective to reduce the configuration free space complexity. This method is not adapted for dynamic environments since a change in the environment causes the reconstruction of the whole graph. Several variants of these methods were proposed: Visibility based PRM [4], Medial axis PRM [5], Lazy PRM [6] and sampling based roadmap of trees [7]. Other methods are used and Helguera et all used a local method to plan paths for manipulator robots and solved the local minima problem by making a search in a graph describing the local environment using and A* algorithm until the local minima are avoided [8]. The path planning becomes more complex when there are inserted obstacles in a given environment. Blackmore and Williams in 2006 presents a complete algorithm by posing the problem as disjunctive programming. They are able to use existing constrained optimization methods to generate optimal trajectories for manipulator path planning with obstacles [9]. Path planning in real-time is introduced by Samir et all in 2006 in the dynamic environment. This approach is based on the constraints method coupled with a procedure to avoid local minima by bypassing obstacles using a boundary following strategy [10] More recent Tavares et all use a double A* algorithm for multiple industrial manipulators. This approach uses one A* algorithm to approach the target and an another A* more refining to reduce the error [11].

3 System architecture

Many efforts have been done to achieve a system that acquires the environment by means of an RGB-D sensor, planning a way for an UR5 manipulator to reach its end point with the ability to avoid obstacles. Figure 1, presents a simplified block diagram of the system.

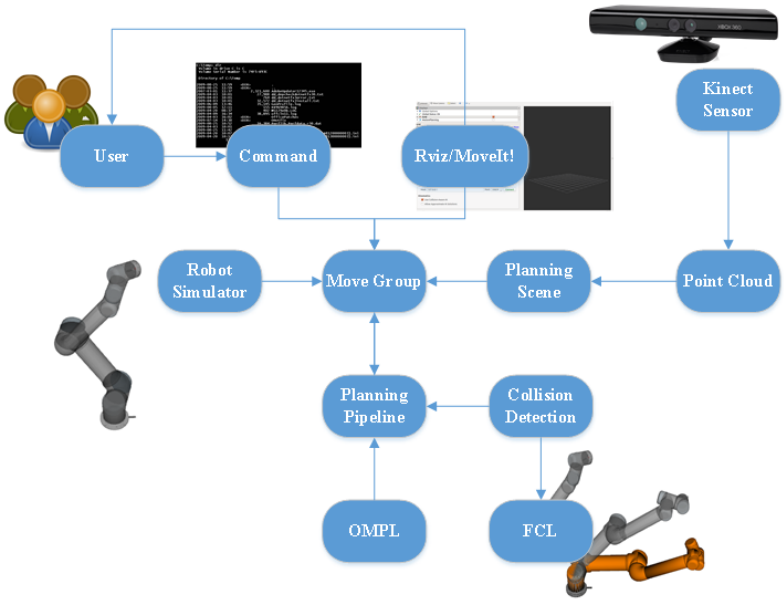

Fig. 1. Diagram of system functions. Image adapted from [22].

3.1 ROS

Robotic Operating System (ROS) is a framework that contains a wide range of use in developing programs for robots. ROS makes interactions between the functionalities of a robot (sensors, locomotion, vision, navigation and location) with contribution of libraries and services, facilitating the robotic application.

The philosophy is to make a piece of software that could work in other robots by making little changes in the code [16]. Several ROS modules are used in the present work. Next subsections address the main ones.

3.2 Rviz

The ROS framework comes with a great number of powerful tools to help the user and developer in the process of debugging the code, and detecting problems with both the hardware and software. This comprises debugging facilities such as log messages as well as visualization and inspection capabilities which allows the user to see what is going on in the system easily [16].

Rviz, presented in Figure 2 is a 3D visualizer to make a virtual simulation of robotic models in ROS. During the simulation it is possible to create scenarios with obstacles, to change positions of pose of a robot or to move them through a virtual "world". It is also possible to insert sensors, such as Kinect, change positions of sensors or robots. This way it is confirmed that the application is ready to be implemented in a real application, avoiding possible problems in real robots.

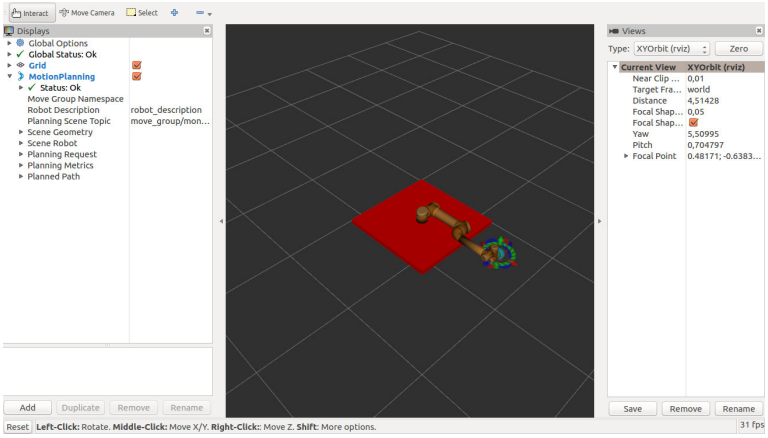

Fig. 2. Screenshot of a Rviz 3D visualizer with a simple application. On the left a display panel, in the middle a simulated UR5 on the table and on the right a views panel.

3.3 RGB-D Sensor

Develop a robotic application requires the use of sensors. There are currently several types of devices that ROS supports. This package is defined in different categories: 2D range finders, 3D sensors, Pose estimation, Cameras, Sensor Interfaces, and other ones [20]. The 3D sensor package, contains the RGB-Depth (RGB-D) sensors such as Kinect.

RGB-D cameras consist of an RGB and a depth sensor that capture color images along with per-pixel depth information (depth map). These features have promoted the wide adoption of low-cost RGB-D cameras in numerous at-home applications, such as body tracking, gait monitoring for tele-rehabilitation, tracking of facial expressions, object and gesture recognition among the others [12].

3.4 MoveIt!

MoveIt! is a well-known software for planning mobile manipulation movements, incorporating the latest advances in motion planning, manipulation, 3D perception, kinematics, control and navigation. It provides an easy-to-use platform for developing advanced robotics applications, evaluating new robot designs and building integrated robotics products for industrial, commercial, R&D and other domains [13]. Figure 3 shows a screenshot of the MoveIt! performing a path planning.

The main node of this software is the move_group that integrates among several other tools. A good example of how move_group works is the path planning, where it is necessary to collect information from a point cloud and turn it into obstacles in the simulation. MoveIt! uses C++ or Phyton language which makes it easy to establish commands and create interface when viewing some movement

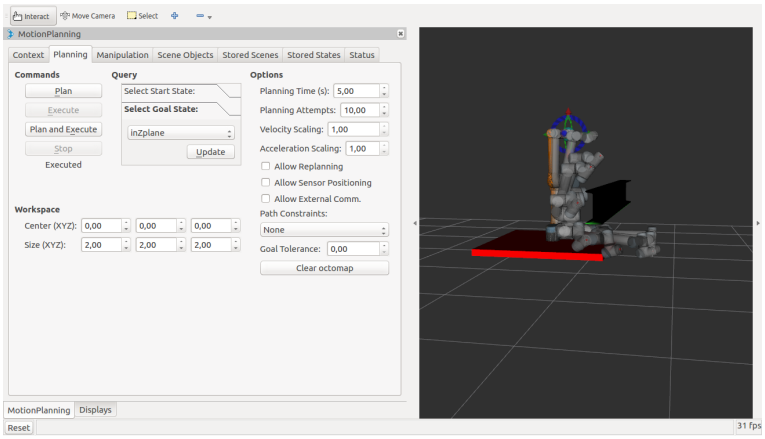

Fig. 3. Screenshot of a MoveIt! with a planned path executed.

in 3D. The algorithms embedded in MoveIt! can be used by many planners: Open Motion Planning Library (OMPL), Stochastic Trajectory Optimization for Motion Planning (STOMP), Search-Based Planning Library (SBPL) and Covariant Hamiltonian Optimization for Motion Planning (CHOMP).

3.5 Camera calibration

The calibration of consumer-grade depth sensors has been widely investigated since the release of the first-generation Kinect in 2010. Various calibration methods, particularly for the depth sensor, have been studied by different research groups [14].

The availability of affordable depth sensors in conjunction with common RGB cameras (even in the same device, e.g. the Microsoft Kinect) provides robots with a complete and instantaneous representation of both the appearance and the 3D structure of the current surrounding environment. This type of information enables robots to perceive and actively interact with other agents inside the working environment. To obtain a reliable and accurate measurements, the intrinsic parameters of each sensors should be precisely calibrated and also the extrinsic parameters relating the two sensors should be precisely known. The calibration must be done because there are no integrated sensors able to provide both color and depth information yet (sensors are separated).

These sensors provide colored point clouds that suffer from a non accurate association between depth and RGB data, due to a non perfect alignment between the camera and the depth sensor. Moreover, depth images suffer from a geometric distortion, typically irregular and position dependent. These devices are factory calibrated, so each sensor is sold with its own calibration parameter set stored inside a non-volatile memory. On the other side, the depth distortion is not modeled in the factory calibration. So, a proper calibration method

for robust robotics applications should precisely estimate the misalignment and both the systematic and distortion errors [15]. Figure 4 shows the calibration procedure.

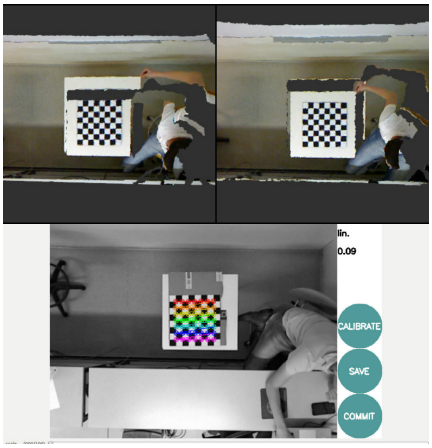

Fig. 4. Calibration procedure to obtain the intrinsic and extrinsic parameters.

4 Path planning

Different methods of path planning can be exploited in an application of robotic manipulators. An interesting planner is the OMPL, a library for many trajectory calculation algorithms. However, to check for collisions, the FCL library (Flexible Collision Library, included in MoveIt!) is used.

The OMPL planner works with two ways to create a path, one uses differential constraints (Control-based planners) and the other establishes a path through the geometric and kinematic constraints of the system (Geometric Planners) which is addressed in this paper [23].

A widely used algorithm is the multiple query of scripts created from the environment, known as PRM (Probabilistic Roadmap Method). These multiple scripts are based on sampling algorithms, which can have a higher cost framework. Another good algorithm is RRT (Rapidly-exploring Random Trees), that is very simple to implement: it has low cost of framework and has good outputs that accomplishes its work by making state trees.

Therefore, with these algorithms it is possible to carry out a path planning from an initial pose to a final pose. The steps to perform this path planning are indicated in Figure 1, so the use of a RGB-D image generated by a Kinect can check for possible obstacles.

Images are introduced into the system through interconnected nodes, which make the calculations necessary to have non-collision paths between the start

and the goal poses. Finally, the execution of the movement is done, if there is a trajectory planning without collisions. Otherwise the MoveIt! informs that it is not possible to carry out the collisions free movement. If the environment changes at any time, for example if obstacles change places or new obstacles are inserted into the work environment, the system (OMPL) will recalculate the trajectory to reach the final pose. By this way, the aim of a system of avoiding collisions dynamically arises, that is, the whole collision avoidance system adapts the planning routes according to the environment.

5 Results

To verify that the dynamic collision avoiding system works, it was made a lab simulation with a Kinect sensor and a virtual model of a UR5 manipulator. In this way, the purpose is to create a scenario with real obstacles to guarantee the operation of the system. The main idea is to make the Kinect sensor to create a point cloud of real obstacles and indicates to MoveIt! where the virtual manipulator can not collide, that is, where the manipulator can move to reach the goal pose. The working environment for the validation of the dynamic collision avoidance system was developed with a simple table as the base for one box that form the real obstacles. The virtual manipulator model has been configured to be fixed to the center of the table, so that the manipulator stays between the box and human, without maintaining contact with them. At this stage the use of the RGB-D sensor was important to generate the point cloud and to be able to calibrate the positioning of all the objects. In order to avoid shadow interference in point cloud generation, the best RGB-D sensor fixture is at the top of the working environment. In Figure 5, it is possible to observe the real working environment and the processes of transformation of this environment to the MoveIt! as a point cloud in order to perform the perception of the real obstacles and to make a simulation of path planning without collisions in a virtual model of the UR5 manipulator.

For the MoveIt! to perform the path planning it is necessary to configure the algorithm that will do the routes of the manipulator. For the tests, two algorithms, PRM and RRT, were chosen. Both algorithms were chosen because they are widely used in path planning, so these algorithms can be inserted into the dynamic collision avoidance system.

In order to guarantee the consistency of the comparison between two algorithms, the same path planning configuration was performed in the tests of each algorithm. Only two parameters were changed, planning time and planning attempt while the other parameters remained with the MoveIt! default configuration.

The planning time has been set to 10 seconds. This parameter indicates the time limit at which the system will take to find a path planning. In the parameter planning attempts was set to 15. This parameter indicates to the system how many path planning should be done within the set time. In case the system does not find a path planning within the timeout, the system will not move the

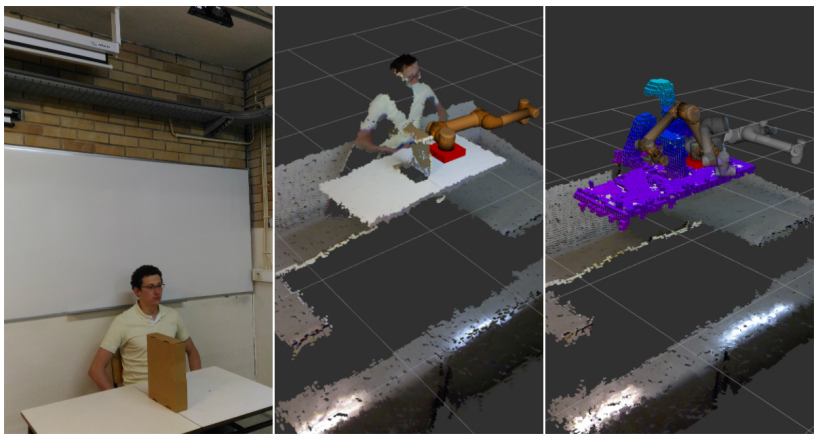

Fig. 5. Figure that shows the scenario created, the obstacles in the real world and the steps to transform the scenario in a way that the software perceives.

manipulator. The same manipulator pose configurations (start and goal poses) were used in both algorithms. The start and goal poses are indicated in Figure 6.

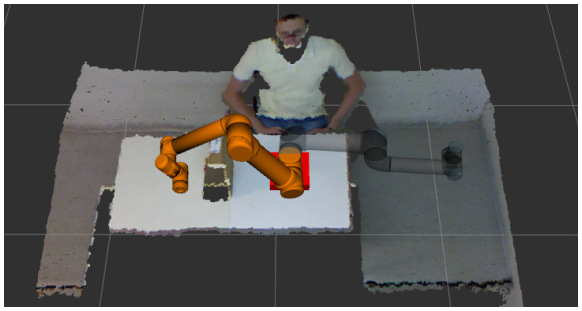

Fig. 6. The state where the UR5 manipulator is transparent is the start pose, as the state where the manipulator is orange is the goal pose.

Therefore, during each test the algorithm must find a route solution within the time limit and create states (or poses) to realize the trajectory. The state sequence that is expected for each algorithm to find can be visualized in Figure 7.

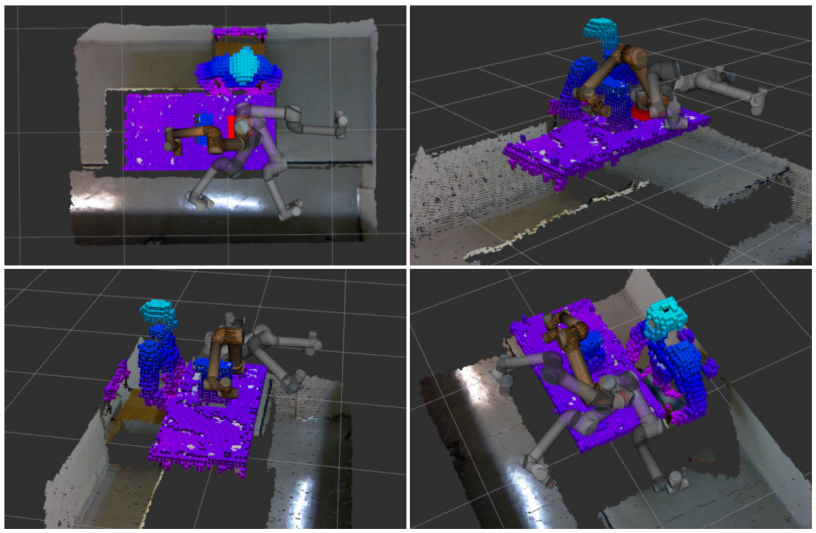

Fig. 7. Example of a sequence of states forming a trajectory.

The first test uses the PRM algorithm. Figure 8 shows the real scenario created and the transformation of this scenario to the perception of the software. It consists of: a point cloud transformed into Octotree (blue and purple boxes), the planning trail (in Gray) and the remaining points are the point cloud not considered for simulation.

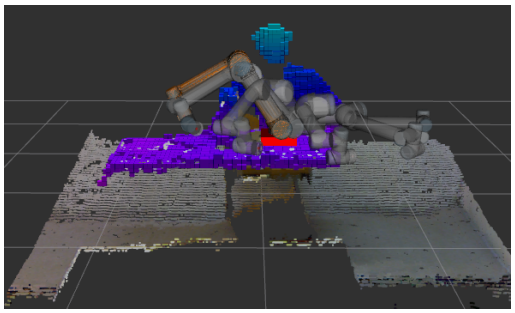

Fig. 8. Path planning using the PRM algorithm.

To ensure that the first test does not interfere with the next test, all nodes have been restarted. The second test, with the RRT algorithm presented in Figure 9, shows the same scenario created and the trail that the planner choose to reach the goal pose.

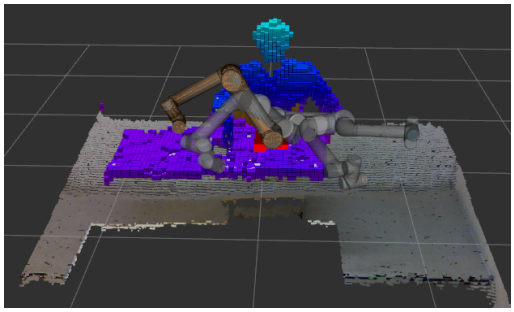

Fig. 9. Path planning using the RRT algorithm.

At the end of each path planning performed by the algorithms through MoveIt!, the time and amount of states (or poses) were used to find a route to the final pose. These data were collected and analyzed in graph format, as shown in Figure 10. Although both algorithms find a certain amount of states to perform the path planning, it is not necessary to use all found states.

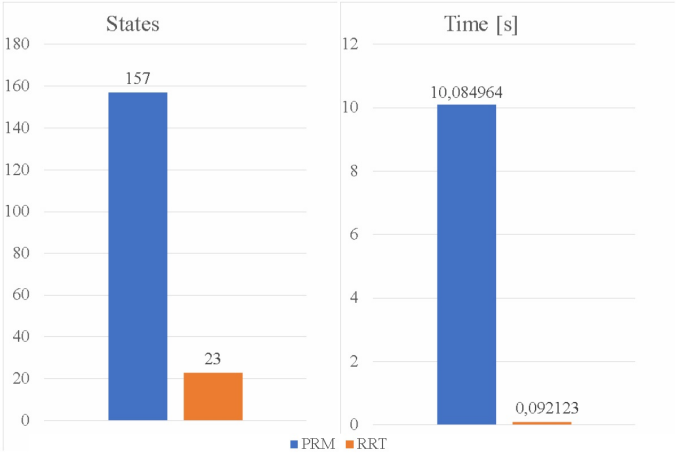

Fig. 10. Results of logs generated by the system.

The logs presented in graph format show that the algorithm PRM use all the time that was configured to find a path planning. This has resulted in a higher value of pose states. However, the RRT algorithm uses less time to find a path planning solution, resulting in smaller amounts of pose states.

6 Conclusion and Future Work

In the presented paper a collaborative manipulator and two path planning algorithms were stressed and compared allowing to develop a system that helps and collaborates with humans, according to the new paradigms of Industry 4.0. The system uses ROS and a RGB-Depth sensor (Kinect) to acquire the environment such as objects and humans positions. The implemented system allows to re-plan the movement avoiding collisions while guaranteeing the execution of operations. The logs generated by the system, show a difference between the analyzed algorithms. While RRT uses agility in finding a solution to plan the path to the goal pose, the PRM uses all the time it has been assigned to find a trajectory. Therefore the use of these algorithms must be adequate to the objective of the project in which it is implemented. The simulation of an UR5 robot with acquired point cloud validates the approach of both algorithms. In future works, it will be possible to optimize the point cloud for the system to have a faster response to the introduction of new objects in the working environment of the UR5 robot.

Acknowledgment

Project "TEC4Growth - Pervasive Intelligence, Enhancers and Proofs of Concept with Industrial Impact/NORTE-01-0145-FEDER-000020" is financed by the North Portugal Regional Operational. Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, and through the European Regional Development Fund (ERDF).

This work is also financed by the ERDF – European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme within project POCI-01-0145-FEDER-006961, and by National Funds through the FCT – Fundaçao para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) as part of project UID/EEA/50014/2013.

References

1. Barraquand J, Latombe JC (1991) Robot motion planning. a distributed representation approach, International Journal of Robotics Research 10(6), 628–649.
2. Ralli E, Hirzinger G (1994) Fast path planning for robot manipulators using numerical potential fields in the configuration space, Vol. 3, pp. 1922-1929.
3. Kavraki L, Svestka P, Latombe JC, Overmars M (1996) Probabilistic roadmaps for path planning in high dimensional configuration spaces. IEEE Transactions on Robotics and Automation, Vol. 12, No. 4, pp. 566–580, ISSN 1042-296X
4. Siméon T, Laumond JP, Nissoux C (2000) Visibility based probabilistic roadmaps for motion planning . Advanced Robotics 14(6): 477–494.
5. Wilmarth S, Amato N, Stiller P, Maprm (1999) A probabilistic roadmap planner with sampling on the medial axis of the free space, in Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1024–1031.

6. Bohlin R, Kavraki LE (2000) Path Planning Using Lazy PRM, in Proceedings of the IEEE International Conference on Robotics and Automation, San Fransisco, vol. 1, pp. 521–528.
7. Plaku E, Bekris KE, Chen BY, Ladd AM, Kavraki LE (2005) Sampling based roadmap of trees for parallel motion planning, IEEE Transactions on Robotics, 21:4, 597–608.
8. Helguera C, Zeghloul S (2000) A local-based method for manipulators path planning in heavy cluttered environments, Proceedings of IEEE International Conference on Robotics and Automation, pp. 3467–3472, San Francisco.
9. Blackmore L, Williams B (2006) Optimal manipulator path planning with obstacles using disjunctive programming. American Control Conference, Minneapolis.
10. Lahouar S, Zeghloul S, Romdhane L (2006) Real-Time Path Planning for Multi-DoF Manipulators in Dynamic Environment, International Journal of Advanced Robotic Systems, Volume: 3 issue: 2.
11. Tavares P, Lima J, Costa P, Moreira AP (2016) "Multiple manipulators path planning using double A*", Industrial Robot: An International Journal, Vol. 43 Issue: 6, pp.657-664, https://doi.org/10.1108/IR-01-2016-0006.
12. Staranowicz A, Brown GR, Morbidi F, Mariottini GL (2014) Easy-to-Use and Accurate Calibration of RGB-D Cameras from Spheres. In: Klette R., Rivera M., Satoh S. (eds) Image and Video Technology. PSIVT 2013. Lecture Notes in Computer Science, vol 8333. Springer, Berlin, Heidelberg.
13. Chitta S (2016) MoveIt!: An Introduction. In: Koubaa A. (eds) Robot Operating System (ROS). Studies in Computational Intelligence, vol 625. Springer, Cham.
14. Walid Darwish W, Tang S, Wenbin L, Chen W (2017) A New Calibration Method for Commercial RGB-D Sensors, Sensors 2017, 17, 1204; doi:10.3390/s17061204.
15. Basso F, Pretto A, Menegatti E (2014) Unsupervised intrinsic and extrinsic calibration of a camera-depth sensor couple, Proceedings of IEEE International Conference on Robotics and Automation (ICRA).
16. Martinez A, Fernández E (2013) Learning ROS for Robotics Programming. Packt Publishing, Birmingham.
17. Joseph L (2015) Mastering ROS for Robotics Programming. Packt Publishing, Birmingham.
18. The Robotic Operation System Wiki, `http://wiki.ros.org/ROS/Tutorials/UnderstandingTopics`
19. ROS Industrial Training Exercises with version Kinetic, `https://github.com/ros-industrial/industrial_training/wiki`
20. Sensor supported by ROS, `http://wiki.ros.org/Sensors`
21. Sucan IA, Chitta S, "MoveIt!", [Online] Available:, `http://moveit.ros.org`
22. Sucan IA, Chitta S, "MoveIt!", [Online] Available:, `http://picknik.io/moveit_wiki/index.php?title=High-level_Overview_Diagram`
23. Sucan IA, Moll M, Kavraki LE (2012), The Open Motion Planning Library, IEEE Robotics & Automation Magazine, [Online] Available:, `http://ompl.kavrakilab.org`