ICONIET 2018

Proceedings of the International Conference on Innovation, Entrepreneurship and Technology,
30-31 October 2018, BSD City, Indonesia,
ISSN: 2477-1538

3D Path Planning for Quadrotor Using Gazebo Simulator

Mohammad Ryan Dirgantara[a*], Eka Budiarto[a], Rusman Rusyadi.[a]

[a]*Swiss German University, Indonesia*

[*]ryandirgantara14@gmail.com

Abstract: This research explores the 3D path planning for quadrotor, which is an unmanned aerial vehicle (UAV) with four rotors. The quadrotor is simulated using robot operating system (ROS) and Gazebo software, and is equipped with camera, GPS sensor, and inertial measurement unit (IMU) sensor to do the mapping of its environment. The packages used in ROS are Hector Quadrotor package, joystick package, Octomap package, and MoveIt package. These packages were modified so that it could be integrated with each other and fulfill the objective of this research. For the 3D path planning, a method called rapidly-exploring random tree (RRT) is explored and implemented. Several experiments regarding the behavior of the quadrotor, the mapping, and the path planning were conducted to find out the performance and limitations of the simulation. This simulation is set up so that it can be used to validate a new design of quadrotor before it is tested with a physical prototype.

Keywords: Quadrotor, robot operating system (ROS), Gazebo simulator, rapidly-exploring random tree (RRT)

1. Introduction

Quadcopter has mostly been used by ungraduated students as a research platform where students tried to implement their newly innovated ideas to a prototyped quadcopter. Quadrotor has quite a complex movement on its own and it relates to swift maneuver in unpredictable and tempestuous weather, a slight mistake on the controller would cause a very high chance for an unintentional sharp movement to occur. To do a trial and error process directly to the quadcopter is not recommended due to the high risk involved, therefore the need to do a simulation first is highly advisable as it saves the production cost, reduce the risk of the quadcopter from being damaged, and would hopefully saves time during troubleshooting. The ideas that is proposed is a simple path planning, the path planning algorithm will then be integrated with the simulated quadrotor to see whether it is possible to be used on a real quadrotor or not.

2. Literature Survey

2.1. Software

The software that is being used for this research is ROS Kinetic and Gazebo ver 7.x which was run using Ubuntu Linux 16.04 as the operating system. The simulation can only be run using a specific Robotic Operating System (ROS) version, ROS Kinetic, as some of the packages cannot be compiled in a diff erent version. The quadcopter simulation compiled in ROS will then be simulated using Gazebo.

ROS has a very handy tool which ease user in terms of creating or developing robots. ROS can be fully used even without graphic user interface. ROS has Robot visualization, RViz, whose function is to visualize the reading of the sensor used in the robot, ROS Qt. rqt. Which provide a Qt based framework mainly used in the creation graphical user interface, Unified Robot Description Format, URDF, which provide an XML representation of the model for the robot (Saputra, 2017).

Gazebo is an open source Robotic simulator which is quite well known throughout the en-gineering society. Gazebo is used since it is resourceful and has many users supports compared to other robotic simulator. It has a lot of plug-in or built-in software specifically chosen for designing and simulating purposes. The format used in Gazebo, SDF, is quite a stable format who is capable enough of depicting and characterized a robot models in the surrounding envi-ronment. It also has the ability to export design from another source such as Blender or Catia, A SketchUp with STL or Collada format.

2.2. Mapping

Mapping in terms of robotics application means the ability of a robot to localize itself in an unknow environment and construct a map of its surrounding. SLAM, which stands for Simul-taneous Localization and Mapping, has been used a lot by robot developer to map an unknown environment since its logarithm can estimate the position from scan matching and combine it with the odometry of the robot. The mapping package that is used for this research is called Octomap as it has been proven that it can map the environment in all 3 axis which is very essential for 3D path planning.

Octomap has been considered as the sate of the art most optimal method for a 3D mapping representation. Octomap recursively decomposes the environment spaces into cubes using an octree and it is represented by an octal tree. The entire observable environment will be divided into eight volumes in which each cube will have the same size of voxels. Occupied space will be represented with a cube and each cube has a diff erent color defining the diff erence in height occupied in the local reference frame. Each cube can be divided recursively into other eight smaller cube and so on, this technique enabled the space to be subdivided up to the desired precision in which every step of recursion it will create a smaller voxel to represent the space and will only stop when size of the volume reaches the minimum size established a priori (Hornung et al., 2013).

2.3. Path Planning

Path planning in robotics is a process where a robot finds the shortest path or the most efficient and optimum ways to get into the intended location between many routes or obstacle. Depend-ing by the dimension of a robot, path planning can be diff erentiated into two types which is 2D path planning and 3D path planning. Since quadrotor can moves in all 3 axes and 6 degree of freedom the suitable path planning would be 3D path planning. Most 3D path planning used a sample-based algorithm as the path planning algorithm, a pre-known information of the whole workspace area (mathematic representation to describe the workspace) will be needed for sampling-based method. Sampling based algorithm sampled the environment as a set of node or cell, then it searches the node randomly till it finally obtains a feasible path (Yang et al., 2016).

RRT will be used as the main path planning algorithm for quadrotor need to have a 3D oriented path and RRT can visualize planning concept in more than two dimensions so it fits with the criteria, and also it is quite easy to implement. According to the pseudo code of RRT, it will start with a search for an empty tree. Then add an initial location (configuration/node) to the search tree. When the tree is searching for the goal, pick a goal location (configuration/node) and the location is usually notated as q_r. Find a vertex in the search tree which is the closest to that random point q_n and add a path in the tree between all q_r and q_n while making sure there will be no collision occurred somewhere in the link.

3. Methodology

3.1. Robotics Operating System Packages

These are the packages used in ROS to fulfil this thesis objective which is to create the 3D path planning simulation:

1. Hector quadrotor package: Contains a mechanical design/URDF of the quadrotor, plugins for the aerodynamic and propulsion which complement the design, dynamical modelling to simulate the behavior of the quadrotor, and a program to control the quadrotor
2. Joy package: Enabled ROS to read data of the joystick from the user

3. MoveIt! Package: it incorporates the latest advance motion planning, manipulation, 3D perception, kinematics, control, and navigatio (Sucan and Chitta, 2016a).
4. Octomap package: Provide mapping algorithm in which its library can implement a 3D occupancy grid mapping
5. Action Controller package: translate geometrical trajectory created by MoveIt! to velocity commands(cmd_vel)

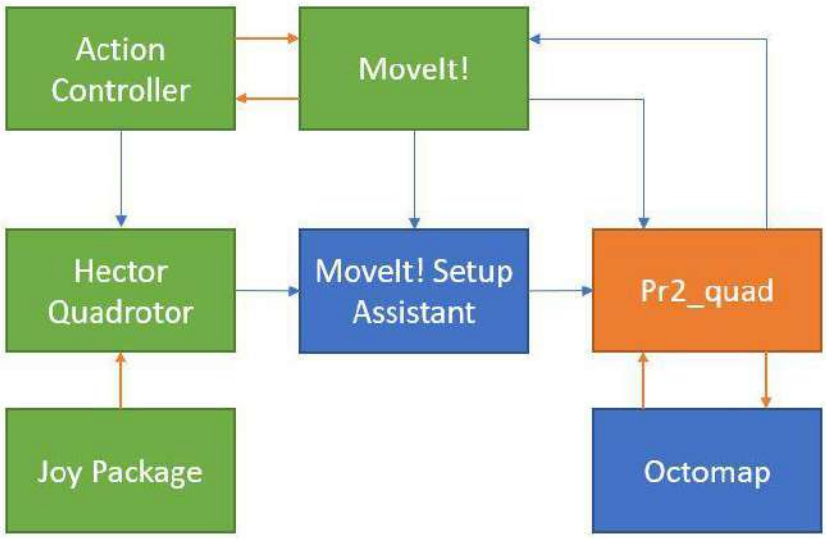

Figure 3.1: Packages Connection.

The relationship between each package can be seen in figure 3.1. The green box indicates the modified packages, the blue box indicates the unmodified packages while the orange box indicates the generated package. Hector quadrotor can interact with MoveIt! once it has been setup by the setup assistant, it will the generate a package which was named 'pr2_quad'. The launch file will then be created in the 'pr2_quad' package which will integrate all the packages. The joy package is connected to hector quadrotor package to move the quadcopter around the area to do the mapping first. Octomap package subscribe the reading from the sensor in the 'pr2_quad' package and publish the mapping result. MoveIt! will then subscribe the movement constraint of the quadrotor and the result from the mapping, it will then generate the path planning trajectories and publish a topic to action controller so that the quadcopter can execute the trajectories created by the integrated path planning algorithm in MoveIt!. Action controller convert this topic so that hector quadrotor can 'understand' the topic and move according to the trajectories.

4. Results and Discussion

Ayush Gaud, an engineer from India has implemented RRT path planning to his own simulated quadrotor without using MoveIt!, however there were no documentation so the result couldn't be compared (Gaud, 2016). Wil Selby and Alessio Tonioni has both implement RRT path planning to their own simulated quadrotor using MoveIt! but in a diff erent ROS version, they both provides an excellent overview on how to integrate quadcopter with MoveIt! however there were no data results shown on the documentation (Selby, 2015c; Tonioni, 2015). So the result from the path planning in the simulated quadrotor discussed later on this paper will not be compared to other existing competition approaches.

4.1. Path Trajectories

Path planning process can be initiated through the RVIZ. The trajectories will be generated once the goal query has been placed around the map within the workspace area. Once the goal query and the start query has been declared select the planning button in the planning tab. During initial position, the whole environment has yet to be mapped, so when a path is requested it will generate the path sccording to the recorded map. As it can be seen in on the left side of figure 4.1, a path was generated even though there

Proceedings of the International Conference on Innovation, Entrepreneurship and Technology,
30-31 October 2018, BSD City, Indonesia,
ISSN: 2477-1538

is a wall there, but since the sensor only read half of the wall, the path planning algorithm does not consider the other half part of the wall. The left side of figure 4.1 shows the reading of the environment while the right side of figure 4.1 shows the actual wall standing in front of the quadrotor.

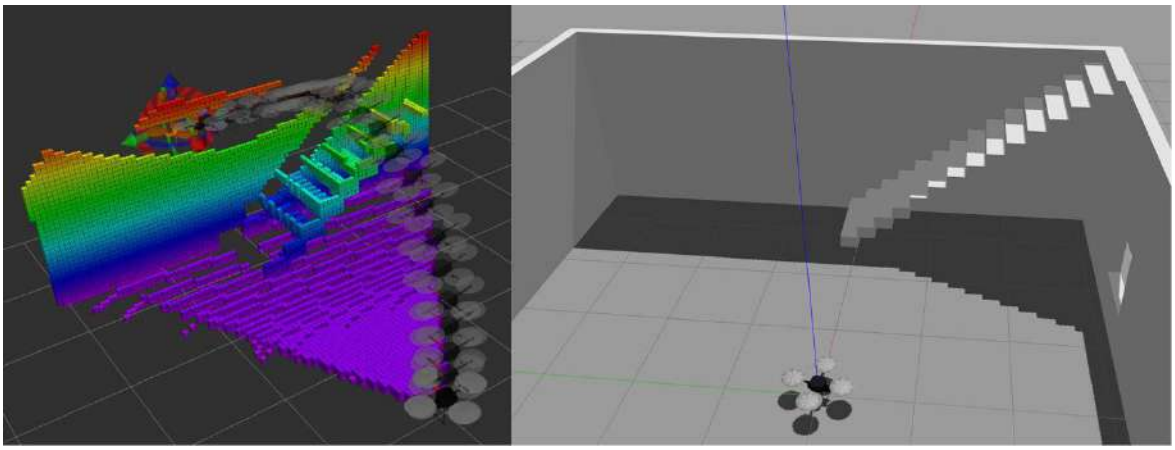

Figure 4.1: Reading of the environment by the quadrotor vs the actual environment

4.2. Path Planning Execution

The values shown by the topic move_gr oup•g oal is the position of the goal set by the user, this result will then be compared to the actual position of the quadrotor to test the accuracy of the path planning. A percentage of error will then be calculated by finding the diff erence between the goal and actual final position of the quadrotor. The position of the quadrotor can be obtain through the topic •gr ound_trut h_to_t f •pose. The test for the execution will be done three times. The first try would be done during the initial position (stationary position), second try would be done when the quadrotor is hovering, third try would be during hovering and positioned the goal state in a narrow area. Due to the computer limitation (incapable of mapping the whole map, the specification of the computer is not strong enough to contain the data from the mapping), the whole test were done without fully map the environment first. The path executed for each trial can be seen in figuree 4.2.

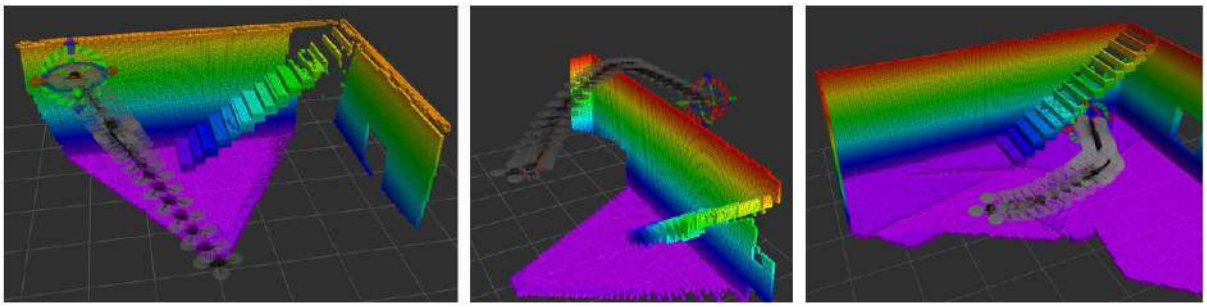

Figure 4.2: First Trial Path Execution vs Second Trial Path Execution vs Third Trial Path Execution.

Table 4.1: Percentage of error for all trial in all 3 translational axis and 3 rotational axis.

	First Trial Percentage of Error	Second Trial Percentage of Error	Third Trial Percentage of Error
Position in X axis	19.12%	4.60%	4.90%
Position in Y axis	0.47%	77.69%	14.60%
Position in Z axis	0.11%	16.00%	0.36%
Orientation in X axis	0.60%	1.20%	2.20%
Orientation in Y axis	0.68%	3.80%	5.50%
Orientation in Z axis	9.09%	12.00%	10.00%

As it can be seen in from table 4.1, the quadrotor has better accuracy on executing the path planning algorithm when it is first in stationary position although it has a high percentage of error in the X axis position. On the second and third trial, the quadcopter has a huge off set in terms of position in the Y axis. The accuracy of the orientation of the quadrotor for all three trials are arguably quite high in the X and Y axis as it only gives an error of less 6% although for some reason the quadrotor was not able to deliver a satisfactory result in the Z axis orientation with an error of 9 to 12%. Although the path planning algorithm has huge error in either X or Y axis, the quadrotor was able to move to the goal position and avoid obstacle by following the path generated.

5. Conclusion

In conclusion the path planning works best during the stationary position. The quadrotor drifts slowly due to numerical inaccuracies in Gazebo's physics engine, the quadrotor slowly rotate clockwise with Z axis as its pivot. The quadrotor rotate due to noise created by the GPS and IMU in the URDF. The quadrotor slowly rotate with an increase velocity of the rotation. When the path is executed there will be extra angular velocity causing it to drift a slightly from the intended goal which explains why in all trial there were no 0% of error and the worst result were obtain when the quadrotor moves from a hovering position since when it hovers the noise becomes worst causing to drift more.

REFERENCES

Cai, G., Chen, B. M., and Lee, T. H. (2011). *Unmanned rotorcraft systems*. Springer.

Conley, K. (2012). Nodes.

Dirgantara, M. R. (2018). *VALIDATING THE MATHEMATICAL MODELLING ON THE HECTOR QUADROTOR IN ROS AND CREATING THE 3D PATH PLANNING SIMULATION USING GAZEBO SIMULATOR*. Swiss German University.

Fatan, M. and Lavi, B. (2013). An adaptive neuro pid for controlling the altitude of quadcopter robot.

Foote, T. (2013). tf: The transform library. In *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, Open-Source Software workshop, pages 1–6.

Gaud, A. (2016). Quadcopter path planning using rrt* and minimum jerk trajectory generation.

Gibiansky, A. (2012). Andrew gibiansky,math.

Hoff mann, G., Rajnarayan, D. G., Waslander, S. L., Dostal, D., Jang, J. S., and Tomlin, C. J. (2004). The stanford testbed of autonomous rotorcraft for multi agent control (starmac). In *The 23rd Digital Avionics Systems Conference (IEEE Cat. No.04CH37576)*, volume 2, pages 12.E.4–121–10 Vol.2.

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*. Software available at http://octomap.github.com.

J., A. K. and Tore, H. (1995). *PID controllers*. Instrument Society of America.

Jarvis, R. A. and Zelinsky, A. (2003). *Robotics research: the tenth international symposium*.

Springer.

Liang, O. (2016). Quadcopter forum archive.

Luis, R. G. C., Alejandro, E. D. L., Lozano, R., and Claude, P. (2013). *Quad Rotorcraft Control Vision-Based Hovering and Navigation*. Springer London.

MAYADAG, A. (2015). *ROS BASED PROGRAMMING AND VISUALIZATION OF QUADRO-TOR HELICOPTERS*. ISTANBUL TECHNICAL UNIVERSITY.

Meyer, J. (2012). Simulated quadrotor is slowly drifting on its own [closed] edit.

Meyer, J. and Kohlbrecher, S. (2011a). Wiki.

Meyer, J. and Kohlbrecher, S. (2011b). Wiki.

Meyer, J., Sendobry, A., Kohlbrecher, S., Klingauf, U., and Stryk, O. V. (2012). *Simulation, Modeling, and Programming for Autonomous Robots Lecture Notes in Computer Science*, page 400–411.

Meyer-Delius, D., Beinhofer, M., and Burgard, W. (2012). Occupancy grid models for robot mapping in changing environments. 3:2024–2030.

Moll, M., ßucan, I. A., and Kavraki, L. E. (2015). Benchmarking motion planning algorithms: An extensible infrastructure for analysis and visualization. IEEE Robotics & Automation Magazine, 22(3):96–102.

Noreen, I., Khan, A., and Habib, Z. (2016). Optimal path planning using rrt* based approaches: A survey and future directions. International Journal of Advanced Computer Science and Applications, 7(11).

Pestana, J., Sanchez-Lopez, J. L., Puente, P. D. L., Carrio, A., and Campoy, P. (2014). A vision-based quadrotor swarm for the participation in the 2013 international micro air vehicle competition. 2014 International Conference on Unmanned Aircraft Systems (ICUAS).

Pradeep, V. and Marder-Eppstein, E. (2009). actionlib.

Raj, A. (2015). Ros 101 : The action server – client model.

Saputra, A. D. (2017). Path Planning Simulation for Quadcopter Using Gazebo Simulator. Swiss German University.

Selby, W. (2015a). Quadrotor control system design - position, attitude, and motor control.

Selby, W. (2015b). Quadrotor system modeling - non-linear equations of motion.

Selby, W. (2015c). Ros integration - quadrotor 3d mapping and navigation.

Sucan, I. A. and Chitta, S. (2016a). Concepts.

Sucan, I. A. and Chitta, S. (2016b). Moveit! motion planning framework.

Thomas, D. (2017). Ros msg.

Tonioni, A. (2015). A simple autopilot for a quadrotor realized using moveit!.

WikiRos (2012). Wiki.

Wu, H.-H. and Bainbridge-Smith, A. (2018). Advantages of using a kinect camera in various applications.

Yang, L., Qi, J., Song, D., Xiao, J., Han, J., and Xia, Y. (2016). Survey of robot 3d path planning algorithms. 2016:1–22.

Young, W. R. (1987). The helicopters. Time-Life Books.