BLG-456E - Robotics

AN EARTQUKE RESEARCH AND RESCUE ROBOT: MAPPING AND DODGING INCOMING OBJECTS WITH A QUADCOPTER

13343 **150150701** Yunus Göngör

13343 **150120123** Ali Rıza Salihoğlu

13343 **150130066** Duhan Cem Karagöz

13343 **150110031** Kerem Er

K1000

Damien J. Duff

Table of contents

1. Abstract

This report explains problem and solution for BLG456E project completed by team K100 in 2017. Our team implemented a discovery drone that avoids incoming objects for research and rescue missions in earthquake zones. However, our team did not have enough time to combine all the modules and project was not fully completed. Each module is implemented properly.

2. The Problem

In this project we aimed to solve the problem of discovery on an area after earthquake with drones. In this problem, biggest issue is incoming or falling objects on the air. We have created a simulated drone for mapping an area and avoiding incoming objects.

Drones are flying robots that can achieve high speeds and altitudes. Also, drones can be very small. But when a drone meets an unpredicted incoming object on the air (in this case, a rock or another unstable piece), a collision might happen. This collision might prevent drone's mission, fling abilities or cause even more damage if other object is also another drone or a plane or a bird or a human. This issue has been researched before and has a lot of solutions produced. Massachusetts Institute of Technology, Computer Science & Artificial Intelligence Lab; Queen Mary University of London, Centre for Intelligent Sensing are only some of groups that considered this problem. MIT Computer Science & Artificial Intelligence Lab has been published drake-crazyflie-tools on github and Poiesi and Cavallaro from Centre for Intelligent Sensing published an algorithm called "Detection of fast incoming objects with a moving camera" (2016).

Our aim in this project is to build a simulated drone that, discovers an area, and avoids incoming objects. However, we have kept our application limited to be able to implement.

3. The Solution

Our team implemented a robot that travels all map without any setbacks caused debris. Best robot suits our needs is a quadcopter, therefore we have decided use hector drone in ROS (robotic operating system). To see and detect our environment we used quadcopter's camera. And mapping of the map is done via a slam module using lidar on the quadcopter.

3.1 Design

This solution includes two different parts, first part is to path planning and mapping. Second part is to avoiding incoming objects. Path planning is planned to be completed using map data and focused on discovering unmapped places. Mapping is planned to be completed using lidar on quadcopter. Camera on quadcopter used to avoid incoming objects. All modules connected with a path follower. Path follower planned to manage quadcopter on planned path and avoid incoming objects using rapid moves. Rapid moves planned according to incoming object vector from object detector module and surroundings from map data. Object detector module planned to use simple image processing to detect an object and create an object vector. Source image is coming from camera on quadcopter.

3.2 Technical Details

This project includes a quadcopter called hector. Hector quadcopter has different modules that runs on ROS (robotic operating system). Those modules referenced in the references section of this report. Our team have modified those modules and added extra modules to implement designed solution.

Mapping module in other words SLAM module on the hector uses various inputs to locate itself on the map while mapping the surrounding area. SLAM does that using lidar on the quadcopter. SLAM

takes the data from lidar topic transforms it according to tf to match the world frame not the quadcopter than SLAM adds the result to rviz so we can visually see and understand the world. Also SLAM keeps track of the movements of the quadcopter so it knows where it currently is. Team tested the SLAM module on many levels most of them are successful. SLAM module fails when quadcopter makes contact with any walls or flies too high above the map level.

Path follower module written by Yunus Güngör since there is no working path follower module for hector quadrotor or our team could not run existing modules. Also, a modification for rapid movement would be necessary. Implementing the module considering that saved some time. This module is called hector_follower and can be run with ` rosrun hector_follower follow.py` command. Consider that, this module requests each point of the path published one by one. And this module is not a `smart` module. If there is an obstacle on the path, quadrotor will not be able to follow path and a new path will be required. Since our design has a separate path planning phase, this is not a big issue for our implementation. Points of path expected to be published on `/waypoint_cmd` topic on ROS. This module also has an interrupt like routine for dodging incoming objects. A rapid move planned according to the map data when an incoming object vector is published on `/incoming' topic. And this rapid move executed quickly. Then quadrotor keeps following the path. When a new vector appeared, process of following path stops and does not continue until rapid movement executed. After execution path following counties.

Exploring the world and path planning for return base is done with hector exploration planner package under hector navigation. Before this one we tried other navigation, packages but could not be able to get successful results from them. To use plan the return path we called makePlan function with parameters robots current pose as starting pose and 0,0,0 point as goal pose. Both exploring world and path planning to return base publish their data to /explore_path topic. Hector exploration planner looks for spaces to explore around itself and plans a path and then publishes it to the /explore_path topic. Also, it can plan a path to a given goal position such as 0,0,0 for returning base. It retrives robots position and calculates costmap and plans the path to the goal pose. Then it publishes the path to the /explore_path topic.

3.3 Evaluation

This project has some unsolved issues. Mapping is complete, and drone maps the environment with no problem. Mapping is done using slam module on the quadcopter therefore error is dependent on noise coming from simulation and laser itself. Exploration, path planning for returning to base and following the path is being done successfully by the quadcopter. Also creating objects with random places is done but writing a client for throwing object could not be done because we encountered several errors while following the tutorial on ROS wiki about creating clients. Thus, detecting and avoiding them could not be tested.

After brief analysis of the original proposal in the light of the final project we do learned to research more and manage time in a more sufficient way.

Solution is not applicable to every situation. Our team implemented solution with limited capabilities to be able to complete the project. Map data might get corrupted while dogging objects due to tilt of lidar sensor, but map fixes itself overtime. So, this problem only causes slow mapping and can be fixed with further versions. Furthermore, any contact with the walls or ceiling of the map will corrupt the map data leading to errors. Also, project was not fully tested so there might be issues we are not aware of.

4. Implementation

Solution implemented in virtual environment of robotic operating system. Gazebo, referenced hector modules and RViz used for implementation. However, RViz is not necessary for system to work. RViz used for development and debugging purposes. Our solution implemented using python and C++.

An installation is required before executing solution. Install the project source code to the ROS workspace folder then use `catkin_make` command to compile the project. After that “roslaunch hector_quadrotor_demo indoor_slam_gazebo.launch “ command will call base of the project and start the main components. “rosservice call /enable_motors true” service call will enable the motors to operate. If motors are not enabled, quadcopter will not be able to move even if all modules work correctly. After that path planning module should be called. Then calling path follower module with `rosrun hector_follower follow.py` command will set quadcopter in motion. We can interrupt the process with throwing an object to the quadcopter. Exploring can be started with rosrun hector_exploration_controller simple_exploration_controller ‘rosrun hector_exploration_controller simple_exploration_controller’ and path planning for returning base can be started with ‘rosrun hector_exploration_node exploration_planner_node’ commands

References

“Detection of fast incoming objects with a moving camera”, Fabio Poiesi, Andrea Cavallaro, (2016)

https://github.com/blandry/crazyflie-tools

https://github.com/tu-darmstadt-ros-pkg/hector_quadrotor, Technische Universität Darmstadt

https://github.com/tu-darmstadt-ros-pkg/hector_localization, Technische Universität Darmstadt

https://github.com/tu-darmstadt-ros-pkg/hector_gazebo, Technische Universität Darmstadt

https://github.com/tu-darmstadt-ros-pkg/hector_models, Technische Universität Darmstadt

https://github.com/tu-darmstadt-ros-pkg/hector_slam, Technische Universität Darmstadt

https://github.com/tu-darmstadt-ros-pkg/hector_navigation, Technische Universität Darmstadt