Universiteit Antwerpen

Progress Report Week 3

Michiel Aernouts
michiel.aernouts@student.uantwerpen.be

May 20, 2017

Abstract

In order to save time and reduce the risk of crashing the ErleCopter, all algorithms should be tested in a simulator before implementing them. For this purpose, we will work with Gazebo. We research multiple SLAM algorithms such as 6D SLAM [1], RGB-D SLAM [2], OctoSLAM [3] and LSD SLAM [4] to map an indoor environment. The data that was gathered using SLAM can be stored in an OctoMap [5].

1 Progress

1.1 Octomap research

OctoMap [5] is a memory-efficient mapping approach based on an octree data structure using probabilistic occupancy estimation. This approach can be useful for our research, as it is important to acquire an accurate 3D map of an environment.
Figure 1 shows a mindmap that I made based on the paper written by Hornung et al [5].
Unfortunately, I could not spend as much time as I hoped on researching and debugging the OctoMap source code, partly because there were a few difficulties when building the source code on my machine. Apparently, the problem on my machine was that I was running Qt5, while the Octomap distribution that is available at `https://github.com/OctoMap/octomap` requires the use of Qt4. Installation instructions for Qt4 can be found here..

EDIT: Another method to build OctoMap is to open the project with CLion and build it from there.

I managed to look at the first example using the 'octovis' tool. In order to to this, I executed:

- `cd ~/projects/octomap/octomap/share/data`

- `octovis geb079.bt`

Next week I will focus on debugging the OctoMap source code and analyzing the examples.

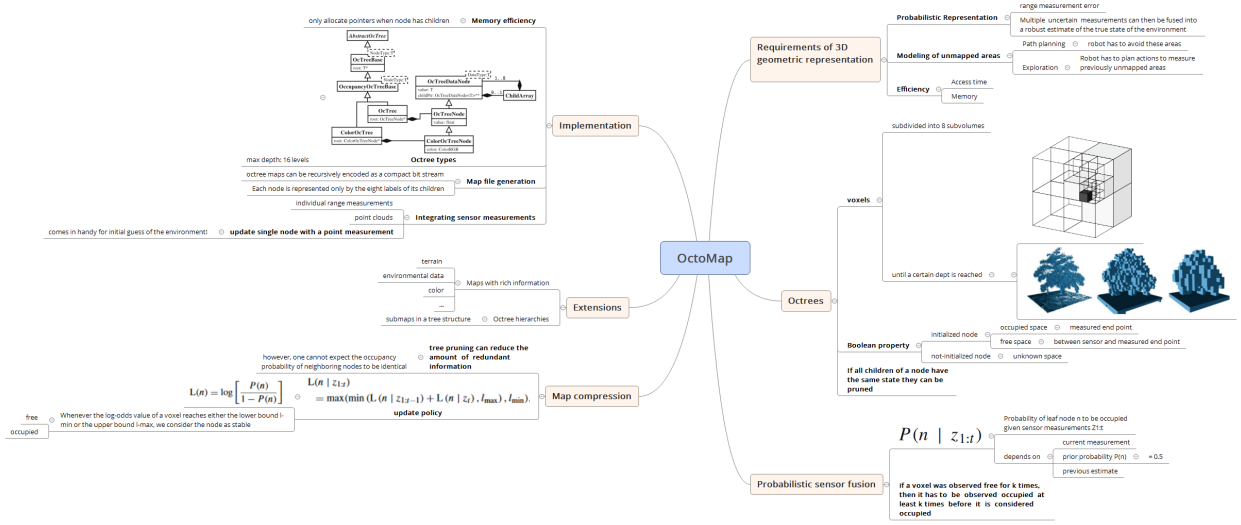

Figure 1: OctoMap

1.2 Erle-Copter calibration

As mentioned in last week's progress report, there were some problems flying the Erle-Copter. Take off was pretty unstable, causing the drone to topple most of the time.
This week, I fixed this problem by applying the following solutions:

- I removed the carbon landing gear to lower the center of gravity.
- I calibrated the Erlecopter using APM Planner tutorials.
- Using APM Planner, I was able to map some flight modes to the RC switches. The instructions to to this can be found here.

The flight of the Erle-Copter is now stable. Next, I will try to implement a fixed trajectory using mavros. This script will be tested with Gazebo, and then on the Erle-Copter itself.

1.3 Rviz

Rviz is a ROS tool that allows us to visualize certain ROS-nodes, such as the odometry or the laser range data of the Erle-Copter.

Visualizing the odometry was no problem, but with the laser range data it did not seem to work at first. Eventually, I was able to create a map of the simulation environment by applying this tutorial on the Erle-Copter.

1.4 Automatic trajectory in simulation

Based on the `ros_erle_takeoff_land` example project, I was able to script linear movements for the Erle-Copter. This script was successfully tested in Gazebo. The test can be recreated by copying the code below under the *'DO STUFF'* section in the `ros_erle_takeoff_land` example.
Unfortunately, I am not yet able to control yaw rotation. I will try to get this working next week. When this works, it might be interesting to write an API that sends position and velocity commands to the ErleCopter.

```
1  sleep(4);
2  //pose publisher
3  ros::Publisher local_pos_pub = n.advertise<geometry_msgs::
       PoseStamped>"mavros/setpoint_position/local",10);
4  geometry_msgs::PoseStamped pose;
5
6  //9 metres forward
7  pose.pose.position.x = 0;
8  pose.pose.position.y = -9;
9  pose.pose.position.z = srv_takeoff.request.altitude;
10 ROS_INFO("Going␣forward␣9␣metres");
11 for(int i = 100; ros::ok() && i > 0; --i){
12         local_pos_pub.publish(pose);
13         ros::spinOnce();
14         r.sleep();
15     }
16
17 sleep(3);
18
19 //5 metres left
20 pose.pose.position.x = 5;
21 pose.pose.position.y = pose.pose.position.y;
22 pose.pose.position.z = srv_takeoff.request.altitude;
23 ROS_INFO("Going␣left␣5␣metres");
24 for(int i = 100; ros::ok() && i > 0; --i){
25         local_pos_pub.publish(pose);
26         ros::spinOnce();
```

```
27          r.sleep();
28      }
29  sleep(3);
```

1.5 Propeller guards

The propeller guards are ready to print. I created 2 models, one for the black legs and one for the yellow legs. This was necessary because the bottom of the black and yellow legs did not have the same dimensions.

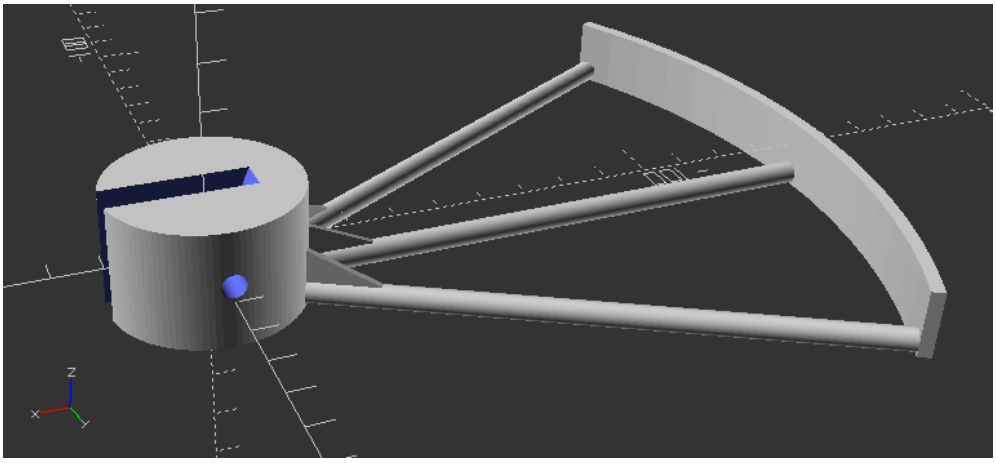

Figure 2: Propeller guard for the ErleCopter

2 Planning week 4

- Continue OctoMap research and debugging
- Test SLAM in simulation
- Complete mavros script
  - Control yaw rotation
  - Create dynamic API to send position and velocity commands
  - Create a GUI. Use `http://wiki.ros.org/rqt`?
- Print propeller guards

References

[1] Andreas Nüchter, Kai Lingemann, Joachim Hertzberg, and Hartmut Surmann. 6D {SLAM}{\textemdash}{3D} mapping outdoor environments. *Journal of Field Robotics*, 24(8-9):699–722, 2007.

[2] Felix Endres, Jurgen Hess, Nikolas Engelhard, Jurgen Sturm, Daniel Cremers, and Wolfram Burgard. An evaluation of the {RGB}-D {SLAM} system. *2012 {IEEE} International Conference on Robotics and Automation*, 2012.

[3] Joscha Fossel, Daniel Hennes, Daniel Claes, Sjriek Alers, and Karl Tuyls. {OctoSLAM}: A {3D} mapping approach to situational awareness of unmanned aerial vehicles. *2013 International Conference on Unmanned Aircraft Systems ({ICUAS})*, 2013.

[4] Jakob Engel, Jorg Stuckler, and Daniel Cremers. Large-scale direct {SLAM} with stereo cameras. *2015 {IEEE}/{RSJ} International Conference on Intelligent Robots and Systems ({IROS})*, 9 2015.

[5] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. {OctoMap}: an efficient probabilistic {3D} mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, 2 2013.