Universiteit Antwerpen

Progress Report Week 4

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. The drones' trajectory can be scripted using mavros. 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

I examine the OctoMap source code using CLion. This IDE can be downloaded here. The installation instructions are simple:

- Unpack the CLion-*.tar.gz file: `tar xfz CLion-*.tar.gz`
- Run CLion.sh from the bin subdirectory

Next, start a new project in CLion by clicking on *'Checkout from Version Control'* → *'Git'* and fill in the GitHub repository URL: `https://github.com/OctoMap/octomap.git`. Fill in the local directory and the project name where OctoMap has to be stored.

1.1.1 simple_example.cpp

The OctoMap library comes with some default examples. In order to fully understand the code, I analyzed these examples by changing parameters and debugging. The 'simple_example' works as follows:

```
1  OcTree tree (0.1);  // create empty tree with resolution 0.1
2
3  // insert some measurements of occupied cells
4  for (int x=-20; x<20; x++) {
5    for (int y=-20; y<20; y++) {
6      for (int z=-20; z<20; z++) {
7        point3d endpoint ((float) x*0.05f, (float) y*0.05f, (
             float) z*0.05f);
8        tree.updateNode(endpoint, true); // integrate 'occupied
             ' measurement
9      }
10   }
11 }
12
13   // insert some measurements of free cells
14 for (int x=-30; x<30; x++) {
15   for (int y=-30; y<30; y++) {
16     for (int z=-30; z<30; z++) {
17       point3d endpoint ((float) x*0.02f-1.0f, (float) y*0.02f
             -1.0f, (float) z*0.02f-1.0f);
18       tree.updateNode(endpoint, false);  // integrate 'free'
             measurement
19     }
20   }
21 }
```

1. Create an empty octree with resolution 0.1
2. Create a point cloud `endpoint`. Update the node, with `endpoint` as occupied cells.
3. Create a new point cloud `endpoint`. Update the node, with `endpoint` as free cells.

The resulting octree is then written into the file `simple_tree.bt`, which can be visualized with octovis:

- `~/projects/octomap/octomap/bin`
- `octovis simple_tree.bt`

1.2 Trajectory script

As shown in the previous progress report, I wrote a mavros script to control the Erle-Copter. Unfortunately, I am still only able to execute linear movements, no rotations. In order to solve this, I contacted Erle-Robotics on their forum and via mail. At the end of this week, I was promised to get a response within the next few days, so I will move this topic to next week, awaiting their response.

1.3 LSD-SLAM

1.3.1 Research

The first SLAM algorithm I will test is LSD-SLAM. Based on the paper written by Engel et al [4], I created the mindmap in figure 1.

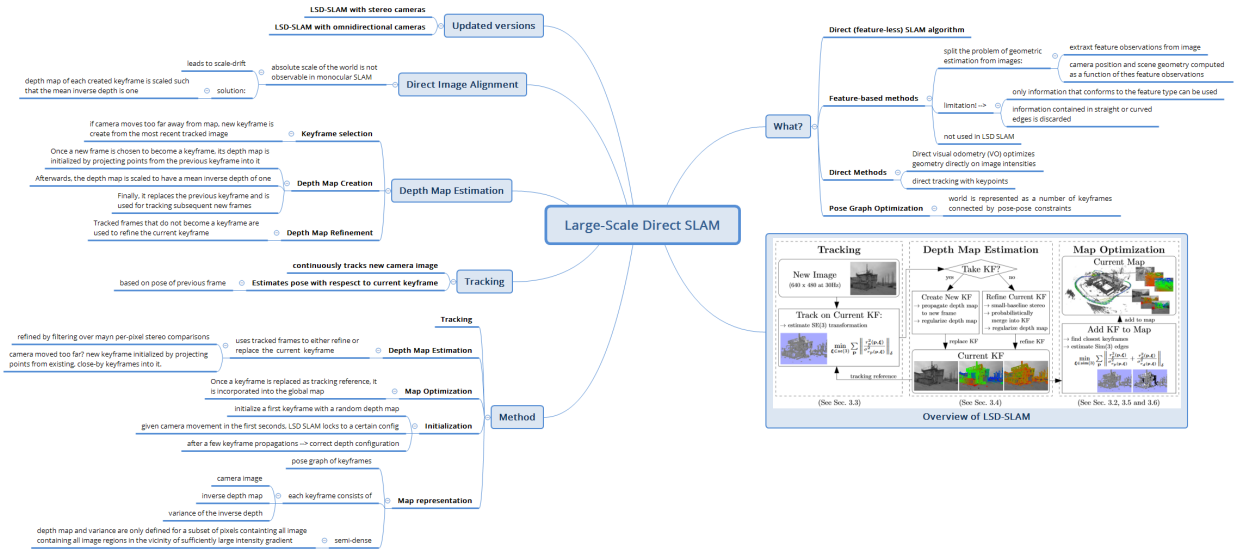


Figure 1: LSD SLAM

Updated versions of LSD SLAM are also available, such as LSD SLAM using omnidirectional cameras [6], and LSD SLAM using stereo cameras [7]. The latter can be interesting for this thesis, as I have access to a stereo camera.

1.3.2 ROS Installation and launch

I successfully implemented LSD-SLAM on my own Gazebo simulation. Installation instructions can be found at `https://github.com/tum-vision/lsd_slam`. In this report, I will give a tutorial on how to recreate this.

- Launch a the Erle-Copter simulation. I configured the `erlecopter.xacro`-file, to just use the generic-camera, as LSD-SLAM only requires a basic RGB-camera.
- Place objects all around the map. **Important:** place lights around the map! These will give more depth for the image. I extended the default Gazebo simulation model, so that it will launch with lights and objects already present on the map.
- Start recording a bagfile using the following command:
  `rosbag record -O lsd_slam_bag /erlecopter/front/image_front_raw /erlecopter/front/camera_front_info`
  - This records the bagfile `lsd_slam_bag.bag` with the topics /erlecopter/front/image_front_raw and /erlecopter/front/camera_front_info
- Launch the `mission1` mavros script to make the Erle-Copter follow a programmed trajectory
- After the trajectory is complete, stop the rosbag recording and the simulation
- Start a new `roscore`
- Run the LSD-SLAM viewer with:
  `rosrun lsd_slam_viewer viewer`
- Start the LSD-SLAM core with the command:
  `rosrun lsd_slam_core live_slam /image:=/erlecopter/front/image_front_raw /camera_info:=/erlecopter/front/camera_front_info`
- cd to the location where you recorded the bagfile and play the bagfile with:
  `rosbag play --clock lsd_slam_bag.bag`
- If the resulting point cloud is not satisfying, you can reconfigure the LSD-SLAM parameters. To do this, run:
  `rosrun rqt_reconfigure rqt_reconfigure`
  After that, it is best to experiment with these two parameters:
  - **KFUsageWeight**: [double] Determines how often keyframes are taken, depending on the overlap to the current keyframe. Larger →more keyframes
  - **KFDistWeight**: [double] Determines how often keyframes are taken, depending on the distance to the current Keyframe. Larger →more keyframes

  In my tests, the best values for these parameters appear to be 4 for KFUsageWeight and 3 for KFDistWeight. A more in depth overview of all parameters can be found at `https://github.com/tum-vision/lsd_slam`.

To ease this procedure and to win some time, I wrote a simple shell script that executes all necessary commands for a successful test.

- Firstly, run `~/scripts/LSD_SLAM_live_demo.sh`
  (Try to run `~/scripts/simulation_launchscripts/launch_mavproxy.sh` first if launching the simulation fails.)
- Wait until the MAVProxy terminal says *'GPS lock at 0 meters'*. Then, open a new terminal and run `rosrun mission2 mission2` to have the Erle-Copter follow a fixed trajectory.
- In the 'Pointcloud viewer', you will see that a point cloud of the environment is generated. You can save this point cloud to a .ply file by typing 'p' in the viewer. The file will be stored in the lsd_slam_viewer folder.

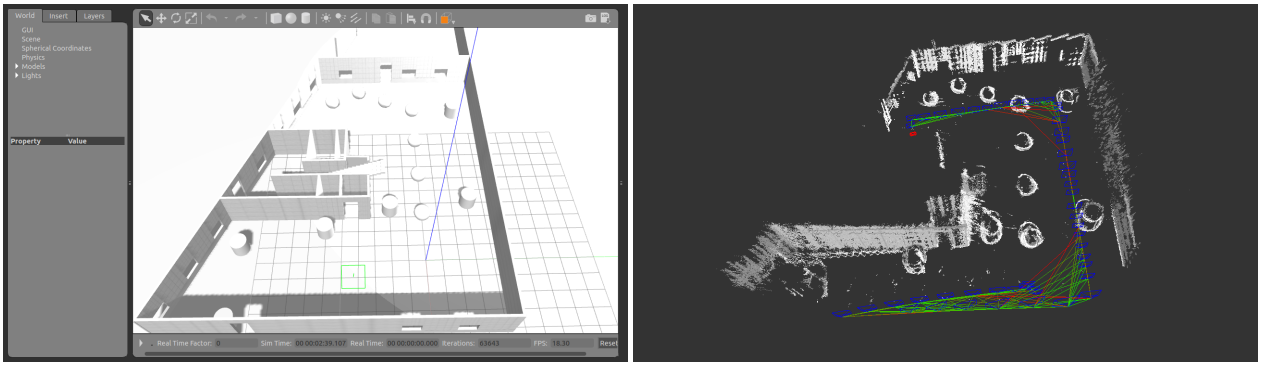
1.3.3 Test results

The simulated Erle-Copter is equipped with a camera that has a 130 degrees field of view and a refresh rate of 80 Hz. I moved the camera a bit forward, so that the propellers would not interfere with the image.
I experimented with different movements to come up with a good trajectory. Apparently, moving backwards too far does not benefit the LSD SLAM algorithm, as the motion tracking will fail pretty quickly. When executing 'mission2', the Erle-Copter follows the following path (expressed in x y z coordinates):

- Go to 0 0 1 (takeoff)
- Go to 10 1 1
- Go to -2 1 1.5
- Go to -2 -11 1.5
- Go to 7 -10 1.5
- Go to 7 -10 0 (land)

These are only linear movements. When I figure out the code how to rotate the Erle-Copter around its z-axis, I will create a new mission that can build a more complete map of the environment.

In figure 2, the result of the LSD SLAM algorithm is visualized. Clearly, the outcome is promising, especially when we take into account the simplicity of the trajectory.



(a) Simulated map in Gazebo (b) LSD SLAM point cloud

Figure 2: Result of LSD SLAM after executing mission2

1.4 Miscellaneous

- Created a custom depth camera for simulation in Gazebo, based on the generic camera module and the kinect camera module. File: `custom_depth_cam.xacro`
- Wrote some simple shell scripts to save time when launching simulations or SLAM algorithms
- Installed Meshlab to visualize point clouds.
- Tested Erle-Copter failsafe. When flying the drone and shutting down the RC:
  - In 'Stabilize' mode, the drone shuts down
  - In 'Alt Hold' mode, the drone remains on the same altitude

  The failsafe can be programmed using APM Planner. I will review this before I will fly the Erle-Copter again.

2 Planning week 5

- SLAM
  - Create a reference bagfile to test all SLAM algorithms with the same variables
  - RGBD-SLAM research and tests
  - OctoSLAM research and tests
- Insert point clouds in an OctoMap
- Yaw rotation with mavros

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, Thomas Schöps, and Daniel Cremers. LSD-SLAM: Large-Scale Direct Monocular SLAM. *Computer Vision ECCV 2014*, pages 834–849, 2014.

[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.

[6] David Caruso, Jakob Engel, and Daniel Cremers. Large-scale direct SLAM for omnidirectional cameras. *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 9 2015.

[7] 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.