Progress Report Week 11

Michiel Aernouts
michiel.aernouts@student.uantwerpen.be

May 20, 2017

Abstract

Optimisation of wireless networks is critical for the localisation of wireless devices. For this purpose, a wave propagation model of the environment can be created. Such a model contains a map of the environment combined with RF measurements that are obtained within that map. In this paper, we compare several visual SLAM algorithms such as LSD SLAM [1] and RGB-D SLAM [2] that can be used to render an accurate 3D map of an indoor environment. In order to test these algorithms, simulation software is used to navigate a drone around a room. A camera that is mounted on the drone provides necessary data for the algorithms. After finishing a SLAM algorithm, the resulting point cloud can be implemented in an OctoMap [3] to generate a volumetric representation.

1 Progress

1.1 Datasets

This week, I had to generate a dataset that can be used for our future research. Because of the dependency problems with the freenect-driver on the Erle-Brain, freenect had to be launched on my laptop. In order to record a dataset with the kinect, i followed these steps:

1. Power on the Erle-Brain and connect the laptop to the erlecopter WiFi network
2. Set the Erle-Brain as ROS master: `export ROS_MASTER_URI=http://10.0.0.1:11311`
3. Set the ROS IP on the laptop to 0.0.0.0: `export ROS_IP=0.0.0.0`
4. Execute the launch script that sets the static transformation of `/camera_link` to `/base_link` by doing `roslaunch kinect_transformer transformer.launch`. This will define the transformation of the Kinect camera relative to the Erle-Copter base.

As the camera is mounted approximately 10 centimeters below the Erle-Copter, I only had to set the z-axis transformation to -0.1 metres. As RGBD SLAM subscribes to /tf by default, the algorithm will get fused sensor data of the visual odometry and the Erle-Copter odometry. This will result in a more accurate trajectory estimate.

5. Launch the freenect driver on the laptop: `roslaunch freenect_launch freenect.launch`
6. Record a bagfile: `rosbag record -b 2048 /topics`. It is recommended to increase the buffersize, as long recordings may cause the buffer to overflow and drop messages. The rosbag has to contain the following topics:
   - /tf
   - /camera/rgb/image_raw
   - /camera/rgb/camera_info
   - /camera/depth/image_raw
   - /camera/depth/camera_info
7. As we are using visual SLAM algorithms, we need to keep as many visual features in the image as possible while recording a dataset. I found that the best way to do this is spinning around 360 degrees in the middle of the room while holding the Erle-Copter high up. After that, do a coastal navigation of the room. Repeat this when entering a new room.

I recorded multiple datasets at home and around room V315 of the Groenenborger campus.

1.2 RGBD SLAM

1.2.1 Methods

There are multiple ways to apply the RGBD SLAM, all of which have their pros and cons. Firstly, SLAM can be performed online. This method only requires the camera driver to be launched, no bagfiles have to be recorded. It is a fast approach to map the environment. However, the algorithm does tend to lose visual features when using this method, which causes RGBD SLAM to stop mapping the environment until it detects a previously observed feature.
Secondly, our previously recorded bagfile can be played out with `rosbag play`. When playing back a bagfile, a parameter can be set the playout rate. If a slow rate is used, RGBD SLAM will have more time to detect visual features. However, this does require more computational resources than online SLAM, especially memory space. The algorithm will crash after a while if there are not enough resources available.
Thirdly, the RGBD SLAM GUI has an option to load a ROS bagfile. This method will iterate

over all messages in the bagfile, so that all information is utilised and many visual features are detected. Nonetheless, loading a bagfile requires even more resources than playing a bagfile, and the method is very slow.
As the last method does seem very interesting, I will try to test it on a computer that can deliver all necessary resources and compare the result to online SLAM and playing a bagfile.

1.2.2 Parameters

In order to obtain an accurate map and trajectory, I set the following para

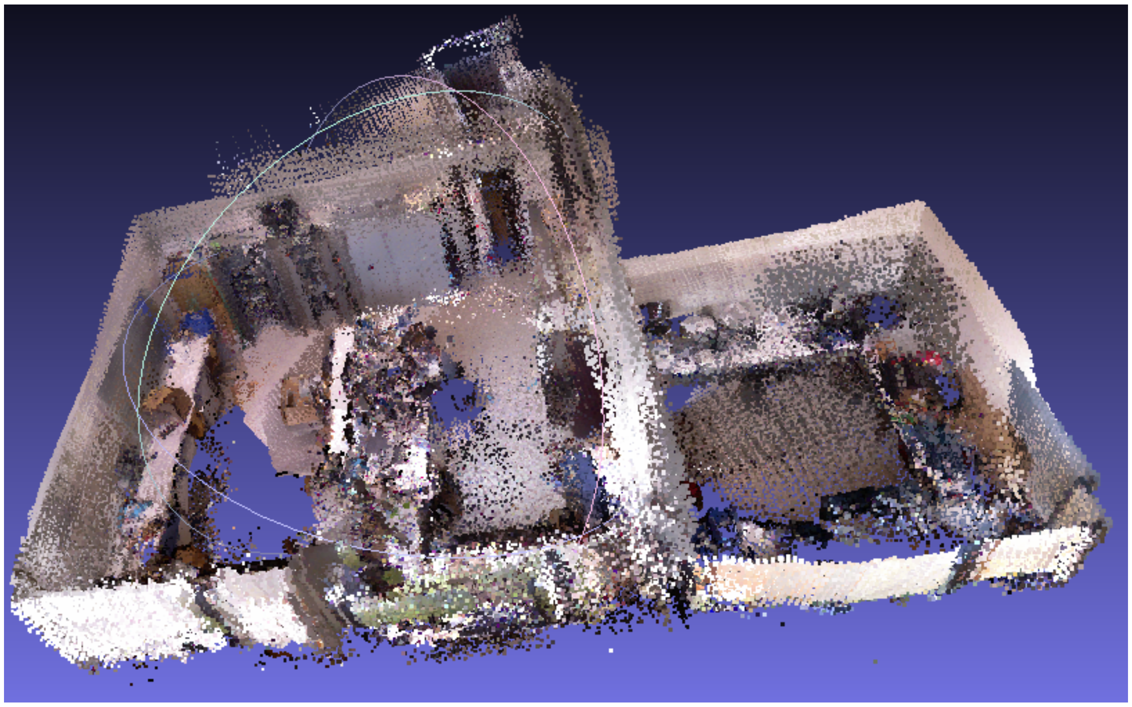
- Filter data between 0.8 and 5 metres to reduce camera noise. The optimal values for these parameters can vary when mapping a different environment.
  - `maximum_depth` = 5
  - `minimum_depth` = 0.8
- Feature detector/extractor: `SIFTGPU` (The maps rendered with ORB are atrocious)
- Matcher type: `FLANN`
- Enable `gicp` as ICP method (Iterative Closest Point)
- Set `cloud_creation_skip_step` to 8 or even 16 to reduce memory usage and prevent RGBD SLAM from crashing

1.2.3 Results

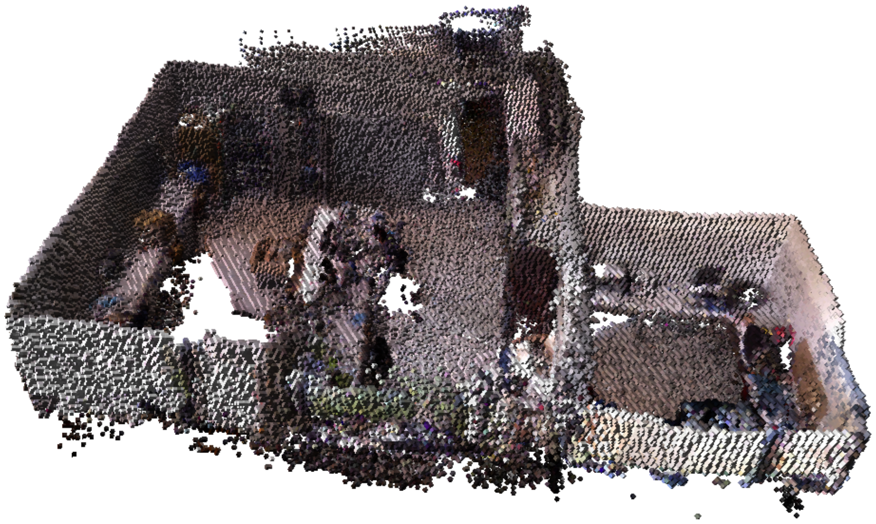
The best result so far was obtained by playing out the bagfile I recorded around room V315. By setting the rosbag playout rate to 0.2 and the cloud_creation_skip_step parameter of RGBD SLAM to 16, many visual features were detected while memory usage remained limited. However, visual features were still lost when entering the hallway, so the resulting map only shows room V315 and a side room. In figures 1a and 1b, the pointcloud and OctoMap results are shown.

1.2.4 Evaluation

Empirical study shows that RGBD SLAM rendered a dense point cloud of the environment (figure 1a). Also it seems that the dimensions of the room are pretty accurate, and the amount of noise is limited. As mentioned before, only a small part of the hallway is included in the map, because the algorithm could not detect visual features at that point. The Octomap in figure 1b shows the same accuracy, but is much more memory efficient (8.2 MB, while the point cloud takes up 122.1 MB).



(a) V315 point cloud



(b) V315 OctoMap

Figure 1

In figure 2, a trajectory was plotted in the xz-plane. Note that this trajectory is not the trajectory that was used in 1. I chose a different trajectory that better demonstrates the issues I am still facing on this topic.
What we should be seeing is a trajectory that takes off at xyz-coordinates 0 0 0, and then moves along the x-axis while approximately keeping the same altitude. However, we see that the trajectory is inverted in the z-axis, and keeps going down instead of keeping the same altitude. This is probably due to the static transformation mentioned in 1.1. The z-axis is inverted because the camera is mounted upside down, and the downward motion is probably due to the camera tilt. I will implement these parameters in the `transformer.launch` file in order to solve these issues.

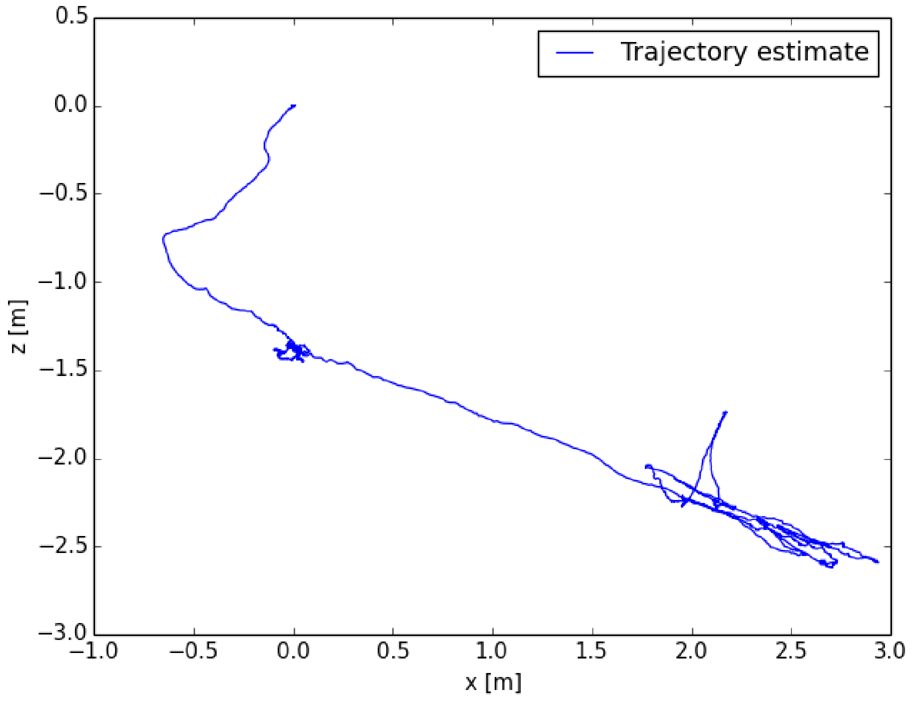


Figure 2: XZ-axis plot of a trajectory generated when loading a bagfile in RGBD SLAM. The trajectory is inverted and appears to be going downwards constantly.

With the RGBD SLAM benchmark tool proposed by Sturm et al [4], the estimated trajectory can be compared to a ground truth trajectory. However, we did not implement a system to generate a ground truth trajectory. For outdoor situations, GPS can be used. As we conduct our research indoor, this is not a viable option. For indoor enviroments, it is possible mount a downward facing camera that detects visual markers. For example, Yu et al [5] implement such a system to autonomously navigate a drone in an environment. We

will not be pursuing this topic, as it is out of our research scope.
We will use the RGBD SLAM benchmark tool as a base for comparing two trajectory estimates, and then empirically determine which trajectory is the best match to the ground truth trajectory. This way, we can decide which method is most suitable to obtain an accurate trajectory.

2 Planning week 12

- Improve static transformation of `/camera_link` to `/base_link`. Right now, the trajectory keeps rising in the z-axis. Solve this by changing the orientation angles in `transformer.launch` (Problem described in 1.2.4, figure 2).
- Run the RGBD SLAM algorithm on a better computer to create a better dataset
- OctoMap
  - Research
  - Initial guess algorithm. Create a point cloud based on the actual dimensions of the environment.

References

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

[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] 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.

[4] Jrgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *IEEE International Conference on Intelligent Robots and Systems*, 2012.

[5] Ethan Yu, Xi Xiong, and Xia Zhou. Automating {3D} wireless measurements with drones. *Proceedings of the Tenth {ACM} International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization - {WiNTECH} {\textquotesingle}16*, 2016.