Progress Report Week 5

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 LSD SLAM [1], RGB-D SLAM [2], ORB SLAM2 [3] and OctoSLAM [4] to map an indoor environment. The data that was gathered using SLAM can be stored in an OctoMap [5].

1 Progress

1.1 RGB-D SLAM

1.1.1 Research

Based on the paper by Endres et al [2], I made a mindmap that lists the main aspects of RGB-D SLAM. This mindmap can be seen in figure 1.

1.1.2 Installation

- Instructions at `https://github.com/felixendres/rgbdslam_v2/tree/indigo`. Follow the instructions below ***'Installation from Scratch'***.
- When building rgbdslam, there were a lot of undefined references to OctoMap functions. This can be solved by doing the following:
  - Below ROS, in `'find_package()'`, add `octomap_msgs` and `octomap_ros`
  - Comment the 4 lines below the line that says `'# Octomap ##########'`

  For more detail, see figures 2a and 2b.

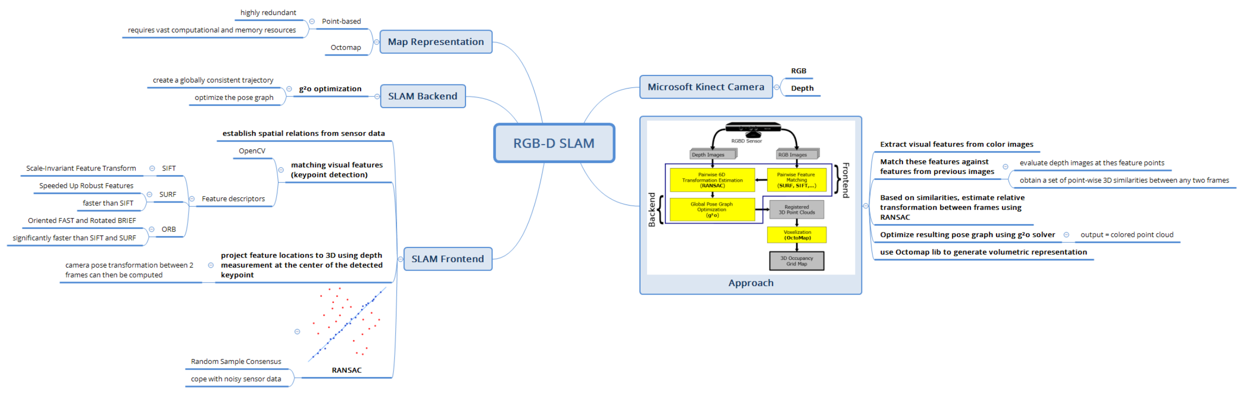
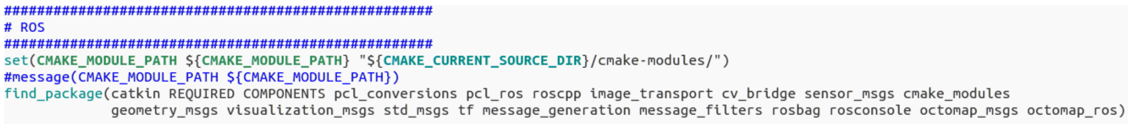
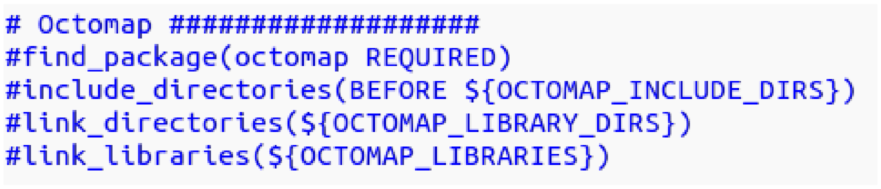

highly redundant
Point-based
requires vast computational and memory resources
Octomap
Map Representation
create a globally consistent trajectory
g²o optimization
optimize the pose graph
SLAM Backend
establish spatial relations from sensor data
OpenCV
matching visual features (keypoint detection)
Scale-Invariant Feature Transform
SIFT
Speeded Up Robust Features
SURF
faster than SIFT
Feature descriptors
Oriented FAST and Rotated BRIEF
ORB
significantly faster than SIFT and SURF
camera pose transformation between 2 frames can then be computed
project feature locations to 3D using depth measurement at the center of the detected keypoint
SLAM Frontend
Random Sample Consensus
RANSAC
cope with noisy sensor data
RGB-D SLAM
Microsoft Kinect Camera
RGB
Depth
RGBD Sensor
Depth Images
RGB Images
Pairwise 6D Transformation Estimation (RANSAC)
Pairwise Feature Matching (SURF, SIFT,...)
Frontend
Backend
Global Pose Graph Optimization (g²o)
Registered 3D Point Clouds
Voxelization (OctoMap)
3D Occupancy Grid Map
Approach
Extract visual features from color images
Match these features against features from previous images
evaluate depth images at thes feature points
obtain a set of point-wise 3D similarities between any two frames
Based on similarities, estimate relative transformation between frames using RANSAC
Optimize resulting pose graph using g²o solver
output = colored point cloud
use Octomap lib to generate volumetric representation

Figure 1: RGB-D SLAM overview



```
##########################################################
# ROS
##########################################################
set(CMAKE_MODULE_PATH ${CMAKE_MODULE_PATH} "${CMAKE_CURRENT_SOURCE_DIR}/cmake-modules/")
#message(CMAKE_MODULE_PATH ${CMAKE_MODULE_PATH})
find_package(catkin REQUIRED COMPONENTS pcl_conversions pcl_ros roscpp image_transport cv_bridge sensor_msgs cmake_modules
              geometry_msgs visualization_msgs std_msgs tf message_generation message_filters rosbag rosconsole octomap_msgs octomap_ros)
```

(a)



```
# Octomap ####################
#find_package(octomap REQUIRED)
#include_directories(BEFORE ${OCTOMAP_INCLUDE_DIRS})
#link_directories(${OCTOMAP_LIBRARY_DIRS})
#link_libraries(${OCTOMAP_LIBRARIES})
```

(b)

Figure 2

1.1.3 Tests

In order to test the algorithm, I had to change my Gazebo model. The default model was completely gray and did not have any objects in it. If I would try RGB-D SLAM in this environment, there would be to few reference points to create a map. The new model can be seen in figure 3.
Next, I mounted the custom kinect-based depth camera that I created last week on the Erle-Copter in Gazebo. In order to create a correct map of the environment, I noticed that it is best to script the trajectory in a way that the camera nearly always has a wall in its line of sight. Therefore I created `mission4`. The trajectory from this mission can be seen in the resulting pointcloud (figure 4a).

RGB-D SLAM can be executed online or with a bagfile. For testing purposes, I chose the latter. This allowed me to test numerous RGB-D SLAM parameters on the exact same

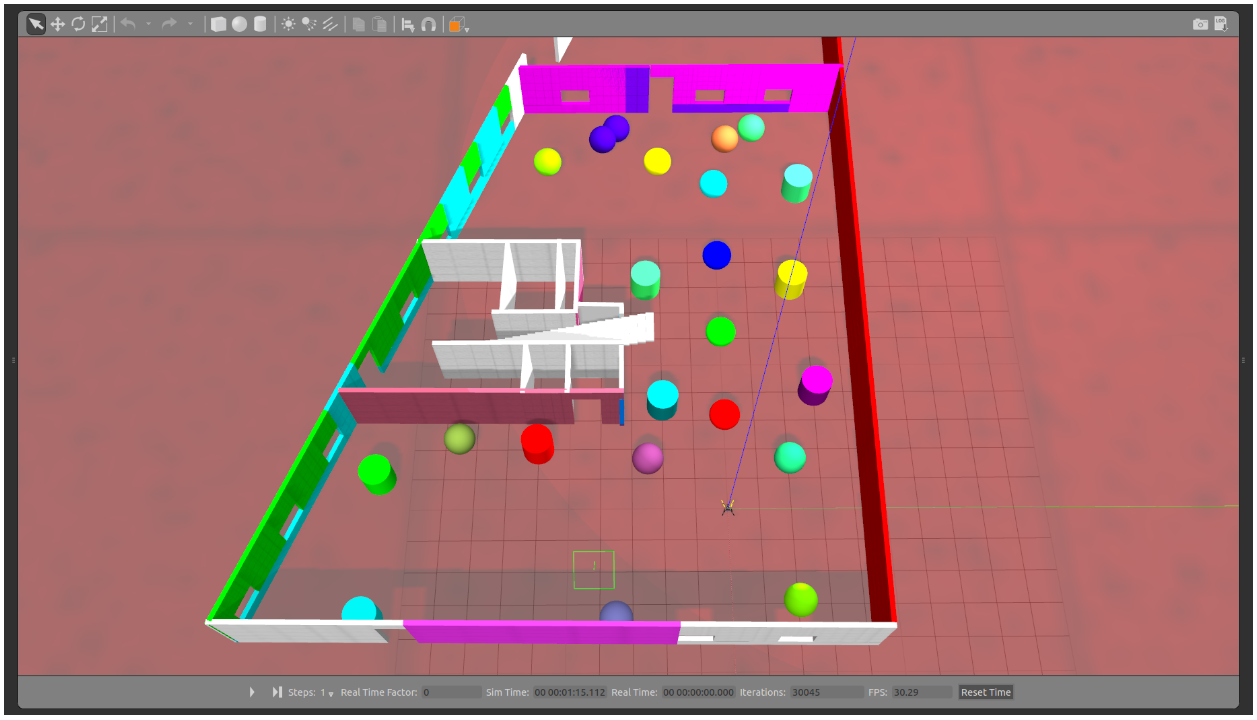

Steps: 1 Real Time Factor: 0 Sim Time: 00 00:01:15.112 Real Time: 00 00:00:00.000 Iterations: 30045 FPS: 30.29 Reset Time

Figure 3: The new simulation model

simulation. Recording the bagfile in combination with simulating in Gazebo was another challenge. The simulated Erle-Copter became very unstable during recording, because the combination of these processes was very though on my CPU. The solution to this problem was to split up the recording in different bagfiles. The command i eventually used was:

```
rosbag record --chunksize=512 -b 256 --split --duration=15 -O kinect_bag4_rgbdslam
/tf /camera/rgb/image_color /camera/rgb/camera_info
/camera/depth_registered/sw_registered/image_rect_raw /camera/depth/camera_info
```

When playing back these bagfiles, all subbags have to be started simultaneously. Due to the timestamps, the bags will be played sequentially. Every bag contains the following topics:

- /tf
- /camera/rgb/image_color
- /camera/rgb/camera_info
- /camera/depth_registered/sw_registered/image_rect_raw

- /camera/depth/camera_info

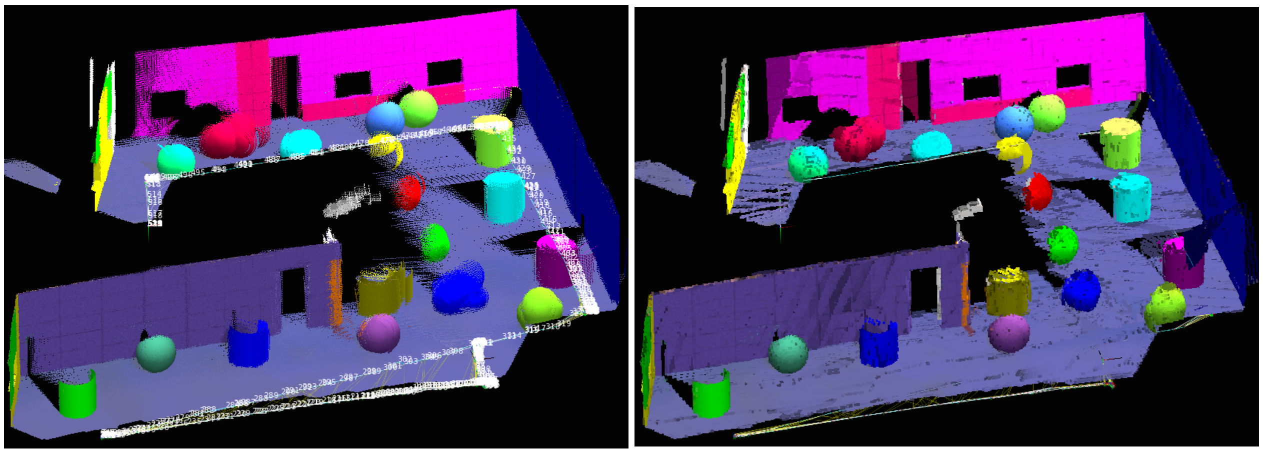
When the bagfile was created, I started the RGB-D SLAM GUI with `roslaunch rgbdslam simulation_test.launch`. This starts a launch-file that I created based on `fast_visual_odometry.launch` example that I found in the github library. When the GUI was ready, i played the rosbags and the pointcloud was formed.

1.1.4 Results

At first, the resulting pointcloud was very abstract, so I empirically changed some parameters in the launch file. In order to obtain the result from figure 4, I tweaked the following parameters:

- max_keypoints = 200
- max_matches = 100
- ransac_iterations = 200
- cloud_creation_skip_step = 4

One of the main advantages of RGB-D SLAM is its OctoMap support. In the GUI, online Octomapping has to be enabled.



(a) RGB-D SLAM point cloud (b) RGB-D SLAM OctoMap

Figure 4: Result of the RGB-D SLAM algorithm

Clearly, figure 4 shows promising results. It has to be noted that the trajectory of the simulation is still not optimal, as I am still working on scripting yaw rotation. When this is achieved, I can create a trajectory that follows the walls, while turning the camera at every corner. Hopefully, this will result in better loop closures.

1.2 ORB SLAM2

1.2.1 Research

In my search for suitable visual SLAM algorithms, I came across ORB SLAM2 [3]. This algorithm supports monocular cameras, RGB-D cameras and stereo cameras.

1.2.2 Installation

Installation instructions for ORB SLAM2 are found at `https://github.com/raulmur/ORB_SLAM2`. Make sure to install all necessary dependencies first! I cloned the repository in the `src` folder of my catkin workspace and then executed `build.sh` and `build_ros.sh`

1.2.3 Tests

First, I studied the ROS Examples (Monocular, RGB-D and Stereo). The results of these examples can be seen at `https://www.youtube.com/watch?v=ufvPS5wJAx0`. Next, I tested Monocular ORB SLAM2 with the `rgb_cam_bag2` bagfile and RGB-D ORB SLAM2 with the `kinect_bag3_rgbdslam` bagfile.

1.2.4 Results

When viewing the example results at `https://www.youtube.com/watch?v=ufvPS5wJAx0`, it is clear that the pointcloud is extremely sparse.
The experiments with my own bagfiles gave no result at all, as ORB SLAM2 was not able to create a decent point cloud. Another problem with this algorithm is that in contrast to LSD SLAM [1] and RGB-D SLAM [2], there are not a lot of parameters that can easily be changed.
I will not examine this algorithm any further, as LSD SLAM and RGB-D SLAM appear to be way more suitable for my thesis.

1.3 OctoSLAM

1.3.1 Research

As I intend to focus on visual SLAM algorithms, OctoSLAM [4] is not eligible for my research. This SLAM algorithm creates a point cloud of the environment using a 2D laser range finder. Therefore, I will not research this algorithm any further.

1.4 Yaw rotation with Mavros

This is still a work in progress. I finally received an answer on my forum topic, but for now I am struggling with RC overrides via mavros.

1.5 ROS Bagfile

Normally, I planned to create a default rosbag that could serve to test different SLAM algorithms. As my trajectory is still not optimal. For now, I use the bagfile `kinect_bag3.bag` to test the algorithms. This bagfile contains the topics mentioned in section 1.1.3. `Mission4` was used as a trajectory.
When I can optimize the trajectory code, I will create a new default bagfile.

2 Planning week 6

- Yaw rotation with Mavros
- RC override with Mavros
- Test indoor flight with Erle-Copter
- Connect camera to Erle-Copter and create a bagfile of a real 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] Raul Mur-Artal and Juan D. Tardos. ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras. 10 2016.

[4] 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.

[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.