1 What I've done

- Implemented the following in python using opencv:
  - Detecting SIFT features, and matching them (using brute force matcher)
  - Method 1. Calculating the essential matrix and using it to find R and t
  - Method 2. Getting a depth associated with feature points, and then aligning them using Kabsch algorithm, to find R and t
  - Graphing trajectories (component-wise) for both positions and quaternions

2 Parts of report to look at

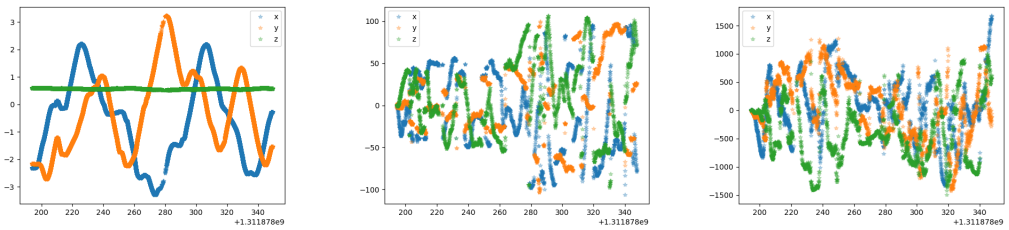
- Nothing new.

3 Questions

-

4 Comments

- Pieter is using the new RealSense on the TX2, so it might be a bit tricky to run code for the R200 on it.
- The timestamps for the ground truth, rgb and depth images are not aligned, and there are not the same number of each. Right now I'm just reading in the rgb and depth images in order and matching them up that way, but I should actually associate a depth image to the rgb using the timestamps.
- I should convert to axis-angle to compare the rotations rather than using the quaternions (although I'm not sure how to graph that).
- The registered trajectories are very different to the ground truth. In particular, the ground truth has almost no variation in z but all components vary in the registered trajectories. I'm not sure if the issue is poor registration or issues converting between the various coordinates (does z in the ground truth correspond to y in the image? – and x to z, y to x).

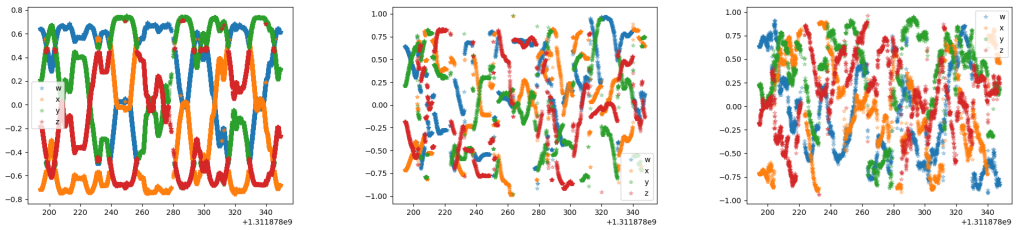


(a) Position trajectory - ground truth

(b) Position trajectory - RGB only registration

(c) Position trajectory - registration using depth

Figure 1: Comparison of position trajectories.



(a) Quaternion trajectory - ground truth

(b) Quaternion trajectory - RGB only registration

(c) Quaternion trajectory - registration using depth

Figure 2: Comparison of quaternion trajectories.

- The position trajectories also have major scaling issues, and the registration with depth increases massively.
- Need to quantify the differences, but will probably need to sort out the time stamp alignment first.
- There are a few frames where the matching couldn't find enough good points (need at least 6) and so I skipped them. I also think my bound for good points might be a bit high. I might swap to a relative measure, as some frames have much better matches than others.

5 Stuff to do

- ~~Fix quadcopter~~
- Adjust quadcopter trajectory so that it can see the top box

- Investigate data
  - ~~Investigate boxes to determine which ones can be picked up as point clouds~~
  - ~~Investigate RealSense cameras, compare the two cameras and using them on TX1/TX2 or Windows/Ubuntu~~
- Investigate registration algorithm
  - Generate 3D object in MATLAB that I can get points from from various camera poses (may also need to get RGB and depth images if we're going with that approach?)
  - Apply registration algorithm to generated point clouds (ground truth known) – without noise first, then add noise. Get error in true and estimated translation and rotation.
- Reading
  - ~~Incorporating RGB and depth images – feature matching (papers you sent me)~~
  - Write up summary of main points (parts of the algorithm, choices with advantages/disadvantages)
  - See if more recent papers have made advancements