1 What I've done

- Compared the two RealSense cameras
- Read through the three papers you sent me
- Had a look at various RGB-D databases

2 Parts of report to look at

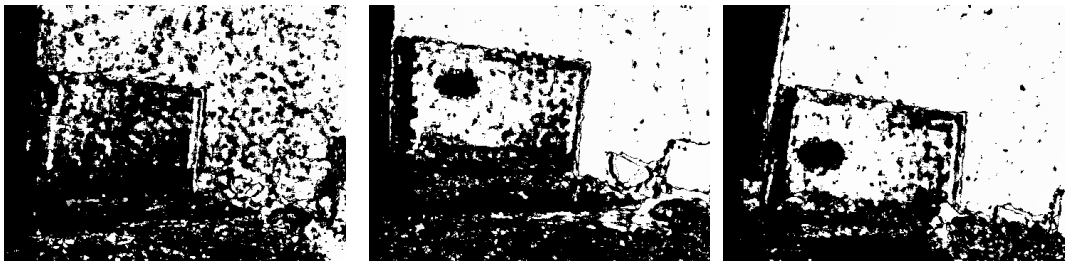
- Nothing new.

3 Questions

-

4 Comments

- None of the papers really seem to incorporate odometery information from the camera mount (one uses visual odometry).
- Generalized main steps in algorithms for each frame:
  1. Estimate change in camera pose, keep track of global pose.
  2. Attempt to see if frame matches a previous keyframe (can apply filtering first, in order to detect loop closure.
  3. If it matches, update the global pose estimate since the last detected loop closure.
  4. Determine if frame is a keyframe.
- These datasets look pretty good:
  1. http://www.scan-net.org/. Indoor scenes, annotated with 3D camera poses, surface reconstructions, and instance-level semantic segmentations
  2. https://vision.in.tum.de/data/datasets/rgbd-dataset. Various scenes, used in one of the papers you sent me. RGB and depth movies and ground truth trajectories. Has some datasets from a kinect mounted to a Pioneer robot, which come with odometry data and point clouds.



(a) Depth image from RealSense that was mounted to quadcopter, with code running on TX1

(b) Depth image from other RealSense, with code running on TX1

(c) Depth image from other RealSense, with code running on Ubuntu

Figure 1: Comparison of depth images for the two RealSense cameras and for running on different platforms.

- As I said in the email I sent you earlier this week, the RealSense camera I had at home seems to produce a less noisy depth image. Also, running from Ubuntu doesn't seem to make a difference, but the any data capture programs die shortly after being started (likely due to the fact I don't have a powered USB hub to plug the camera into).

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