Laavanye Bahl

Team D: CuBi

Teammates:
Jorge Anton, Paulo Camasmie, Changshen Shen, Nithin Subbiah Meganathan

ILR08

October 9, 2019

Individual Progress

The main focus for this review was the validation of a successful pickup.

Following things were done:

1. Comparison of what information/ sensor to use for this validation.
2. Obtaining the current image for validation.
3. Coding the comparison algorithm.
4. Making it robust to light, environment and position invariant.
5. Coming up with the unit test.
6. Integration with ROS.
7. Integration with state machine.

Many sensors were considered for this problem like the use of ultrasonic sensors, rgb camera and stereo camera. The basic idea was to make the manipulator go to certain position for validation and collect the rgb image / point cloud or depth image of the empty tray. Then at whatever time step you want to validate the pickup, you simply command the manipulator to go to that validation position again, collect the current data for comparison by telling the difference with the data of the empty tray and setting a threshold for validation.

Three approaches were considered:

1. **Depth images for comparison**
   Depth images were first tried for this purpose, since it invariant to the color of the object picked.
   Drawbacks of depth images are described in the individual challenges.

2. **Depth images for segmenting out the tray from the background and rgb was comparison**
   This approach was tried to get position invariance while selecting the region for comparison from the rgb images as shown in Fig 1.

The disadvantage for this was objects on ground which were too close to the manipulator when it is at the validation position as well the NaN points.

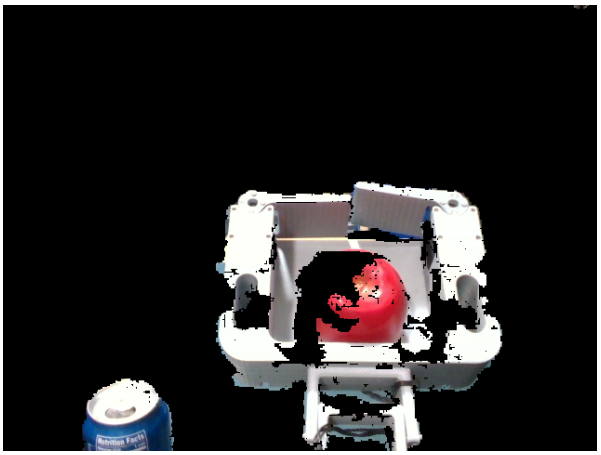

Fig 1. Using depth for masking tray

3. **Using rgb for comparison**

   After trying out the first two solution, the following approach was selected.

- Command manipulator to go to the fixed validation position on boot up.
- Collect image of the empty tray as shown in Fig 1a.
- Intelligently select the corners of the region you want to crop and compare (the center of the tray).
- Warp this cropped portion and rectify it for better comparison.
- Save this rectified image for the empty tray.
- Obtain current image when comparison is required as shown in Fig 1b.
- Follow the same procedure as the empty tray.
- Now for comparison the simples solution is to subtract the default image from the current one and analyze the histogram of color intensities for R, G, B channels and compare the norm of the difference of this array between the current and default images.

Fig 1a. Image of empty tray

Fig 1b. Image of tray with object

ROS integration:

Built a ROS service to achieve this task. The state machine can simply call this and with type as input (save default or comparison) and receives a difference score and a validation decision.

Challenges

Individual challenges:

1. For comparison one simple solution was to subtract the default image from the current one and analyze the difference, but it very sensitive to the pixel positions.
   Solution: Obtain the histogram of color intensities for R, G, B channels and compare the norm of the difference of this array between the current and default images.

2. The disadvantage of comparing the depth images were the NaN points in the depth image as seen in Fig3.
   Solution: Hole filling algorithms were also tried as a solution but, did not return very accurate results which could differentiate empty from occupied.

3. Depth images were affected by the environment and the background as the depth image is normalized by the furthest value to the range 0 -255 as shown in Fig 3.

Solution: Work with the unnormalized values instead of gray scale pixels.

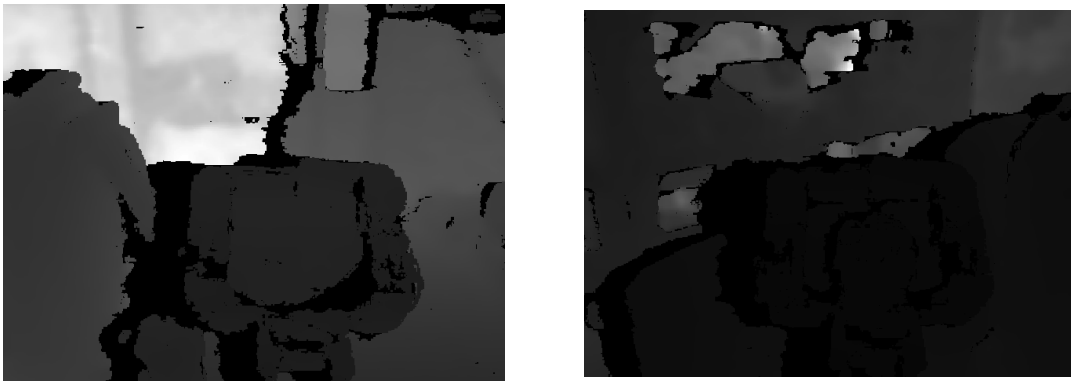

Fig 3. Depth images obtained of the same manipulator in different environments.

Team challenges:

Following were the team challenges:

1. Finalizing the SVD replication:
   This was a bit of challenge as the new code contained new added features. We worked together to finalize the code and integrate everything.

2. Integration of state machine
   This requires effort. We sat with Jorge to manage this nicely.

3. This PR was actually the best with respect to team challenges. There were not many. We planned efficiently in advance with precise sub tasks and did achieve them before time.

Teamwork

For this PR, we had meetings for task division. So, I actively participated in the planning.

Following describes the work done by the team members and how I interacted with them:

Paulo:

Worked on fail safe operation of the manipulator. If the gripper gets stuck picking up an object then it can be recovered by monitoring the torque values of the motors and detecting an anomaly. Also, Paulo was this PR's presenter, so I interacted with him by explaining the work done by me in detail.

Bobby:

Bobby lead the website updation and worked with Jorge on coming up with exploration strategies. I interacted with him regarding the website and wiki updates and comparison of the LiDAR and Stereo base maps.

Jorge:

Jorge lead the replication of SVD and removal of errors and worked with Bobby for initial exploration strategies. I interacted with him to help me get the manipulator to the validation position and the whole module could be added to the state machine. I decided to create a service call for validation.

Nithin:

Nithin worked further on conducting the decided subtask for base map creation. He replicated the map creation pipeline that I used in the previous PR, limited the range of the stereo camera to see improvements. I interacted with him by helping to run the previous PR algorithm and solving errors. I also had discussions with him to decide various sub-tasks. I will further be working with him on this task.

Future Work

Individual plans:

1. Lead the obstacle detection task, plan and divide work.
2. Develop an initial obstacle detection pipeline.
3. Work with Nithin on making the decision for the base map creation.
4. Work with Jorge to perfect the state machine.
5. Version the code on Github after the SVD replication and the added features.

Team plans:

1. Work together to finalize code after the added features.
2. Perform SVD with exploration.
3. Demonstrate initial obstacle detection separately.
4. Tune gains for more rapid execution (aiming for 20-30% speed-up).