Laavanye Bahl

Team D: CuBi

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

ILR09

October 23, 2019

Individual Progress

The main focus of this review was improving validation pickup and its integration with the whole system and research for obstacle detection.

Grasp Validation
For grasp validation, the process was further improved by making a robust service that was persistent for multiple requests.

Following pipeline was used:

- Command manipulator to go to the fixed validation position on boot up.
- Collect the image of the empty tray as shown in Fig 1a and store it.
- Get the pre-defined corners of the region to crop.
- Warp this cropped portion with inverse perspective mapping and rectify it for better comparison as shown in Fig 2a and 2b.
- Obtain the current image when the comparison is required as shown in Fig 1b.
- Follow the same procedure as the empty tray.
- Now for comparison, subtract the histogram of color intensities for R, G, B channels (as shown in Fig 3a and 3b) and compare the norm of the difference of this array between the current and empty tray cropped images with a defined threshold.

Fig 1a. Image of empty tray

Fig 1b. Image of the tray with object

Fig 2a. Cropped and warped image

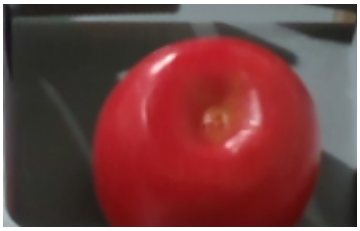

Fig 2b. Cropped and warped image

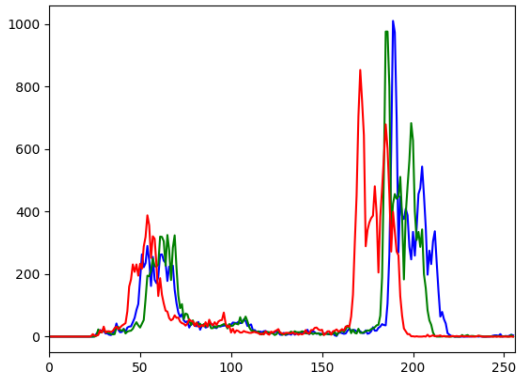

Fig 3a. Histogram of colors of empty tray

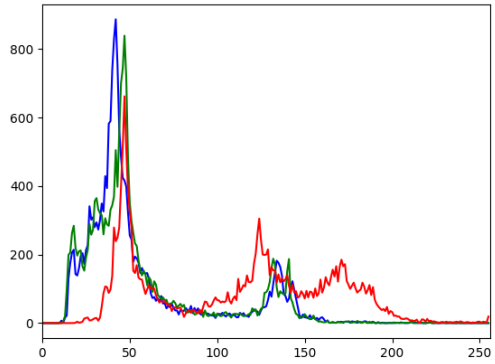

Fig 3b. Histogram of colors of the object

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.

State machine integration:

Added the service client in the state machine, which has to call the validation function once on boot-up and then every time we want to validate the grasp. I added a validation position in the state machine which on boot-up executes the command to go to the validation position and saves the data for the empty tray to get good images for the current environment. When validation is required the service's function is simply called to get a validation result. This state was added in the state machine. CuBi will attempt to grasp the object again if it was unsuccessful for the first time and if it fails again then it starts to explore some other area.

This validation will give us huge speed advantage as compared to the previous state machine in which CuBi always goes to the base to drop off the toy after every pick up attempt even if it was unsuccessful.

Obstacle Detection:

I read a lot of papers, blogs, and projects for doing a literature survey. I also met and discussed with Jorge and Nithin about the potential approaches we could take for this task.

Following things were considered:

- Is deep learning required?
- Can we do obstacle detection and SLAM together and build the most updated map always?
- Do we lock obstacles? How do we get obstacle position invariance? How to do efficient obstacle tracking?
- How will the planner receive information about the obstacles? Should the obstacles be plotted on the map?
- How object detection pipeline fits in with obstacle detection? Is it enough to differentiate obstacles based on size?
- Would the elevation of the camera pose a problem?
- Do we need to set a threshold for the field of view if using point clouds?
- Do we define certain obstacles and develop the program just for them or make a general pipeline like object detection?
- How to handle false-positives?
- Do we need to label obstacles?
- Come up with metrics for obstacle detection.

After reading about a lot of approaches, I decided to try a few of them.

One interesting approach is explained in Fig 4.

- It uses the Intel real-sense stereo camera, taking both point cloud and RGB images
- We can then exploit deep learning for detecting obstacles using RGB images.
- Object tracking is performed on these detections.
- Then we use point clouds for accurate localization and getting 3D information of the obstacles which is essential to estimate its size and position in the real world coordinates.

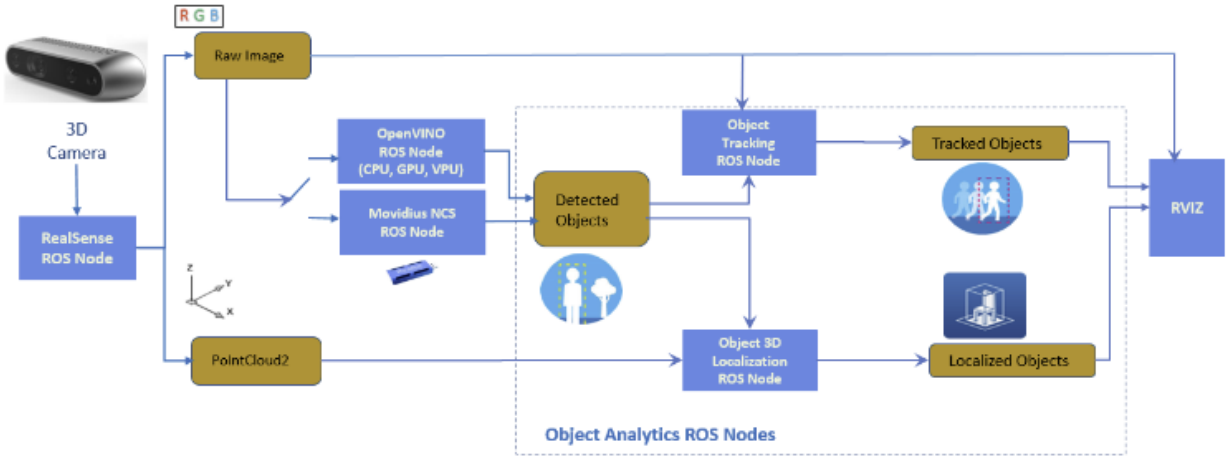

Fig 4. Potential obstacle detection pipeline

Challenges

Individual challenges:

1. Installing new packages like Open CV on Jetson takes a lot of time.
2. Understanding different modules of the code takes a lot of time and effort when working on integration.
3. I spent a lot of time debugging an issue where my service for grasp validation was crashing without returning an error. The problem was with not closing a plot of the histogram of colors with plt.close() command before calling the service again.

Team challenges:

Following were the team challenges:

1. Integration and updating state machine:

   When there are a lot of changes made my different members of the team, it requires a lot of time and effort to add new functionality to the state machine.

2. Finalizing a testing location:

   As it takes a lot of time to move things and map different rooms, we had to finalize a room. This was a challenge because of the availability, lighting conditions, furniture, and distance from MRSD lab. We decided on the lab next to the MRSD lab on floor B where team Delta is working.

Teamwork

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

Paulo:

Paulo worked on adaptive gains tuning and doubled CuBi's speed. I interacted with him for understanding the working of the state machine for efficiently merging our codes together with the system and not hurt the current implementation.

Bobby:

Bobby worked on performing localization with the hardware by building the LiDAR map of the new room that we finalized for our demos. He also worked with Jorge on exploration strategies. I interacted with him for deciding the output type for the map and how it can be used for various applications of localization/ exploration.

Jorge:

Jorge worked along with Bobby for exploration. He also improved the state machine further and worked with Paulo on integrating fail-safe grasping with the state machine. I interacted with him for obstacle detection algorithms and integrating grasp validation with the state machine.

Nithin:

Nithin worked on mapping the new location with Bobby and obstacle detection. I interacted with him regarding the same to discuss different strategies and what parameters to consider to tackle the task of obstacle detection.

Future Work

Individual plans:

1. Develop obstacle detection pipeline.
2. Get some results for obstacle detection.
3. Maintain and merge code on GitHub.
4. Test current object detection pipeline in the new environment which will fail. Therefore, come up with new strategies to tackle that.
5. Robust object detection – Add better locking for objects while moving towards it.

Team plans:

1. Further speedups in mobility controls.
2. Continuous integration and improvements.
3. Making CuBi work in the new environment.
4. Improving localization.
5. Showing exploration.