2/24/2016

Progress Review 9

Individual Lab Report #8

Abhishek Bhatia

Team D: Team HARP (Human Assistive Robotic Picker)
Teammates: Alex Brinkman, Feroze Naina, Lekha Mohan, Rick Shanor

I. Individual Progress

For this progress review, our team's major task was to get ready for the UR5 arm, wait and pray that the arm gets delivered on time, and finally if the arm arrives, setup the arm physically, interface the arm with ROS and access the APC configuration space. Luckily, the arm got delivered just a day before the progress review and we were able to complete the physical and software setup smoothly and were able to access the complete APC configuration space.

After the last progress review, the first thing that I worked on for this progress review was fixing the lab machine. During my last progress review, I had mentioned about the driver issue making the Kinect_bridge to run only on CPU and not on GPU. Since, I was unable to fix this issue with re-installing libfreenect drivers, I thought the fastest way was to reset the machine, install Ubuntu from the scratch and reinstall all the drivers. This was a clean approach and fixed the driver issue. To overcome such issues in future, we have decided to not install any auto updates and install only the necessary drivers and updates.

After this, I worked on exploring color based object identification approaches. The idea behind this was that since some of the items from the item dictionary had very distinct colors, like the *oreo's* with a lot of blue pixels, or the *cheezeit* box with a lot of red pixels, or the *duck toy* (which gave us difficulty in identification using just the depth based method last semester) with a lot of yellow pixels etc, it was probably worth for us to explore some simple *rgb* based object identification techniques that can give us some prediction percentage which we can combine with our current depth based metric to solidify our prediction. For this, I started with a simple approach of making color histograms and comparing them with the training dataset.

The first step was to read the point cloud and parse the *rgb* color values corresponding to each pixel and convert it into the *Lab* color space. A *Lab* color space is a color-opponent space with dimension L for lightness and a and b for the color-opponent dimensions, based on nonlinearly compressed coordinates. The most important attributes of the L*a*b*-model are device independence and is considered to be more robust in situations with variable lighting conditions. The space is mapped onto a three-dimensional integer space for device-independent digital representation, and for these reasons, the L*, a*, and b* values are usually absolute, with a pre-defined range. The lightness, L*, represents the darkest black at L* = 0, and the brightest white at L* = 100. The color channels, a* and b*, will represent true neutral gray values at a* = 0 and b* = 0. The red/green opponent colors are represented along the a* axis, with green at negative a* values and red at positive a* values. The yellow/blue opponent colors are represented along the b* axis, with blue at negative b* values and yellow at positive b* values, as shown below in Figure 1. The scaling and limits of the a* and b* axes will depend on the specific implementation of Lab color, but they often, and for our application run in the range of ±100 or −128 to +127 [1].

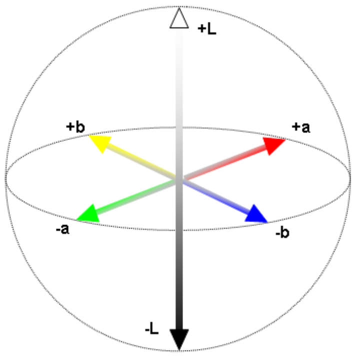

Figure 1: Image displaying the color distribution along different axes for *Lab* color space

I wrote a simple parser in C++ to read a point cloud in *rgb* color space and wrote a function to convert it to *Lab*. After this, I wrote a function to generate histogram bins. I kept the bin width variable for all the three channels. The idea behind keeping the bin width variable is to leave slack for experimentation and fine tuning. Finally, I wrote a function to compare histograms. The compare function takes 2 histograms as input and returns the comparison error, the less the error, the better the comparison. I researched a lot of methods for histogram comparison and concluded on using the Quadratic-Chi histogram method for bin-to-bin and cross-bin comparisons. The bin-to-bin comparison was simple root mean square error comparison, but for the cross-bin comparison, I averaged the errors using variable weights for comparisons within the bins that are not adjacent. Further details about the approach can be figured out through this research paper: The Quadratic-Chi Histogram Distance Family [2].

After this, Rick and I worked together to integrate *rgb* based histogram generation and comparison as part of our original vision pipeline. The vision pipeline now uses the data from both the depth based ICP approach and *rgb* based histogram approach to generate scores for identifications of various objects. We still need to generate a huge database of test images and test the integrated framework extensively, but for now, Rick generated a database of 20 images for some 4-5 objects from the APC item dictionary and we tested the integrated framework which seemed to generate satisfactory results. Figure 2 displays the results from our testing, we have 3 different items and 2 different images per item. The figure displays 20 bins generated, first 10 bins for ‘a’ color space and next 10 bins for ‘b’ color space. As it is visible from the figure, each object type had very similar histograms for images in different orientations and lighting conditions. We ignored comparison between the ‘L’ space as it mostly concerns with the lighting condition.

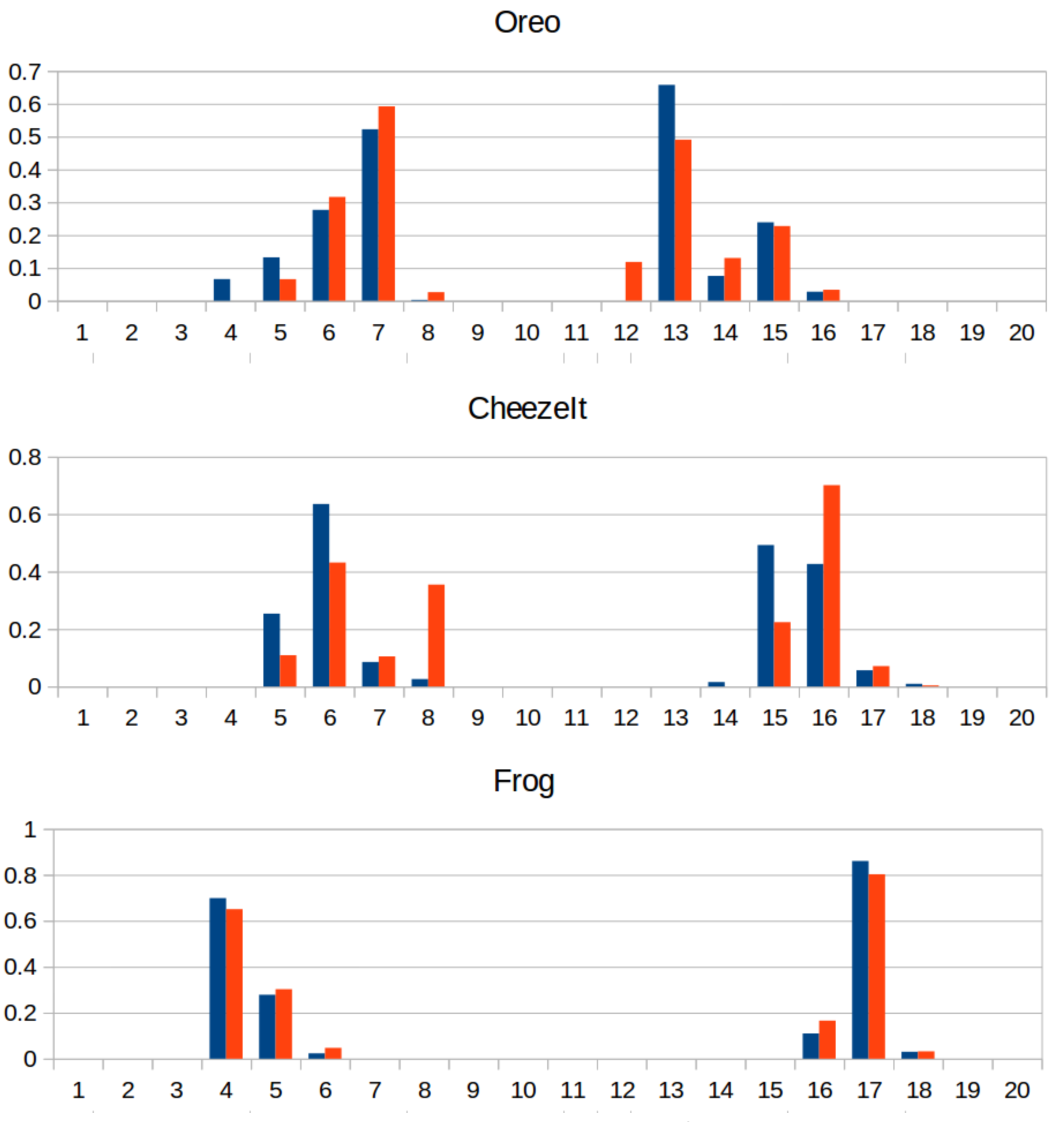

Figure 2: Image displaying histograms generated for various test images

After this, the last thing I worked on before the progress review was to setup the UR5 arm as soon as we received it on Tuesday. Alex, Rick and I worked together to setup the base and bolted the UR5 securely on the base, Figure 3. Later, Alex and I focussed on setting up the arm to work with ROS. I installed the Universal Robots UR5 bring-up ROS package. First I tested with the base package and standard test scripts, once I had that working, Alex and I worked together to setup our teleoperation scripts and other scripts to test the APC configuration space.

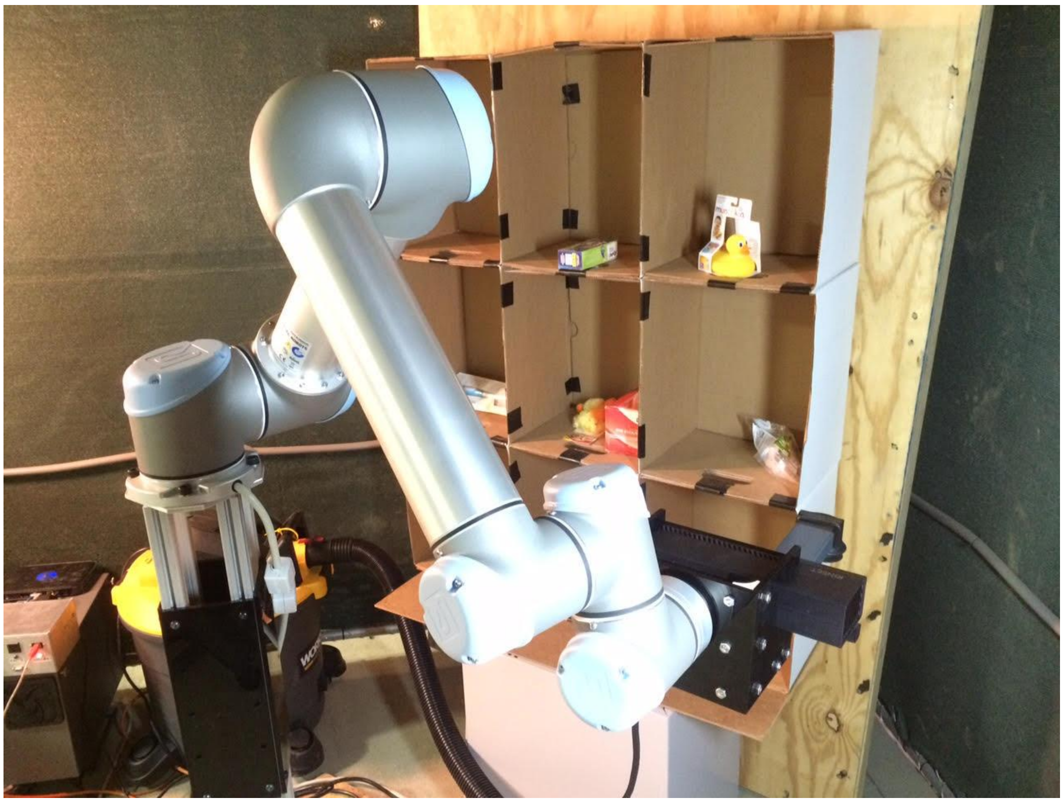
Figure 3: UR5 arm mounted on the base and 9-bin shelf visible in the background

II. Challenges

The biggest challenge we faced this week was while setting up the UR5 arm to work with ROS. While trying to work with the base UR5 bring-up package, we discovered there was a bug with the UR5 bring up package. It was interfering with the updated firmware that our UR5 arm had come with. One option was to downgrade the UR5 firmware, but we instantly decided against it as we thought it was not a good idea to mess with the firmware on the first day itself. Some further research and debugging landed us on a bug report online that mentioned this issue was fixed with another unofficial release of the UR5 bring-up package. We installed this updated package and it fixed this issue for us. Besides, to make the UR5 work with our existing scripts, we had to do a lot of workarounds and hacks, but we were able to make everything work eventually.

III. Teamwork

For this week's progress review, we worked together to get ready and have all the dependencies sorted out such that we could setup the UR5 arm seamlessly, as soon as we receive it. Besides this, we worked towards trying out having a *rgb* based object identification approach along with our current framework to improve the prediction accuracy. We also tried out off the shelf perception packages such as simtrack [3] for identifying partially occluded objects, and aruco hand-in-eye [4] ros package for Kinect camera calibration.

Alex: Alex wrote scripts for teleoperation of UR5 and other scripts to setup UR5 and test APC configuration space once we setup the arm. He also worked on Kinect camera calibration using the aruco eye-in-hand Kinect camera calibration package.

Feroze: Feroze worked on setting up simtrack and modelled some items from APC item dictionary.

Lekha: Lekha continued with her work on the grasp planner. She generated grasping models for 5 items from APC item dictionary.

Rick: Rick worked on setting up PERCH for some items from the APC dictionary. He also worked on setting up the vision pipeline to include the *rgb* based identification functions generated by me. He also worked on making minor modifications to the UR5 base to increase its stability.

Abhishek: I worked on exploring and generating color histogram based object identification techniques and later worked with Rick to integrate them with our existing vision pipeline.

IV. Future Plans

My major targets for the next Performance Review are to continue working with *rgb* histogram based object identification approach. There are multiple things to be done in this respect that includes database generation and extensive testing, and parameter fine-tuning. Besides this, Rick and I will also work to explore different machine learning techniques that we can include as part of our vision pipeline to further improve the object identification accuracy and look for techniques that can work well with partially occluded scenes. One major bottleneck with these tasks is an updated list of APC item dictionary for this year, which we hope to get by end of this week.

IV. References

1) https://en.wikipedia.org/wiki/Lab_color_space
2) http://www.ariel.ac.il/sites/ofirpele/publications/ECCV2010.pdf
3) http://www.karlpauwels.com/simtrack/
4) https://github.com/jhu-lcsr/aruco_hand_eye