Progress Review 12

Rick Shanor
Team D: Human Assistive Robotic Picker
Teammates: Alex Brinkman, Feroze Naina, Abhishek Bhatia, Lekha Mohan
ILR11
April 12, 2016

Individual Progress

Over the last two weeks, I finished implementing and integrating the entire vision pipeline. First, I finished training a CNN that identifies superpixels. Next, I implemented a conditional random field algorithm that determines item centers and grasp points based on adjacent probabilities. Finally, I integrated the entire vision pipeline with the robot state machine. Using this vision process, we were able to make the robot pick up items of interest from the shelf. In addition to this work, I also helped with system integration and testing.

As previously demonstrated, the first step in the identification pipeline is to mask the shelf. The masking operation is performed by using the SegNet CNN model to label each pixel shelf or item. The results of Segnet can be seen in figure 1.

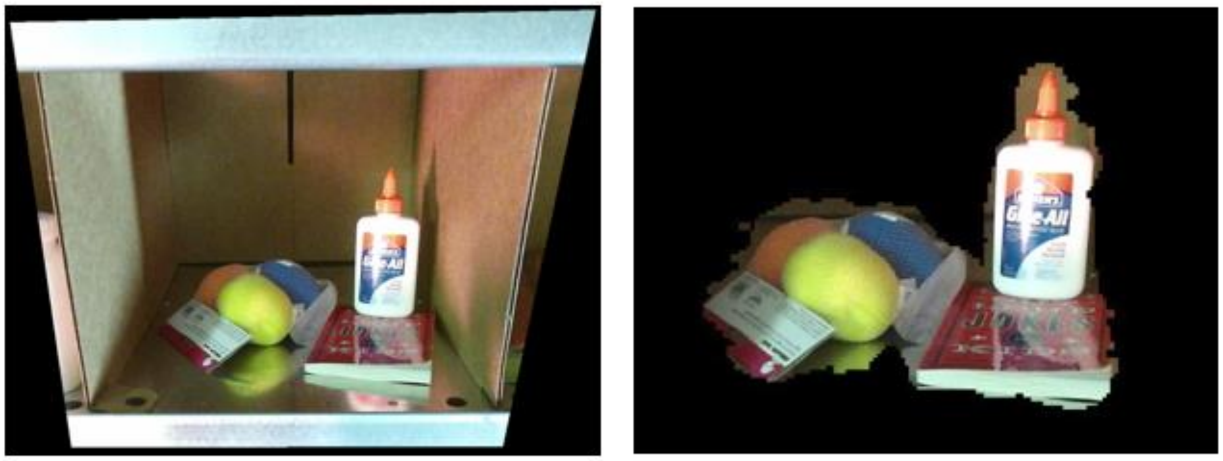

Figure 1: Shelf Image (left) and Filter Output (right)

Next, I apply the SLIC algorithm to the filtered image. SLIC divides an image into approximately K desired superpixels based on edges, distances, and colors. I iterate through each superpixel, and save the ones that are not black. In addition, I calculate the edge region of each segment and cache its neighbors.

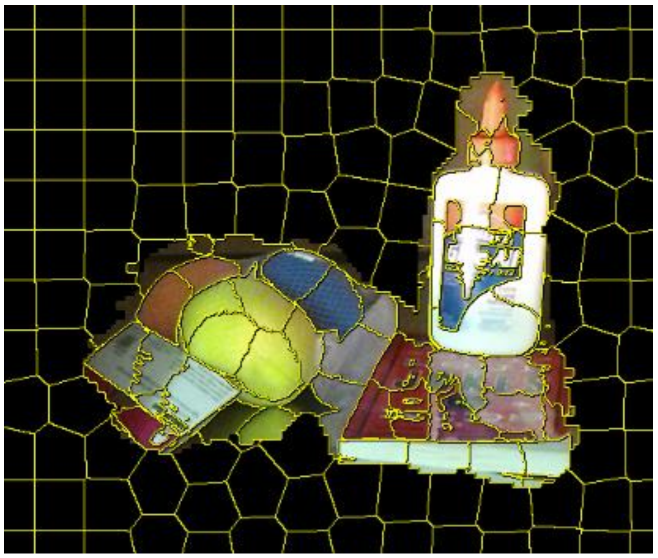

Figure 2: SLIC superpixeled image

After deciding upon a superpixel algorithm, the next step was to train a CNN based on the AlexNet architecture. To train the network, we captured about 100 images of each item on a turntable. Then, we applied the superpixel algorithm to the training data, generating almost 250,000 training images in total for the 38 items. The network is then trained for almost 12 hours on an Nvidia 980 until losses converge, indicating the network has achieved optimal training.

During test time, after breaking the image (figure 2) into superpixels, we classify each superpixel using the CNN. Figure 3 show a few examples of classified output. Each superpixel takes around .2 seconds to be classified. The probabilities are normalized after setting the probabilities of items not on the shelf to zero. By reducing the probability vector based on knowledge of shelf items, we are able to make much more accurate predictions.

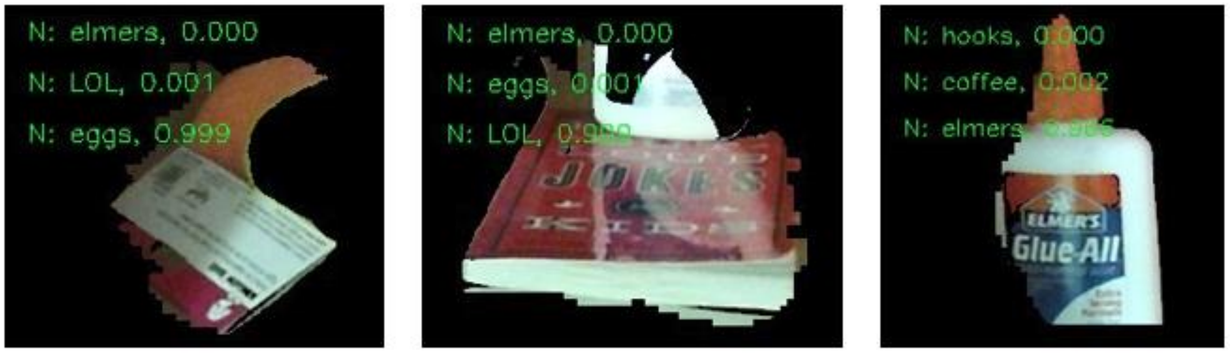

Figure 3: Labeled superpixels

After calculating a probability for each superpixel, a graph is generated connecting neighboring superpixels. An example of such a graph is shown in figure 4 below. After generating a graph, each superpixel is scored based on its predicted probabilities as well as its neighbors predicted probabilities. This enforces smoothness and rewards similar classifications in a region. Finally, a single super pixel is labeled for each item as the maximum of the calculations above.

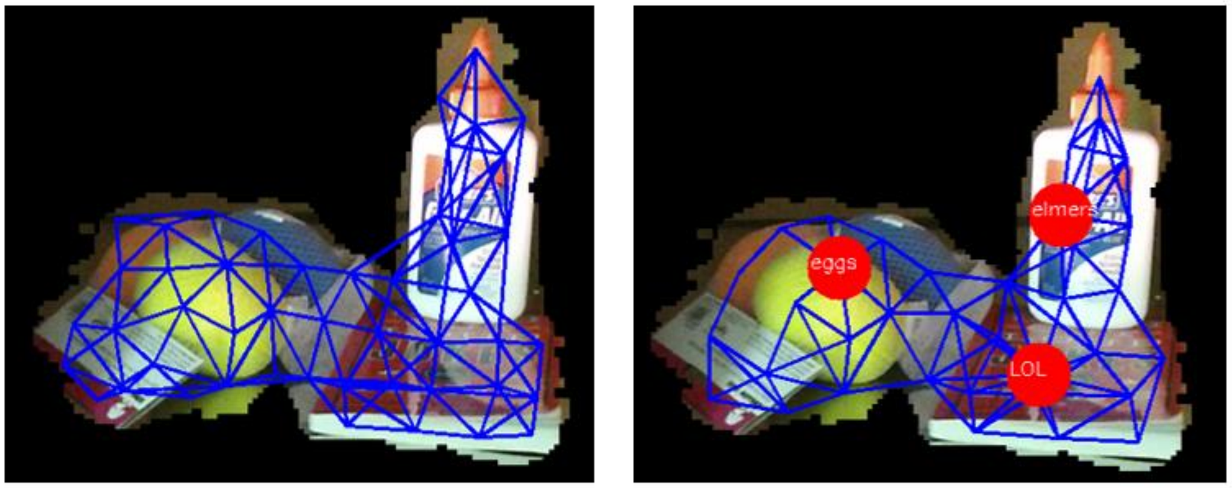

Figure 4: Superpixel graph and labeled nodes

Finally, after identifying the centroid label, we apply a floodfill-like algorithm that expands out and labels neighboring pixels based on their predicted probabilities. This allows us to label each pixel in the image. Figure 5 shows a fully labeled image. The eggs, the Elmer's glue, and the LOL joke book are all accurately labeled. After labeling each pixel in the image, these results are passed to the grasp planning algorithm, which determines the optimal way to pick up the item of interest.

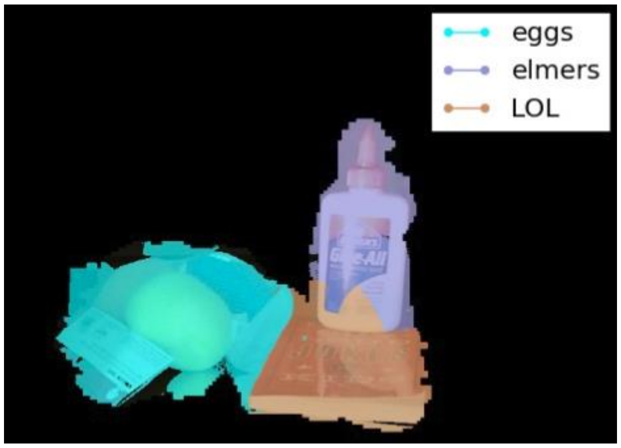

Figure 5: Fully labeled identification results

Challenges

After weeks of hard work, we finally achieved multi-item picking from the Kiva shelf. As a team, grasping in the new, more confined space was difficult to achieve. Personally, my biggest challenge was getting this entire vision pipeline running. I tried many different superpixeling methods, training scenarios, and global scene optimization techniques. Several iterations later, I finally finished the above pipeline. On test cases, the algorithm identifies items with at least 85% accuracy.

Teamwork

Alex has worked hard on robot control and motion planning. He has debugged many problems, including localization and shelf collisions. In addition, he integrated a longer end-effector so that we can reach items in the back corners of the shelf. Both Alex and Feroze worked on grasping. They iterated on our normal-based approach so that we have more control over end-effector placement. Abhishek helped me with perception tasks, including algorithm testing. In addition, he worked on high level scripting. He and I began investigating a verification identification algorithm after we remove an item from the shelf before we place it in the bin. Finally, Lekha and I built 3D models for items with robust geometries for use in PERCH.

Plans

Before the SVE, I plan to expand the vision algorithm from 11 items to all 38 items. This will require retraining the CNN with our expanded collection of turntable images. However, I don't anticipating scaling to have any large impact on accuracy, since it is still an N choose 1 problem where N is the number of items on the shelf. After this, I plan to incorporate error checking into the perception system. We will not want to attempt item pick up until our perception confidence is high. Finally, the current perception pipeline runs in about 30 seconds. I am going to benchmark our code and try to speed up the pipeline. In addition, we will probably parallelize processes, so we can do grasping and motion planning on one shelf while processing images from another.