Carnegie Mellon University

# Individual Lab Report #6

Author:
Yuqing Qin

Team C: COBORG
Gerald D'Ascoli | Jonathan Lord-Fonda | Yuqing Qin | Husam Wadi
Feng Xiang

Sep. 15, 2021

## Table of Content

## 1 Individual Process

The Coborg platform is a wearable robotic arm that can help people hold objects overhead. My work in the Coborg project mainly focused on the Perception (Vision) Subsystem design and implementation. From last semester, I have completed the design and development process for the vision system, and have also evaluated a set of testing cases during SVD and SVD encore. The general idea is to localize the hand positions and average the 3D hand positions which could be used to indicate the target position. In the past few weeks, I migrated the whole vision system from the x86 system to ARM-based Jetson and did the integration with other subsystems.

### 1.1 Migrate Vision System to Jetson Xavier

From last semester, I have already made the vision system running on the x86 system (my laptop with RTX2070). The hand detection algorithm running on my laptop is YOLO v3, which has 106 layers in total. With CUDA10.2 and cuDNN 7.6.5, it works in 30 FPS (real-time). Since our Coborg platform should be untethered, we migrate all the subsystems to NVIDIA Jetson Xavier.

For the vision system, I went through a few iterations on setting up the correct environment to make it work. Since Jetson is an ARM-based computer, which makes the whole migration process more complicated. To make future setup easier, I wrote down the versions for the required packages to run the vision node. Also, updated the README file on GitHub to memorize the whole migration process. The current environment is running CUDA 10.0, cuDNN 7.6, and OpenCV 3.4.6.

Once finished with the migration, I worked with Jason to integrate the subsystems and tested the subsystem integration on Jetson. Below Figure 1 shows the simulation results of our integration. The yellow bounding box shows the position of my hand, which is the target position that the robot arm should reach. From the simulation, it is clearly showing that our integration works as expected.

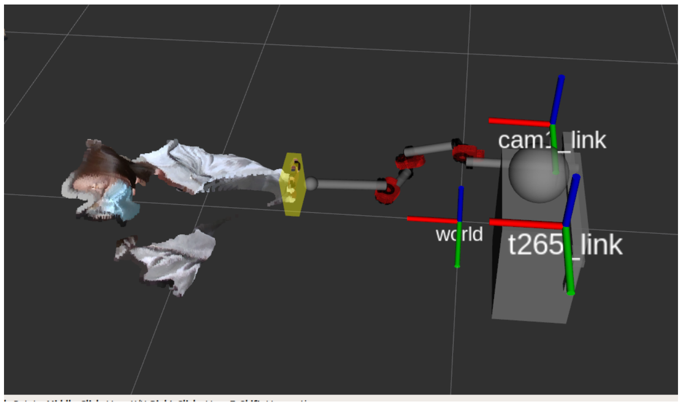

Figure1. Integration Simulation Demonstration

## 1.2 Optimize Vision System Workflow

During the last few weeks, I also optimized the vision workflow to save the memory and time to load the vision system. Since Jetson is not as powerful as my laptop, the vision system performance is not as good as the one we showed during SVD. The main concern is about the run time. Currently, the YOLO v3 running on Jetson is about 10 FPS. This is acceptable for our system since the system starts from voice recognition, which also takes some time to obtain the voice command. During the processing of the voice node, it should be sufficient for the vision system to grab the goal position.

However, to remain some buffer on the vision system, I further optimized the whole vision pipeline when doing post-processing on the goal position. Previously, I post-processed the 3D bounding boxes by calculating the averaged position and surface normal. However, during the surface normal calculation, it would first extract the 2D center position of hands and look at the corresponding 3D position, which is a similar process to get our final goal position. Therefore, there is no need to redo the process in another node which will take more time and memory to handle the 3D bounding boxes. Instead, the vision system will extract the 3D goal position directly from the 2D center position at the time when it estimates the surface normal. The below flow charts (Figure 2 and Figure 3) demonstrate the improvements in our vision pipeline.

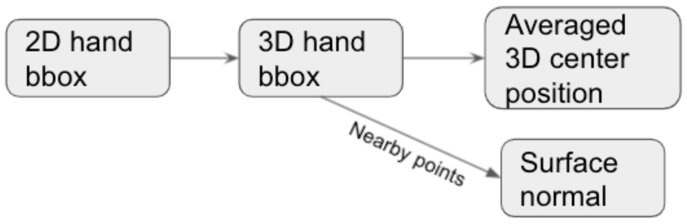

Figure 2. Previous vision pipeline

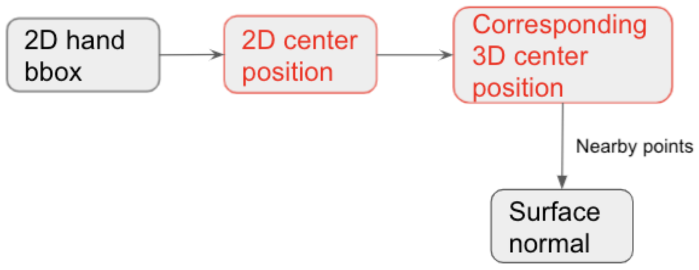

Figure 3. Current vision pipeline

## 2 Challenges

The main challenges I faced are about the Jetson setup. It took me a lot of time to figure out the correct versions for CUDA, cuDNN, and OpenCV. We have tried different combinations of these packages, and also done a lot of research on which version would work. Before setting up Jetson, I was considering the combination I used in my laptop (x86). However, it turned out that the Jetson (ARM-based) had much fewer choices on the versions of the developing packages. Since there are some dependencies on Jetson, we cannot even install the packages(CUDA, cuDNN, OpenCV) separately. We have to install the fixed combination of these packages by using NVIDIA internal SDK manager. It limited the versions that we could work on, and I also need to downgrade OpenCV to 3.4.6 manually to make the darknet_ros running.

## 3 Teamwork

| **Team Member** | **Teamwork Progress** |
|---|---|
| Feng Xiang | - Migrated actuated manipulation system to Jetson<br>- Worked with me to test initial integration on Jetson<br>- Tested the proposals of adding DoF to our existing system |
| Jonathan Lord-Fonda | - Worked on smart manipulation development<br>- Worked on writing scripts for evaluating the task space<br>- Wrote the requirements for the task space |
| Gerry D'Ascoli | - Proposed the PCB v2.0 improvements<br>- Improved the voice system<br>- Integrated all the subsystem nodes into one launch file |
| Husam Wadi | - Project management work<br>- Worked with Gerry on the PCB<br>- Worked on the CAD for the camera holder |

Table 2. Teamwork for Coborg

## 4 Plans

For the next two weeks, I will mainly focus on the task space evaluation and adjustment for the vision system. I will work with Jonathan to identify the workspace that the D435i could capture. Later, I will adjust the camera positions with the new camera holder (3D printed by Husam) to test out the proper location for D435i. I will also prepare the backup plan for using two depth cameras in case that one camera cannot cover our full task space.