Carnegie Mellon University

Individual Lab Report #9

Author:
Yuqing Qin

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

Oct. 26, 2021

Table of Content

1 Individual Process

The Coborg platform is a wearable robotic arm that can help people hold objects overhead. In this semester, my work in the Coborg project mainly focuses on the Perception (Vision) Subsystem upgrade. From the last progress review, I have upgraded the hardware part for the vision system. Using two cameras, we can extend the FOV from 42 degrees to 70 degrees. For this progress review, I continue working on the vision system upgrade plan, mainly focusing on the software upgrade.

1.1 Vision System Software Upgrade

From the last progress review, I have shown the upgraded sensor system with two cameras stacked together, which could extend the FOV from 42 degrees to 70 degrees. Since adding one more camera, we are currently running two YOLO ROS nodes at the same time. Either one camera would have the hands being detected. By saying this, we also need to deal with the edge cases that hands show in both camera frames. To do this, I created a new ROS node to handle all of the edge cases, and also reorganized the whole pipeline for the vision system.

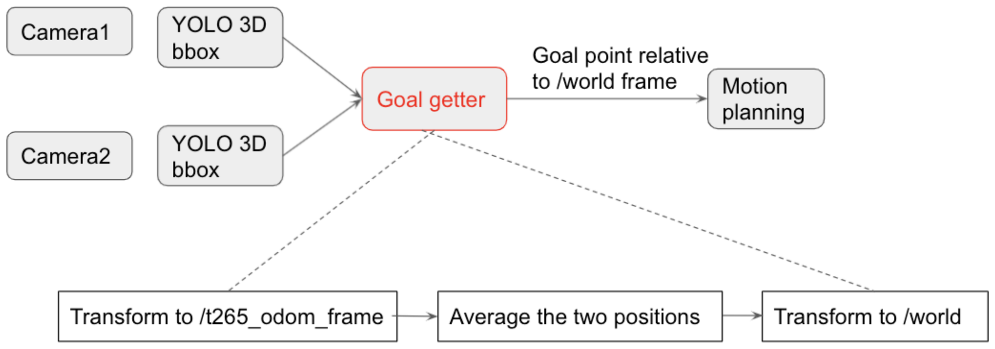
Camera1
YOLO 3D bbox
Camera2
YOLO 3D bbox
Goal getter
Goal point relative to /world frame
Motion planning
Transform to /t265_odom_frame
Average the two positions
Transform to /world

Figure1. Reorganized vision system and manipulation system

As shown in Figure 1 above, the ‘goal-getter’ node takes two YOLO v3 nodes at the same time as the inputs. Before post-processing on the positions relative to different camera frames, we first need to transform them to the single t265 odom frame. Therefore, we can further do the post-processing based on the same frame. Regarding the post-processing on the edge cases, if both hands are shown in the camera frames, the goal-getter will take the average of these two 3D positions. If hands are only shown in the single frame, we just simply take it. Then, we stored this post-processed goal position (relative to /t265_odom_frame) in memory, since this is the static goal position we have to keep track of during the process. To indicate the relative position for the robot arm to move, we further transform this static position to the position relative to the

/world frame (which is changed based on the current location of the person). The goal-getter will constantly publish this goal position (relative to /world frame) so that the smart manipulation system could constantly update the trajectory. We also publish the tf frame that can be easily visualized by using rviz for debug usage. Figure 2 below demonstrates the goal-getter output. The x-axis(in red) is the surface normal. The origin of the goal pose is the center of the hand position.

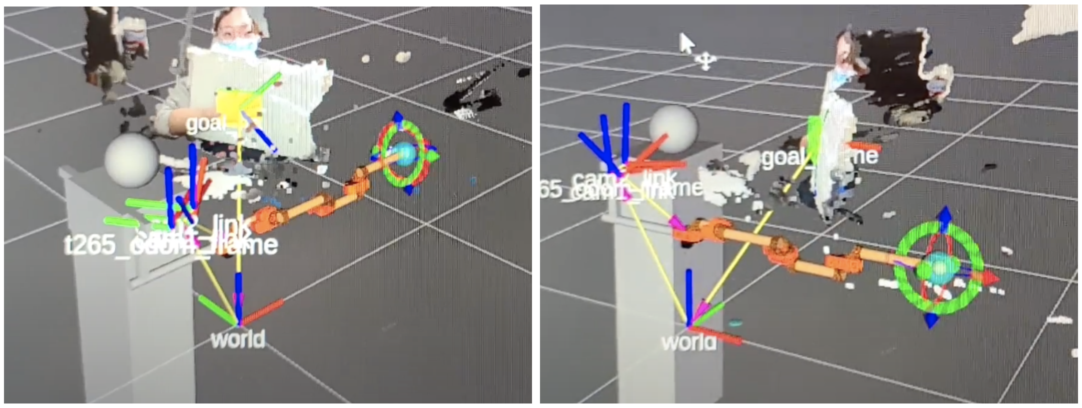

Figure 2. Rviz visualization for goal frame. Front view (left), side view (right)

After testing the node with hands put on different locations, we found out that when hands were on the edge of two cameras, the bounding box detections were unstable. This issue would affect the goal-getter's performance a lot. To solve this problem, I decided to use the 'moving average' on the adjacent frames to smooth the detection results. Figure 3 below shows the unstable detection and the generated goal pose frame using the 'moving average' approach. From the figures, we can see that the first frame detects the upper part of the hand while the second frame detects the lower part of the hand. If without moving average, the final pose frame is most likely off the center of the hand. By applying the moving average, the final pose frame is at the center of the hand.

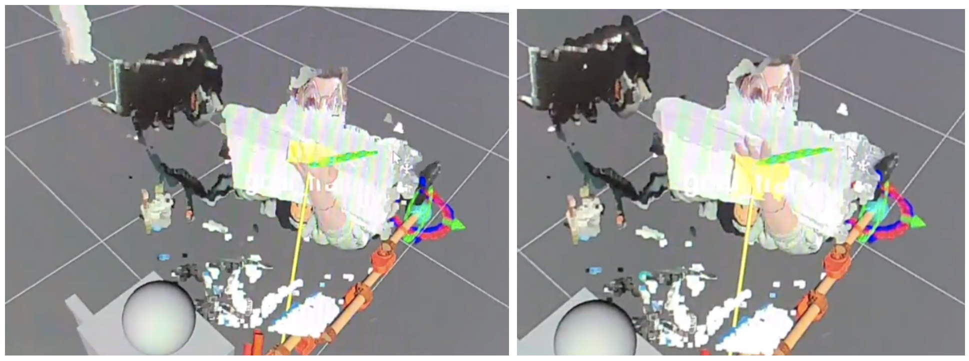

Figure 3. Unstable detections using moving average to generate goal pose frame

2 Challenge

The main challenges I faced are about the design of the goal-getter node. At the beginning of this semester, I removed the goal-getter node since there was only one camera stream. However, when we have two camera streams, we have to put a lot of post-processing workload on the manipulation side (i.e. average, transforms) if without a goal-getter node. After the discussion with the whole team, we decided to isolate the post-processing and the tf transforms to be a new node. In the end, I brought the goal-getter node back and added the tf transforms as well.

3 Teamwork

Team Member	**Teamwork Progress**
Feng Xiang	- Worked with me on vision upgrade plan - Worked with Gerry on resolved rate - Worked with Jonathan on smart manipulation implementation and testing
Jonathan Lord-Fonda	- Worked on debugging the smart manipulation code - Worked with Jason on smart manipulation testing
Gerry D'Ascoli	- Worked with Jason on resolved rate implementation and testing - Worked on integrated resolved rate with smart manipulation - Worked on new hardware assembly
Husam Wadi	- Worked on hardware - Worked on 3D printed parts - Worked on the assembly of new hardware design

Table 2. Teamwork for Coborg

4 Plans

In the next two weeks, I will mainly focus on the integration of the vision node with all other subsystems. I will test the goal-getter node further to ensure its run time performance. Since I added a moving average, there will be a larger latency compared to the one without the moving average. I will also iterate on different window sizes for the moving average to improve the performance. Since the next PR is the rehearsal of FVD, I would mainly focus on the integration to ensure the full use case can be run smoothly.