Carnegie Mellon University

Individual Lab Report #3

Author:
Yuqing Qin

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

March 18, 2021

Table of Content

1 Individual Process

The Coborg platform is a wearable robotic arm that can help people hold the target objects overhead. My work in the Coborg project mainly focused on the Vision Subsystem design and implementation. Our team came up with the hand detection idea to indicate the target object location when people hold the objects. In the first progress review, I have already implemented the 2D YOLO detection node combined with the depth camera node. During the last two weeks, I further implemented the 3D YOLO hand detection node and set up the launch file for both the depth camera(D435i) and the tracking camera(T265).

1.1 3D YOLO ROS Node

To extract the 3D position of the hands putting on the object, I developed the 3D YOLO ROS node by using the open-source package, called “darknet_ros_3d”. This node is connected to the “darknet_ros” node, which is the 2D version of the YOLO model I developed in the first progress review. By updating the hand detection model weights and configurations, the output from “darknet_ros” is the bounding box information of hands showing in the camera frame. The 2D bounding box then goes into the “darknet_ros_3d” node. The 3D YOLO node checks the depth-registered point cloud and extracts the 3D bounding box coordinates from the point cloud. Because the point cloud contains all of the 3D positions relative to the depth camera link, the output of 3D bounding box coordinates are also relative to the depth camera link. This relative 3D position is useful for the motion planning system to generate the trajectory later when we combine the two systems. Figure 1 shows the output from the “darknet_ros_3d” node.

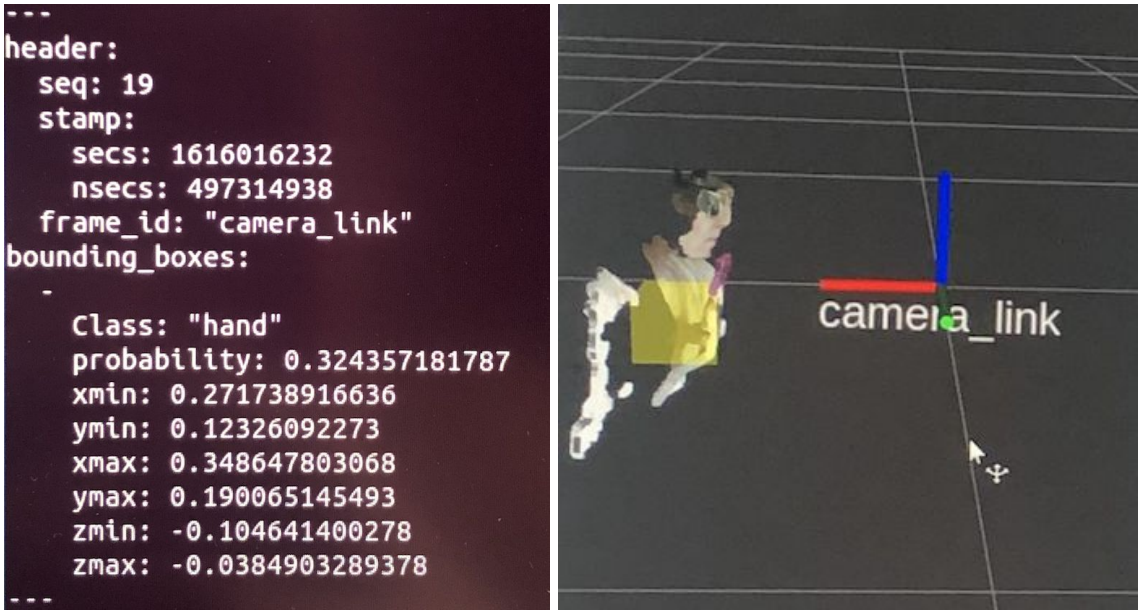

Figure 1. 3D hand detection ROS node implementation (left: 3D bounding box coordinates, right: 3D bounding box showing in yellow in rviz)

1.2 Launch File Combination

Our project will use two cameras. D435i is the depth camera used for hand detection, while the T265 is the tracking camera used for motion planning. Since these two cameras are all developed by the Intel Realsense package, it is convenient for us to combine these two cameras by creating a new launch file.

The D435i launch file contains general information about the depth and point cloud. The pre-processing of the point cloud is done by deploying another package ("rgbd_launch") in the launch file. This preprocessing step will give the depth-registered point cloud, which is used in the 3D hand detection node. Moreover, the T265 launch file contains similar parameters as the depth camera. By combining these two launch files, we also defined the relationship between these two cameras. As a result, the two cameras can have the correct relative distance in the world frame when they are launched. Figure 2 shows the three cameras' frames when I initialize the cameras. The camera_link and t265_link are fixed as initialization, and the t265_odom_frame is a moving frame as I move the t265 camera. The motion planning is built on the t265_odom_frame, and the vision system is built on the camera_link.

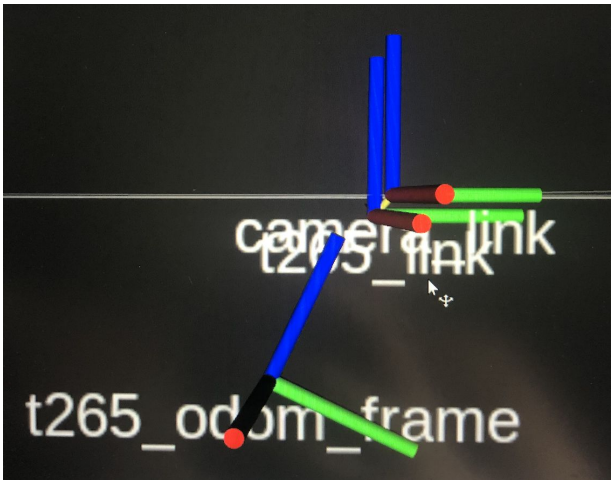

Figure 2. Launched camera frames

2 Challenges

The main challenge I faced in the past two weeks was the 3D bounding box extraction. The D435i provides a default point cloud, which contains fewer points compared to the RGB frame. Therefore, when I matched the 2D pixel to the 3D point, it was not the one-to-one correspondence. To solve this problem, I found the depth registered point

cloud, which used the preprocessing package I mentioned earlier in the launch file. It turns out that the depth registered point cloud gives a one-to-one correspondence on the RGB pixel and the 3D real-world coordinates. By using this point cloud, the 3D positions can be extracted correctly.

The current challenge I am facing is to ensure the accuracy of the 3D position. Considering that the motion planning system may also have some noises, I need to make the hand detection as much as precise before passing it into the motion planning.

3 Teamwork

Team Member	**Teamwork Progress**
Feng Xiang	-Tied T265 and D435 cameras to URDF model -Able to move URDF model live relative to global odom frame in RViz simulator
Jonathan Lord-Fonda	-Connected and implemented main_state_machine node with voice_recog node -Semantically wrote out all nodes -Updated ROS Node Map with proposal -Worked with Jason on Actuated Manipulation -Read about Elastic Bands -Finished setting up Linux and personal ROS -Met with Kelvin to review ROS Node Map
Gerry D'Ascoli	-Integrated new microphone for improved single user voice recognition -Helped Jonathan develop the main_state_machine node -Developed voice_recog node ROS wrapper for voice recognition to feed commands to main_state_node -Tested successful communication and proper functionality between the voice_recog node and main_state_node -Updated website -Developed conceptual design for CoBorg PCB with Husam
Husam Wadi	-Assisted with ROS main node development -Assisted Jason with T265 tracking camera output -Ensured team used Github

Table 1. Teamwork for Coborg

4 Plans

For the next two weeks, I will first work on measuring the accuracy of the vision system. By applying the vision system in different circumstances, I need to make sure the accuracy is good enough before integrating it with motion planning. I will test the system

with different angles of the camera, multiple hands situation, and also the accuracy with the real robot arm.

After ensuring accuracy, I will transfer the vision system from my laptop to the robot PC. I need to set up the GPU environment and launch all of the nodes I implemented before.