

Contact point visualization:

<https://youtu.be/J3-h318al8>

### Camera Calibration

The camera will not be used during the data collection experiment. It is only used for acquiring the object pose and then aligning the point cloud data with the ground truth material annotations. The camera is fixed on the platform, so it's an eye-to-hand problem. The goal is to know the transformation between camera frame C and robot base frame B. The checkerboard will mount on the end-effector of the robot arm and the relationship between checkerboard P and end-effector frame E doesn't matter.

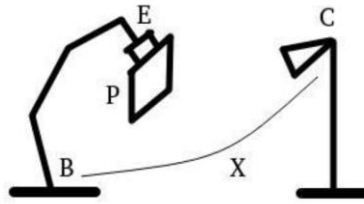


Figure 17: A diagram of eye-to-hand problem in camera calibration

The relationship between coordinate B, E, P, C can be established by:

$${}^B T_i {}^E T_P = {}^C T_P T_i$$

Also, we have:

$${}^E T_i {}^B T_j {}^C T_P T_i = {}^E T_P = {}^E T_j {}^B T_C {}^C T_P T_j$$

Therefore, we can eliminate  ${}^E T_P$  and get:

$${}^B T_j {}^E T_i {}^B T_C = {}^C T_P T_j {}^C T_i$$

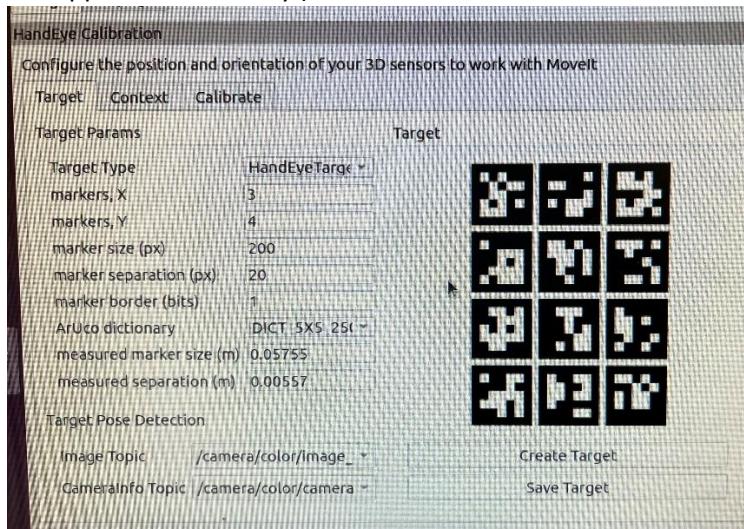
The translation of  ${}^B T_j {}^E T_i$  can be known from the robot arm and the translation of  ${}^C T_P T_i$  can be known from the algorithms on the checkerboard. So, the only unknown variable is  ${}^B T_C$  and the equation is in the form of  $AX=XB$  which can be solved by various of method such as Kronecker Product.

Now we understand camera calibration from a mathematic perspective. In practice, there are lots of tools we can use to do camera calibration, such as MATLAB, OpenCV, and ROS. Since we already made MoveIt work, we kept using the MoveIt plugins and graphical interface for camera calibration. A more specific introduction can be found [here](#). Essentially, we need to attach a checkerboard on the robot end-effector, use MoveIt to move the arm in several poses, use the calibration plugins to take 12 photos as shown in figure 18 and finally get the transformation matrix results as shown in figure 19.

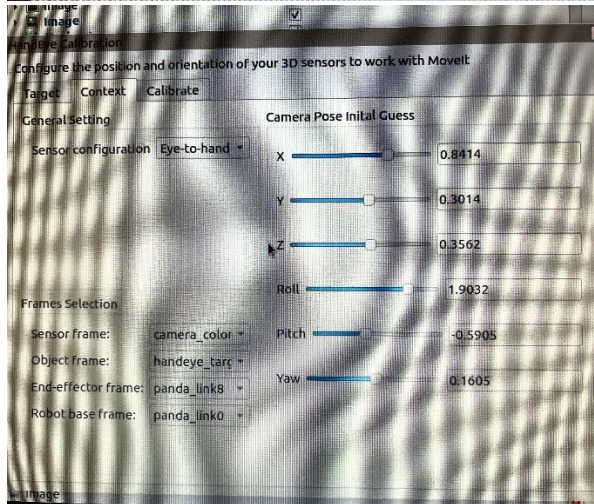
Steps:

1. Launch realsense camera: `roslaunch realsense2_camera rs_camera.launch`
2. Launch franka\_control and Rviz
3. Add>>By topic>>handeye\_calibration>>image>>compressed
4. Add>>By display type>>HandEyeCalibration

5. Add>> By display type>>MarkerArray>> set it to listen to the “/rviz\_visual\_tools” topic (It may not appear immediately.)



- 6.



7. (\*sensor frame: camera\_color\_optical\_frame)

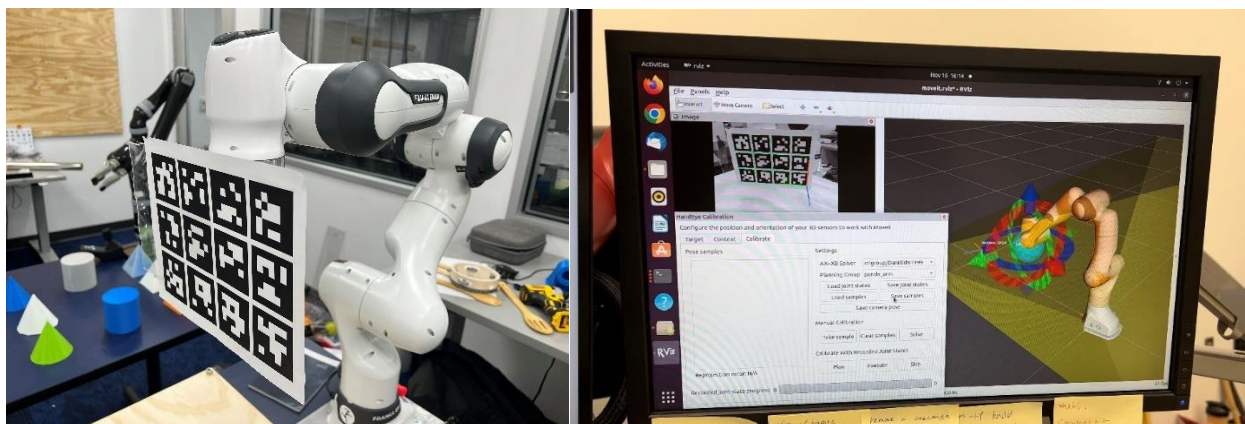
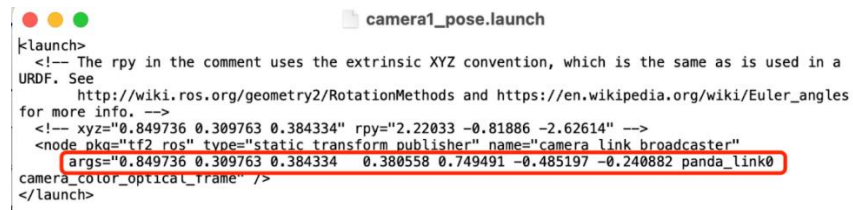


Figure 18: The way how the checkerboard was attached on the arm (on the left) and the MoveIt camera calibration plugin (on the right)



```

camera1_pose.launch
|
| launch>
| <!-- The rpy in the comment uses the extrinsic XYZ convention, which is the same as is used in a
| URDF. See
| http://wiki.ros.org/geometry2/RotationMethods and https://en.wikipedia.org/wiki/Euler_angles
| for more info. -->
| <!-- xyz="0.849736 0.309763 0.384334" rpy="2.22033 -0.81886 -2.62614" -->
| <node pkg="tf2_ros" type="static transform publisher" name="camera link broadcaster"
|   args="0.849736 0.309763 0.384334 0.380558 0.749491 -0.485197 -0.240882 panda_link0
| camera_color_optical_frame" />
| </launch>

```

Figure 19: The transformation matrix obtained from calibration

## Learning Algorithms

After having the ground truth and tapping data, we can consider what tasks can we do. The primary task we would like to do is the 3D shape reconstruction and material segmentation. Due to the limited time and energy, only the 3D object classification task has been done. It's a preliminary practice of implementing deep learning on 3D objects and the 3D reconstruction and material segmentation work will be done later on.

### 3D object classification

PointNet is mainly studied, and the architecture is shown in figure 20. The blue portion of the network is for the classification task. It takes  $n$  points with  $x$ ,  $y$ , and  $z$  as input. Then the input point cloud data will go through input and feature transformations to make the point cloud feature become invariant to its pose in the Euclidean space, and then the features will be processed by max pooling to produce a global feature. Finally, the output of the network is  $k$  scores for corresponding labels.