# Lab 06: Camera Calibration

By Sean Lu, last modified on 04/13, 2020

The objective of this tutorial is to introduce the camera calibration process. The main purpose of calibration is to measure the intrinsic and extrinsic matrix of your camera.

In topic 1 we will use the tool from ‘camera\_calibration’ to measure the intrinsic matrix of your laptop camera. In topic 2, we are going to align thermal image and color image. To do so, we have to measure not only intrinsic but extrinsic matrix. In topic 3, we will use a simple example to do hand-eye calibration, which is an important process in robotic manipulation.

## Hardware and Software Setup

**Laptop setup**

**laptop$ cd ~/sis\_lab\_all\_2020**

**laptop$ git checkout devel-[student\_id] # switch to your branch**

**laptop$ git stash**

**laptop$ git pull origin master**

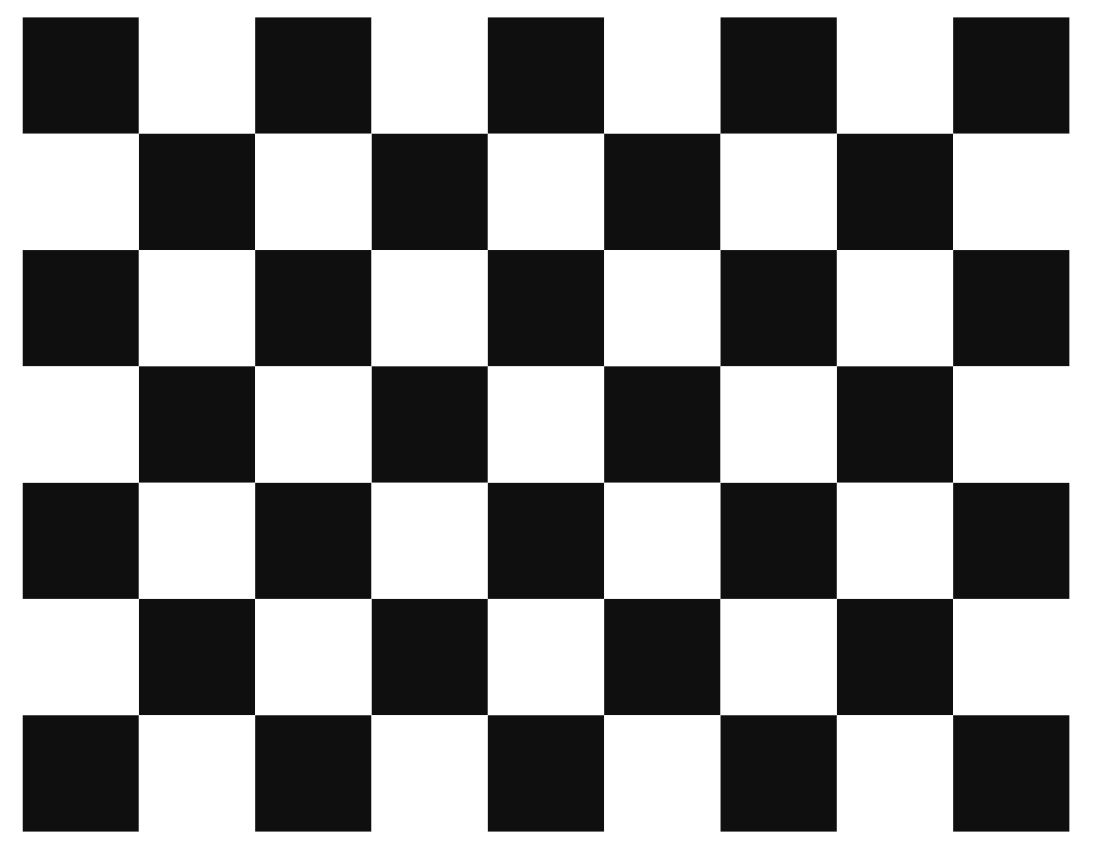
**laptop$ git stash pop**

**laptop$ cd 06-Camera\_Calibration/catkin\_ws && catkin\_make**

**laptop$ source devel/setup.bash # Do this every time you open a new terminal**

**laptop$ cd ../script/topic2/cali\_thermal && sh auto\_make.sh**

And you will get a chessboard pattern likes the following.

****

[**Lab 06: Camera Calibration**](#_74gram21yqax) **1**

[Hardware and Software Setup](#_chc57ow4x0s1) 1

[Overview](#_x5puhc1p2jjo) 2

[Topics and Activities](#_g0fw0bgwxhdt) 2

[Topic/Activity 1 Intrinsic Calibration of Your Laptop Camera](#_aadm982kyolo) 2

[Topic 1.1 Calibrate Your Camera](#_rl15slrm10vt) 2

[Topic/Activity 2 Align Thermal Image to Color Image](#_sf5mhl8ztpdm) 6

[Topic 2.1 Thermal-Color Alignment](#_v4sjn3hb2lab) 7

[Topic/Activity 3 Hand-Eye Calibration in Fixed-Eye Configuration](#_d3flxg22g2i3) 8

[Topic 3.1 Mathematic Background](#_9j2e1dd265kn) 8

[Assignment Tasks](#_kack8n2wqwv3) 9

[Task 1. Hand-Eye Calibration](#_b88a8cdwhf8c) 10

[Reference](#_uwwglharlfb) 11

## Overview

Estimated Time to Finish: 1 hours

After completing this tutorial you should

* Know how to get intrinsic matrix of your camera with camera\_calibration package
* Know how to align image from one camera to the other
* Know how to do hand-eye calibration in fixed-eye configuration
* Create your own package and write an executable node

## Topics and Activities

### Topic/Activity 1 Intrinsic Calibration of Your Laptop Camera

Camera calibration is an important process, you can correct for lens distortion or measure the object size in real-world unit. In this topic, we introduce how to measure the intrinsic matrix of your camera with [camera\_calibration](http://wiki.ros.org/camera_calibration) package from ROS.

#### **Topic 1.1 Calibrate Your Camera**

(Estimated Time to Finish: 40 minutes)

The main concept of calibration is illustrated in [here](https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html). Fortunately, there are already out-of-the-box ROS package which can save your time.

Open your laptop camera

laptop $ **roslaunch usb\_cam usb\_cam-test.launch**

If you get garbled image in image\_view window, try:

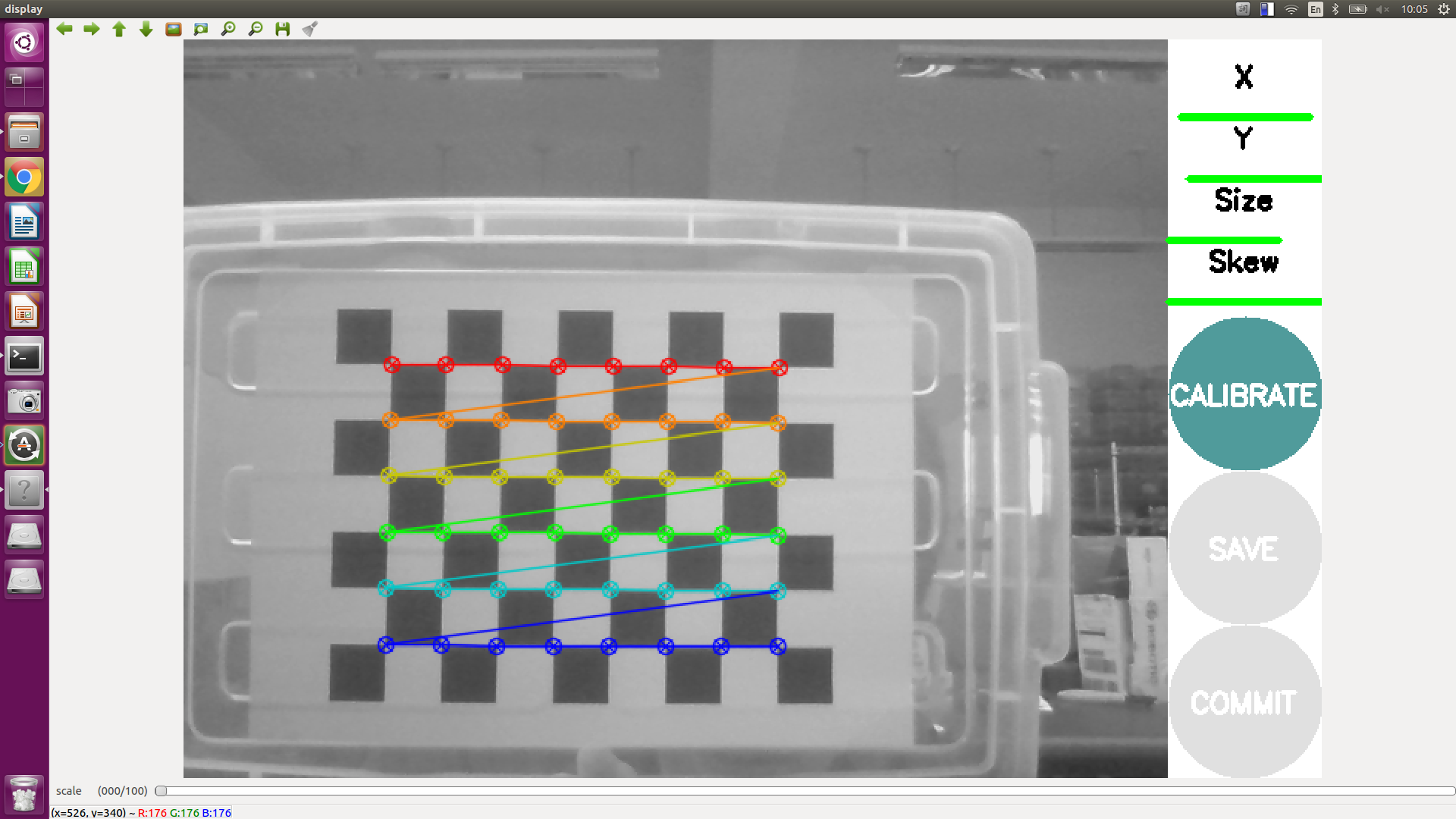
laptop $ **roslaunch usb\_cam usb\_cam-test.launch pixel\_format:=mjpeg**

Turn on calibration GUI

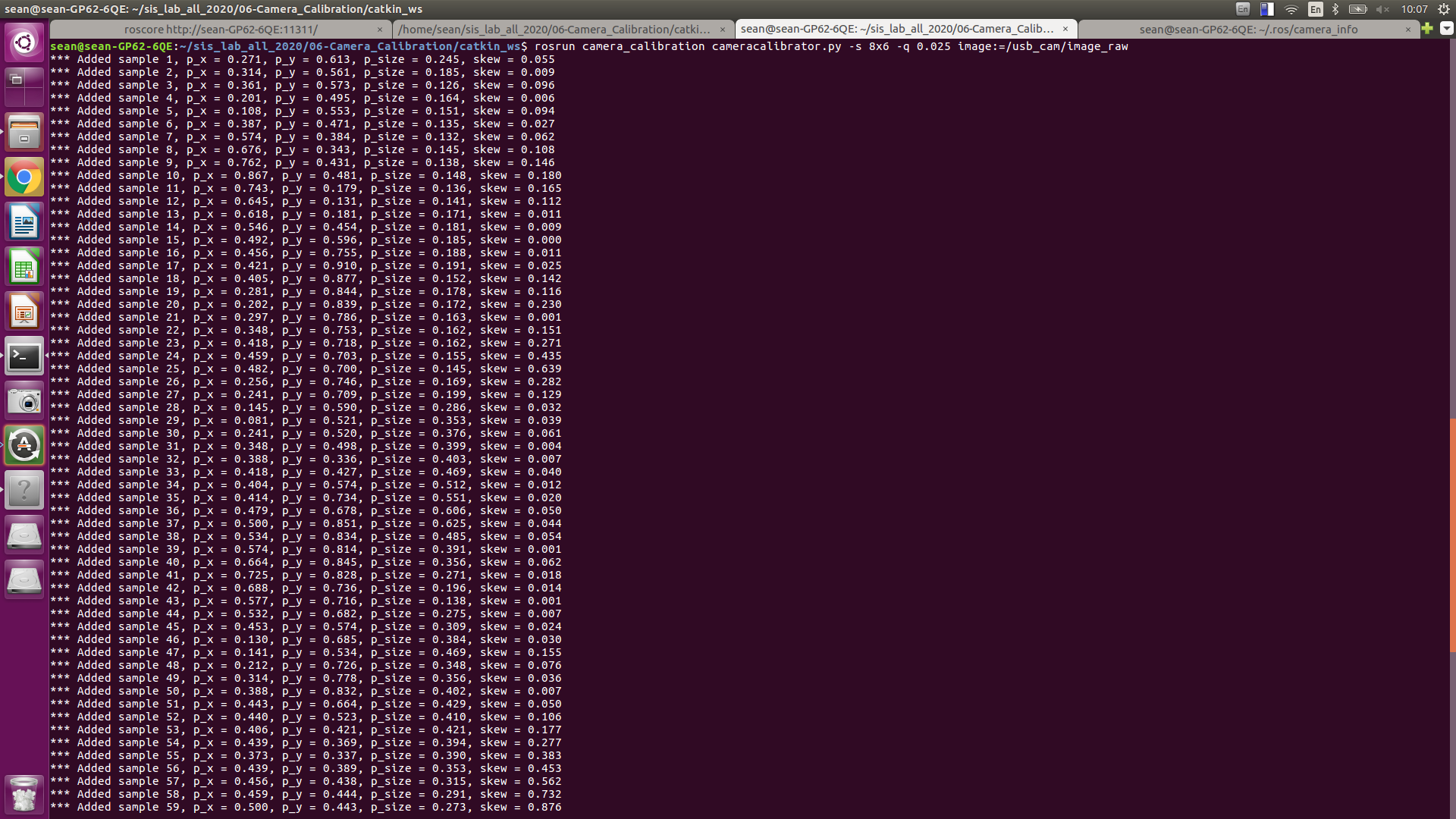
laptop $ **rosrun camera\_calibration cameracalibrator.py -s 8x6 -q 0.025 -c laptop\_camera --no-service-check image:=/usb\_cam/image\_raw**

You will see a window show up likes this

Put the chessboard pattern in the field of view of your camera **in different poses until the calibrate button is clickable.**

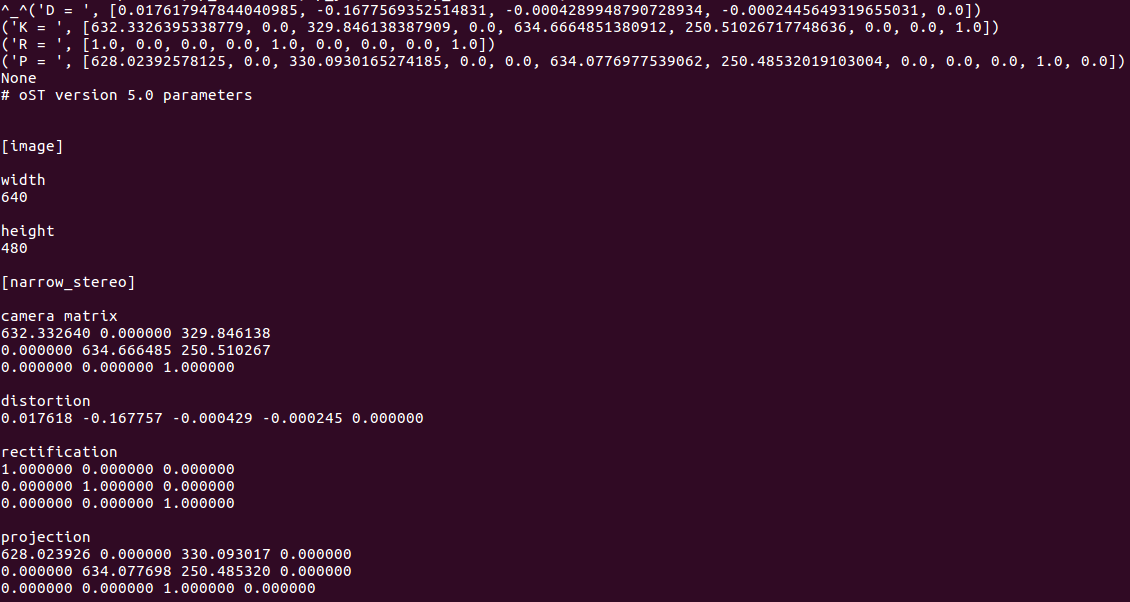


In the terminal, you will see each added data point

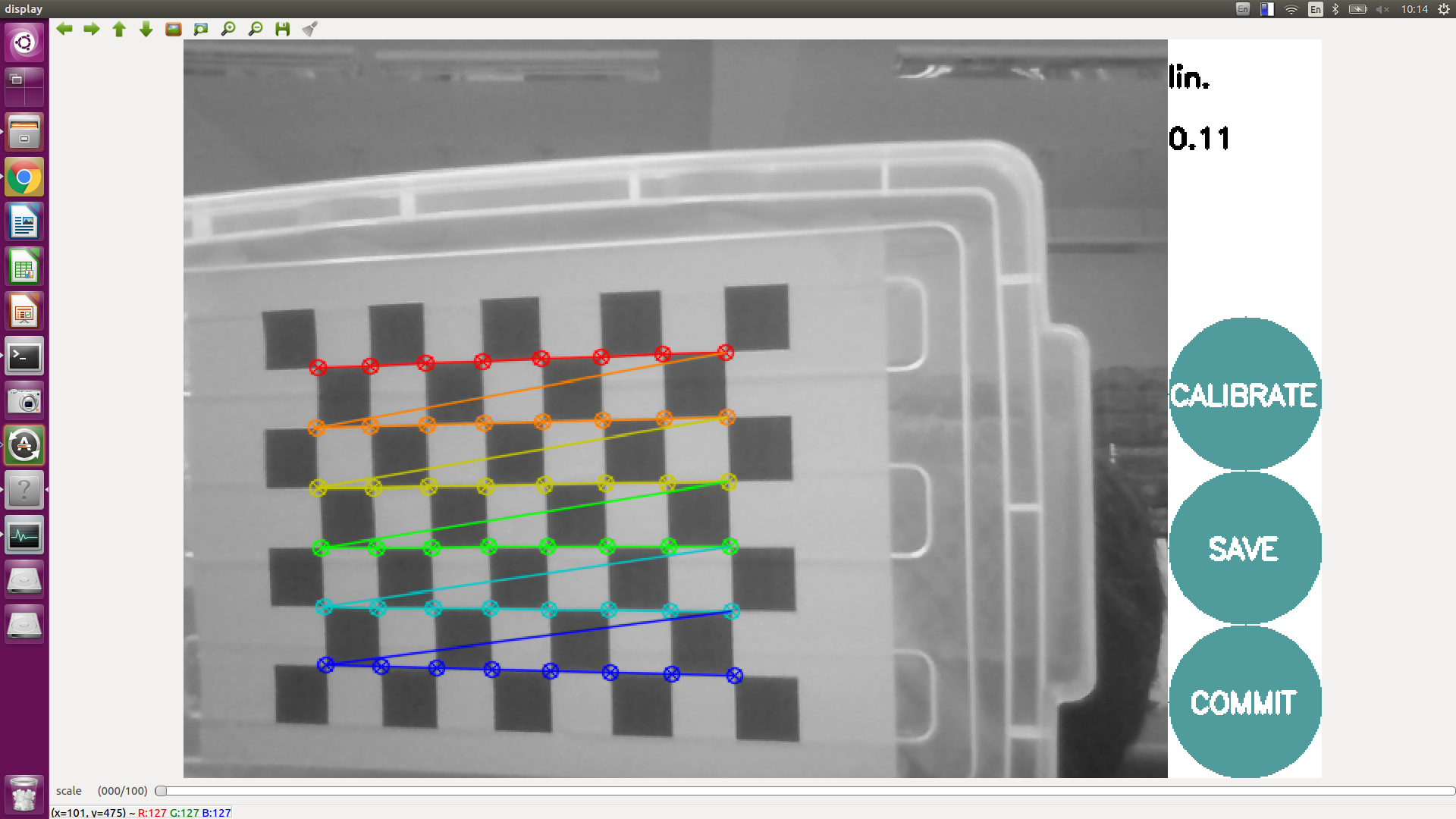


Now, click calibrate and wait for a while, the screen will turn into gray but it is still alive! The more data you collect, the more computation time it takes.

After the computation is complete, the window will return to its original state and the terminal will show the result



If you put the pattern in the FOV, you will see



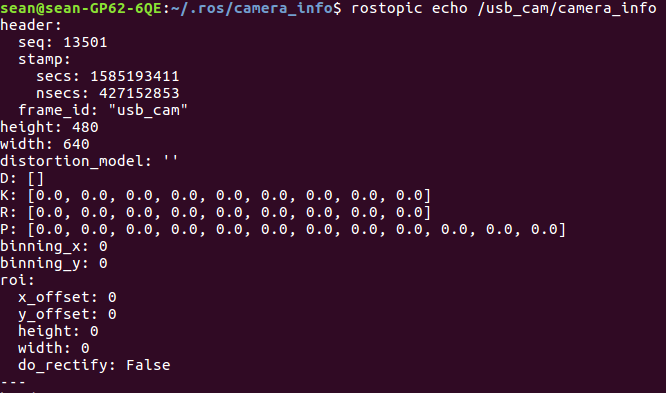
the value under `lin.` stands for the root-mean-square error in pixel, where the error means the distance from each middle corners to the line formed by two endpoint corners in each row.

Press save then all calibration images will be saved into '**/tmp/calibrationdata.tar.gz**', as well as the camera matrix with YAML format. Extract `**ost.yaml**` into `**~/.ros/camera\_info/**`

Now, press CTRL+C in terminal which you run cameracalibrator to stop the node.

Let’s take a look at the default intrinsic matrix of your camera

laptop **$ rostopic echo /usb\_cam/camera\_info**

The result will be like 

This is not a valid intrinsic coefficients, since the node manager can’t find the intrinsic file when we launch the node.

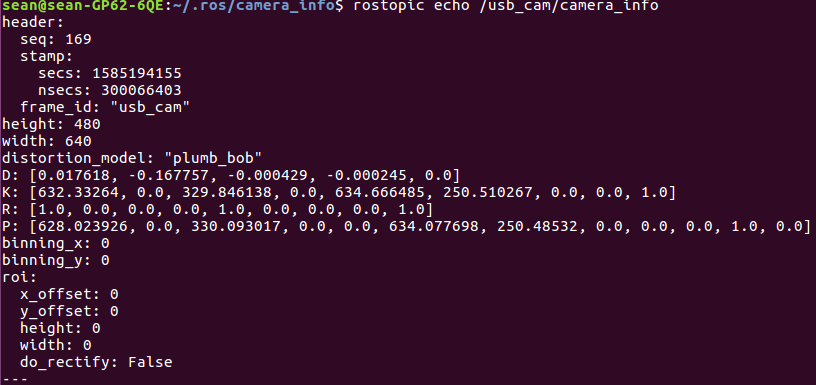
We have the intrinsic file after calibration, so let’s start the node again. CTRL+C in the terminal where you launch the usb\_cam node and relaunch it.

laptop **$** **roslaunch usb\_cam usb\_cam-test.launch**

And in another terminal

laptop **$ rostopic echo /usb\_cam/camera\_info**

You will see the intrinsic coefficients as same as you got in the calibration process



**Discussion:**

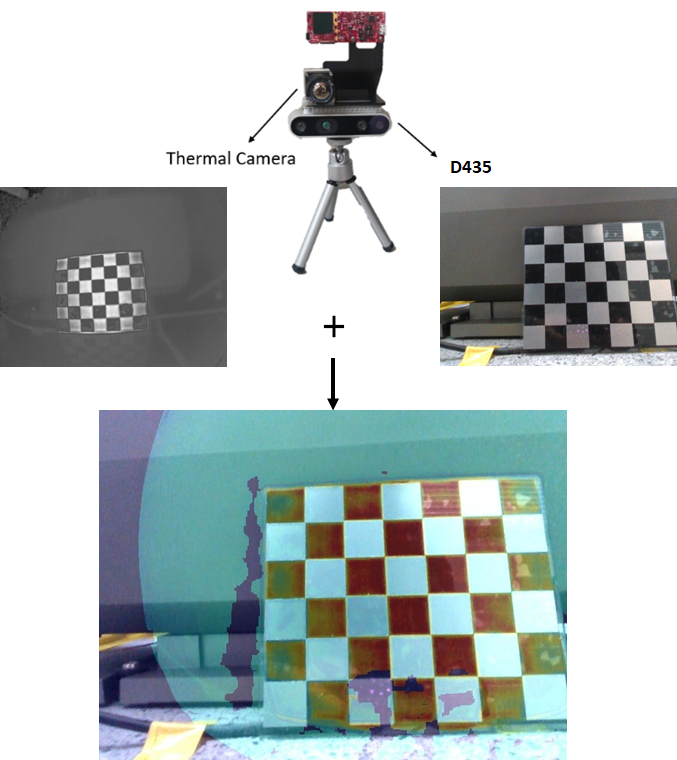
1. Launch **topic1\_discussion.launch** in **apriltags\_ros** and using the tag printing we gave in Lab 4, compare the result of tag position with valid and invalid intrinsic matrix.

What have you found?

**Hint: after tag in camera FOV, try: $ rosrun tf tf\_echo usb\_cam tag\_0**

### Topic/Activity 2 Align Thermal Image to Color Image

You can align thermal image to color image plane and feed into neural network for providing more information [1]. To do so, you have to measure intrinsic and extrinsic matrix simultaneously. In this topic, we try to align thermal image to color image with the help of custom-made chessboard.



#### **Topic 2.1 Thermal-Color Alignment**

(Estimated Time to Finish: 10 minutes)

Run the commands

laptop **$** **cd ~/sis-lab-all-2020/06-Camera-Calibration/script/topic2/cali\_thermal/align\_thermal\_to\_rgb/bin**

laptop **$** **./get\_extrinsic \**

**../example\_input/rgb\_intrinsic.txt \**

**../example\_input/thermal\_intrinsic.txt \**

**../example\_input/rgb0000.jpg \**

**../example\_input/depth0000.png \**

**../example\_input/thermal0000.jpg \**

**7**

I will not explain the source codes in depth but only introduce the concepts inside. Let’s take a look at **~/sis-lab-all-2020/06-Camera-Calibration/script/topic2/cali\_thermal/align\_thermal\_to\_rgb/src/get\_extrinsic.cpp**

From Line 50 to 84, it try to detect all chessboard corners in the color image, and using pixels of each corners, intrinsic coefficients and depth image to get there position w.r.t. D435 frame.

And from Line 85 to 108, it try to find all corners in the thermal image and check if the same number of corners in both color and thermal image; while this time we will not be able find the position of each corner w.r.t. thermal camera frame, since we don’t have depth information.

Then, the position in D435 frame and the pixel in thermal image plane of each detected corner is stored into `obj` and `img` respectively.

Now, it call the key function in this script, `**cv::calibrateCamera**`, with 2/3D information, initial guess of thermal camera intrinsic (including distortion coefficients) and extrinsic placeholder (rvec and tvec). The output will be stored in `thermal\_intrinsic`, `rvec` and `tvec`. (`rvec` is one representation of rotation matrix and can be converted into tf::Matrix3x3)

Both matrices are wrote into `camera\_model.txt`. Aligned result and the overlaid image are also saved in `thermal\_to\_rgb.jpg` and `combined.jpg` respectively.

And, of course, we can continue adding different views of images to make the result more accurate.

**Discussion:**

1. For pixels no depth information available, what method was used to estimated it in the code? See `**getPosition**` in `**helper.hpp**`

### Topic/Activity 3 Hand-Eye Calibration in Fixed-Eye Configuration

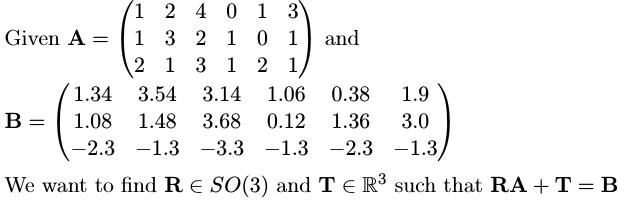
Hand-eye calibration is an important process in robotic manipulation, you can’t make your robot arm reach where your perception system predicted without hand-eye calibration.

Hand-eye calibration is the process measuring the transformation matrix from your camera coordinate to the robot arm coordinate, or the extrinsic matrix. There are many configuration for your camera setup, for example, it may be fixed on a frame, mounted in the wrist of the arm or even equipped on a rotatable motor. In this topic, we focus on fixed-eye configuration.

#### **Topic 3.1 Mathematic Background**

(Estimated Time to Finish: 10 minutes)

Consider the follow question:



The algorithm for finding such R and T in least square can be found in [2], I have wrote a simple Python script for you that implement the algorithm to solve the question.

laptop **$** **cd sis-lab-all-2020/06-Camera-Calibration/script/topic3 & python point\_registration.py**

In this example, we do know the correspondences between points, i.e., and its corresponding . However, in most cases we do not know such correspondences but we want to find the transformation that best match two sets. The problem becomes **iterative closest points**, or **ICP,** we have to predefine some convergent criteria, and start the iterative process: guess the correspondences, use this correspondences to find the transformation. The process will stop until at least one convergent criterion is meet.

To find the extrinsic matrix from camera to robot arm, we have to design a calibration tool, paste a calibration pattern on it and mount it on the robot. We have to design the board because we have to know the position of each corners on the pattern with respect to the end effector frame. The following tool is an example.

[2]

The pattern in this figure is called **ChAruco**, the combination of chessboard and Aruco, it can provide position of each numbered corner with high precision, and it is [open source](https://docs.opencv.org/3.4/df/d4a/tutorial_charuco_detection.html).

(And, of course, you can also use Apriltag)

~~~~

To collect data, we put the pattern in the FOV of camera in different poses, record the position of each corner in both camera and robot arm frame, then we can find the transformation of this two point sets. To get the transformation matrix from camera to robot arm, the source set is the position in camera frame and target set is the position in robot arm frame.

In assignment, you are asked to find the matrix from the data we collected.

## Assignment Tasks

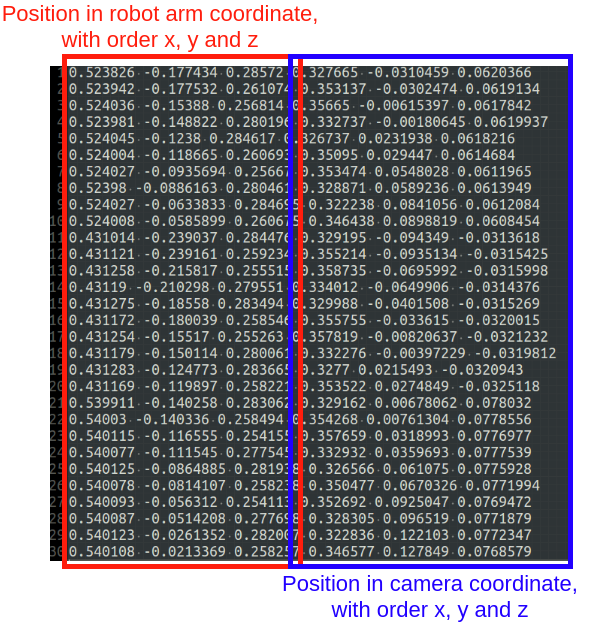
### Task 1. Hand-Eye Calibration

In this assignment, you have to create a ROS package, named **hand\_eye\_calibration**, write a C++ node which implement the algorithm in [2] with the data we provided and find the extrinsic matrix from camera to robot arm.

Eigen is a widely-used linear algebra library in C++, so you don’t have to worry about how to do singular value decomposition.

You don’t have to link library when you use Eigen since it is a header-only library. All you have to do is to include the header files and make the compiler know where they are. (Typically they located at `**/usr/include/eigen3/Eigen**`)

The collected data are wrote in `sis\_lab\_all\_2020/06-Camera\_Calibration/data/calibration.txt` and formatted as the following:



**Rules:**

* **YOU HAVE TO WRITE A FUNCTION TO PARSE THE FILE, HARDCODING THE DATA POINTS IS NOT ALLOWED.**
* **Notice also that YOU SHOULD NOT ASSUME THE NUMBER OF DATA POINTS IS KNOWN.**
* **YOU SHOULD EDIT `CMakeLists.txt` so that after you run**

**>> catkin\_make command, you get the executable file in devel/lib/hand\_eye\_calibration**

**Expected Result:**

[[ 0.00609639, 0.00918406, 0.99993924, 0.46044048],

[-0.04944292, 0.99873755, -0.00887158, -0.13002717],

[-0.99875835, -0.04938583, 0.00654278, 0.61125747],

[ 0.00000000, 0.00000000, 0.00000000, 1.00000000]]

**HINT**: [Creating a ROS Package](http://wiki.ros.org/ROS/Tutorials/CreatingPackage)

## Reference

[1] Shivakumar, Shreyas S., et al. "PST900: RGB-Thermal Calibration, Dataset and Segmentation Network." *arXiv preprint arXiv:1909.10980* (2019).

[2] Arun, K. Somani, Thomas S. Huang, and Steven D. Blostein. "Least-squares fitting of two 3-D point sets." *IEEE Transactions on pattern analysis and machine intelligence* 5 (1987): 698-700.

[3] F. C. Park and B. J. Martin, "Robot sensor calibration: solving AX=XB on the Euclidean group," in *IEEE Transactions on Robotics and Automation*, vol. 10, no. 5, pp. 717-721, Oct. 1994. (**Camera-on-hand configuration**)