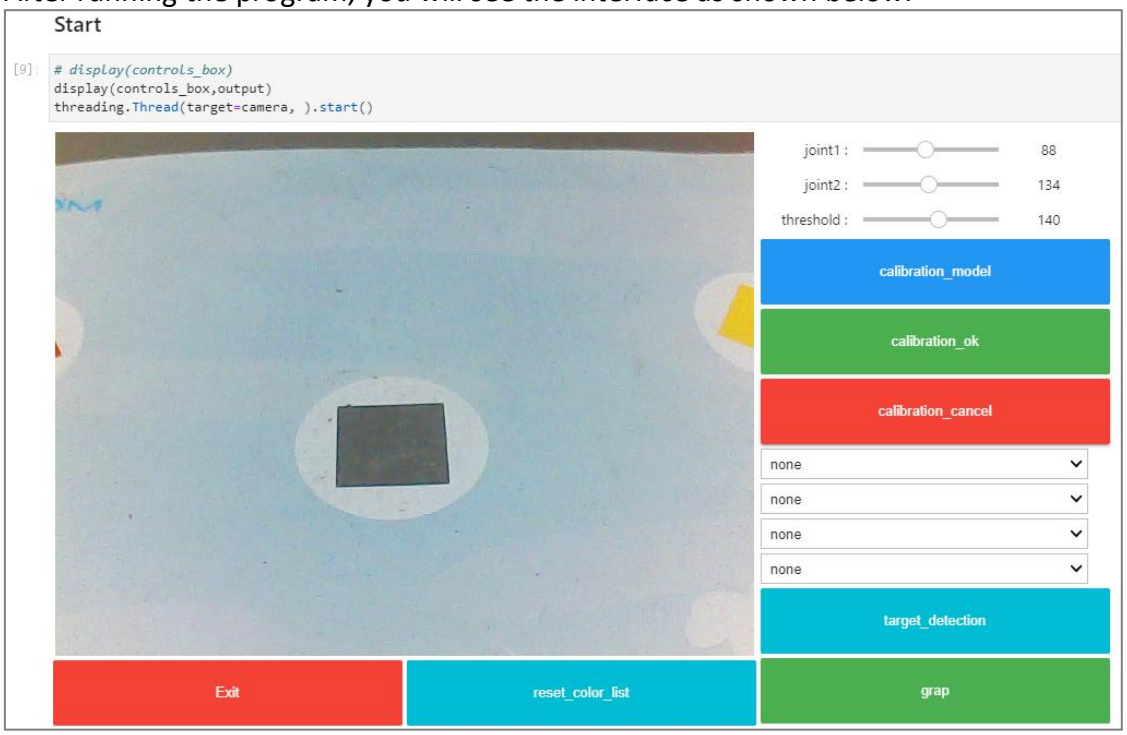
Path: dofbot_ws/src/dofbot_color_identify/scripts/dofbot_config.py

1. Box calibration

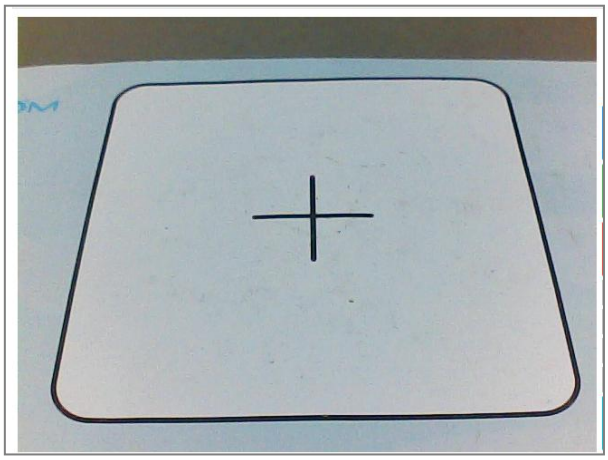
In order to convert the position of the center of the building block in the camera's field of view into the actual distance from the base coordinate of the robot arm.

We need to perform box calibration.

After running the program, you will see the interface as shown below.



1) Place map as shown below.

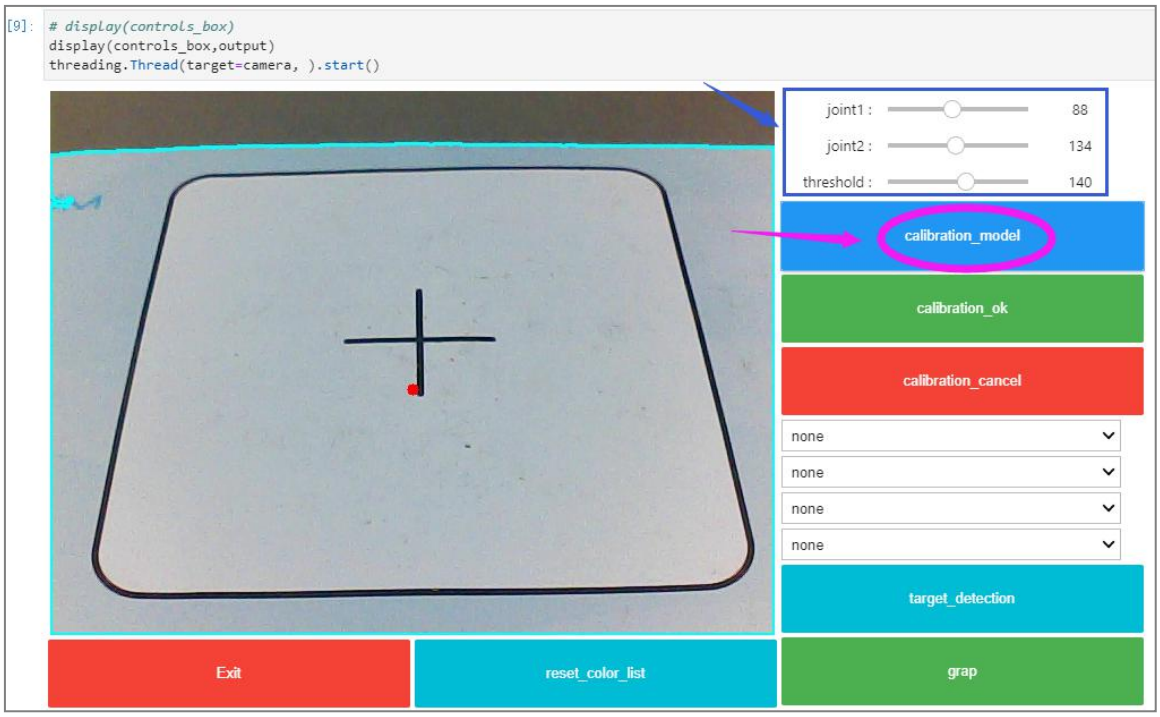


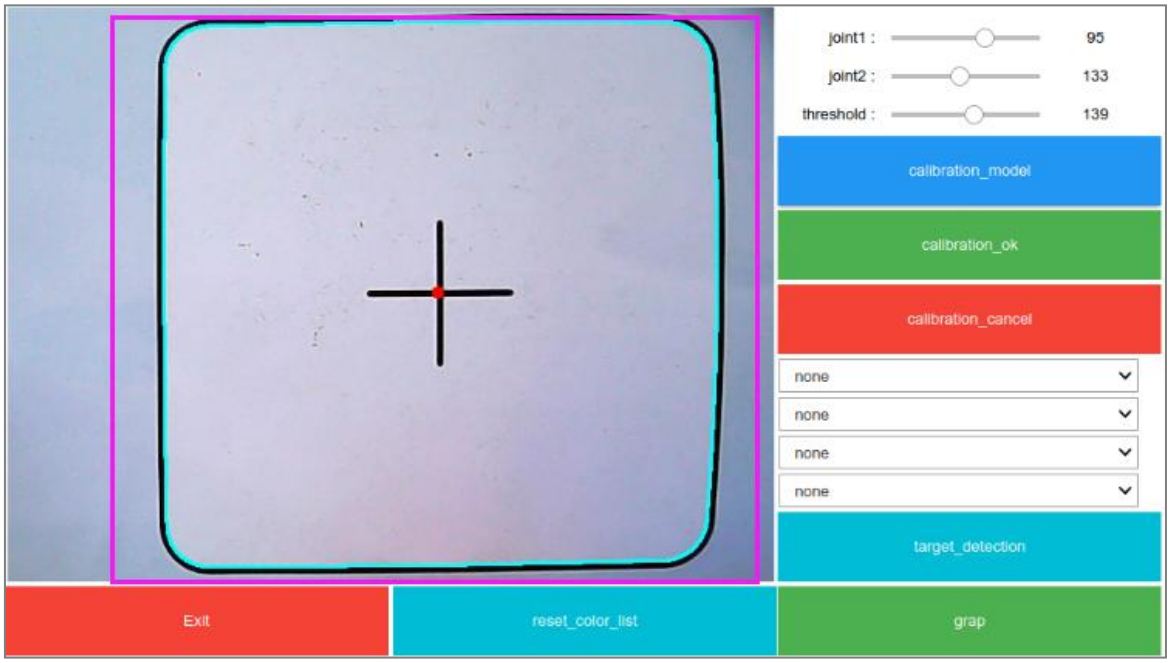
2) Click “Calibration_model”, adjust the slider above to ensure that the camera can accurately detect the edge of the map, as shown below.

[joint1] Control the left and right adjustment of the No. 1 servo.

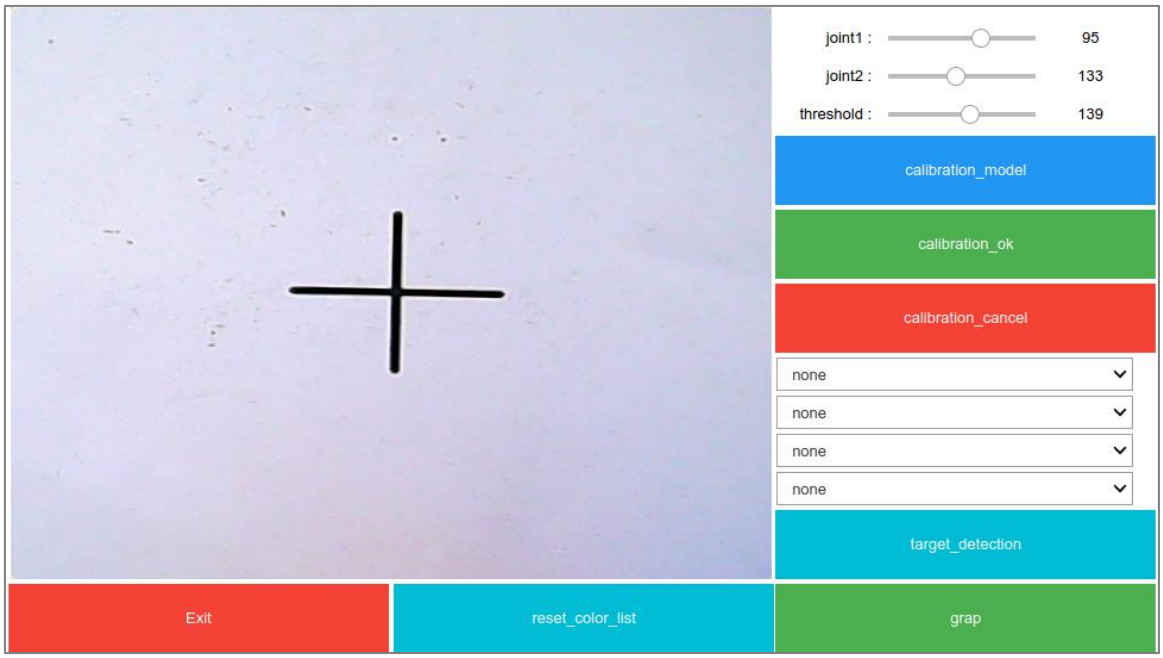
[joint2] Control the up and down adjustment of No. 2 servo.

[Threshold] Adjust image edge binarization detection threshold.





3) Click “Calibration_OK”. You will following interface.



If an error occurs during the calibration, click the [calibration_cancel] button to cancel the calibration result, and then perform the calibration again.

2. About code

2.1 Image processing, extract contour points

```
gray = cv.cvtColor(self.image, cv.COLOR_BGR2GRAY)

gray = cv.GaussianBlur(gray, (5, 5), 1)

ref, threshold = cv.threshold(gray, self.threshold_num, 255, cv.THRESH_BINARY)

kernel = np.ones((3, 3), np.uint8)

blur = cv.morphologyEx(threshold, cv.MORPH_OPEN, kernel, iterations=4)

contours, hierarchy = cv.findContours(blur, mode, method)
```

2.2 Draw contour, return the edge point of contour

```
for i, c in enumerate(contours):

    area = cv.contourArea(c)

    if h * w / 2 < area < h * w:

        mm = cv.moments(c)
        if mm['m00'] == 0:
```

```
        continue
        cx = mm['m10'] / mm['m00']
        cy = mm['m01'] / mm['m00']

        cv.drawContours(self.image, contours, i, (255, 255, 0), 2)

        dp = np.squeeze(cv.approxPolyDP(c, 100, True))

        cv.circle(self.image, (np.int(cx), np.int(cy)), 5, (0, 0, 255), -1)
```

2.3 Perspective transformation, transform the detected outline perspective into (640,480) image

```
for i in range(len(dp)):
if dp[i][0]<320 and dp[i][1]<240: upper_left=dp[i]
   if dp[i][0]<320 and dp[i][1]>240: lower_left=dp[i]
   if dp[i][0]>320 and dp[i][1]>240: lower_right=dp[i]
   if dp[i][0]>320 and dp[i][1]<240: upper_right=dp[i]

pts1 = np.float32([upper_left, lower_left, lower_right, upper_right])

pts2 = np.float32([[0, 0], [0, 480], [640, 480], [640, 0]])

M = cv.getPerspectiveTransform(pts1, pts2)

Transform_img = cv.warpPerspective(image, M, (640, 480))
```