Color stacking

1. Color recognition instructions

Color recognition uses the HSV color recognition function. The path where the HSV color calibration file is saved is ~/jetcobot_ws/src/jetcobot_color_identify/scripts/HSV_config.txt. If the color recognition is not accurate enough, please recalibrate the HSV value of the block color according to the [Color threshold adjustment color block calibration] course. After the calibration operation is completed, it will be automatically saved to the HSV_config file and the program will be rerun without additional code modification.

The path where the file for the robot position calibration is saved is ~/jetcobot_ws/src/jetcobot_color_identify/scripts/XYT_config.txt. After calibration, restart the program and click the calibration mode to automatically read the file information to reduce repeated calibration actions.

2. Important code explanation

Code path: ~/jetcobot_ws/src/jetcobot_color_identify/scripts/color_sorting_stacking.ipynb

~/jetcobot_ws/src/jetcobot_utils/src/jetcobot_utils/grasp_controller.py

Since the camera may have deviations in the gripping position of the building block, it is necessary to increase the deviation parameter to adjust the deviation value of the robot arm to the recognition area. The type corresponding to color sorting and color stacking is "color", so it is necessary to change the offset parameter under type == "color". The X offset controls the front and back offset, and the Y offset controls the left and right offset.

```
# Get the XY offset according to the task type,
    def grasp_get_offset_xy(self, task, type):
        offset_x = -0.012
        offset_y = 0.0005
        if type == "garbage":
            offset_x = -0.012
            offset_y = 0.002
        elif type == "apriltag":
            offset_x = -0.012
            offset_y = 0.0005
        elif type == "color":
            offset_x = -0.012
            offset_y = 0.0005
        return offset_x, offset_y
```

The coordinate value of the stacking area where the object is placed. If the coordinate value of the placement position is inaccurate, you can modify it appropriately.

```
# stacking
    def goStackingNum1Pose(self):
        coords = [140, -160, 110, -180, -2, -43]
        self.go_coords(coords, 3)

    def goStackingNum2Pose(self):
```

```
        coords = [145, -160, 145, -180, -2, -43]
        self.go_coords(coords, 3)

    def goStackingNum3Pose(self):
        coords = [145, -160, 175, -180, -2, -43]
        self.go_coords(coords, 3)

    def goStackingNum4Pose(self):
        coords = [145, -160, 205, -180, -2, -43]
        self.go_coords(coords, 3)
```

3. Start the program

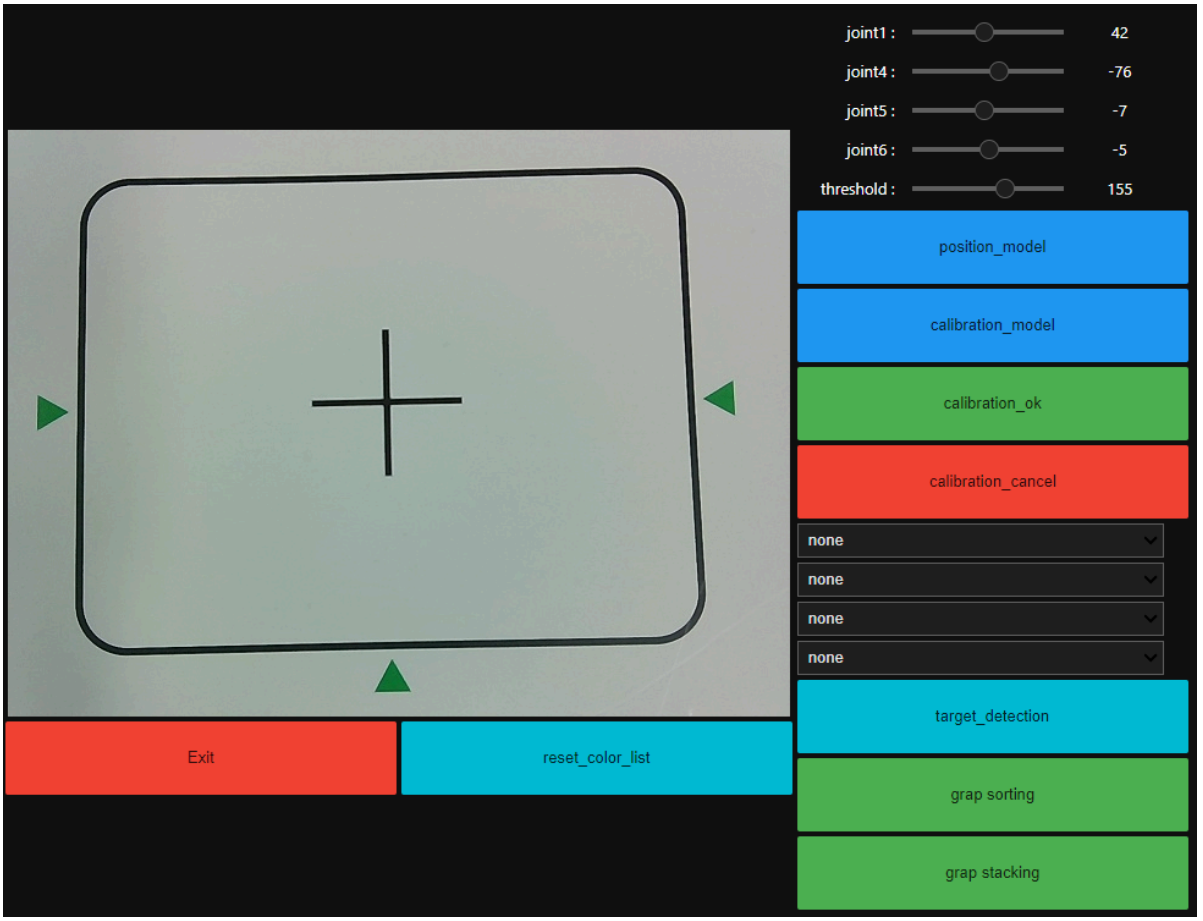
Start the program

Open the jupyterlab webpage and find the corresponding .ipynb program file.

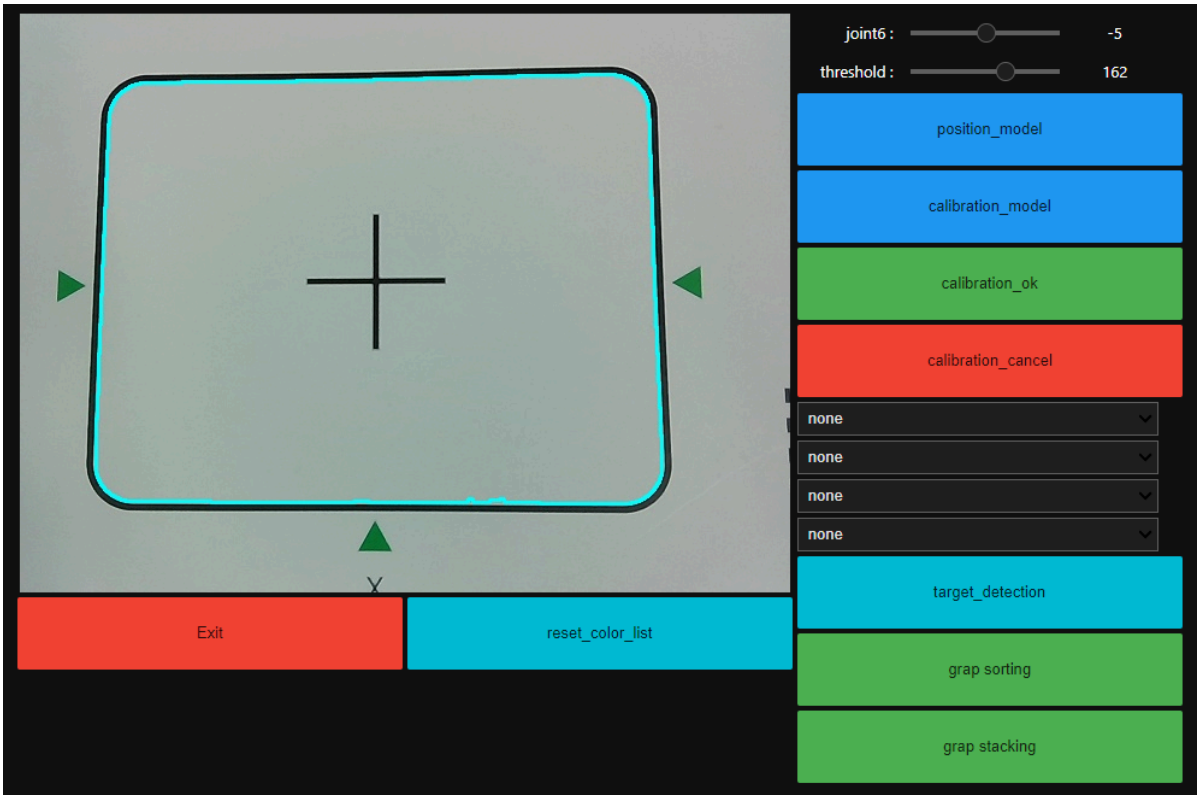
Then click Run All Commands.

4. Experimental operation and effect

After the program is running, the jupyterlab webpage will display the control, the camera screen on the left, and the functions of the relevant buttons on the right.

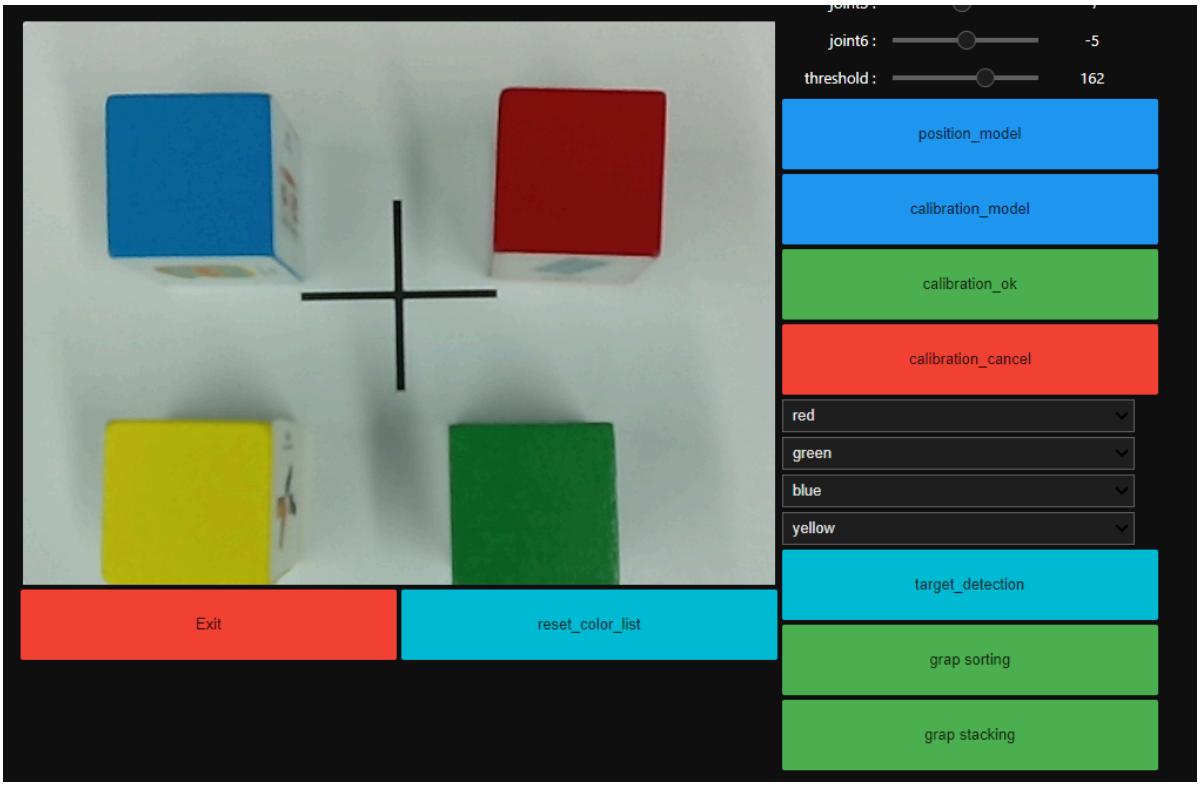


Click the [position_model] button, drag the joint angle above, update the position of the robot arm, and make the recognition area in the middle of the entire image. Then click [calibration_model] to enter the calibration mode, and adjust the robot arm joint slider and threshold slider above to overlap the displayed blue line with the black line of the recognition area.

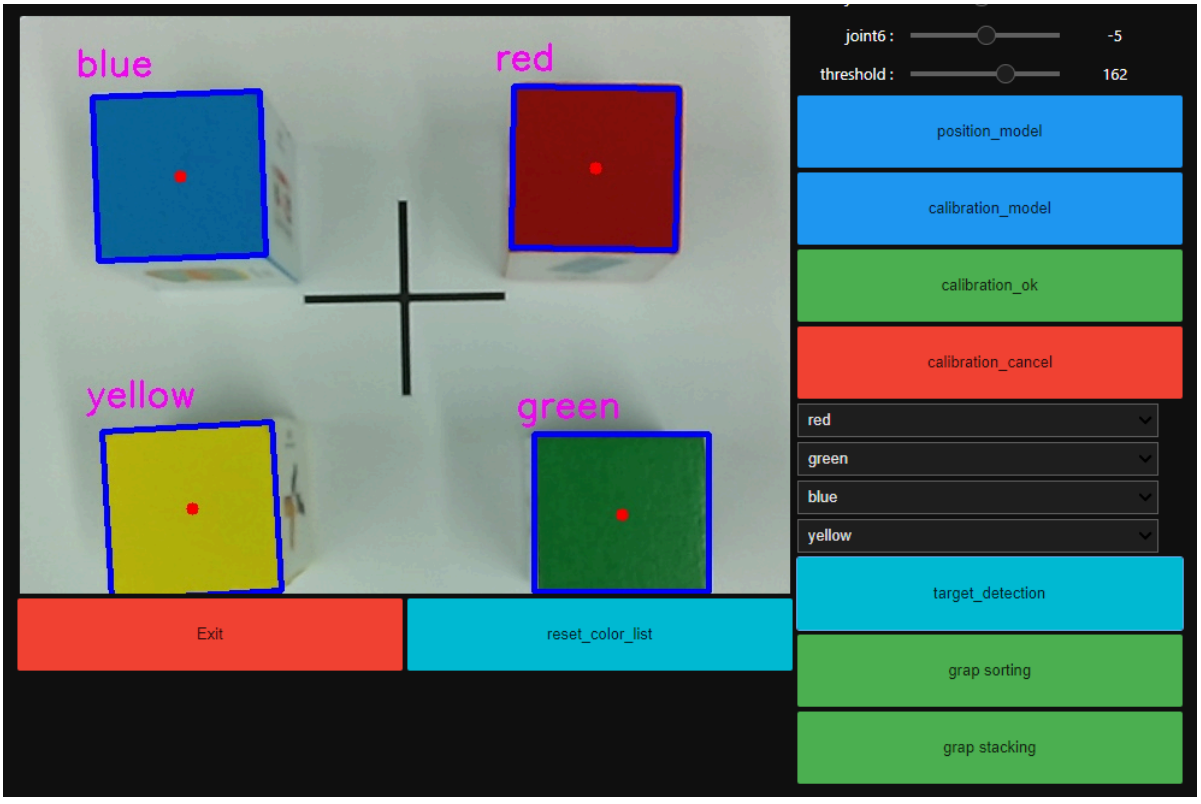


Click [calibration_ok] to calibrate OK, and the camera screen will switch to the recognition area view.

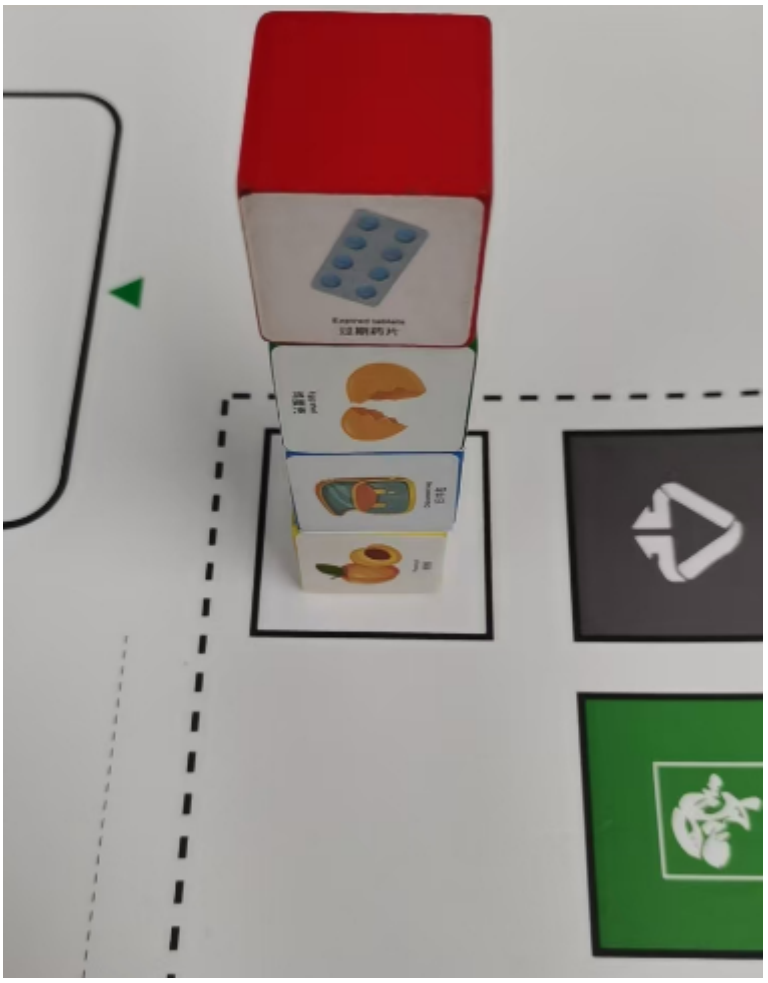
Place the building blocks in the recognition area, and then select the color order on the right.



Then click [target_detection] to start color recognition. If the color recognition is inaccurate, please calibrate the color first and then run the program again.



Then click the [grap stacking] button to start stacking. The system will grab and stack the recognized colors in order.



If you need to exit the program, please click the [Exit] button.