# apriltag stacking

## 1. apriltag sorting instructions

The file path for robot arm position calibration 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, reducing repeated calibration actions.

## 2. About code

Code path: ~/jetcobot_ws/src/jetcobot_apriltag/scripts/apriltag_sorting_stacking.ipynb

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

Since the camera may have deviations in the position of the building block, it is necessary to add deviation parameters to adjust the deviation value of the robot arm to the recognition area.

The type corresponding to apriltag sorting and apriltag stacking is "apriltag", so it is necessary to change the offset parameter under type == "apriltag".

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.

If the coordinate of the placement position is inaccurate, you can modify this coordinate value 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 program

**Start roscore**

- If you are using Jetson Orin NX/Jetson Orin Nano board. You need to enter the Docker environment using the following command.
- Then, run roscore

```
sh ~/start_docker.sh
roscore
```

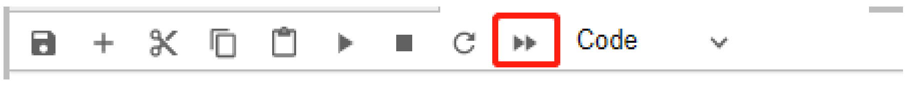
- If you are using Jetson  Nano board. You need to enter the following command directly.

```
roscore
```

**Start the program**

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

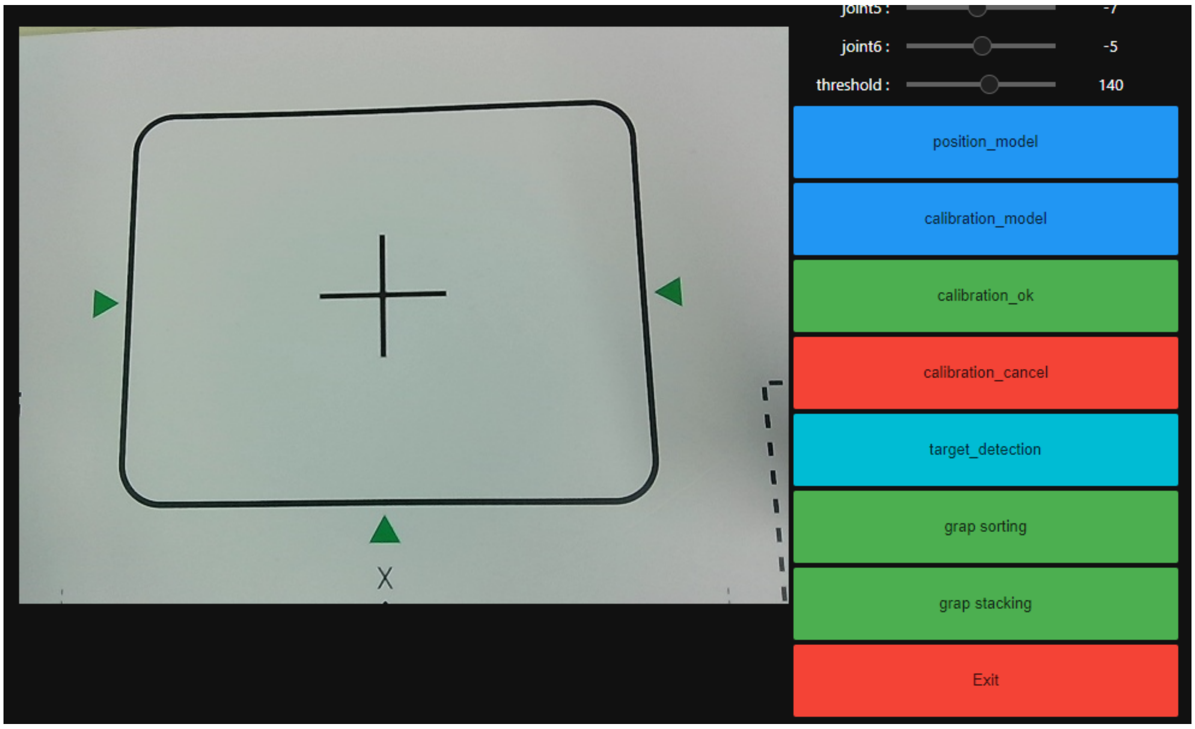
Click the Run button to run the entire code.



## 4. Experimental operation and results

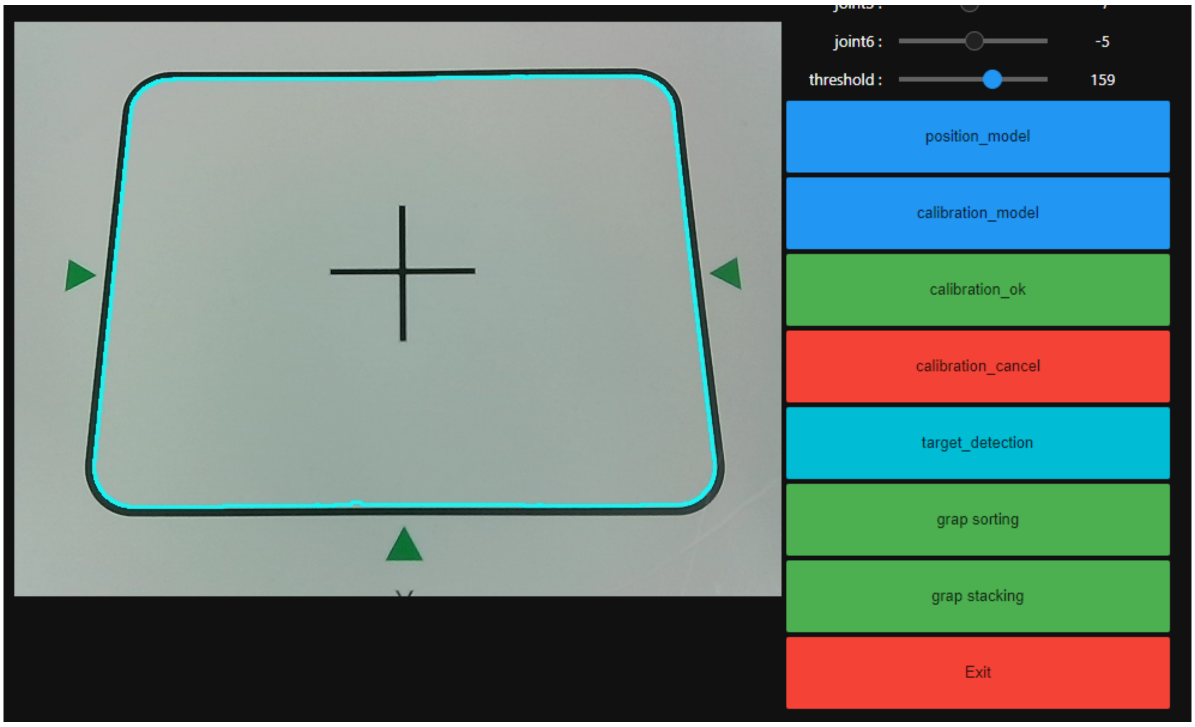
After the program runs, the Jupyterlab webpage will display the control button.

Camera image on the left side, related buttons on the right side.



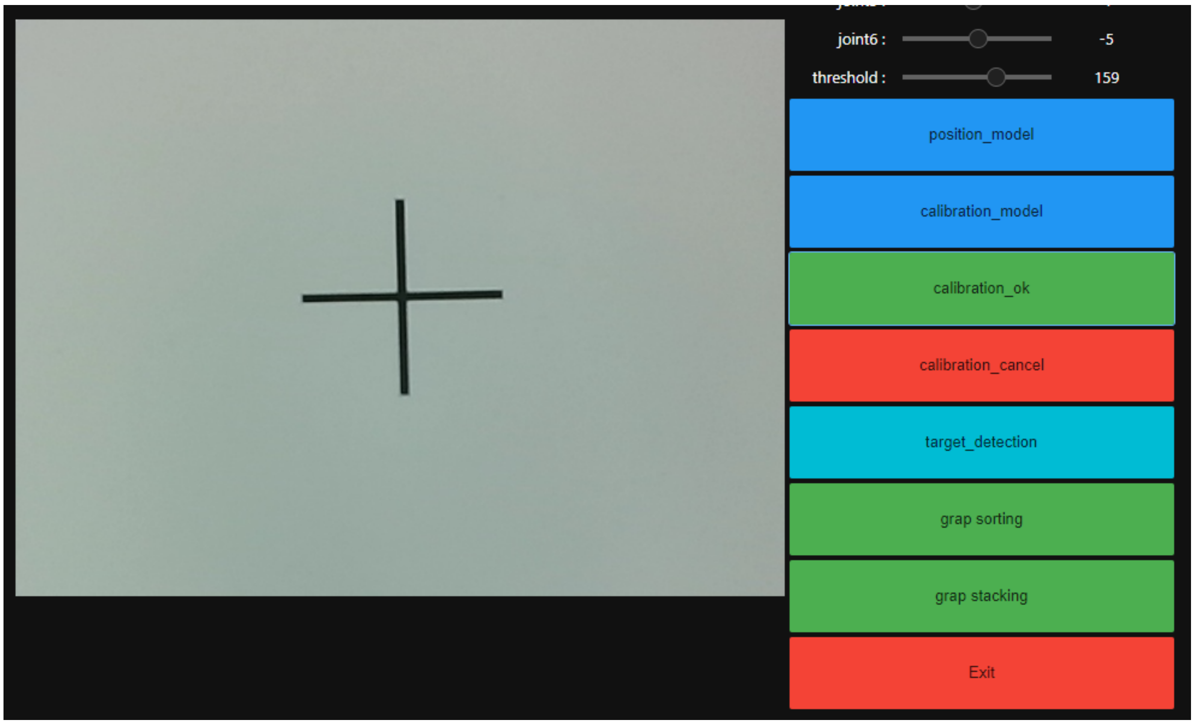
Click the 【position_model】button, drag the joint angle above, update the position of the robot arm, and make the recognition area in the middle of the entire image.

Then, click 【calibration_model】 to enter the calibration mode, and adjust the robot arm joint slider and threshold slider above to make the displayed blue line overlap with the black line of the recognition area.

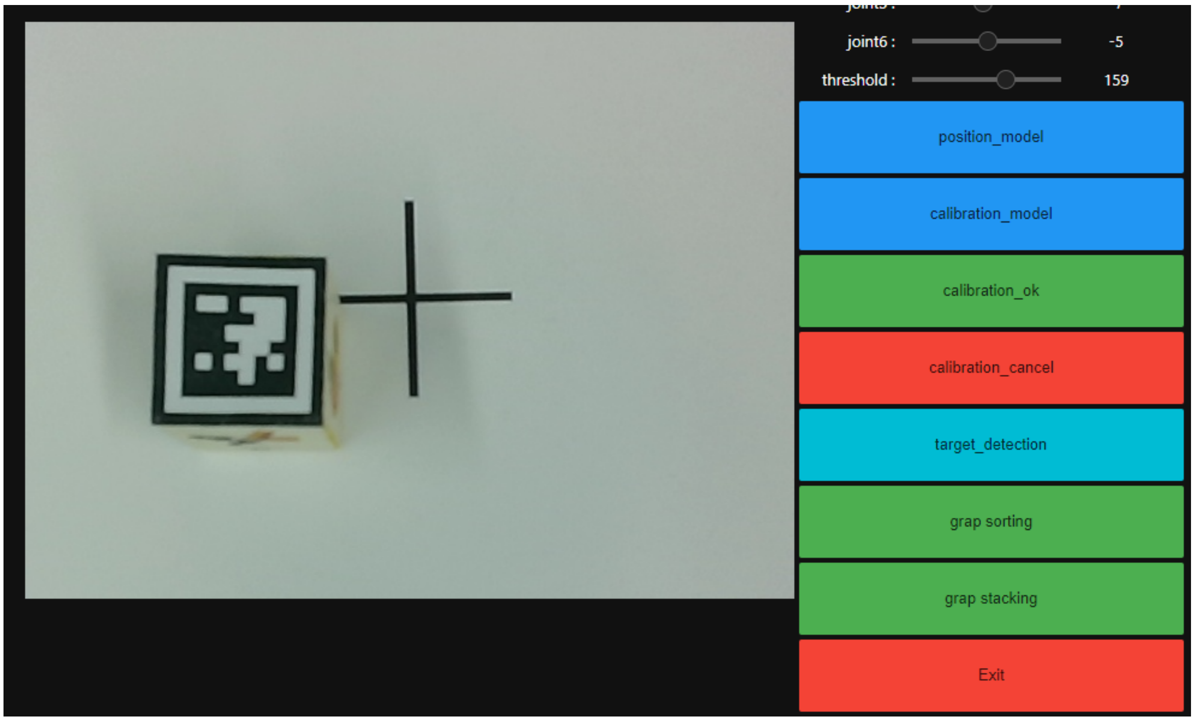


Click 【calibration_ok】 to calibrate OK.

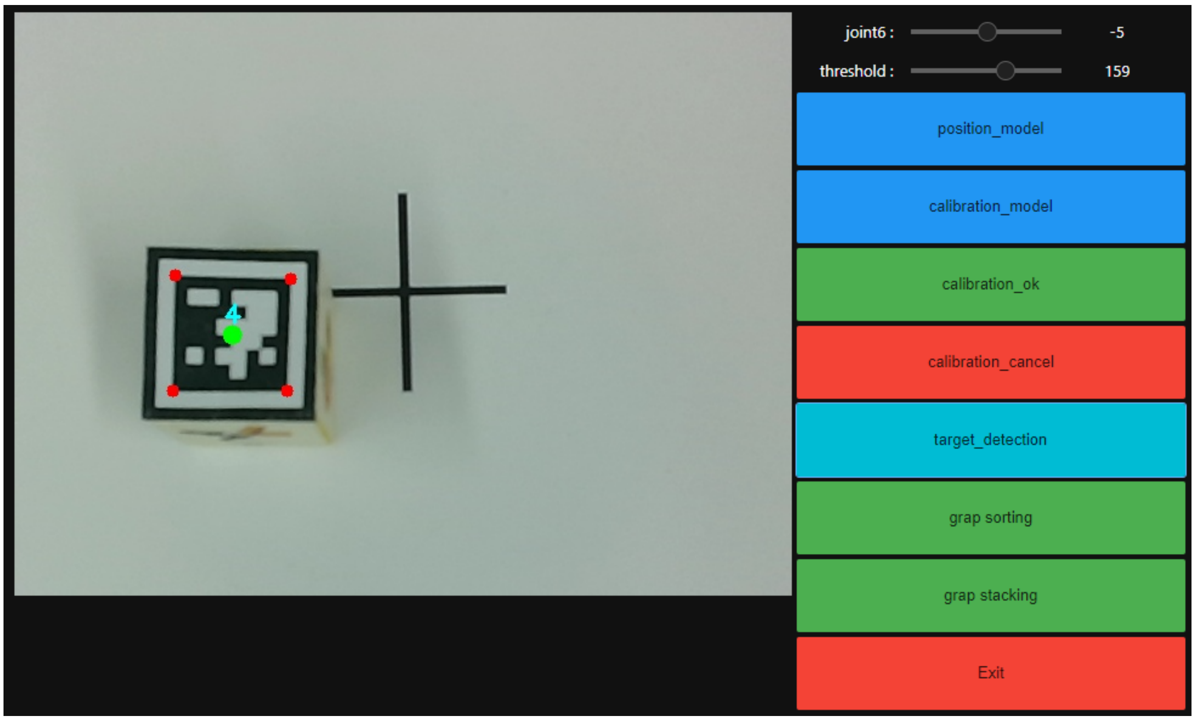
The camera screen will switch to the recognition area perspective.



Place the blocks into the recognition area and select the color order on the right.



Click 【target_detection】to start label code recognition.



Then, click the [grap stacking] button to start stacking.

The system will identify the ID number of the tag code and grab the building blocks to the stacking area according to the tag number and stack them.

If you need to exit the program, please click the 【Exit】 button.