Color block volume measurement

This lesson uses the Raspberry Pi as an example. For Raspberry Pi and Jetson-Nano boards, you need to open a terminal on the host computer and enter the command to enter the Docker container. Once inside the Docker container, enter the commands mentioned in this lesson in the terminal. For instructions on entering the Docker container from the host computer, refer to **[01. Robot Configuration and Operation Guide] -- [4.Enter Docker (For JETSON Nano and RPi 5)]**. For RDKX5 and Orin boards, simply open a terminal and enter the commands mentioned in this lesson.

1. Program Functionality

> After running the program, it will recognize the shape of the selected color block, retrieve relevant data about the block, and calculate its volume.

2. Code Reference Path

```
~/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_depth/yahboomcar_depth/Basic/calculate_volume.py
~/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_depth/yahboomcar_depth/astra_common.py
```

- calculate_volume.py

  Calculates the data of the color block based on its center point and the volume of the color block based on the recognized shape.

- astra_common.py
  - Color tracking (color_follow class): Identifies and tracks objects of a specific color using the HSV color space
  - Image processing tools: Includes multi-image stitching (manyimgs function) and shape detection (is_regular_square function)
  - File read/write functions: Used to save/read HSV color ranges

3. Program startup

3.1. Startup commands

For Raspberry Pi and Jetson-Nano boards, you need to enter the Docker container first. For RDKX5 and Orin controllers, this is not necessary.

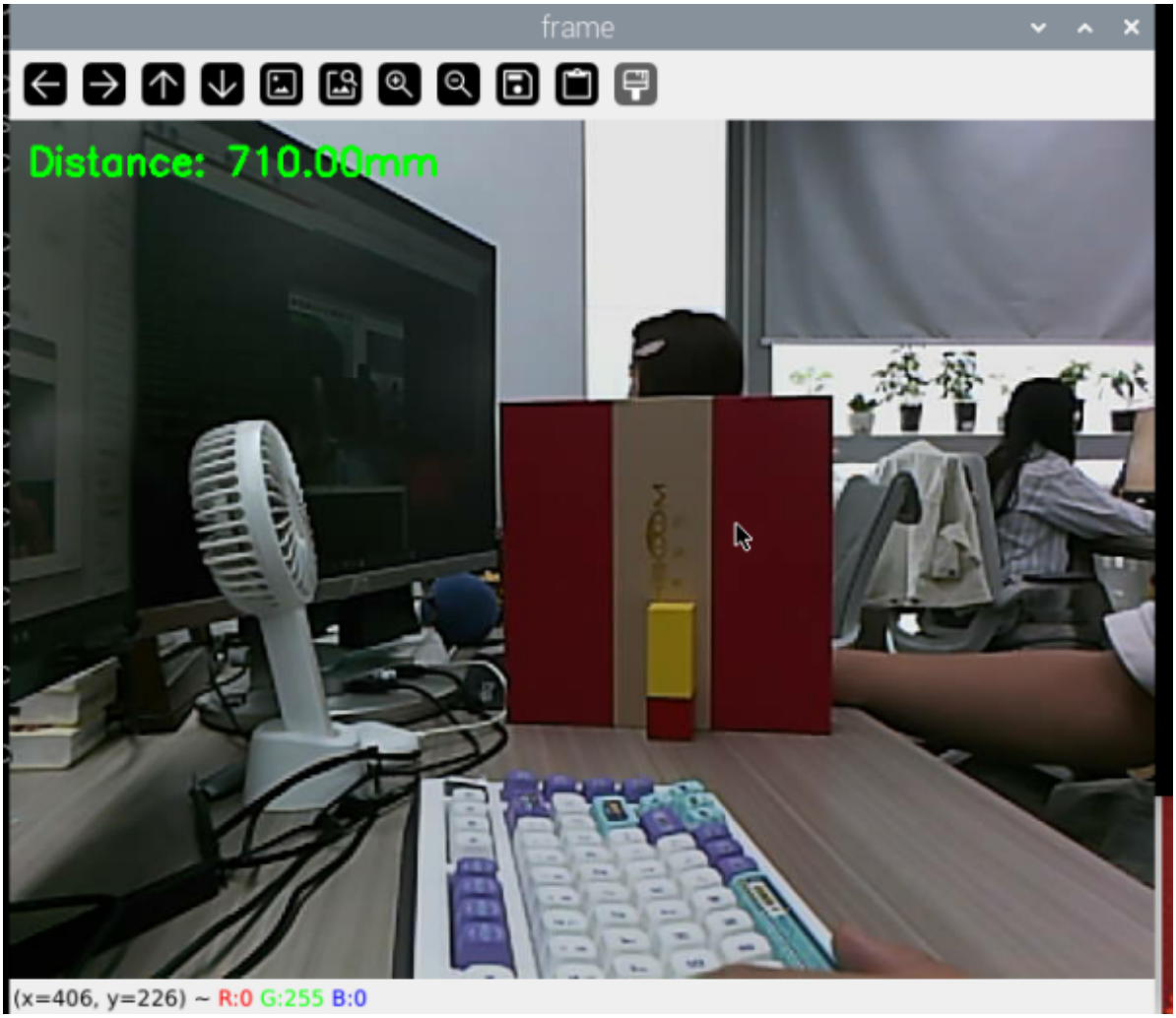
Enter the Docker container (see [Docker course] --- [4. Docker Startup Script] for steps).

All the following commands must be executed within the same Docker container **(see [Docker course] --- [3. Docker Submission and Multi-Terminal Access] for steps).**

Terminal input:

```
ros2 launch ascamera hp60c.launch.py
ros2 run yahboomcar_depth calculate_volume
```

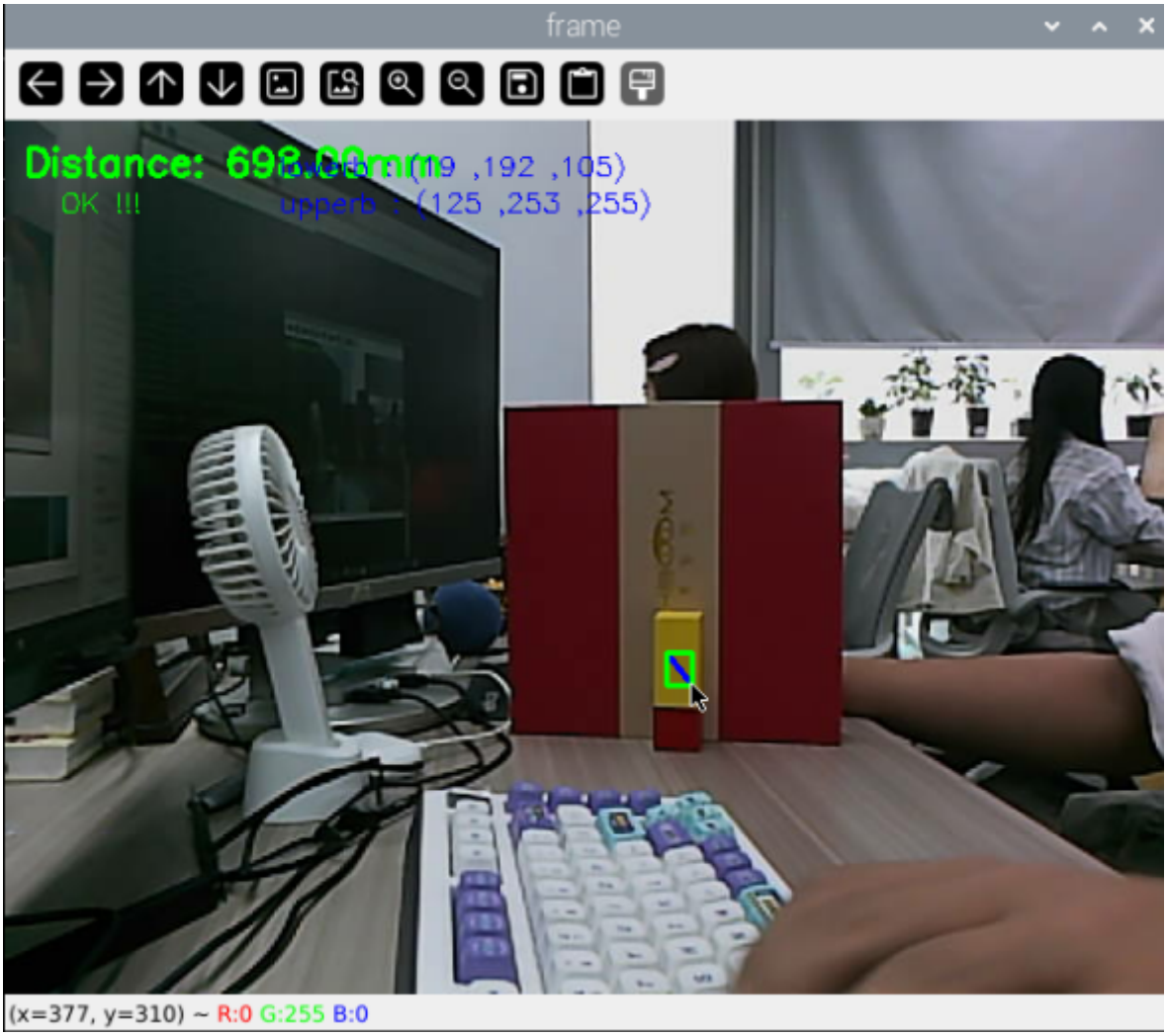
After the program runs, click the mouse at the calibration distance of the object to be calibrated, and press the space bar to calibrate.



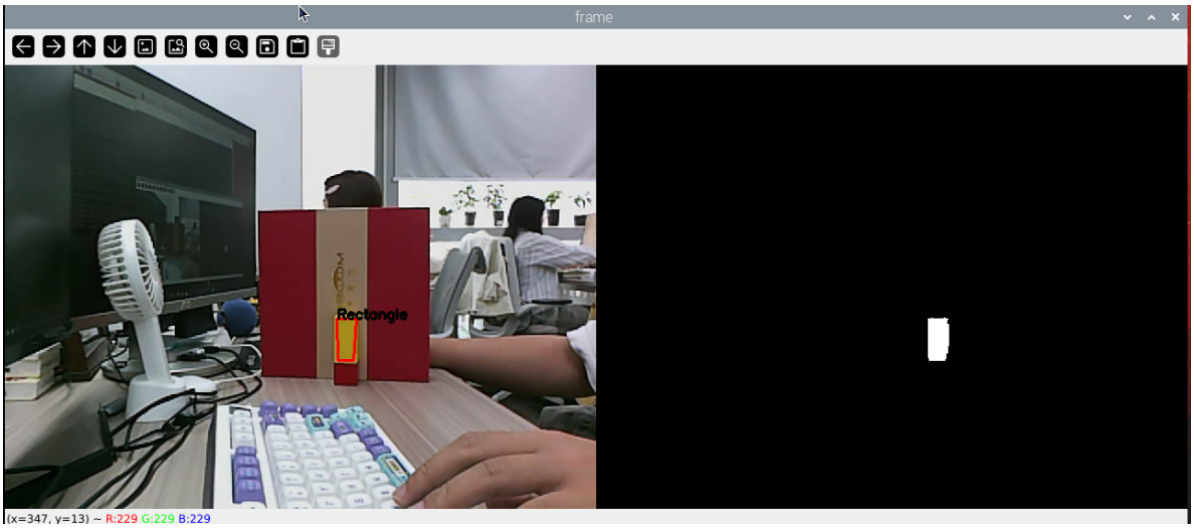
The terminal prints the calibrated coordinates and distance.

```
root@raspberrypi:/# ros2 run yahboomcar_depth calculate_volume
获取深度值： (426, 211) = 0.702
```

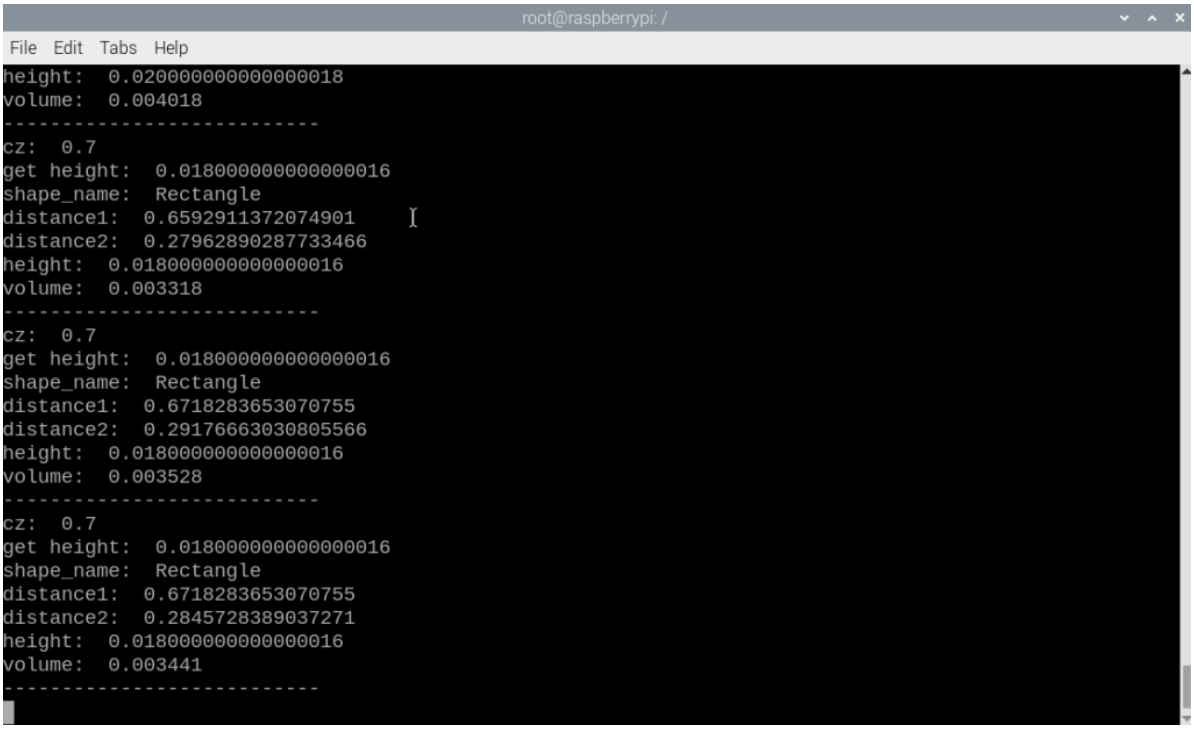
After the program starts, place the color swatch in front of the calibrated object. Select an area within the color swatch with the mouse and obtain the HSV value.



Resulting image. **Note: The position of the color swatch needs to be larger than the blind zone of the depth camera; a distance greater than 0.3 meters is recommended.**



The program uses the HSV value to filter out other colors, retaining only colors within the HSV value range. The program identifies the shape of the color block, calculates its data based on its center point, and then calculates its volume based on the identified shape. (The volume is not necessarily accurate; it's a measurement.)



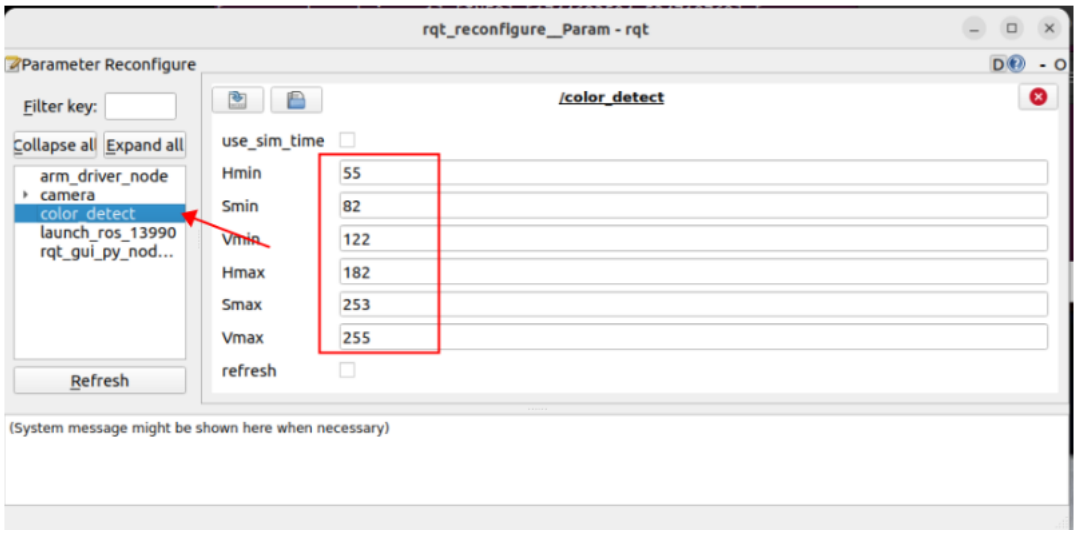
3.2 Dynamic Parameter Adjustment

If the HSV values of the selected color do not correctly identify the color block shape, you may need to fine-tune the HSV values. Enter the following command in the terminal to start the dynamic parameter adjuster.

In the terminal, enter:

```
ros2 run rqt_reconfigure rqt_reconfigure
```

You can modify the HSV values by changing the corresponding values.



☑ After modifying the parameters, click a blank area in the GUI to write the parameter values. Note that this will only take effect for the current boot. To make it permanent, you need to modify the parameters in the source code.

As shown in the figure above,

4. Core Code

4.1. calculate_volume.py

This program has the following main functions:

- Opens the camera and acquires an image;
- Receives keyboard and mouse events for mode switching and color selection;
- Processes the image to calculate the volume of color blocks based on depth and distance information;
- Adjustable HSV color values

Some of the core code is as follows.

```
def ComputeVolume(self,color_frame,depth_frame):
    # 转换彩色图像为OpenCV格式 Convert color images to OpenCV format
    rgb_image = self.rgb_bridge.imgmsg_to_cv2(color_frame,'bgr8')
    result_image = np.copy(rgb_image)

    # 转换深度图像并调整尺寸 Convert depth image and resize
    depth_image = self.depth_bridge.imgmsg_to_cv2(depth_frame, encoding[1])
    frame = cv.resize(depth_image, (640, 480))
    depth_image_info = frame.astype(np.float32)

    # 处理彩色图像得到物体轮廓（具体实现在self.process中）Process the color image to
get the object outline (specifically implemented in self.process)
    action = cv.waitKey(10) & 0xFF
    result_image = cv.resize(result_image, (640, 480))
    result_frame, binary= self.process(result_image,action)

    if self.color.shape_cx!=0 and self.color.shape_cy!=0:
         #矩形/圆柱体处理 Rectangle/cylinder processing
        if self.color.shape_name == "Rectangle" or self.color.shape_name ==
"Cylinder":
            # 获取三个角点坐标 Get the coordinates of three corner points
            x1, y1 = self.get_corner[0], self.get_corner[1]
            x1 = min(max(0, x1), 640 - 1)
            y1 = min(max(0, y1), 480 - 1)
            x1, y1 = int(round(x1)), int(round(y1))

            z1 = depth_image_info[y1,x1]/1000

            # 获取点1在相机坐标系中的真实位置 Get the real position of point 1 in the
camera coordinate system
            if z1 != 0:
                c1_pose_T = self.get_pos(x1,y1,z1)
            else:
                self.error = True

            # 同样处理点2和点3 The same process is done for points 2 and 3
            x2, y2 = self.get_corner[2], self.get_corner[3]
            z2 = depth_image_info[y2,x2]/1000


            x3, y3 = self.get_corner[4], self.get_corner[5]
            z3 = depth_image_info[y3,x3]/1000
```

```
            # 获取中心点深度并计算高度 Get the center point depth and calculate the
height
            cx, cy = self.color.shape_cx, self.color.shape_cy
            cz = depth_image_info[cy,cx]/1000
            if cz != 0:
                self.center_pose_T = self.get_pos(cx,cy,cz)
                self.height = self.center_pose_T[2]*100  # 高度转换为厘米 Convert
height to centimeters

            # 如果没有错误则计算体积 If there are no errors, calculate the volume
            if not self.error:
                # 计算点到中心的距离（转换为厘米） Calculate the distance from the
point to the center (convert to centimeters)
                point1 = np.array([c1_pose_T[0], c1_pose_T[1], c1_pose_T[2]])

                c_point = np.array([self.center_pose_T[0],
self.center_pose_T[1], self.center_pose_T[2]])

                r = np.linalg.norm(point1 - c_point)*100

                # 计算边到边距离（转换为厘米） Calculate side-to-side distance
(convert to centimeters)
                distance1 = np.linalg.norm(point1 - point3)*100
                distance2 = np.linalg.norm(point3 - point2)*100

                # 根据形状计算体积 Calculate volume from shape
                if self.color.shape_name == "Rectangle":
                    volume = distance1 * distance2 * self.height

                if self.color.shape_name == "Cylinder":
                    volume = math.pi * r * r * self.height  # πr²h
            #立方体处理 Cube processing
            if self.color.shape_name == "Square":
                # 获取中心点坐标 Get the center point coordinates
                cx, cy = self.color.shape_cx, self.color.shape_cy
                cx, cy = int(round(cx)), int(round(cy))

                # 获取中心点深度 Get the center point depth
                cz = depth_image_info[cy,cx]/1000

                if cz != 0:
                    self.center_pose_T = self.get_pos(cx,cy,cz)
                    self.height = self.center_pose_T[2]*100  # 立方体边长Cube side
length

                    # 立方体体积（边长³） Volume of a cube (side length³)
                    volume = self.height * self.height * self.height
```