Color block color sorting

Before starting this function, you need to close the process of the big program and APP. If you need to start the big program and APP again later, start the terminal,

```
bash ~/dofbot_pro/APP_DOFBOT_PRO/start_app.sh
```

1. Function description

After the program is started, the robot arm will perform color recognition according to the set HSV value. After pressing the space bar, the robot arm will clamp the recognized robot arm with its lower claws and place it at the set position after clamping; after placement, it returns to the recognized posture.

2. Start and operate

2.1. Start command

Enter the following command in the terminal to start,

```
#Start the camera:
ros2 launch orbbec_camera dabai_dcw2.launch.py
#Start the underlying control:
ros2 run dofbot_pro_driver arm_driver
#Start the inverse solution program:
ros2 run dofbot_pro_info kinemarics_dofbot
#Start the color recognition program:
ros2 run dofbot_pro_color color_detect
#Start the robot gripping program:
ros2 run dofbot_pro_driver grasp
```

2.2. Operation process

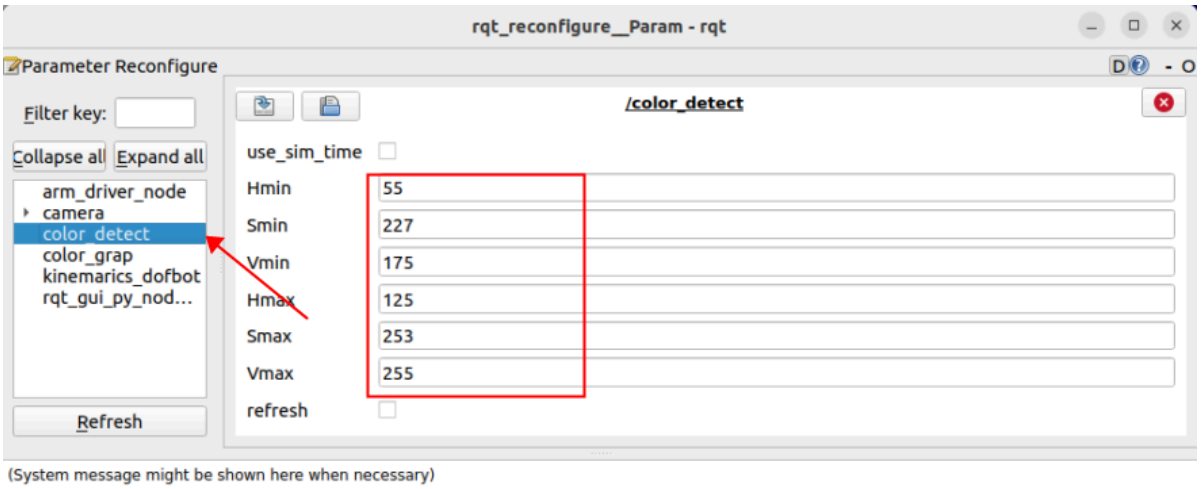
After the terminal is started, place blocks of different colors, the camera captures the image, and press the following buttons to process the image:

- [i] or [I]: Enter the color recognition mode, directly load the HSV value calibrated by the last program for recognition;
- 【r】or 【R】: reset program parameters, enter color selection mode, select a certain area of the color block with the mouse, obtain the HSV value of this area, release the mouse to enter color recognition mode
- Spacebar: start to grab the recognized color block

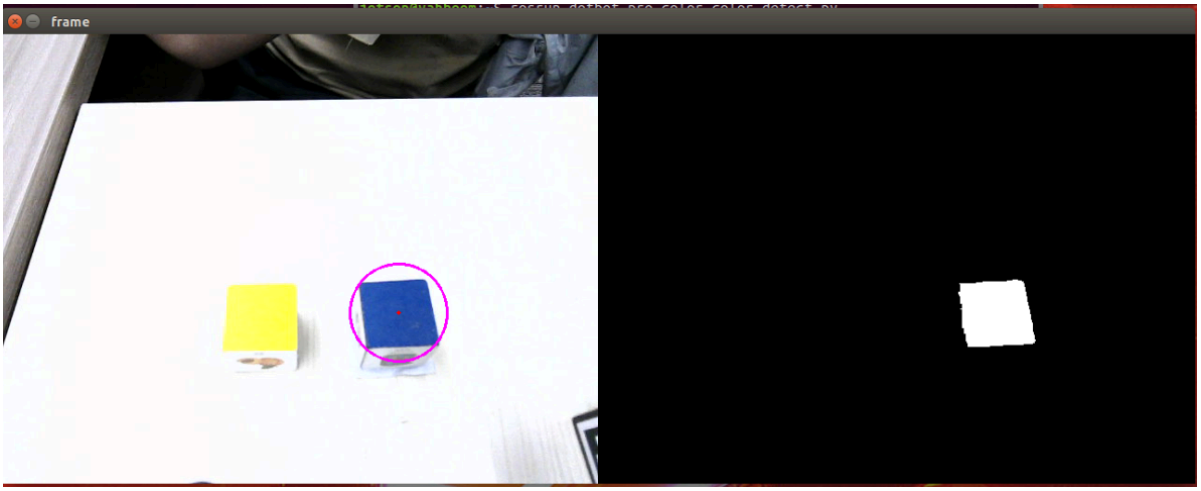
After entering the color recognition mode, if the current HSV value still cannot filter out other colors, you can use the dynamic parameter adjuster to fine-tune HSV. Enter the following command in the terminal to start the dynamic parameter adjuster.

```
ros2 run rqt_reconfigure rqt_reconfigure
```

You can modify the HSV value through the slider

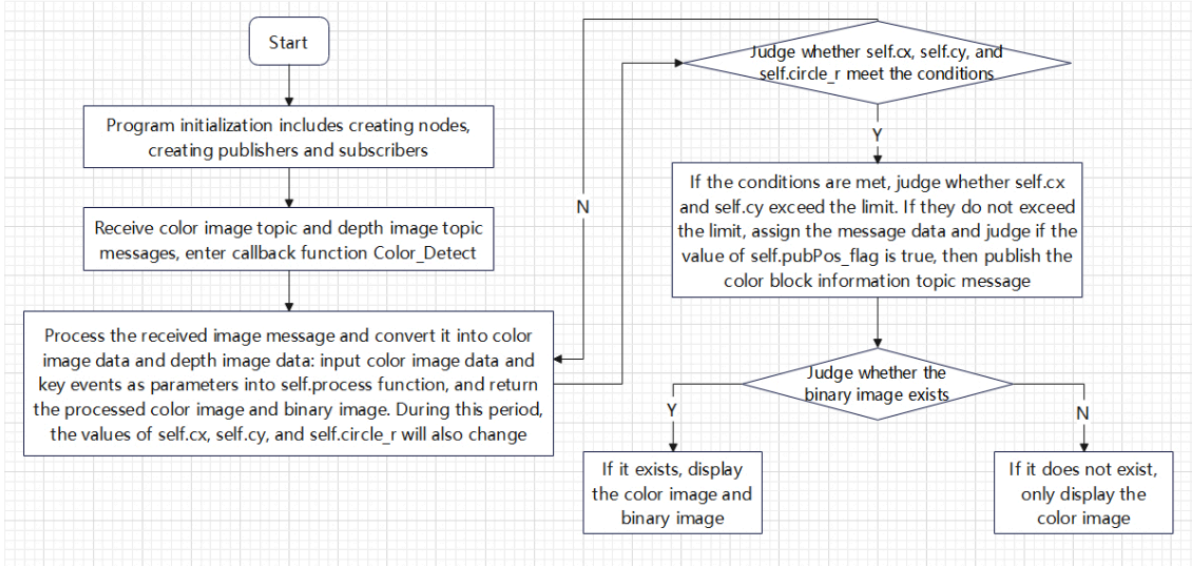


As shown in the figure below, when the image on the left (binarization) only shows the only recognized color, click the color image box and press the spacebar, and the robot will lower its claw to grab the recognized color block.



3. Program flow chart

color_detect.py



4、核心代码解析

4.1、color_detect.py

代码路径：

```
/home/jetson/dofbot_pro_ws/src/dofbot_pro_color/dofbot_pro_color/color_detect.py
```

astra_common代码库路径：

```
/home/jetson/dofbot_pro_ws/src/dofbot_pro_color/dofbot_pro_color/astra_common.py
```

导入必要的库，

```
import cv2
import rclpy
from rclpy.node import Node
import numpy as np
from message_filters import ApproximateTimeSynchronizer, Subscriber
from sensor_msgs.msg import Image
from std_msgs.msg import Float32, Bool
from cv_bridge import CvBridge
import cv2 as cv

import time
import math
import os
encoding = ['16UC1', '32FC1']
import tf_transformations as tf
import transforms3d as tfs
#导入自定义的图像处理库
#Import custom image processing library
from dofbot_pro_color.astra_common import *
from dofbot_pro_interface.msg import *
```

Initialize program parameters, create publishers, subscribers, etc.

```
def __init__(self):
    super().__init__('color_detect')
    self.declare_param()
    #Robot arm recognizes the posture of the color block
    self.init_joints = [90.0, 120, 0.0, 0.0, 90, 90]
    #Create a publisher for publishing color block information
    self.pub_ColorInfo = self.create_publisher(AprilTagInfo, "PosInfo", 1)
    self.pubPoint = self.create_publisher(ArmJoint, "TargetAngle", 1)
    #Create a subscriber to subscribe to gesture recognition results
    self.grasp_status_sub = self.create_subscription(Bool, 'grasp_done',
self.GraspStatusCallback, 1)
    #Create two subscribers to subscribe to the color image topic and the depth
image topic
    self.depth_image_sub = Subscriber(self, Image, "/camera/color/image_raw",
qos_profile=1)
```

```
    self.rgb_image_sub = Subscriber(self, Image, "/camera/depth/image_raw",
qos_profile=1)
    #Time synchronization of color and depth image subscription messages
    self.TimeSynchronizer = ApproximateTimeSynchronizer([self.depth_image_sub,
self.rgb_image_sub],queue_size=10,slop=0.5)
    self.TimeSynchronizer.registerCallback(self.Color_Detect)
    #Save the xy coordinates of the center of the color block
    self.y = 0
    self.x = 0
    #Create a bridge for color and depth image topic message data to image data
    self.rgb_bridge = CvBridge()
    self.depth_bridge = CvBridge()
    #Initialize the region coordinates
    self.Roi_init = ()
    #Initialize the HSV value
    self.hsv_range = ()
    #Initialize the information of the color block, which represents the center x
coordinate, center y coordinate and minimum circumscribed circle radius r of the
color block
    self.circle = (0, 0, 0)
    #Flag for dynamic parameter adjustment, if True, dynamic parameter adjustment
is performed
    self.dyn_update = True
    #Flag for mouse selection
    self.select_flags = False
    self.gTracker_state = False
    self.windows_name = 'frame'
    self.Track_state = 'init'
    #Create color detection object
    self.color = color_detect()
    #Initialize the row and column coordinates of the region coordinates
    self.cols, self.rows = 0, 0
    #Initialize the xy coordinates of the mouse selection
    self.Mouse_XY = (0, 0)
    #The center coordinate xy of the color block after image processing
    self.cx = 0
    self.cy = 0
    #The path of the default HSV threshold file, which stores the last saved HSV
value
    self.hsv_text =
"/home/jetson/dofbot_pro_ws/src/dofbot_pro_color/dofbot_pro_color/colorHSV.text"

    #The minimum circumscribed circle radius of the color block obtained after
image processing
    self.circle_r = 0 #Prevent misidentification of other messy points
    #The flag for publishing machine code information, when True, publishes
/PosInfo topic data
    self.pubPos_flag = False
```

Mainly look at the image processing function Color_Detect,

```
def Color_Detect(self,color_frame,depth_frame):
    #rgb_image
    #接收到彩色图像话题消息，把消息数据转换成图像数据
```

```
    #Receive the color image topic message and convert the message data into
image data
    rgb_image = self.rgb_bridge.imgmsg_to_cv2(color_frame,'bgr8')
    result_image = np.copy(rgb_image)
    #depth_image
    #接收到深度图像话题消息，把消息数据转换成图像数据
    #Receive the depth image topic message and convert the message data into
image data
    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)
    action = cv.waitKey(10) & 0xFF
    result_image = cv.resize(result_image, (640, 480))
    #把得到的彩色图像，作为参数传入process中，并且同时传入键盘事件action
    # Pass the obtained color image as a parameter to the process, and also pass
it to the keyboard event action
    result_frame, binary = self.process(result_image,action)
    #判断self.cx、self.cy不为0说明色块被处理且当外接圆半径大于30的时候，说明是有检测到符合
HSV阈值的色块
    # If self.cx and self.cy are not 0, it means that the color block has been
processed and when the radius of the circumscribed circle is greater than 30, it
means that a color block that meets the HSV threshold has been detected.
    if self.cx!=0 and self.cy!=0 and self.circle_r>30:
        if self.cx<=640 or self.cy <=480:
            center_x, center_y = self.cx,self.cy
            self.x = int(center_x)
            self.y = int(center_y)
            #创建消息计算深度信息并且给里边的数据赋值
            #Create a message to calculate the depth information and assign
values to the data inside
            pos = AprilTagInfo()
            pos.x = center_x
            pos.y = center_y
            pos.z = depth_image_info[self.y,self.x]/1000
            #判断self.pubPos_flag的值是否为True，为True则说明可发布消息
            #Judge whether the value of self.pubPos_flag is True. If it is True,
it means that the message can be published.
            if self.pubPos_flag == True:
                self.pub_ColorInfo.publish(pos)
                self.pubPos_flag = False
    #判断二值化图像是否存在，存在的话现在彩色和二值化图像，否则只显示彩色图像
    # Check if the binary image exists. If it does, display the color and binary
images. Otherwise, only display the color image.
    if len(binary) != 0: cv.imshow(self.windows_name, ManyImgs(1, ([result_frame,
binary])))
    else:
        cv.imshow(self.windows_name, result_frame)
```

Image processing function self.process,

```
def process(self, rgb_img, action):
    rgb_img = cv.resize(rgb_img, (640, 480))
    binary = []
    #判断按键事件，空格按下时，改变信息发布的标识状态，self.pubPos_flag为True表示可以发布信
息话题
```

```
    #Judge the key event. When the spacebar is pressed, change the information
publishing flag status. self.pubPos_flag is True, indicating that the information
topic can be published.
    if action == 32: self.pubPos_flag = True
    #判断按键事件，i或者I按下的时候，改变状态，更改为识别模式
    #Judge the key event. When i or I is pressed, change the state to recognition
mode
    elif action == ord('i') or action == ord('I'): self.Track_state = "identify"
    #判断按键事件，r或者R按下的时候，重设所有参数，进入选色模式
    #Judge the key event. When r or R is pressed, reset all parameters and enter
the color selection mode
    elif action == ord('r') or action == ord('R'): self.Reset()
    #判断状态值，如果为init，则表示为初始状态值，此时可以用鼠标选择区域
    #Judge the state value. If it is init, it means the initial state value. You
can use the mouse to select the area.
    if self.Track_state == 'init':
        cv.namedWindow(self.windows_name, cv.WINDOW_AUTOSIZE)
        #在规定的窗口内进行选择一片区域的颜色
        #Select the color of an area within the specified window
        cv.setMouseCallback(self.windows_name, self.onMouse, 0)
        #判断选色标识，为true表示可以选色
        #Judge the color selection flag, true means you can select the color
        if self.select_flags == True:
            cv.line(rgb_img, self.cols, self.rows, (255, 0, 0), 2)
            cv.rectangle(rgb_img, self.cols, self.rows, (0, 255, 0), 2)
            #判断选择的区域是否存在
            # Check if the selected area exists
            if self.Roi_init[0] != self.Roi_init[2] and self.Roi_init[1] !=
self.Roi_init[3]:
                #调用创建的颜色检测对象self.color里边的Roi_hsv函数，返回的是处理后的彩色图
像和HSV的值
                #Call the Roi_hsv function in the created color detection object
self.color, and return the processed color image and HSV value
                rgb_img, self.hsv_range = self.color.Roi_hsv(rgb_img,
self.Roi_init)
                self.gTracker_state = True
                self.dyn_update = True
            else: self.Track_state = 'init'
    #判断状态值，如果为"identify"，则表示可以进行颜色识别
    #Judge the status value. If it is "identify", it means that color recognition
can be performed.
    elif self.Track_state == "identify":
        #判断是否存在HSV阈值文件，存在的话读取里边的值赋值给hsv_range
        # Check if there is an HSV threshold file. If so, read the value in it
and assign it to hsv_range
        if os.path.exists(self.hsv_text): self.hsv_range =
read_HSV(self.hsv_text)
        #不存在则改变状态为init进行选色
        #If it does not exist, change the state to init to select the color
        else: self.Track_state = 'init'
    if self.Track_state != 'init':
        #判断self.hsv_range值的长度，也就是判断该值是否存在，长度不为0的时候，进入颜色检测函
数
        #Judge the length of the self.hsv_range value, that is, whether the value
exists. When the length is not 0, enter the color detection function
        if len(self.hsv_range) != 0:
```

```
            #调用创建的颜色检测对象self.color里边的object_follow函数，传入彩色图像以及self.hsv_range，也就是hsv的阈值，返回的是处理后的彩色图像，二值化图像和存储符合hsv阈值图形的信息，包括中心点坐标和其最小外接圆的半径
            #Call the object_follow function in the created color detection object self.color, pass in the color image and self.hsv_range, which is the hsv threshold, and return the processed color image, the binary image, and the information that stores the hsv threshold graphic, including the center point coordinates and the radius of its minimum circumscribed circle
            rgb_img, binary, self.circle,_= self.color.object_follow(rgb_img, self.hsv_range)
            #把返回值赋值给存储中心值的self.cx和self.cy,最小外切圆的半径赋值给self.circle_r
            #Assign the return value to self.cx and self.cy that store the center value, and assign the radius of the minimum circumscribed circle to self.circle_r
            self.cx = self.circle[0]
            self.cy = self.circle[1]
            self.circle_r = self.circle[2]
            #判断动态参数更新的标识，为True表示可以对hsv_text文件进行更新和对参数服务器上值进行修改
            #The flag for determining dynamic parameter updates. True means that the hsv_text file can be updated and the value on the parameter server can be modified.
            if self.dyn_update == True:
                write_HSV(self.hsv_text, self.hsv_range)
                params = {'Hmin': self.hsv_range[0][0], 'Hmax': self.hsv_range[1][0],
                          'Smin': self.hsv_range[0][1], 'Smax': self.hsv_range[1][1],
                          'Vmin': self.hsv_range[0][2], 'Vmax': self.hsv_range[1][2]}
                self.dyn_client.update_configuration(params)
                self.dyn_update = False
    return rgb_img, binary
```

4.2、grasp.py

Please refer to the content of [grasp.py] in section 4.2 of the tutorial [Three-dimensional space sorting and gripping\1. Machine code ID sorting].