Volume measurement

Before starting this function, you need to close the process of the big program and APP. Enter the following program in the terminal to close the process of the big program and APP.

```
sh ~/app_Arm/kill_YahboomArm.sh
sh ~/app_Arm/stop_app.sh
```

If you need to start the big program and APP again later, start the terminal.

```
sudo systemctl start yahboom_arm.service
sudo systemctl start yahboom_app.service
```

1. Function description

After the program runs, the program will identify the shape of the selected color block, obtain the relevant data of the color block, and calculate the volume of the color block.

2. Start and operation

2.1. Start command

Enter the following command in the terminal to start,

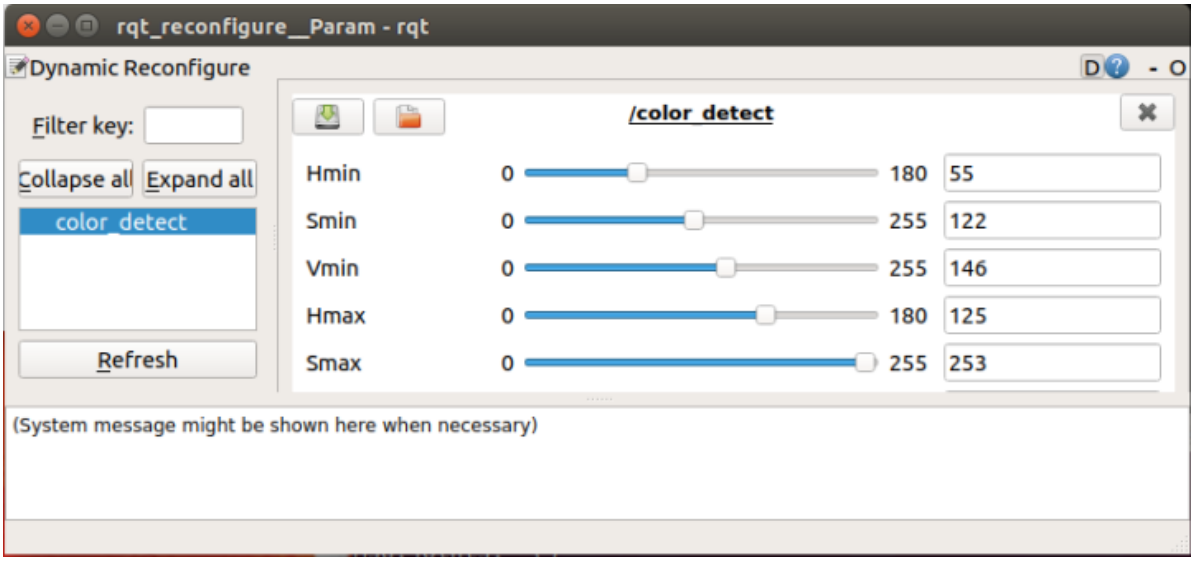
```
#启动相机 Launch the camera
roslaunch orbbec_camera dabai_dcw2.launch
#启动底层控制机械臂 Start the underlying control robot
rosrun dofbot_pro_info arm_driver.py
#启动逆解算程序 Start the inverse solver
rosrun dofbot_pro_info kinemarics_dofbot_pro
#启动体积测试程序 Start the volume test program
rosrun dofbot_pro_RGBDCam calculate_volume.py
```

2.2, Operation

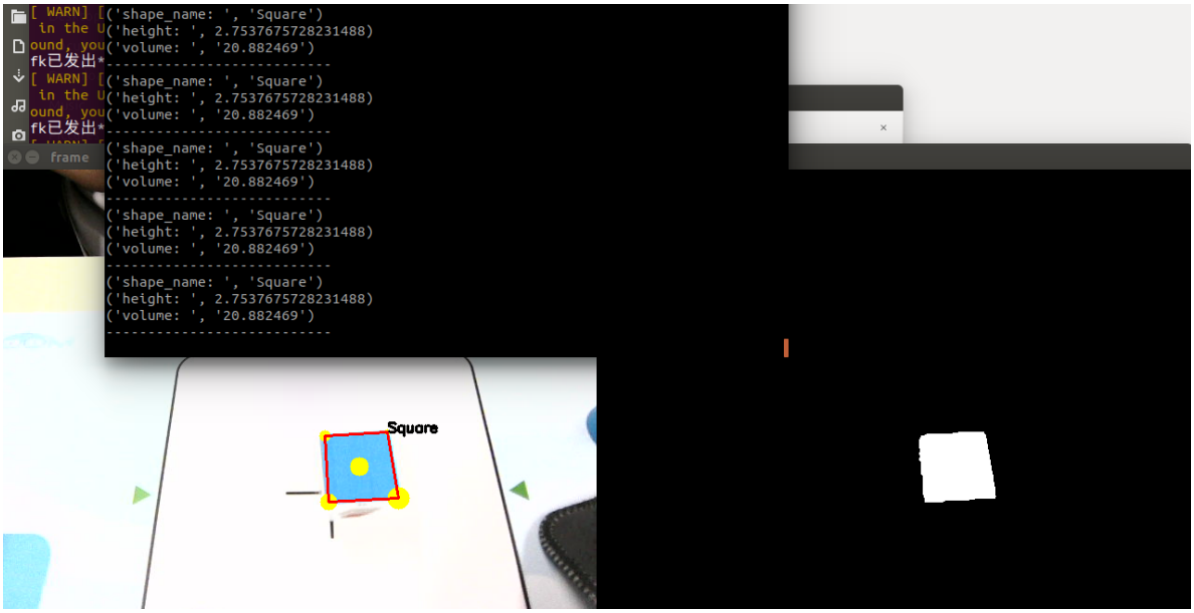
After the program starts, put the color block into the image. Use the mouse to select an area in the color block and obtain the HSV value. If the HSV value of the selected color cannot correctly identify the shape of the color block, you may need to fine-tune the HSV value. Enter the following command in the terminal to start the dynamic parameter regulator,

```
rosrun rqt_reconfigure rqt_reconfigure
```

You can modify the HSV value through the slider,

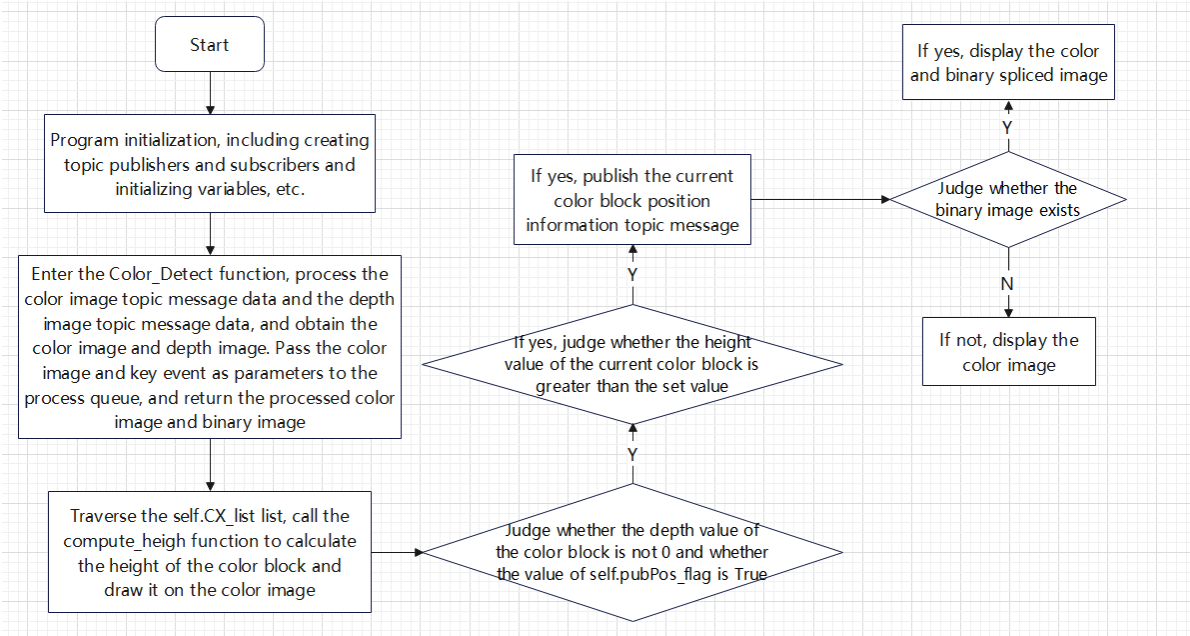


The program uses the HSV value to filter out other colors and only retains colors that meet the HSV value range. The program will identify the shape of the color block, and calculate the data of the color block based on the center point of the color block, and calculate the volume of the color block based on the identified color block shape.



3. Program flow chart

calculate_volume.py



4. Core code analysis

Code path: /home/jetson/dofbot_pro_ws/src/dofbot_pro_RGBDCam/scripts/calculate_volume.py

Import necessary libraries,

```
import rospy
import numpy as np
from sensor_msgs.msg import Image
import message_filters
from std_msgs.msg import Float32,Bool
import os
from cv_bridge import CvBridge
import cv2 as cv
encoding = ['16UC1', '32FC1']
import time
import os
#导入自定义的图像处理库
#Import custom image processing library
from astra_common import *
from dynamic_reconfigure.server import Server
from dynamic_reconfigure.client import Client
import rospkg
from dofbot_pro_color.cfg import ColorHSVConfig
import math
from dofbot_pro_info.msg import *
from dofbot_pro_info.srv import *
#导入transformations处理和计算三维空间中的变换，包括四元数和欧拉角之间的转换
#Import transformations to process and calculate transformations in three-dimensional space, including conversions between quaternions and Euler angles
import tf.transformations as tf
#导入transforms3d 库用于处理三维空间中的变换，执行四元数、旋转矩阵和欧拉角之间的转换，支持三维几何操作和坐标转换
#Import the transforms3d library to handle transformations in three-dimensional space, perform conversions between quaternions, rotation matrices, and Euler angles, and support three-dimensional geometric operations and coordinate conversions
```

```
import transforms3d as tfs
```

Program initialization, creation of publishers, subscribers, etc.

```
def __init__(self):
    nodeName = 'color_detect'
    rospy.init_node(nodeName)
    #机械臂识别色块姿态 Robot arm recognizes color block posture
    self.init_joints = [90.0, 120, 0.0, 0.0, 90, 90]
    #创建客户端，调用逆解算服务 Create a client and call the inverse solution service
    self.client = rospy.ServiceProxy("get_kinemarics", kinemarics)
    #创建两个订阅者，订阅彩色图像话题和深度图像话题
    #Create two subscribers to subscribe to the color image topic and the depth
image topic
    self.depth_image_sub =
message_filters.Subscriber('/camera/depth/image_raw',Image)
    self.rgb_image_sub =
message_filters.Subscriber('/camera/color/image_raw',Image)
    #将彩色和深度图像订阅的消息进行时间同步 Time synchronization of color and depth
image subscription messages
    self.TimeSynchronizer =
message_filters.ApproximateTimeSynchronizer([self.rgb_image_sub,self.depth_image
_sub],10,0.5)
    #处理同步消息的回调函数ComputeVolume，回调函数与订阅的消息连接起来，以便在接收到新消息时
自动调用该函数
    #The callback function ComputeVolume that handles the synchronization message
is connected to the subscribed message so that the function is automatically
called when a new message is received
    self.TimeSynchronizer.registerCallback(self.ComputeVolume)
    #创建彩色和深度图像话题消息数据转图像数据的桥梁
    #Create a bridge for converting color and depth image topic message data to
image data
    self.rgb_bridge = CvBridge()
    self.depth_bridge = CvBridge()
    #初始化区域坐标 Initialize the area coordinates
    self.Roi_init = ()
    #初始化HSV的值 Initialize HSV value
    self.hsv_range = ()
    #初始化识别到色块的信息，这边分别表示色块中心x坐标、中心y坐标和最小外接圆半径r
    #Initialize the information of the recognized color block, which represents
the x coordinate of the center of the color block, the y coordinate of the
center, and the minimum circumscribed circle radius r
    self.circle = (0, 0, 0)
    #动态参数调节的标志位，为True则进行动态参数调节
    #The flag for dynamic parameter adjustment. If it is True, dynamic parameter
adjustment will be performed.
    self.dyn_update = True
    #鼠标选择的标志位 Mouse selection flag
    self.select_flags = False
    self.gTracker_state = False
    self.windows_name = 'frame'
    #初始化状态值 Initialize state value
    self.Track_state = 'init'
    #创建颜色检测的对象 Create a color detection object
    self.color = color_detect()
    #初始化区域坐标的行坐标和列坐标 Initialize the row and column coordinates of the
region coordinates
```

```
    self.cols, self.rows = 0, 0
    #初始化鼠标选择的xy坐标 Initialize the xy coordinates of the mouse selection
    self.Mouse_XY = (0, 0)
    #默认的HSV阈值文件的路径，该文件存储了上次的保存的HSV值
    #The default path of the HSV threshold file, which stores the last saved HSV
value
    self.hsv_text = rospkg.RosPack().get_path("dofbot_pro_color") +
"/scripts/colorHSV.text"
    #加载颜色HSV的配置文件，配置动态参数调节器
    #Load the color HSV configuration file and configure the dynamic parameter
regulator
    Server(ColorHSVConfig, self.dynamic_reconfigure_callback)
    self.dyn_client = Client(nodeName, timeout=60)
    #当前识别到的色块的中心xy坐标和色块最小外切圆的半径
    #The xy coordinates of the center of the currently identified color block and
the radius of the minimum circumscribed circle of the color block
    self.cx = 0
    self.cy = 0
    self.error = False
    self.circle_r = 0 #防止误识别到其他的杂乱的点 Prevent misidentification of other
messy points
    #机械臂当前姿态下的末端位置位姿
    #The end position of the robot arm in its current posture
    self.CurEndPos = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
    #相机内参 Camera internal parameters
    self.camera_info_K = [477.57421875, 0.0, 319.3820495605469, 0.0,
477.55718994140625, 238.64108276367188, 0.0, 0.0, 1.0]
    #机械臂末端与相机的旋转变换矩阵，描述了两者之间的相对位置和位姿
    #The rotation transformation matrix of the end of the robotic arm and the
camera describes the relative position and posture between the two
    self.EndToCamMat =
np.array([[1.00000000e+00,0.00000000e+00,0.00000000e+00,0.00000000e+00],
                                  [0.00000000e+00,7.96326711e-04,9.99999683e-
01,-9.90000000e-02],
                                  [0.00000000e+00,-9.99999683e-01,7.96326711e-
04,4.90000000e-02],

[0.00000000e+00,0.00000000e+00,0.00000000e+00,1.00000000e+00]])
    #获取机械臂当前姿态下的末端位姿
    #Get the end position of the robot arm under the current posture
    self.get_current_end_pos()
    #初始化角点坐标列表
    # Initialize the corner point coordinate list
    self.get_corner = [0,0,0,0,0,0]
    exit_code = os.system('rosservice call /camera/set_color_exposure  50')
```

Image processing function ComputeVolume,

```
def ComputeVolume(self,color_frame,depth_frame):
    #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)
    #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 deep 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)
    #判断当前色块的中心值的坐标是否不为0，如果成立，则说明有检测到色块
    #Judge whether the coordinates of the center value of the current color block
are not 0. If so, it means that a color block has been detected
    if self.color.shape_cx!=0 and self.color.shape_cy!=0:
        #判断色块的形状，如果是长方体或者圆柱，则需要计算角点坐标值
        #Judge the shape of the color block. If it is a cuboid or a cylinder,
you need to calculate the coordinates of the corner points
        if self.color.shape_name == "Rectangle" or self.color.shape_name ==
"Cylinder":
            x1 = self.get_corner[0]
            y1 = self.get_corner[1]
            z1 = depth_image_info[y1,x1]/1000
            if z1!=0:
                c1_pose_T = self.get_pos(x1,y1,z1)
            else:
                self.error = True
                print("z1 invalid Distance!")

            x2 = self.get_corner[2]
            y2 = self.get_corner[3]
            z2 = depth_image_info[y2,x2]/1000
            if z1!=0:
                c2_pose_T = self.get_pos(x2,y2,z2)
            else:
                self.error = True
                print("z2 invalid Distance!")


            x3 = self.get_corner[4]
            y3 = self.get_corner[5]
            z3 = depth_image_info[y3,x3]/1000
            if z1!=0:
                c3_pose_T = self.get_pos(x3,y3,z3)
            else:
                self.error = True
                print("z3 invalid Distance!")

            cx = self.color.shape_cx
            cy = self.color.shape_cy
            cz = depth_image_info[cy,cx]/1000
            if cz!=0:
                center_pose_T = self.get_pos(cx,cy,cz)
                #计算色块高度 Calculate the color block height
                height = center_pose_T[2]*100
                print("get height: ",height)

            if self.error!=True:
                # 定义两个点的坐标 Define the coordinates of two points
                point1 = np.array([c1_pose_T[0], c1_pose_T[1], c1_pose_T[2]])
```

```
                point2 = np.array([c2_pose_T[0], c2_pose_T[1], c2_pose_T[2]])
                point3 = np.array([c3_pose_T[0], c3_pose_T[1], c3_pose_T[2]])
                c_ponit = np.array([center_pose_T[0], center_pose_T[1],
center_pose_T[2]])
                #计算半径 Calculate Radius
                r = np.linalg.norm(point1 - c_ponit)
                # 计算欧几里得距离（矩形边长） Calculate Euclidean distance
(rectangle side length)
                distance1 = np.linalg.norm(point1 - point3)
                distance2 = np.linalg.norm(point3 - point2)

                #长方体的体积等与长乘宽乘高 The volume of a cuboid is equal to length
times width times height
                if self.color.shape_name == "Rectangle":
                    print("shape_name: ",self.color.shape_name)
                    print("distance1: ",distance1)
                    print("distance2: ",distance2)
                    print("height: ",height)
                    volume = distance1 * distance2 * height
                    print("volume: ",format(volume, 'f'))
                    print("----------------------------")
                 #圆柱体的体积等与π乘半径的平方乘高 The volume of a cylinder is equal
to π times the radius squared times the height
                if self.color.shape_name == "Cylinder":
                    print("r: ",r)
                    print("height: ",height)
                    print("shape_name: ",self.color.shape_name)
                    volume = math.pi*r*r* height
                    print("volume: ",format(volume, 'f'))
                    print("----------------------------")

        if self.color.shape_name == "Square":
            print("shape_name: ",self.color.shape_name)
            cx = self.color.shape_cx
            cy = self.color.shape_cy
            cz = depth_image_info[cy,cx]/1000
            if cz!=0:
                center_pose_T = self.get_pos(cx,cy,cz)
                height = center_pose_T[2]*100
                print("height: ",height)
                #正方体的体积等于任一边长的三次方 The volume of a cube is equal to the
cube of the length of any side
                volume = height * height * height
                print("volume: ",format(volume, 'f'))
                print("----------------------------")

        self.error = False
    if len(binary) != 0: cv.imshow(self.windows_name, ManyImgs(1,
([result_frame, binary])))
    else:
        cv.imshow(self.windows_name, result_frame)
```

Get the position function in the world coordinate system get_pos,

```
def get_pos(self,x,y,z):
    #进行坐标系的转换，最终得到点在世界坐标下的位置pose_T和位姿pose_R
    #Convert the coordinate system to get the position pose_T and pose pose_R of
the point in the world coordinate system
    camera_location = self.pixel_to_camera_depth((x,y),z)
    PoseEndMat = np.matmul(self.EndToCamMat,
self.xyz_euler_to_mat(camera_location, (0, 0, 0)))
    EndPointMat = self.get_end_point_mat()
    WorldPose = np.matmul(EndPointMat, PoseEndMat)
    pose_T, pose_R = self.mat_to_xyz_euler(WorldPose)
    return pose_T
```