Mediapipe gesture recognition machine code ID 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

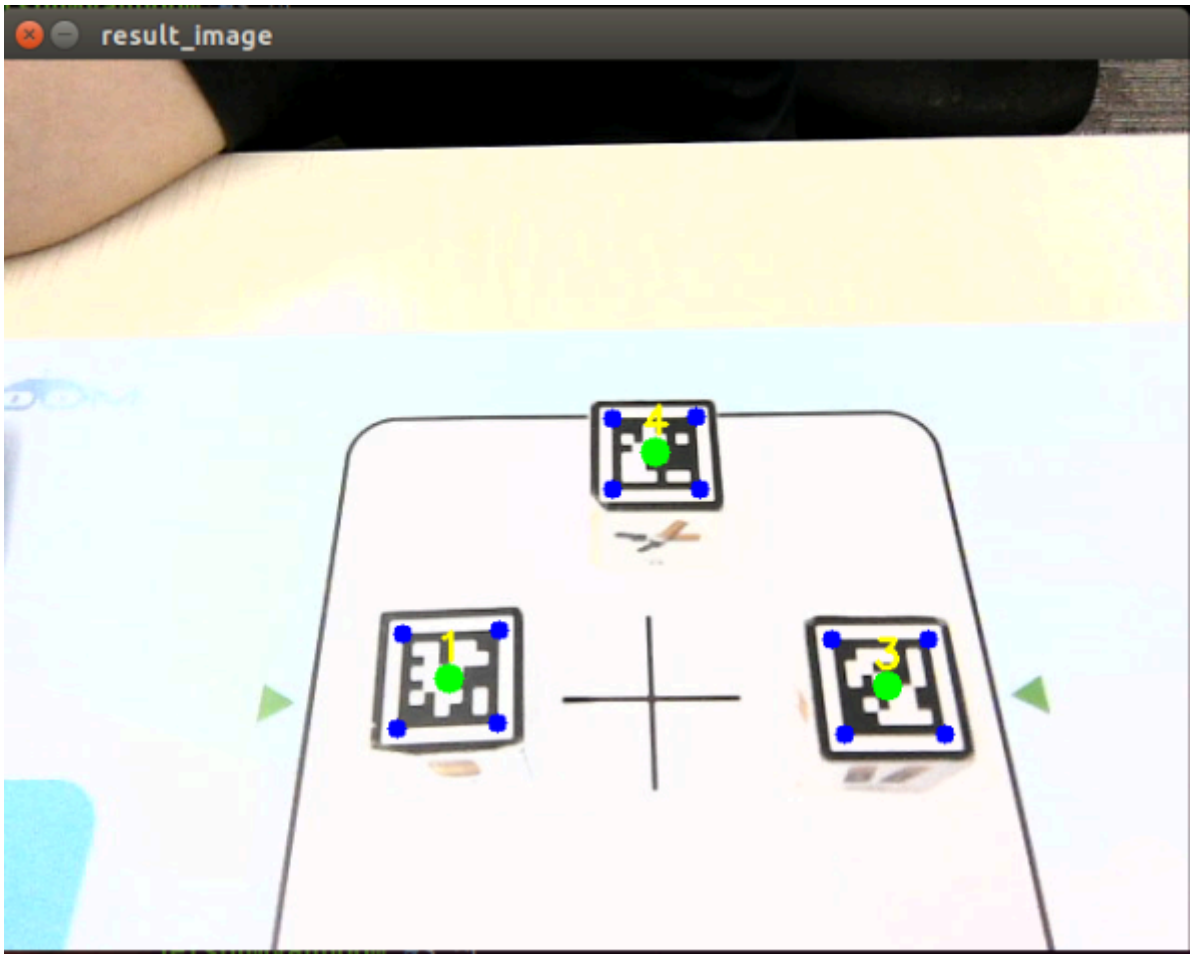
After the program runs, the camera captures the image and recognizes the gesture. The gesture is 1 to 4. The ID value of the target machine code is given through the recognized gesture; then the robot arm will change its posture to find the machine code of the target ID. If it is detected and recognized, it will be clamped and placed in the set position, and then return to the recognized posture and continue to recognize; if the target machine code is not detected, the robot arm will return to the posture of the recognized gesture.

2. Start and operation

2.1. Start command

Enter the following command in the terminal,

```
#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 image conversion program:
ros2 run dofbot_pro_apriltag msgToimg
#Start the machine code recognition program:
ros2 run dofbot_pro_apriltag apriltagID_finger_detect
#Start the robot gripping program:
ros2 run dofbot_pro_driver grasp
#Start the Mediapipe gesture recognition program:
ros2 run dofbot_pro_apriltag MediapipeGesture
```

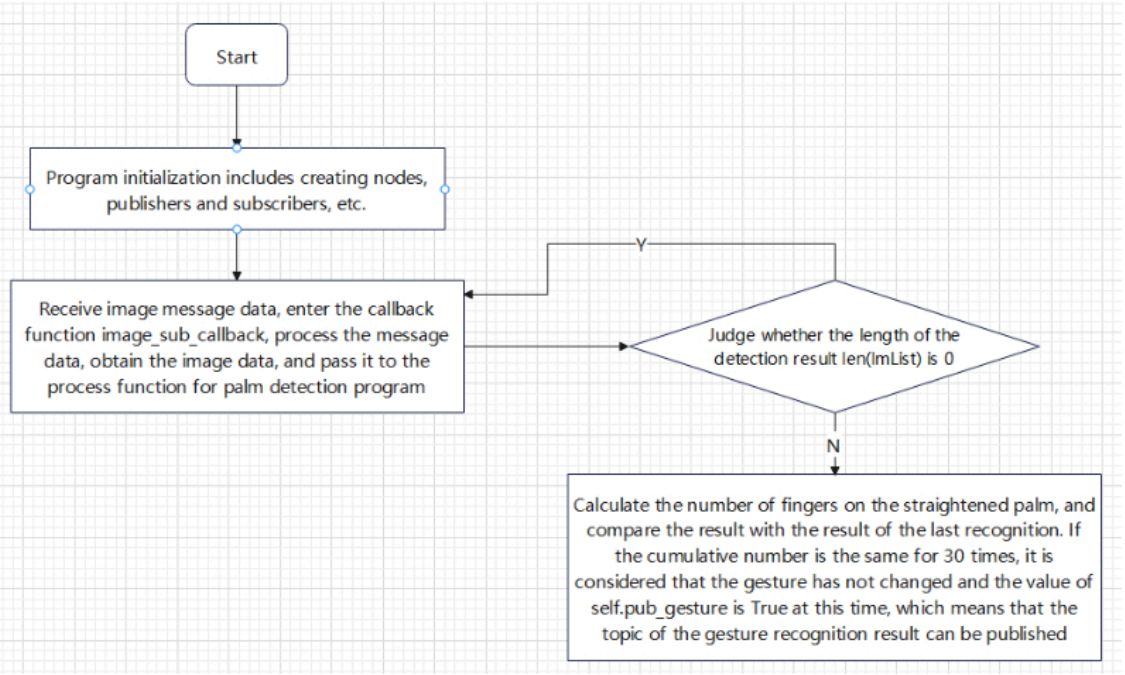


2.2、Operation

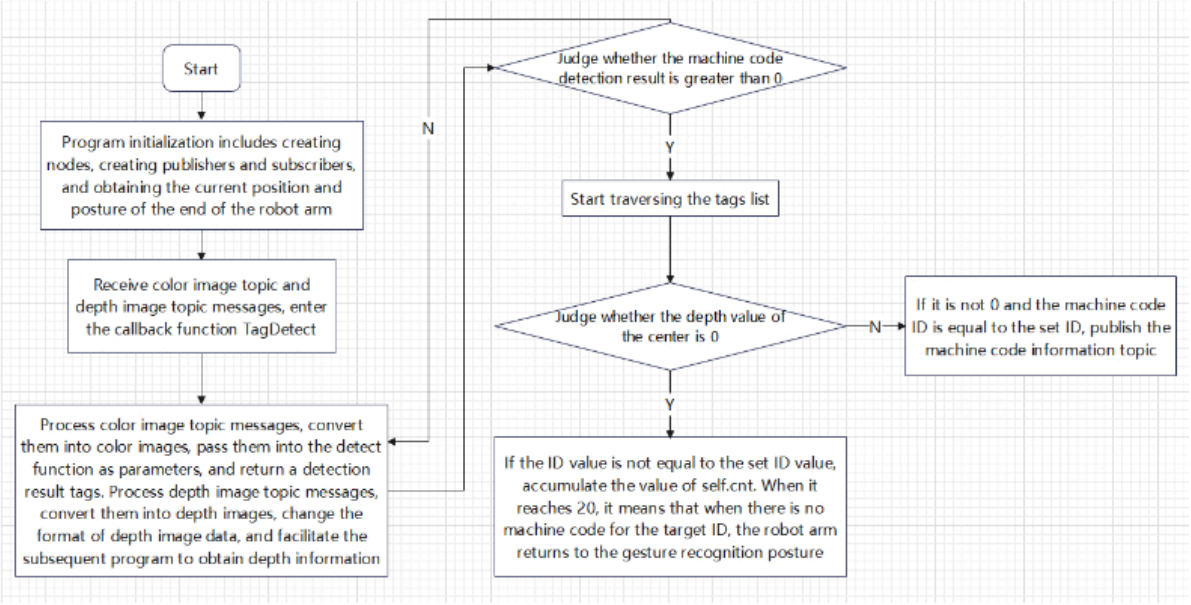
After the program is started, the robot arm will begin to show the gesture recognition posture. There are one to four gestures that can be recognized. The corresponding gestures are as follows; gesture recognition waits for about 3 seconds, waiting for the machine code to change its posture, and becomes the posture of detecting and recognizing the machine code. Press the space bar to start recognition; if the target machine code is recognized, it will clamp it with the lower claw and then place it in the set position; if the machine code that matches the target ID value is not recognized, the robot arm returns to the posture of gesture recognition at the beginning.

3、Program flow chart

MediapipeGesture.py



apriltagID_finger_detect.py



4. Core code analysis

4.1. MediapipeGesture.py

Code path:

```
/home/jetson/dofbot_pro_ws/src/dofbot_pro_apriltag/dofbot_pro_apriltag/MediapipeGesture.py
```

Import necessary library files

```
import rclpy
from rclpy.node import Node
from rclpy.qos import QoSProfile
import numpy as np
from dofbot_pro_apriltag.media_library import *
from collections import defaultdict
import threading
from std_msgs.msg import Float32, Int8, Bool
from dofbot_pro_interface.msg import *
from time import sleep, time
```

Initialize program parameters and create publishers and subscribers

```
def __init__(self):
    super().__init__('detect_gesture_node')
        self.hand_detector = HandDetector()
        self.pub_gesture = True

        qos = QoSProfile(depth=10)
        self.img_sub = self.create_subscription(ImageMsg,
"/image_data",self.image_sub_callback, qos)
        self.grasp_sub=
self.create_subscription(Bool,'grasp_done',self.GraspStatusCallback, qos)
        self.pubPoint = self.create_publisher(ArmJoint, "TargetAngle", qos)
        self.pub_targetID = self.create_publisher(Int8, "TargetId", qos)
        self.pTime = self.cTime = 0
        self.cnt = 0
        self.last_sum = 0
        self.detect_gesture = Int8()
        self.pTime = 0
        self.detect_gesture_joints = [90.0, 150.0, 12.0, 20.0, 90.0, 30.0]
        self.img = np.zeros((480, 640, 3), dtype=np.uint8)
        self.pubTargetArm(self.detect_gesture_joints)
```

Image processing function image_sub_callback

```
def image_sub_callback(self,msg):
    # 将自定义图像消息转化为图像
    # Convert custom image messages into images
    image = np.ndarray(shape=(msg.height, msg.width, msg.channels),
dtype=np.uint8, buffer=msg.data)
    # 将rgb 转化为opencv的bgr顺序
    # Convert rgb to opencv bgr order
    self.img[:,:,0],self.img[:,:,1],self.img[:,:,2] =
image[:,:,2],image[:,:,1],image[:,:,0]
    frame = self.img.copy()
    #把转化得到图像传入process函数，进行图像处理识别函数
    #Put the converted image into the process function to perform image
processing and recognition function
    self.process(frame)
```

Image processing function process

```
def process(self, frame):
    #把图像传入创建的hand_detector对象的findHands函数中，该函数是用来检测实时手部跟踪和识别的
    # Pass the image into the findHands function of the created hand_detector object, which is used to detect real-time hand tracking and recognition
    frame, lmList, bbox = self.hand_detector.findHands(frame)
    #检测结果如果不为0，说明检测到手掌了，把检测的结果传递给Gesture_Detect_threading线程函数，并且启动线程
    #If the detection result is not 0, it means that the palm is detected. Pass the detection result to the Gesture_Detect_threading thread function and start the thread
    if len(lmList) != 0:
        threading.Thread(target=self.Gesture_Detect_threading, args=(lmList,bbox)).start()
    #计算帧率
    #Calculate frame rate
    self.cTime = time()
    fps = 1 / (self.cTime - self.pTime)
    self.pTime = self.cTime
    text = "FPS : " + str(int(fps))
    #把帧率绘制在彩色图像上
    # Draw the frame rate on the color image
    cv.putText(frame, text, (20, 30), cv.FONT_HERSHEY_SIMPLEX, 0.9, (0, 0, 255), 1)
    if cv.waitKey(1) & 0xFF == ord('q'):
        cv.destroyAllWindows()
        rospy.signal_shutdown("exit")
    cv.imshow('frame', frame)
```

Gesture recognition thread function Gesture_Detect_threading

```
def Gesture_Detect_threading(self, lmList,bbox):
    #把检测手掌的结果作为参数传入到fingersUp函数中，该函数会返回手指伸直的列表，列表的下标对应5根手指，哪些手指伸直，则该数组下标对应的值为1，否则为0
    # Pass the result of palm detection as a parameter to the fingersUp function, which will return a list of straightened fingers. The subscripts of the list correspond to the 5 fingers. If the fingers are straightened, the value corresponding to the array subscript is 1, otherwise it is 0
    fingers = self.hand_detector.fingersUp(lmList)
    #计算fingers列表值的总和，来判断有多少根手指是伸直的状态，并且把该置赋值给self.last_sum，对比上一次的手指伸直的总和
    #Calculate the sum of the values ••in the fingers list to determine how many fingers are stretched out, and assign the value to self.last_sum to compare with the sum of the last fingers stretched out
    self.last_sum = sum(fingers)
    print(self.pub_gesture)
    if sum(fingers) == self.last_sum:
        print("--------------------------")
        self.cnt = self.cnt + 1
        print("cnt: ",self.cnt)
        #如果当前的手指总和数与上一次的相等则累计计数，当相同次数到30次后，则认为手势没有变化，且此时self.pub_gesture为True时，可以给消息赋值并且发布/TargetId话题
```

```
        #If the current total number of fingers is equal to the previous one, the
count is accumulated. When the number of the same times reaches 30, it is
considered that the gesture has not changed. If self.pub_gesture is True at this
time, the message can be assigned a value and published to the /TargetId topic
        if self.cnt==30 and self.pub_gesture == True:
            print("sum of fingers: ",self.last_sum)
            #改变self.pub_gesture状态值，防止误识别后又发布话题
            #Change the self.pub_gesture status value to prevent publishing
topics after misidentification
            self.pub_gesture = False
            #赋值给创建的消息数据，self.last_sum的值对应的就是机器码的ID
            #Assign to the created message data, the value of self.last_sum
corresponds to the machine code ID
            self.detect_gesture.data = self.last_sum
            self.pub_targetID.publish(self.detect_gesture)
            #恢复数据，以便下次进行识别
            #Restore data for identification next time
            self.last_sum = 0
```

4.2、apriltagID_finger_detect.py

Code path:

```
/home/jetson/dofbot_pro_ws/src/dofbot_pro_apriltag/dofbot_pro_apriltag/apriltagID
_finger_detect.py
```

Import necessary library files

```
import cv2
import rclpy
from rclpy.node import Node
from rclpy.qos import QoSProfile
import numpy as np
from message_filters import ApproximateTimeSynchronizer, Subscriber
from sensor_msgs.msg import Image
from std_msgs.msg import Float32, Int8, Bool
from dt_apriltags import Detector
from dofbot_pro_apriltag.vutils import draw_tags
from cv_bridge import CvBridge
import cv2 as cv
from dofbot_pro_interface.srv import *
from dofbot_pro_interface.msg import ArmJoint, AprilTagInfo
import pyzbar.pyzbar as pyzbar
import time
import queue
import os
encoding = ['16UC1', '32FC1']
```

程序参数初始化，创建发布者、订阅者，

```
def __init__(self):
    super().__init__('apriltag_detect_node')  # ROS2

        self.detect_joints = [90.0, 150.0, 12.0, 20.0, 90.0, 30.0]
```

```
        self.init_joints = [90.0, 120.0, 0.0, 0.0, 90.0, 90.0]
        self.search_joints = [90.0, 120.0, 0.0, 0.0, 90.0, 30.0]

        # ROS2 pub
        self.pubGraspStatus = self.create_publisher(Bool, "grasp_done", 1)
        self.tag_info_pub = self.create_publisher(AprilTagInfo, "PosInfo", 1)
        self.pubPoint = self.create_publisher(ArmJoint, "TargetAngle", 1)

        # ROS2 sub
        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)
        self.TimeSynchronizer =
ApproximateTimeSynchronizer([self.depth_image_sub,
self.rgb_image_sub],queue_size=10,slop=0.5)
        self.TimeSynchronizer.registerCallback(self.TagDetect)

        # ROS2 sub
        self.grasp_status_sub = self.create_subscription(Bool, 'grasp_done',
self.GraspStatusCallback, 1)
        self.sub_targetID = self.create_subscription(Int8, "TargetId",
self.GetTargetIDCallback, 1)

        self.rgb_bridge = CvBridge()
        self.depth_bridge = CvBridge()
        self.pubPos_flag = False

        # AprilTag
        self.at_detector = Detector(
            searchpath=['apriltags'],
            families='tag36h11',
            nthreads=8,
            quad_decimate=2.0,
            quad_sigma=0.0,
            refine_edges=1,
            decode_sharpening=0.25,
            debug=0
        )

        self.pr_time = time.time()
        self.target_id = 0
        self.cnt = 0
        self.detect_flag = False
        self.pub_arm(self.detect_joints)
```

Mainly look at the image processing function TagDetect,

```
def TagDetect(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,'rgb8')
    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 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)
    #调用detect函数，传入参数，
    #Call the detect function and pass in parameters.
    '''
    cv2.cvtColor(rgb_image, cv2.COLOR_RGB2GRAY): Converts an RGB image to a
grayscale image for label detection.
    False: Indicates that the label's pose is not estimated.
    None: Indicates that no camera parameters are provided and only simple
detection may be performed.
    0.025: May be the set label size (usually in meters) to help the detection
algorithm determine the size of the label.
    1.Returns a detection result, including information such as the location, ID,
and bounding box of each label.
    '''
    tags = self.at_detector.detect(cv2.cvtColor(rgb_image, cv2.COLOR_RGB2GRAY),
False, None, 0.025)
    #给tags里边的各个标签进行排序，非必须步骤
    #Sort the tags in tags, not a necessary step
    tags = sorted(tags, key=lambda tag: tag.tag_id) # 貌似出来就是升序排列的不需要手动
进行排列   It seems that the output is in ascending order and does not need to be
sorted manually
    #调用draw_tags函数，作用是在彩色图像上描绘出识别的机器码相关的信息，包括角点，中心点和id值
    draw_tags(result_image, tags, corners_color=(0, 0, 255), center_color=(0,
255, 0))
    #等待键盘的输入，32表示空格按下，按下后改变self.pubPos_flag的值，表示可以发布机器码相关信
息了
    #Call the draw_tags function to draw the information related to the
recognized machine code on the color image, including corner points, center
points and id values
    key = cv2.waitKey(10)
    if key == 32:
        self.pubPos_flag = True
    if self.target_id!=0:
        print("Get th target id,start to search it.")
        self.pub_arm(self.search_joints)
        #time.sleep(0.5)
        #判断tags的长度，大于0则表示有检测到机器码
        #Judge the length of tags. If it is greater than 0, it means that the
machine code has been detected.
        if len(tags) > 0 :
            #遍历机器码
            #Traverse the machine code
            for i in range(len(tags)):
                #self.tag_info_pub.publish(tag)
                if self.pubPos_flag == True:
                    center_x, center_y = tags[i].center
                    cv.circle(result_image, (int(center_x),int(center_y)), 10,
(0,210,255), thickness=-1)
                    print("tag_id: ",tags[i].tag_id)
                    print("center_x, center_y: ",center_x, center_y)
```

```
                    print("depth:
",depth_image_info[int(center_y),int(center_x)]/1000)
                    #创建机器码信息的消息数据
                    #Create message data for machine code information
                    tag = AprilTagInfo()
                    tag.id = tags[i].tag_id
                    tag.x = center_x
                    tag.y = center_y
                    tag.z = depth_image_info[int(center_y),int(center_x)]/1000
                    #如果当前的机器码的中心点的深度距离大于0且当前的机器码ID等于我们目标ID
值，则发布话题
                    #If the depth distance of the center point of the current
machine code is greater than 0 and the current machine code ID is equal to our
target ID value, then publish the topic
                    if tag.z>0 and tag.id == self.target_id:
                        self.tag_info_pub.publish(tag)
                        self.pubPos_flag = False
                        self.cnt = 0
                    else:
                        if tag.z!=0:
                            print("Invalid distance.")
                        if tag.id != self.target_id:
                            print("Do not find the target id tag.")
                            self.cnt = self.cnt + 1
                            #如果不是我们目标的ID，则累加，累加次数到达20，说明没有检测到符
合的ID，回到手势识别的姿态
                            #If it is not the ID of our target, then accumulate.
When the cumulative number reaches 20, it means that no matching ID is detected,
and return to the gesture recognition posture
                            if self.cnt == 20:
                                self.cnt = 0
                                #回到手势识别的姿态
                                #Return to the gesture recognition posture
                                self.pub_arm(self.detect_joints)
                                self.target_id = 0
        else:
            self.cnt = self.cnt + 1
            #如果没有检测到任何机器码，则累加计数，计数到100后，机械臂回到手势识别的姿态
            #If no machine code is detected, the count is accumulated. After the
count reaches 100, the robot arm returns to the gesture recognition posture
            if self.cnt == 100:
                self.cnt = 0
                self.pub_arm(self.detect_joints)
                self.target_id = 0
    result_image = cv2.cvtColor(result_image, cv2.COLOR_RGB2BGR)
    cv2.imshow("result_image", result_image)
    key = cv2.waitKey(1)
```

Look at the two callback functions.

```
#夹取结果的回调函数
#Callback function for clipping results
def GraspStatusCallback(self,msg):
    #改变self.pubPos_flag，以便于下次可以发布话题消息
    #Change self.pubPos_flag so that you can publish topic messages next time
```

```
    if msg.data == True:
        self.pubPos_flag = True
#获取目标ID的回调函数
#Get the callback function of the target ID
def GetTargetIDCallback(self,msg):
    #改变self.target_id的值
    #Change the value of self.target_id
    self.target_id = msg.data
    print("Get th traget is ",self.target_id)
```

4.3、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].