Identify gesture machine code height sorting

Before starting this function, you need to close the process of large programs and APPs. Enter the following program in the terminal to close the process of large programs and APPs.

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

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

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

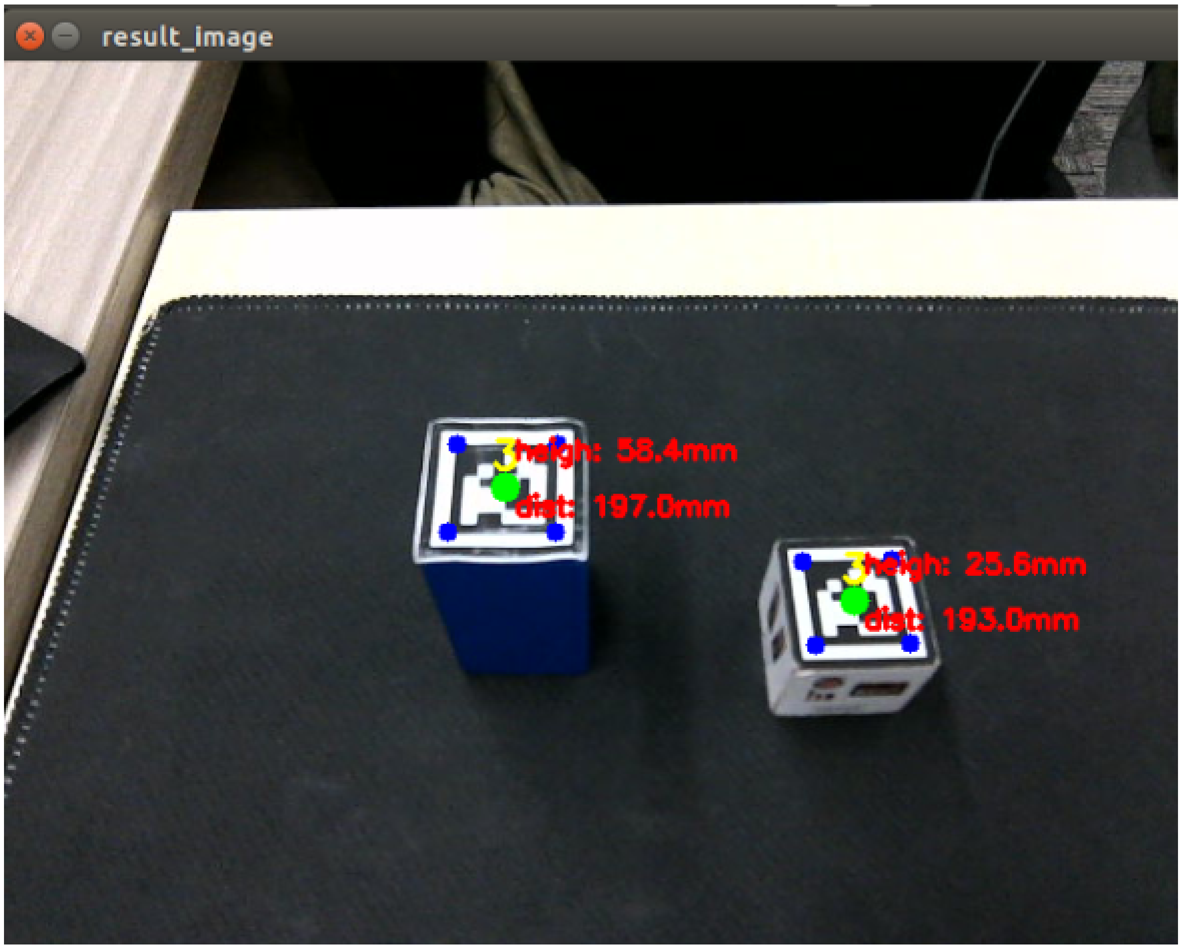
1. Functional Description

After the program is started, the camera captures the image and recognizes the gestures, which are 1 to 5. The height threshold is calculated through the recognized gestures; the robot arm will change its posture to detect the machine code in the image and calculate their height. If it exceeds the height threshold, the robot arm will clamp it with its lower claws and place it at the set position, and then return to the posture of detecting the machine code to continue recognition; if it does not detect a machine code that exceeds the height threshold, the robot arm will make a "shaking head" action group and the buzzer will sound, and then the robot arm will return to the posture of recognizing gestures.

2. Start and operate

2.1. Start command

```
#Start the camera
roslaunch orbbec_camera dabai_dcw2.launch
#Start the underlying control
rosrun dofbot_pro_info arm_driver.py
#Start the reverse settlement program
rosrun dofbot_pro_info kinemarics_dofbot_pro
#Start the image conversion program
rosrun dofbot_pro_apriltag msgToimg.py
#Start the machine code recognition program
rosrun dofbot_pro_apriltag apriltag_list_Hight.py
#Start the robot arm gripping program
rosrun dofbot_pro_info grasp.py
#Start the Mediapipe gesture recognition program
rosrun dofbot_pro_apriltag MediapipeGesture.py
```



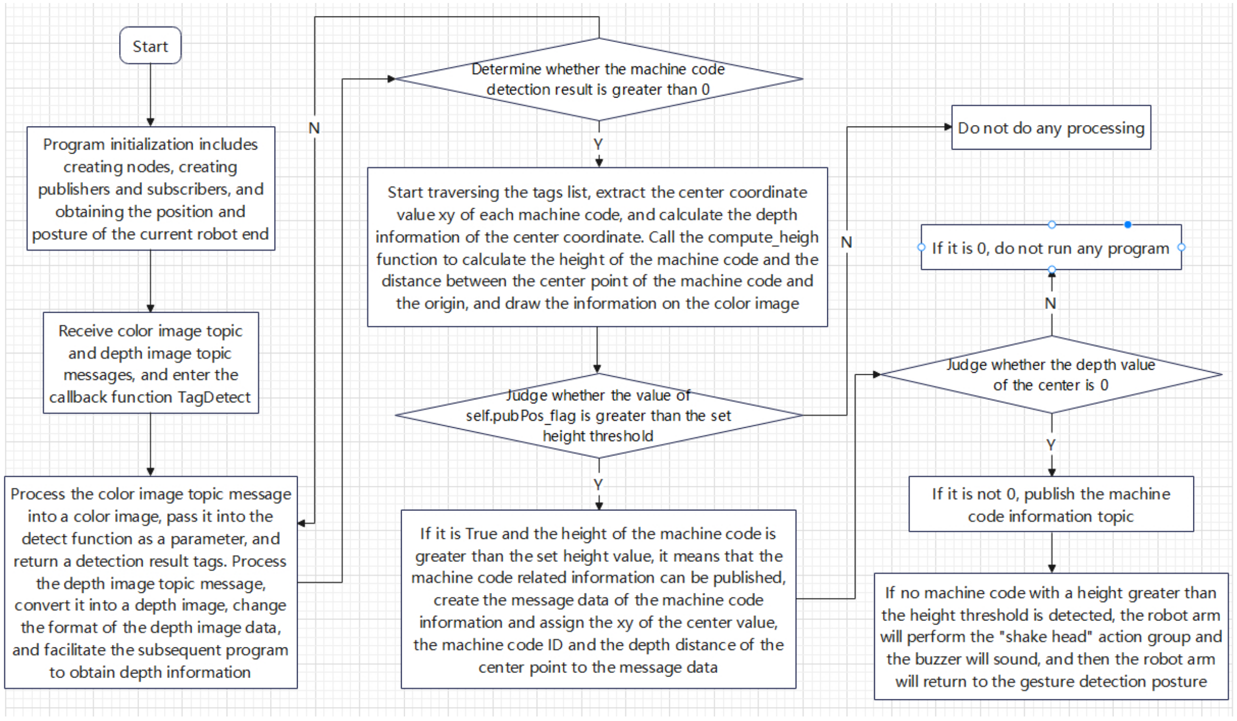
2.2、Operation

After the program is started, the robot arm will begin to show the gesture recognition posture. The number of recognizable gestures is one to five. Gesture recognition Wait for about 3 seconds, wait for the machine code to change its posture, and become the posture of detecting and recognizing the machine code. Press the space bar to start recognition; if the height of the recognized machine code is higher than the calculated threshold, the robot arm will lower its claws to grab the wooden block of the machine code and place it in the set position; after placement, the robot arm returns to the posture of recognizing the machine code, and the space bar needs to be pressed again for the next recognition. If the machine code higher than the set height threshold is not recognized, the robot arm will make a "shaking head" action group and the buzzer will sound, and then the robot arm returns to the posture of recognizing gestures.

Height threshold calculation: 30 + gesture recognition result * 10

3. Program flow chart

apriltag_list_Hight.py


Start
Program initialization includes creating nodes, creating publishers and subscribers, and obtaining the position and posture of the current robot end
Receive color image topic and depth image topic messages, and enter the callback function TagDetect
Process the color image topic message into a color image, pass it into the detect function as a parameter, and return a detection result tags. Process the depth image topic message, convert it into a depth image, change the format of the depth image data, and facilitate the subsequent program to obtain depth information
N
Determine whether the machine code detection result is greater than 0
Y
Start traversing the tags list, extract the center coordinate value xy of each machine code, and calculate the depth information of the center coordinate. Call the compute_heigh function to calculate the height of the machine code and the distance between the center point of the machine code and the origin, and draw the information on the color image
Judge whether the value of self.pubPos_flag is greater than the set height threshold
Y
If it is True and the height of the machine code is greater than the set height value, it means that the machine code related information can be published, create the message data of the machine code information and assign the xy of the center value, the machine code ID and the depth distance of the center point to the message data
N
Do not do any processing
If it is 0, do not run any program
N
Judge whether the depth value of the center is 0
Y
If it is not 0, publish the machine code information topic
If no machine code with a height greater than the height threshold is detected, the robot arm will perform the "shake head" action group and the buzzer will sound, and then the robot arm will return to the gesture detection posture

4. Core code analysis

4.1. MediapipeGesture.py

Code path:

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

You can refer to the tutorial [3D space sorting and clamping\3. Mediapipe gesture recognition machine code ID sorting] in part 4.1 [MediapipeGesture.py] content.

4.2, apriltag_list_Hight.py

Code path:

`/home/jetson/dofbot_pro_ws/src/dofbot_pro_apriltag/scripts/apriltag_list_Hight.py`

Import necessary library functions,

```
import cv2
import rospy
import numpy as np
from sensor_msgs.msg import Image
import message_filters
from std_msgs.msg import Float32,Int8,Bool
#导入绘制机器码库
#Import drawing machine code library
from vutils import draw_tags
#导入机器码库
#Import machine code library
from dt_apriltags import Detector
from cv_bridge import CvBridge
import cv2 as cv
```

```
from dofbot_info.srv import kinemarics, kinemaricsRequest, kinemaricsResponse
from dofbot_pro_info.msg import *
from std_msgs.msg import Float32,Bool
encoding = ['16UC1', '32FC1']
import time
import os
import queue
#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
#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
```

Initialize program parameters, create publishers and subscribers,

```
def __init__(self):
    rospy.init_node('apriltag_detect')
    #机械臂识别机器码的姿态
    #The robot arm recognizes the posture of the machine code
    self.init_joints = [90.0, 120, 0, 0.0, 90, 90]
    #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)
    #创建发布夹取结果的订阅者
    #Create a subscriber to publish the fetch results
    self.pubGraspStatus = rospy.Publisher("grasp_done", Bool, queue_size=1)
    #创建发布蜂鸣器话题的发布者
    #Create a publisher for the buzzer topic
    self.pub_buzzer = rospy.Publisher("Buzzer", Bool, queue_size=1)
    #创建发布机器码信息的发布者
    #Create a publisher that publishes machine code information
    self.tag_info_pub = rospy.Publisher("PosInfo",AprilTagInfo,queue_size=1)
    #创建发布机械臂目标角度的发布者
    #Create a publisher that publishes the target angle of the robotic arm
    self.pubPoint = rospy.Publisher("TargetAngle", ArmJoint, queue_size=1)
    #创建订阅手势识别结果的订阅者
    #Create a subscriber to subscribe to gesture recognition results
    self.sub_targetID = rospy.Subscriber("TargetId",Int8,self.GetTargetIDCallback,
queue_size=1)
    #将彩色和深度图像订阅的消息进行时间同步
    #Synchronize the time of color and depth image subscription messages
    self.TimeSynchronizer = 
message_filters.ApproximateTimeSynchronizer([self.rgb_image_sub,self.depth_image_sub
],1,0.5)
    #创建订阅夹取结果的订阅者
    #Create a subscriber to subscribe to the fetch results
    self.grasp_status_sub = rospy.Subscriber('grasp_done', Bool, 
self.GraspStatusCallback, queue_size=1)
    #处理同步消息的回调函数TagDetect，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 TagDetect that handles the synchronization message is 
connected to the subscribed message so that it can be automatically called when a 
new message is received
    self.TimeSynchronizer.registerCallback(self.TagDetect)
    #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()
    #发布机器码信息的标识，为True时发布/TagInfo话题数据
    #The flag for publishing machine code information. When it is True, it will 
publish the /TagInfo topic data.
    self.pubPos_flag = False
    self.done_flag = True
    #初始化设定的高度阈值为0.0
    #The initial height threshold is set to 0.0
    self.set_height = 0.0
    #初始化设定的距离阈值为0.0
    #The initial distance threshold is set to 0.0
    self.set_dist = 0.0
    self.detect_flag = False
    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.target_id = 31
    self.cnt = 0
    self.Center_x_list = []
    self.Center_y_list = []
    #机械臂识别手势的姿态
    #Robotic arm recognizes gestures
    self.search_joints = [90,150,12,20,90,30]
    #当前的机械臂末端的位置和位姿
    #The current position and posture of the end of the robotic arm
    self.CurEndPos = [-0.006,0.116261662208,0.0911289015753,-1.04719,-0.0,0.0]
    #相机内置参数
    # Camera built-in 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]])
    exit_code = os.system('rosservice call /camera/set_color_exposure  50')
```

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 posture is not estimated.
    None: Indicates that no camera parameters are provided, and only simple detection may be performed.
    0.025: It may be the set label size (usually in meters), which is used 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
    #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
    draw_tags(result_image, tags, corners_color=(0, 0, 255), center_color=(0, 255,
0))
    key = cv2.waitKey(10)
    #定义self.Center_x_list和self.Center_y_list的长度
    #Define the length of self.Center_x_list and self.Center_y_list
    self.Center_x_list = list(range(len(tags)))
    self.Center_y_list = list(range(len(tags)))
    #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
    #Wait for keyboard input, 32 means the space key is pressed, after pressing it,
the value of self.pubPos_flag is changed, indicating that the machine code related
information can be released
    if key == 32:
        self.pubPos_flag = True
    #判断tags的长度，大于0则表示有检测到机器码以及夹取机器码完成的标识
    #Judge the length of tags. If it is greater than 0, it means that the machine
code has been detected and the machine code has been captured.
    if len(tags) > 0 and self.done_flag == True:
        #遍历机器码
        #Traverse the machine code
        for i in range(len(tags)):
            #机器码的中心xy值存在Center_x_list列表和Center_y_list列表中
            #The center xy values of the machine code are stored in the
Center_x_list list and the Center_y_list list
            center_x, center_y = tags[i].center
            self.Center_x_list[i] = center_x
            self.Center_y_list[i] = center_y
            cx = center_x
            cy = center_y
            #计算中心坐标的深度值
            #Calculate the depth value of the center coordinate
            cz = depth_image_info[int(cy),int(cx)]/1000
            #调用compute_heigh函数，计算机器码的高度，传入的参数是机器码的中心坐标和中心点的深度
值，返回的是一个位置列表，pose[2]表示z值，也就是高度值
            #Call the compute_heigh function to calculate the height of the machine
code. The parameters passed in are the center coordinates of the machine code and
the depth value of the center point. The returned value is a position list. pose[2]
represents the z value, which is the height value.
            pose = self.compute_heigh(cx,cy,cz)
            #对高度值进行放大运算，把单位换算成毫米
            #Enlarge the height value and convert the unit into millimeters
            heigh_detect = round(pose[2],4)*1000
            heigh = 'heigh: ' + str(heigh_detect) + 'mm'
            #计算机器码离基坐标系的距离值，对该值进行放大运算，把单位换算成毫米
            #Calculate the distance between the machine code and the base coordinate
system, enlarge the value, and convert the unit into millimeters
            dist_detect = math.sqrt(pose[1] ** 2 + pose[0]** 2)
            dist_detect = round(dist_detect,3)*1000
            dist = 'dist: ' + str(dist) + 'mm'
            #高度和距离值使用opencv绘制在彩色图像上
            #Height and distance values ••are drawn on the color image using opencv
            cv.putText(result_image, heigh, (int(cx)+5, int(cy)-15), 
cv.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0), 2)
            cv.putText(result_image, dist, (int(cx)+5, int(cy)+15), 
cv.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0), 2)
            #如果检测到的机器码高度大于设定的高度阈值且self.pubPos_flag 的值为True且设定的高度阈值不为0
            #If the detected machine code height is greater than the set height 
threshold and the value of self.pubPos_flag is True and the set height threshold is 
not 0
            if heigh_detect>=self.set_height and self.set_height!=0 and 
self.pubPos_flag == True:
                print("self.set_height: ",self.set_height)
                print("heigh_detect: ",heigh_detect)
                #改变self.detect_flag的值，为True则表示识别到了有高于设定阈值的机器码
                #Change the value of self.detect_flag. If it is True, it means that 
a machine code higher than the set threshold has been identified.
                self.detect_flag  = True
                #给消息数据赋值，id值为机器码的id，x和y为机器码的中心值，z为中心点的深度值，这里做了按比例缩小1000倍数，单位是米
                #Assign values to the message data. The id value is the id of the 
machine code. x and y are the center values ••of the machine code. z is the depth 
value of the center point. Here, it is scaled down by 1000 times. The unit is meter.
                tag = AprilTagInfo()
                tag.id = tags[i].tag_id
                tag.x = self.Center_x_list[i]
                tag.y = self.Center_y_list[i]
                tag.z = depth_image_info[int(tag.y),int(tag.x)]/1000
                #如果深度信息不为0，说明为有效数据，然后发布机器码信息的消息
                #If the depth information is not 0, it means it is valid data, and 
then publish the message of machine code information
                if tag.z!=0 :
                    self.tag_info_pub.publish(tag)
                    self.pubPos_flag = False
                    self.done_flag = False
                else:
                    print("Invalid distance.")
        #如果self.detect_flag为False表示没有识别到高于高度阈值的机器码且设定的高度阈值不为0以及使能发布机器码消息，符合三者条件说明没有识别到高于高度阈值的机器码。
        #If self.detect_flag is False, it means that no machine code higher than the 
height threshold is recognized, the set height threshold is not 0, and the release 
of machine code messages is enabled. If these three conditions are met, it means 
that no machine code higher than the height threshold is recognized.
        if self.detect_flag != True and  self.set_height!=0 and 
self.pubPos_flag==True:
            print("----------------------------------------")
            self.set_height!=0
            #机械臂做出“摇头”的动作组
            #The robot arm makes a "shaking head" action group
            self.shake()
            #time.sleep(2)
            #回答识别手势的姿态，准备识别下一次的手势
            #Answer the gesture recognition posture and prepare to recognize the next gesture
            self.pub_arm(self.search_joints)
            #发布夹取完成的话题，以便于下次手势识别节点程序发布手势识别的结果
            #Publish the topic of completed clamping so that the gesture recognition node program can publish the results of gesture recognition next time
            grasp_done = Bool()
            grasp_done.data = True
            self.pubGraspStatus.publish(grasp_done)
        self.pubPos_flag = False
#如果按下空格后没有识别到任何机器码，则同样机械臂做出“摇头”的动作组，蜂鸣器响，然后回到识别手势的姿态
#If no machine code is recognized after pressing the space bar, the robot arm will make the "shake head" action group, the buzzer will sound, and then return to the gesture recognition posture
    elif self.pubPos_flag == True and len(tags) == 0:
        self.shake()
        self.pub_arm(self.search_joints)
        grasp_done = Bool()
        grasp_done.data = True
        self.pubGraspStatus.publish(grasp_done)
    result_image = cv2.cvtColor(result_image, cv2.COLOR_RGB2BGR)
    cv2.imshow("result_image", result_image)
    key = cv2.waitKey(1)
```

The callback function GetTargetIDCallback of the gesture recognition result,

```
def GetTargetIDCallback(self,msg):
    print("msg.data: ",msg.data)
    #计算距离阈值单位是毫米mm，最小是160mm，最大是20mm
    #The distance threshold unit is mm, the minimum is 160mm and the maximum is 20mm
    self.set_dist = 150 + msg.data*10
    #计算高度阈值单位是毫米mm，最小是40mm，最大是80mm
    #The unit of height threshold calculation is millimeter, the minimum is 40mm and the maximum is 80mm
    self.set_height = 30 + msg.data*10
    print("self.set_height: ",self.set_height)
    #接收到消息后，改变机械臂的姿态，呈现识别机器码的姿态
    #After receiving the message, change the posture of the robot arm to present the posture of recognizing the machine code
    self.pub_arm(self.init_joints)
```

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].