Program Analysis for Tag Recognition

1. File Path

The program file is stored in:

/home/ubuntu/armpi_pro/src/visual_processing/scripts/visual_processing_node.py

(Image processing)

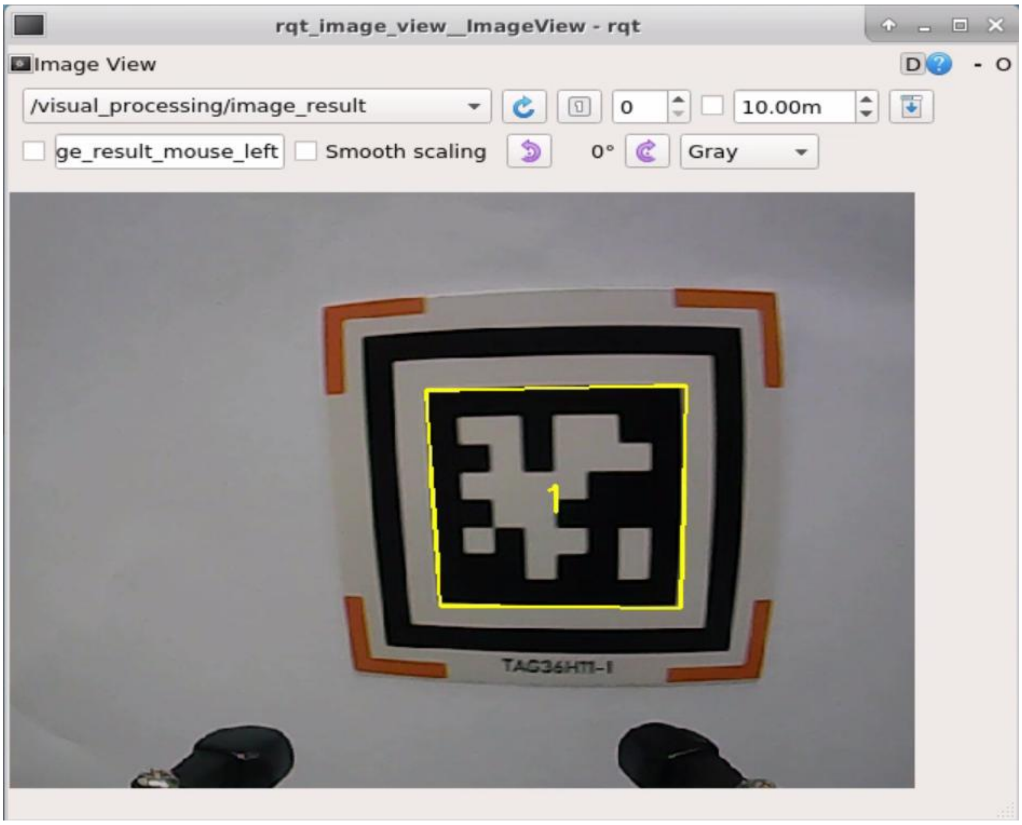
/home/ubuntu/armpi_pro/src/apriltag_detect/scripts/apriltag_detect_node.py

(Chassis Control)

2. Program Performance

After starting the game, the robotic arm will recognize the tag ID. Then, we can see that the tag ID will be framed in rqt tool after recognition and the robotic arm will perform the corresponding action.

Tag ID	**Corresponding action**
1	Drawing a triangle.
2	Drawing a circle
3	Drifting performance



3. Program Analysis

Note: please back up the initial program before making any modifications. It is prohibited editing the source code files directly to prevent making changes in an incorrect manner that could lead to robot malfunctions, rendering them irreparable.

3.1 Import Parameter Module

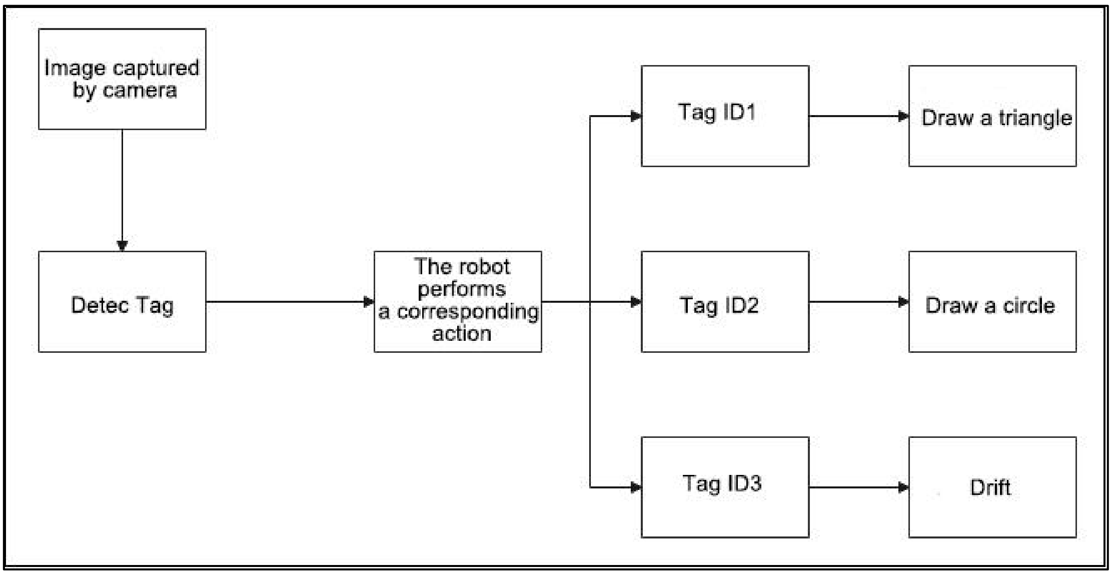
Imported Module	Function
import sys	The sys module of Python is imported to access to system-related functionalities and variables.
import cv2	The OpenCV library of Python is imported to perform image processing and computer

	vision-related functions.
import time	The time module of Python is imported to perform time-related functionalities, such as delay operations.
import math	The math module of Python is imported to perform mathematical operations and functions.
import rospy	The Python library rosy is imported for communication and interaction with ROS.
import numpy as np	The NumPy library is imported and is renamed as np for performing array and matrix operations.
from armpi_pro import Misc	The Misc module is imported from arm_pi_pro package to handle the recognized rectangular data.
from armpi_pro import apriltag	The apriltag module is imported from arm_pi_pro package to perform Apriltag recognition and processing.
from threading import RLock, Timer	The “RLock” class and “Timer” class is imported from the threading module of Python for thread-related operations.
from std_srvs.srv import *	All service message types are imported from the std_srvs in ROS for defining and

	using standard service messages.
from std_msgs.msg import *	All message types are imported form the std_msgs package in ROS for defining and using standard messages.
from sensor_msgs.msg import Image	The image message type is imported from the sensor_msgs packages for processing image data.
from visual_processing.msg import Result	The Result message type is imported from the visual_processing package for the message of image processing results.
from visual_processing.srv import SetParam	The SetParam service type is imported from the visual_processing packages for using customs service related to parameter settings.
from sensor.msg import Led	The Led message type is imported form the sensor.msg module for controlling or representing the LED status on a sensor.
from chassis_control.msg import *	All message types are imported from the chassis_control.msg module, which indicated that all message types defined in this module is imported to perform the chassis control.
from visual_patrol.srv import SetTarget	The SetTarget service type is imported from the visual_patrol.srv module is used to

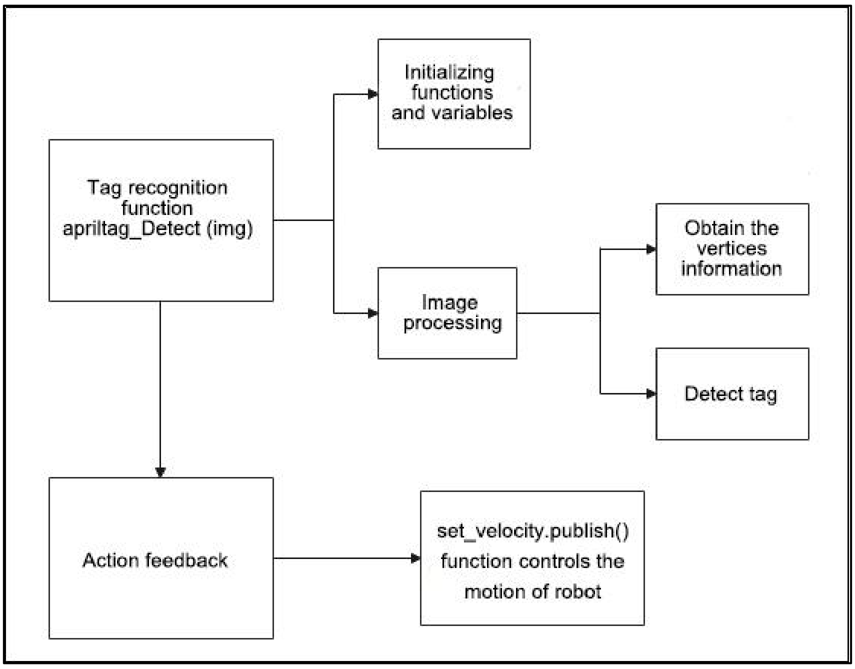
	set a target for line following.
from hiwonder_servo_msgs.msg import MultiRawIdPosDur	The MultiRawIdPosDur message type is imported from the hiwonder_servo_msgs.msg module for controlling servos.
from armpi_pro import PID	The PID class is imported from the armpi_pro module to perform PID algorithm.
from armpi_pro import bus_servo_control	The bus_servo_control module is imported from the armpi_pro module, including the functions and methods related to the servo control.
from kinematics import ik_transform	The ik_transform function is imported from the kinematics module to perform conversion of inverse kinematics.

3.2 Program Logic



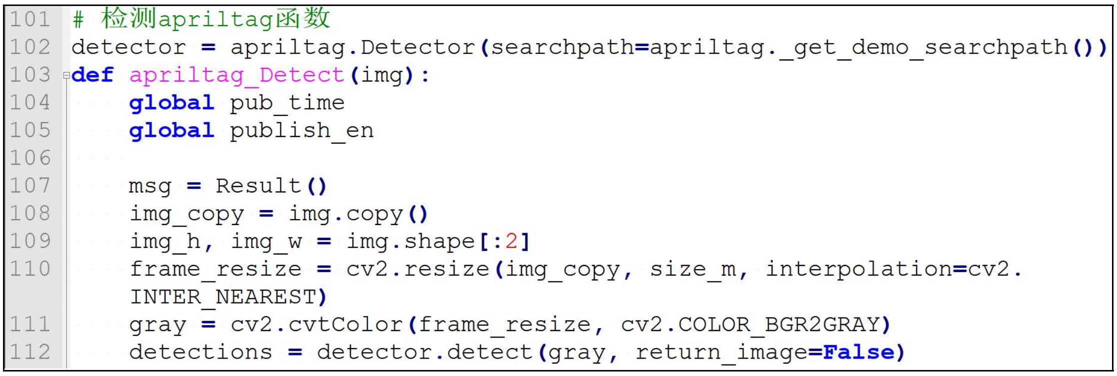
By capturing image information through the camera, the robot performs different actions for various detected labels. When tag ID 1 is detected, the robot moves to create a triangular shape. When tag ID 2 is detected, the robot moves to create a circular shape. And when tag ID 3 is detected, the robot demonstrates drifting maneuvers.

3.3 Code Analysis


Initializing functions and variables
Tag recognition function apriltag_Detect (img)
Obtain the vertices information
Image processing
Detect tag
Action feedback
set_velocity.publish() function controls the motion of robot

3.3.1 Image Processing

Initializing functions and variables

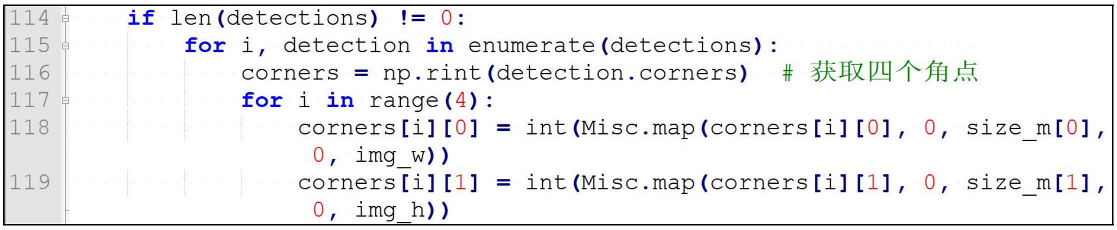


```
# 检测apriltag函数
detector = apriltag.Detector(searchpath=apriltag._get_demo_searchpath())
def apriltag_Detect(img):
    global pub_time
    global publish_en

    msg = Result()
    img_copy = img.copy()
    img_h, img_w = img.shape[:2]
    frame_resize = cv2.resize(img_copy, size_m, interpolation=cv2.INTER_NEAREST)
    gray = cv2.cvtColor(frame_resize, cv2.COLOR_BGR2GRAY)
    detections = detector.detect(gray, return_image=False)
```

Obtain the vertices information

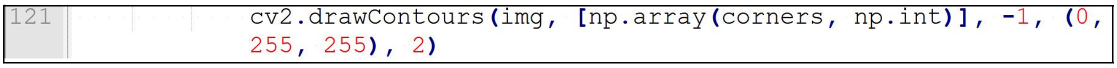
Get the four vertices of tag with np.rint() function.



```
    if len(detections) != 0:
        for i, detection in enumerate(detections):
            corners = np.rint(detection.corners)  # 获取四个角点
            for i in range(4):
                corners[i][0] = int(Misc.map(corners[i][0], 0, size_m[0], 0, img_w))
                corners[i][1] = int(Misc.map(corners[i][1], 0, size_m[1], 0, img_h))
```

Detect tag

After obtaining the vertices information, the tag is recognized by calling drawContours() function in cv2 library.



```
            cv2.drawContours(img, [np.array(corners, np.int)], -1, (0, 255, 255), 2)
```

The meaning of parameters in parentheses is as follow:

The first parameter “img” is the input image.

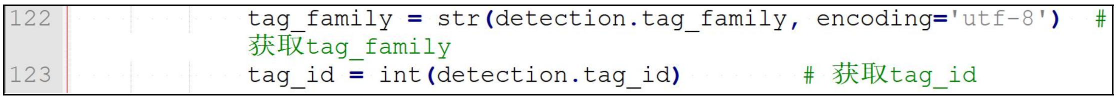
The second parameter “[np.array(corners, np.int)]” is the contour which is list in Python.

The third parameter “-1” is the index of the contour, where the value represents all contours in the drawn contour list.

The fourth parameter “(0, 255, 255)” is the color of contour and its order is B, G and R.

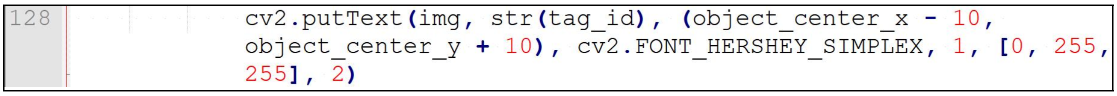
The fifth parameter “2” is the width of contour.

1) The type of the obtained tag (tag_family) and ID (tag_id).



```
            tag_family = str(detection.tag_family, encoding='utf-8')  # 获取tag_family
            tag_id = int(detection.tag_id)        # 获取tag_id
```

2) Print the tag ID and type in the transmitted image by calling putText() function in cv2 library.



```
            cv2.putText(img, str(tag_id), (object_center_x - 10, object_center_y + 10), cv2.FONT_HERSHEY_SIMPLEX, 1, [0, 255, 255], 2)
```

The meaning of parameters in parentheses is as follow:

The first parameter “img” is the input image.

The second parameter “str(tag_id)” is the display content.

The third parameter “(object_center_x - 10, object_center_y + 10)” is the display position.

The fourth parameter “cv2.FONT_HERSHEY_SIMPLEX” is the type of font.

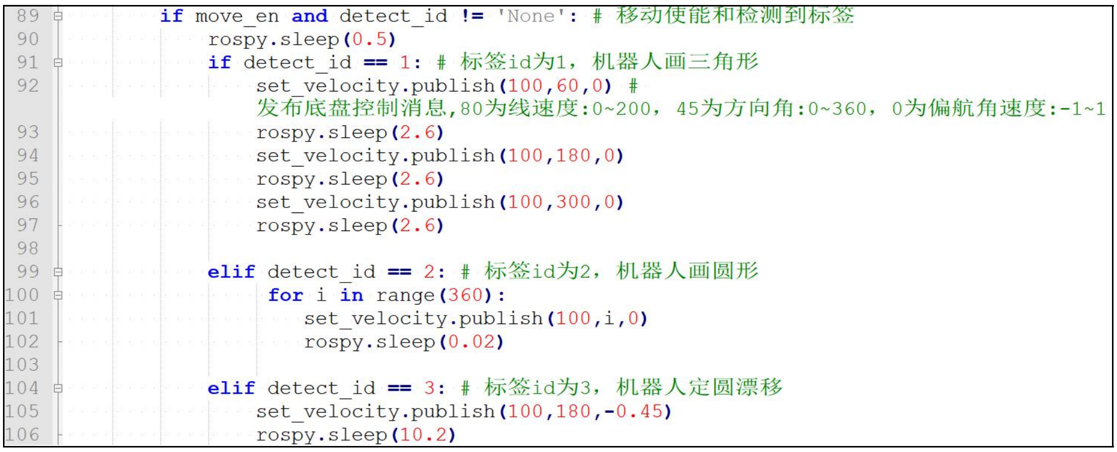
The fifth parameter “1” is the size of font.

The sixth parameter “[0, 255, 255]” is the color of font and its color is B, G and R. Here is yellow.

The seventh parameter “2” is the thickness of font.

3.3.2 Control action

After obtaining ID, control ArmPi Pro to perform corresponding action by calling set_velocity.publish() function in hiwonder_servo_msgs.msg library.



```
        if move_en and detect_id != 'None': # 移动使能和检测到标签
            rospy.sleep(0.5)
            if detect_id == 1: # 标签id为1，机器人画三角形
                set_velocity.publish(100,60,0) #
                发布底盘控制消息,80为线速度:0~200， 45为方向角:0~360， 0为偏航角速度:-1~1
                rospy.sleep(2.6)
                set_velocity.publish(100,180,0)
                rospy.sleep(2.6)
                set_velocity.publish(100,300,0)
                rospy.sleep(2.6)

            elif detect_id == 2: # 标签id为2，机器人画圆形
                for i in range(360):
                    set_velocity.publish(100,i,0)
                    rospy.sleep(0.02)

            elif detect_id == 3: # 标签id为3，机器人定圆漂移
                set_velocity.publish(100,180,-0.45)
                rospy.sleep(10.2)
```

Motor control is illustrated by the code example "set_velocity.publish(100, 60, 0)," where the meanings of the parameters within the parentheses are as follows:

The first parameter "100" represents the linear velocity, indicating the motor's speed in millimeters per second. The range is "-100 to 100," and when the value is negative, the motor rotates in reverse.

The second parameter "60" denotes the orientation angle, representing the direction of the vehicle's movement in degrees. The range is "0 to 360." Here, 90 degrees corresponds to forward movement, 270 degrees is backward, 0 degrees is right, and 180 degrees is left. Other angle values represent corresponding directions.

The third parameter "dx" stands for the yaw angular velocity, indicating the rate of deviation for the vehicle. It is measured in 5 degrees per second. In the program, the range is set as "-0.8 to 0.8." Positive values result in clockwise rotation, while negative values lead to counterclockwise rotation.