9. AprilTag code recognition

9. AprilTag code recognition
    9.1 Introduction
    9.2 Startup
        Preparation before starting the program
        Program Description
        Program start
        Source code

9.1 Introduction

Apriltag is a coded mark commonly used in machine vision. It has a high recognition rate and reliability and can be used for various tasks, including augmented reality, robotics, and camera calibration.

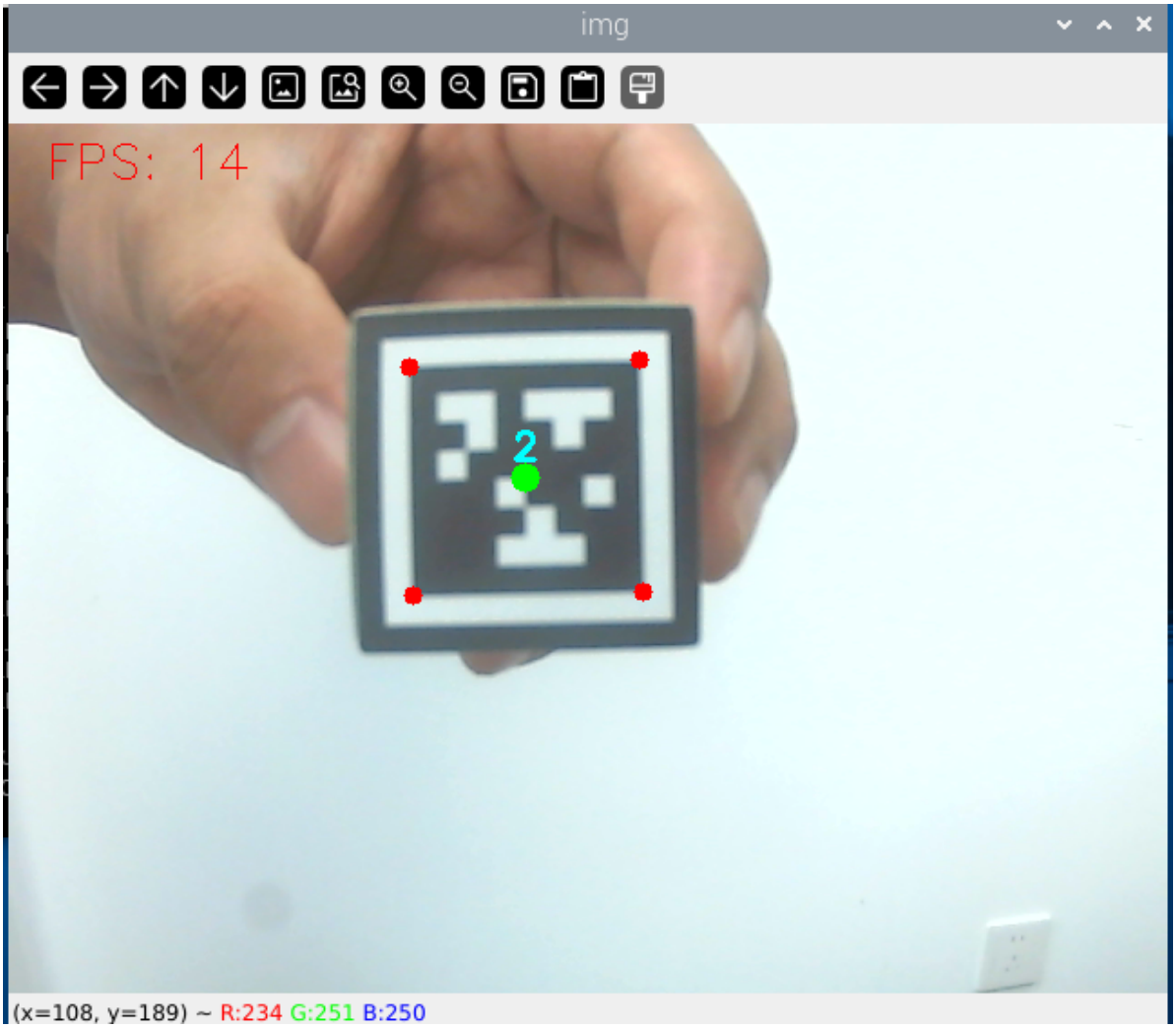
9.2 Startup

Preparation before starting the program

This apriltag tag code uses the TAG36H11 format. The factory has been equipped with the relevant tag code and attached to the building block. You need to take out the building block and place it under the camera screen for identification.

Program Description

After the program is started, the camera captures the image, put the tag code into the camera screen, the system will identify and frame the four vertices of the tag code, and display the ID number of the tag code.

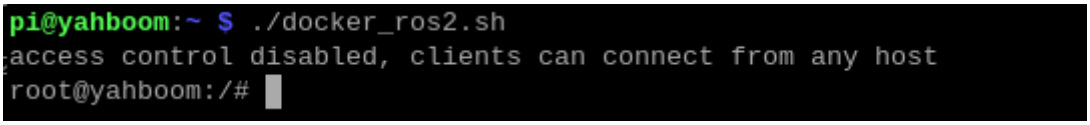


Program start

Open a terminal and enter the following command to enter docker,

```
./docker_ros2.sh
```

The following interface appears, which means that you have successfully entered docker



Enter the following command in the docker terminal to start the program

```
ros2 run yahboomcar_apriltag apriltag_identify
```

Source code

After entering the docker container, the source code of this function is located at,

Code path:
/root/yahboomcar_ws/src/yahboomcar_apriltag/yahboomcar_apriltag/apriltag_identify.py

```
#!/usr/bin/env python3
# encoding: utf-8
import cv2 as cv
import time
```

```python
from dt_apriltags import Detector
from yahboomcar_apriltag.vutils import draw_tags
import logging
import yahboomcar_apriltag.logger_config as logger_config
import rclpy
from rclpy.node import Node
from std_msgs.msg import String, Float32MultiArray

class ApriltagIdentify(Node):
    def __init__(self):
        super().__init__('apriltag_identify_node')
        logger_config.setup_logger()
        self.image = None
        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)

        # AprilTag positions
        self.publisher_ = self.create_publisher(Float32MultiArray,
'apriltag_positions', 10)
        # AprilTag ID
        self.single_tag_id_publisher_ = self.create_publisher(String,
'single_apriltag_id', 10)

    def getApriltagPosMsg(self, image):
        self.image = cv.resize(image, (640, 480))
        msg = Float32MultiArray()
        try:
            tags = self.at_detector.detect(cv.cvtColor(
                self.image, cv.COLOR_BGR2GRAY), False, None, 0.025)
            tags = sorted(tags, key=lambda tag: tag.tag_id)
            if len(tags) > 0:
                for tag in tags:
                    point_x = tag.center[0]
                    point_y = tag.center[1]
                    (a, b) = (round(((point_x - 320) / 4000), 5),
                              round(((480 - point_y) / 3000) * 0.7+0.15, 5))
                    msg.data.extend([tag.tag_id, a, b])

                self.image = draw_tags(self.image, tags, corners_color=(
                    0, 0, 255), center_color=(0, 255, 0))
        except Exception as e:
            logging.info('getApriltagPosMsg e = {}'.format(e))

        return self.image, msg

    def getSingleApriltagID(self, image):
        self.image = cv.resize(image, (640, 480))
        tagId = ""
        try:
```

```
            tags = self.at_detector.detect(cv.cvtColor(
                self.image, cv.COLOR_BGR2GRAY), False, None, 0.025)
            tags = sorted(tags, key=lambda tag: tag.tag_id)
            if len(tags) == 1:
                tagId = str(tags[0].tag_id)
                self.image = draw_tags(self.image, tags, corners_color=(
                    0, 0, 255), center_color=(0, 255, 0))
        except Exception as e:
            logging.info('getSingleApriltagID e = {}'.format(e))

        return self.image, tagId

    def detect_and_publish(self):
        capture = cv.VideoCapture(0)
        capture.set(cv.CAP_PROP_FRAME_WIDTH, 640)
        capture.set(cv.CAP_PROP_FRAME_HEIGHT, 480)

        t_start = time.time()
        m_fps = 0
        try:
            while capture.isOpened():
                action = cv.waitKey(10) & 0xFF
                if action == ord('q'):
                    break
                ret, img = capture.read()
                img, data = self.getApriltagPosMsg(img)

                # AprilTag positions
                self.publisher_.publish(data)

                # AprilTag ID
                _, single_tag_id = self.getSingleApriltagID(img)
                if single_tag_id:
                    single_tag_msg = String()
                    single_tag_msg.data = single_tag_id
                    self.single_tag_id_publisher_.publish(single_tag_msg)

                m_fps += 1
                fps = m_fps / (time.time() - t_start)
                if (time.time() - t_start) >= 2000:
                    t_start = time.time()
                    m_fps = fps
                text = "FPS: " + str(int(fps))
                cv.putText(img, text, (20, 30), cv.FONT_HERSHEY_SIMPLEX, 0.9,
(0, 0, 255), 1)
                cv.imshow('img', img)
        except Exception as e:
            logging.error('Exception occurred: {}'.format(e))
        finally:
            capture.release()
            cv.destroyAllWindows()

def main(args=None):
    rclpy.init(args=args)
    apriltag_identify = ApriltagIdentify()
```

```
    try:
        apriltag_identify.detect_and_publish()
    except KeyboardInterrupt:
        pass

if __name__ == '__main__':
    main()
```