11. Apriltag code recognition

11.1. 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.

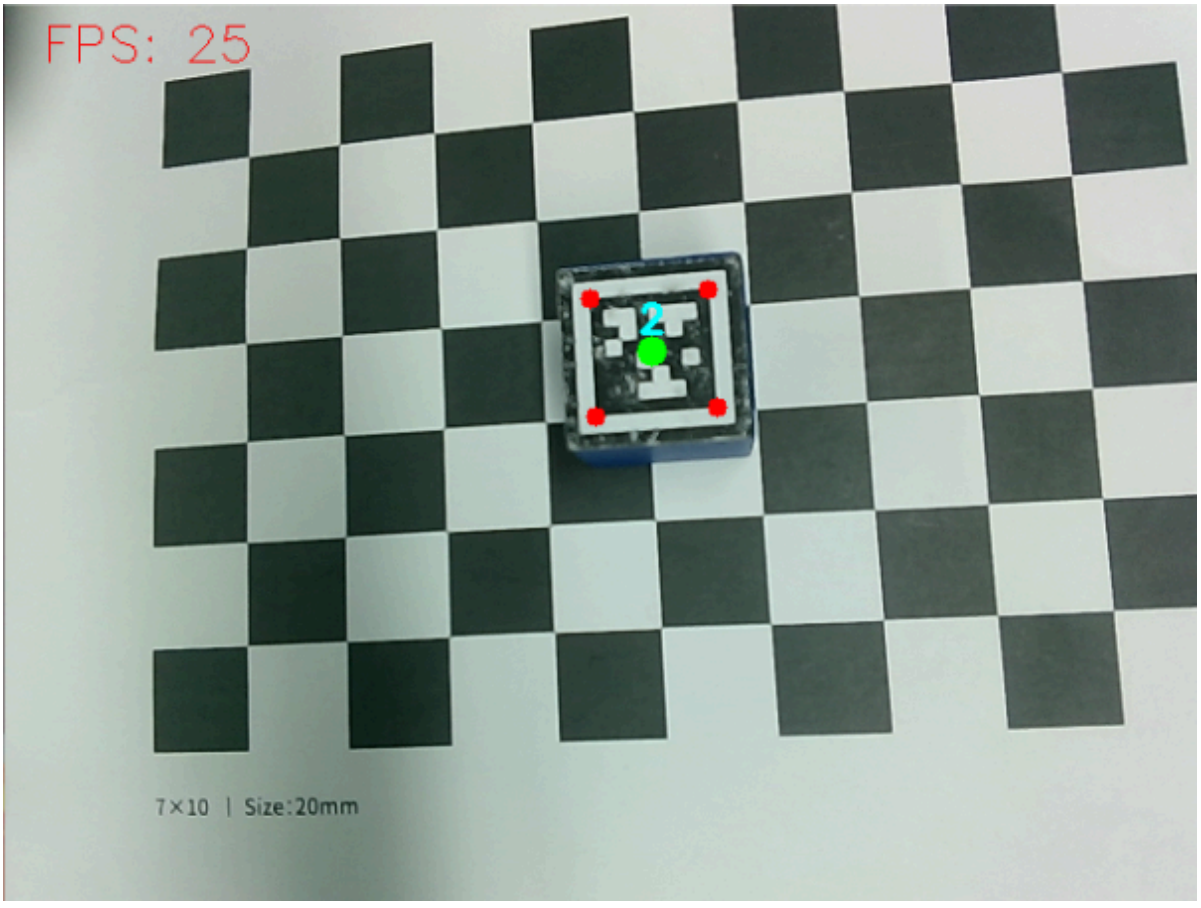
11.2. Startup

11.2.1. Preparation before program startup

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

11.2.2. Program description

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



11.2.3, Program Startup

- Enter the following command to start the program

```
python3 ~/jetcobot_ws/src/jetcobot_apriltag/scripts/apriltag_identify.py
```

11.2.4, Source Code

Code path: ~/jetcobot_ws/src/jetcobot_apriltag/scripts/apriltag_identify.py

```
#!/usr/bin/env python3
# encoding: utf-8
import cv2 as cv
import time
from dt_apriltags import Detector
from jetcobot_utils.vutils import draw_tags
import logging
import jetcobot_utils.logger_config as logger_config

class ApriltagIdentify:
    def __init__(self):
        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)

    def getApriltagPosMsg(self, image):
        self.image = cv.resize(image, (640, 480))
        msg = {}
        try:
            tags = self.at_detector.detect(cv.cvtColor(
                self.image, cv.COLOR_RGB2GRAY), 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) * 0.8+0.15, 5))
                    msg[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_RGB2GRAY), 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

if __name__ == '__main__':
    capture = cv.VideoCapture(0)
    capture.set(cv.CAP_PROP_FRAME_WIDTH, 640)
    capture.set(cv.CAP_PROP_FRAME_HEIGHT, 480)
    tag_identify = ApriltagIdentify()

    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 = tag_identify.getApriltagPosMsg(img)

            m_fps = 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:
        pass
    capture.release()
    cv.destroyAllWindows()
```