В течении разработки проекта \_я\_ разрабатывал систему компьютерного зрения для робота под управлением raspberry pi. Испытания работоспособности программы с использованием python-библиотеки OpenCV производились на различных камерах и одинаково хорошо выполняли поставленную задачу. Для разработки и отладки кода использовалась IDE: Visual studio code, т.к. она позволяет модифицировать свои возможности и упрощает разработку. Для нахождения квадратной метки определенного цвета я использовал черно-белую маску или сетку, а затем искал квадратные контуры и выделял их координаты. При попадании центра квадрата правее вертикальной оси изображения робот должен подворачивать вправо, также и для случая с левосторонним расположением объекта. Для того, чтобы другие предметы не мешали обнаружению, используется функция, которая считает размер найденного объекта и поэтому не реагирует на них. Если все условия соблюдены, то робот направляется к метке и если оказывается непосредственно перед меткой, то останавливается и выполняет необходимые операции с клиентом. Ниже представлен фрагмент кода, который непосредственно запускает веб-сервер с помощью фреймворка flask и транслирует изображение с камеры благодаря OpenCV.

from flask import Flask, render\_template, Response

from camera import VideoCamera

import numpy as np

import cv2

app = Flask(\_\_name\_\_)

@app.route('/')

def index():

    return render\_template('index.html')

def gen(camera):

    while True:

        USLOVNAYA\_PEREMENNAYA = 0

        RECTCOLOR = (0, 255, 0)

        RTHICK = 2

        if(USLOVNAYA\_PEREMENNAYA == 0):

            hsv\_min = np.array((40,100,100), np.uint8)

            hsv\_max = np.array((80,255,255), np.uint8)

        else:

            hsv\_min = np.array((110,120,120), np.uint8)

            hsv\_max = np.array((130,255,255), np.uint8)

        BLOBSIZE\_STOP = 40000

        BLOBSIZE = 700

        my\_color = (0,0,255)

        crange = [0,0,0, 0,0,0]

        def checkSize(w, h):

            if w \* h > BLOBSIZE:

                return True

            else:

                return False

        def stop(w,h):

            if w\*h>BLOBSIZE\_STOP:

                return True

            else:

                return False

        height = 720

        width = 1280

        width\_center=width/2;

        img = camera.get\_frame()

        hsv = cv2.cvtColor(img, cv2.COLOR\_BGR2HSV )

        thresh = cv2.inRange(hsv, hsv\_min, hsv\_max )

        contours, hierarchy = cv2.findContours(thresh.copy(), cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

        kernel = np.ones((5, 5), np.uint8)

        cv2.dilate(thresh, kernel, iterations = 1)

        cv2.erode(thresh, kernel, iterations = 1)

        cv2.drawContours(img, contours, -1, (255, 0, 0), 2, cv2.LINE\_AA, hierarchy, 0)

        cv2.drawContours(img, contours, -1, (255, 0, 0), 2, cv2.LINE\_AA, hierarchy, 2)

        center\_rectangle = 2

        x,y,w,h = 1,2,3,4

        if len(contours) != 0:

            c = max(contours, key = cv2.contourArea)

            x,y,w,h = cv2.boundingRect(c)

            if checkSize(w, h):

                # выводим его str("UP left")str("UP right")str("down left")str("down right")

                cv2.rectangle(img, (x, y), (x+w, y+h), RECTCOLOR, RTHICK)

                cv2.putText(img, str(x)+" "+str(y), (x,y), cv2.FONT\_HERSHEY\_COMPLEX\_SMALL, 1, my\_color, 2, cv2.LINE\_AA)

                cv2.putText(img, str(x+h)+" "+str(y), (x+h,y), cv2.FONT\_HERSHEY\_COMPLEX\_SMALL, 1, my\_color, 2, cv2.LINE\_AA)

                cv2.putText(img, str(x)+" "+str(y+w), (x,y+w), cv2.FONT\_HERSHEY\_COMPLEX\_SMALL, 1, my\_color, 2, cv2.LINE\_AA)

                cv2.putText(img, str(x+h)+" "+str(y+w), (x+h,y+w), cv2.FONT\_HERSHEY\_COMPLEX\_SMALL, 1, my\_color, 2, cv2.LINE\_AA)

                center\_rectangle = x + h/2;

                if stop(w,h):

                    print("STOP")

                else:

                    if(width\_center < center\_rectangle and abs(width\_center-center\_rectangle)>width\_center/2):

                        print("GO RIGHT")

                    else:

                        if(width\_center > center\_rectangle and abs(width\_center-center\_rectangle)>width\_center/2):

                            print("GO LEFT")

                        else:

                            print("GO FORWARD")

        ret, jpeg = cv2.imencode('.jpg', img)

        yield (b'--frame\r\n'

               b'Content-Type: image/jpeg\r\n\r\n' + jpeg.tobytes() + b'\r\n\r\n')

@app.route('/video\_feed')

def video\_feed():

    return Response(gen(VideoCamera()),

                    mimetype='multipart/x-mixed-replace; boundary=frame')

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(host='localhost',port = 80, debug=True)

Подключаемый модуль:

import cv2

class VideoCamera(object):

    def \_\_init\_\_(self):

        # Using OpenCV to capture from device 0. If you have trouble capturing

        # from a webcam, comment the line below out and use a video file

        # instead.

        height = 720

        width = 1280

        self.video = cv2.VideoCapture(0)

        self.video.set(3, width)

        self.video.set(4, height)

        # If you decide to use video.mp4, you must have this file in the folder

        # as the main.py.

        # self.video = cv2.VideoCapture('video.mp4')

    def \_\_del\_\_(self):

        self.video.release()

    def get\_frame(self):

        success, image = self.video.read()

        # We are using Motion JPEG, but OpenCV defaults to capture raw images,

        # so we must encode it into JPEG in order to correctly display the

        # video stream.

        ret, jpeg = cv2.imencode('.jpg', image)

        return image