import numpy as np

import cv2

import RPi.GPIO as GPIO

from time import sleep

video\_capture = cv2.VideoCapture(-1)

video\_capture.set(0, 700)

video\_capture.set(0, 700)

GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)

AN1 = 19

AN2 = 26

DIG1 = 16

DIG2 = 13

# Setup Output Pins

GPIO.setup(AN1, GPIO.OUT)

GPIO.setup(AN2, GPIO.OUT)

GPIO.setup(DIG1, GPIO.OUT)

GPIO.setup(DIG2, GPIO.OUT)

sleep(1)

p1 = GPIO.PWM(AN1, 100)

p2 = GPIO.PWM(AN2, 100)

while(True):

# Capture the frames

ret, image = video\_capture.read()

img = image[0:700, 0:700]

# Convert to grayscale

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

# Gaussian blur

blur = cv2.GaussianBlur(gray,(5,5),0)

# Color thresholding

ret,thresh = cv2.threshold(blur, 125,255,cv2.THRESH\_BINARY\_INV)

# Find the contours of the frame

\_, contours,hierarchy= cv2.findContours(thresh, 1, cv2.CHAIN\_APPROX\_NONE)

# Find the biggest contour (if detected)

if len(contours) > 0:

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

M = cv2.moments(c)

cx = int(M['m10']/M['m00'])

cy = int(M['m01']/M['m00'])

cv2.line(thresh,(cx,0),(cx,720),(255,0,0),3)

cv2.line(thresh,(0,cy),(1280,cy),(255,0,0),3)

cv2.line(img,(cx,0),(cx,720),(255,0,0),3)

cv2.line(img,(0,cy),(1280,cy),(255,0,0),3)

cv2.drawContours(img, contours, -1, (0,255,0), 3)

print (cx)

print (cy)

if cx >= 500:

GPIO.output(DIG1, GPIO.HIGH)

GPIO.output(DIG2, GPIO.HIGH)

p1.start(30)

p2.start(20)

sleep(1)

if cx >= 400:

GPIO.output(DIG1, GPIO.HIGH)

GPIO.output(DIG2, GPIO.HIGH)

p1.start(30)

p2.start(20)

sleep(1)

if cx < 400 and cx > 300:

GPIO.output(DIG1, GPIO.HIGH)

GPIO.output(DIG2, GPIO.HIGH)

p1.start(20)

p2.start(20)

sleep(1)

if cx <= 300:

GPIO.output(DIG1, GPIO.HIGH)

GPIO.output(DIG2, GPIO.HIGH)

p1.start(20)

p2.start(30)

sleep(1)

if cx <= 200:

GPIO.output(DIG1, GPIO.HIGH)

GPIO.output(DIG2, GPIO.HIGH)

p1.start(20)

p2.start(30)

sleep(1)

else:

p1.start(0)

p2.start(0)

#Display the resulting frame

cv2.imshow('frame',img)

cv2.imshow('blur',gray)

cv2.imshow('thresh',thresh)

if cv2.waitKey(1) & 0xFF == ord('q'):

break