import numpy as np

import cv2

import RPi.GPIO as IO

from gpiozero import AngularServo

import time

IO.setwarnings(False)

IO.setmode(IO.BCM)

servo = AngularServo(25, *min\_pulse\_width*=0.0006, *max\_pulse\_width*=0.0021)

IO.setup(23,IO.OUT)

IO.setup(24,IO.OUT)

IO.setup(18,IO.OUT)

speed = IO.PWM(18,100)

video\_capture = cv2.VideoCapture(0)

speed.start(0)

def forward\_90():

    speed.ChangeDutyCycle(100)

    IO.output(24,1)

    IO.output(23,0)

def forward\_30():

    speed.ChangeDutyCycle(30)

    IO.output(24,1)

    IO.output(23,0)

def forward\_50():

    speed.ChangeDutyCycle(60)

    IO.output(24,1)

    IO.output(23,0)

def forward\_60():

    speed.ChangeDutyCycle(60)

    IO.output(24,1)

    IO.output(23,0)

def stop():

    speed.ChangeDutyCycle(0)

    IO.output(24,0)

    IO.output(23,0)

def \_map(*x*, *in\_min*, *in\_max*, *out\_min*, *out\_max*):

    return int((x - in\_min) \* (out\_max - out\_min) / (in\_max - in\_min) + out\_min)

while(True):

    ret, frame = video\_capture.read()

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

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

    ret,thresh = cv2.threshold(blur,45,255,cv2.THRESH\_BINARY\_INV)

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

    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'])

        if cx>190 and cx<270:

            turn = \_map(cx,270,50,0,60)

            if turn < 90:

                servo.angle = turn

                forward\_50()

        if cx>310 and cx<390:

            turn = \_map(cx,310,440,-10,-60)

            if turn > -90:

                servo.angle = turn

                forward\_50()

        if cx > 390:

            turn = \_map(cx,310,600,-10,-90)

            if turn > -90:

                servo.angle = turn

                forward\_30()

        if cx < 190:

            turn = \_map(cx,270,10,0,90)

            if turn < 90:

                servo.angle = turn

                forward\_30()

        if cx>270 and cx<310:

            servo.angle = -10

            if cy>280:

                forward\_90()

            else:

                forward\_60()

2.0

import numpy as np

import cv2

import RPi.GPIO as IO

from gpiozero import AngularServo

import time

import board

import neopixel

pixels = neopixel.NeoPixel(board.D10, 10,*brightness*=0.2)

IO.setwarnings(False)

IO.setmode(IO.BCM)

servo = AngularServo(25, *min\_pulse\_width*=0.0006, *max\_pulse\_width*=0.0021)

IO.setup(23,IO.OUT)

IO.setup(24,IO.OUT)

IO.setup(18,IO.OUT)

speed = IO.PWM(18,100)

buttonIn = 17

IO.setup(buttonIn,IO.IN,*pull\_up\_down*=IO.PUD\_UP)

pixels.fill((0, 0, 0))

video\_capture = cv2.VideoCapture(0)

speed.start(0)

def forward\_90():

    IO.output(24,1)

    IO.output(23,0)

    for x in range(50,90,1):

        speed.ChangeDutyCycle(x)

def forward\_30():

    speed.ChangeDutyCycle(50)

    IO.output(24,1)

    IO.output(23,0)

def forward\_50():

    speed.ChangeDutyCycle(50)

    IO.output(24,1)

    IO.output(23,0)

def forward\_60():

    speed.ChangeDutyCycle(60)

    IO.output(24,1)

    IO.output(23,0)

def stop():

    speed.ChangeDutyCycle(0)

    IO.output(24,0)

    IO.output(23,0)

def \_map(*x*, *in\_min*, *in\_max*, *out\_min*, *out\_max*):

    return int((x - in\_min) \* (out\_max - out\_min) / (in\_max - in\_min) + out\_min)

while(True):

    if(IO.input(buttonIn)==0):

        time.sleep(2)

        while(True):

            ret, frame = video\_capture.read()

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

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

            ret,thresh = cv2.threshold(blur,45,255,cv2.THRESH\_BINARY\_INV)

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

            if len(contours) > 0:

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

                M = cv2.moments(c)

                a = M['m00']+1

                cx = int(M['m10']/a)

                cy = int(M['m01']/a)

                if cx>=190 and cx<=270:

                    turn = \_map(cx,270,10,0,60)

                    if turn < 90:

                        servo.angle = turn

                        forward\_50()

                if cx >= 310 and cx<=390:

                    turn = \_map(cx,310,570,-10,-60)

                    if turn > -90:

                        servo.angle = turn

                        forward\_50()

                if cx > 390:

                    turn = \_map(cx,310,620,-10,-90)

                    if turn > -90:

                        servo.angle = turn

                        forward\_30()

                if cx < 190:

                    turn = \_map(cx,270,-20,0,90)

                    if turn < 90:

                        servo.angle = turn

                        forward\_30()

                if cx>270 and cx<310:

                    servo.angle = -10

                    if cy>20:

                        forward\_90()

                    else:

                        forward\_60()

PID

import numpy as np

import cv2

import RPi.GPIO as IO

from gpiozero import AngularServo

import time

import board

import neopixel

import math

pixels = neopixel.NeoPixel(board.D10, 2,*brightness*=0.1)

IO.setwarnings(False)

IO.setmode(IO.BCM)

servo = AngularServo(25, *min\_pulse\_width*=0.0006, *max\_pulse\_width*=0.0021)

IO.setup(23,IO.OUT)

IO.setup(24,IO.OUT)

IO.setup(18,IO.OUT)

speed = IO.PWM(18,100)

buttonIn = 17

IO.setup(buttonIn,IO.IN,*pull\_up\_down*=IO.PUD\_UP)

pixels.fill((0, 0, 0))

video\_capture = cv2.VideoCapture(0)

speed.start(0)

def forward\_90():

    IO.output(24,1)

    IO.output(23,0)

    for x in range(50,90,1):

        speed.ChangeDutyCycle(x)

def forward\_30():

    speed.ChangeDutyCycle(50)

    IO.output(24,1)

    IO.output(23,0)

def forward\_50():

    speed.ChangeDutyCycle(50)

    IO.output(24,1)

    IO.output(23,0)

def forward\_60():

    speed.ChangeDutyCycle(50)

    IO.output(24,1)

    IO.output(23,0)

def stop():

    speed.ChangeDutyCycle(0)

    IO.output(24,0)

    IO.output(23,0)

def show():

    pixels[1] = (100,255,0)

    pixels[0] = (100,255,0)

    pixels.show()

def hide():

    pixels[1] = (0,0,0)

    pixels[0] = (0,0,0)

    pixels.show()

def forward(*x*):

    speed.ChangeDutyCycle(x)

    IO.output(24,1)

    IO.output(23,0)

def \_map(*x*, *in\_min*, *in\_max*, *out\_min*, *out\_max*):

    return int((x - in\_min) \* (out\_max - out\_min) / (in\_max - in\_min) + out\_min)

while(True):

    hide()

    if(IO.input(buttonIn)==0):

        time.sleep(2)

        show()

        while(True):

            ret, frame = video\_capture.read()

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

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

            ret,thresh = cv2.threshold(blur,45,255,cv2.THRESH\_BINARY\_INV)

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

            if len(contours) > 0:

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

                M = cv2.moments(c)

                a = M['m00']+1

                cx = int(M['m10']/a)

                cy = int(M['m01']/a)

                current = cx

                if(current<290):

                    error2 = int(math.floor((290 - current)/2.9))

                    if(error2<80):

                        servo.angle = (-10+error2)

                        forward\_50()

                if(current>290):

                    error1 = int(math.floor((290 - current)/3.7))

                    if(error1>-80):

                        servo.angle = (-10+error1)

                        forward\_50()

                if(current==290):

                    servo.angle = -10

                    forward\_90()

import numpy as np

import cv2

import RPi.GPIO as IO

from gpiozero import AngularServo

import time

import board

import neopixel

pixels = neopixel.NeoPixel(board.D10, 3,*brightness*=0.2)

IO.setwarnings(False)

IO.setmode(IO.BCM)

servo = AngularServo(25, *min\_pulse\_width*=0.0006, *max\_pulse\_width*=0.0021)

IO.setup(23,IO.OUT)

IO.setup(24,IO.OUT)

IO.setup(18,IO.OUT)

speed = IO.PWM(18,100)

buttonIn = 17

IO.setup(buttonIn,IO.IN,*pull\_up\_down*=IO.PUD\_UP)

pixels.fill((0, 255, 0))

video\_capture = cv2.VideoCapture(0)

speed.start(0)

def forward\_90():

    IO.output(24,1)

    IO.output(23,0)

    for x in range(30,100,1):

        speed.ChangeDutyCycle(x)

def forward\_30():

    speed.ChangeDutyCycle(40)

    IO.output(24,1)

    IO.output(23,0)

def forward\_50():

    speed.ChangeDutyCycle(50)

    IO.output(24,1)

    IO.output(23,0)

def forward\_60():

    speed.ChangeDutyCycle(70)

    IO.output(24,1)

    IO.output(23,0)

def stop():

    speed.ChangeDutyCycle(0)

    IO.output(24,0)

    IO.output(23,0)

def \_map(*x*, *in\_min*, *in\_max*, *out\_min*, *out\_max*):

    return int((x - in\_min) \* (out\_max - out\_min) / (in\_max - in\_min) + out\_min)

while(True):

    if(IO.input(buttonIn)==0):

        time.sleep(2)

        while(True):

            ret, frame = video\_capture.read()

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

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

            ret,thresh = cv2.threshold(blur,45,255,cv2.THRESH\_BINARY\_INV)

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

            if len(contours) > 0:

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

                M = cv2.moments(c)

                a = M['m00']+1

                cx = int(M['m10']/a)

                cy = int(M['m01']/a)

                if cx>=190 and cx<=270:

                    turn = \_map(cx,270,10,0,60)

                    if turn < 90:

                        servo.angle = turn

                        forward\_50()

                if cx >= 310 and cx<=390:

                    turn = \_map(cx,310,570,-10,-60)

                    if turn > -90:

                        servo.angle = turn

                        forward\_50()

                if cx > 390:

                    turn = \_map(cx,310,620,-10,-90)

                    if turn > -90:

                        servo.angle = turn

                        forward\_30()

                if cx < 190:

                    turn = \_map(cx,270,-20,0,90)

                    if turn < 90:

                        servo.angle = turn

                        forward\_30()

                if cx>270 and cx<310:

                    servo.angle = -10

                    if cy>260:

                        forward\_90()

                    else:

                        forward\_60()