```python
from Raspi_MotorHAT import Raspi_MotorHAT, Raspi_DCMotor

import time
import atexit
from PID import calcOBSPID, calcLinePID
import cv2
import numpy as np
from obstacleAvoid import avoidObs
from LineFollower import LineFollower


### Open a file for data output ###
f = open("demo4_t_pm_lerr_ms_lm_rm_run8.csv",'w')

### Set CONST variables ###
MAX_SPEED = 120
FRAME_RATE = 30
TIMECOUNT = 0


### Obstacle PID Constants ###
obs_sp = 30
dt = 1/FRAME_RATE
old_err = 0
ObsPIDgains = [8.35,0.15,.0000008]


### Line Follow PID Constants ###
line_sp = 160
old_line_err = 0
LinePIDgains = [.85,0,.000000025]


### MotorHAT Initialization ###
mh = Raspi_MotorHAT(addr=0x6f)


### Create a function to turn off motors on program exit ###
def turnOffMotors():
    mh.getMotor(1).run(Raspi_MotorHAT.RELEASE)
    mh.getMotor(2).run(Raspi_MotorHAT.RELEASE)
    mh.getMotor(3).run(Raspi_MotorHAT.RELEASE)
    mh.getMotor(4).run(Raspi_MotorHAT.RELEASE)


### Register turnOffMotors to run at exit ###
atexit.register(turnOffMotors)


### Define variables for left and right motors ###
leftMotor = mh.getMotor(1)
rightMotor = mh.getMotor(3)


### Make sure the motors are off ###
leftMotor.run(Raspi_MotorHAT.RELEASE)
rightMotor.run(Raspi_MotorHAT.RELEASE)


### Default values for contours ###
h = 0
w = 0


### Setup Webcam as video input ###
cap = cv2.VideoCapture(0)
cap.set(3,320.0)                    # Resize the image x-direction
```

```
68  cap.set(4,240.0)                    # Resize the image y-direction
69  cap.set(5,FRAME_RATE)               # Set the webcam frame rate
70
71
72  ### Grab two frames to throw away ###
73  for i in range(0,2):
74      flag, trash = cap.read()
75
76  ### Main Loop, Will be broken into functions ###
77  while True:
78
79      ### Zero the x-centroid and percent_makeup variables every loop ###
80      cx = 0
81      percent_makeup = 0
82
83      ### Set the motor directions ###
84      leftMotor.run(Raspi_MotorHAT.FORWARD)
85      rightMotor.run(Raspi_MotorHAT.FORWARD)
86
87      ### Read one frame from the webcam ###
88      flag, frame = cap.read()
89
90      ### Resize the frame to remove borders ###
91      frame = frame[60:180, 0:320]
92
93      ### Find the size of the x- and y-direction ###
94      xsize = int(frame.shape[1]/2)
95      ysize = int(frame.shape[0]/4)
96
97      ### Find the BGR value of sample pixel ###
98      px = np.array(frame[int(ysize), int(xsize)])
99
100     ### Enter the avoidObs function ###
101     percent_makeup = avoidObs(frame, px, w, h)
102
103     ### Enter the LineFollower function ###
104     cx = LineFollower(frame)
105
106     ### Calculate the PID value for motor speed ###
107     obs_motor_speed = int(calcOBSPID(obs_sp, percent_makeup, old_err, ObsPIDgains, dt))
108
109     ### Print values to the terminal ###
110     print("cx: "+str(cx)+"% makeup: "+str(percent_makeup)+" PID Motor Speed:
        "+str(obs_motor_speed))
111
112     ### Cap the motor speed ###
113     if obs_motor_speed > MAX_SPEED:
114         obs_motor_speed = MAX_SPEED
115
116     ### Turn the motors backwards if less than zero ###
117     if obs_motor_speed < 0:
118         leftMotor.run(Raspi_MotorHAT.BACKWARD)
119         rightMotor.run(Raspi_MotorHAT.BACKWARD)
120         obs_motor_speed = abs(obs_motor_speed)
121     else:
122         leftMotor.run(Raspi_MotorHAT.FORWARD)
123         rightMotor.run(Raspi_MotorHAT.FORWARD)
124
125
126     ### Initialize the differential between motors ###
127     line_motor_diff = 0
128
129     ### If the centroid is not NaN or Inf ###
130     if not np.isinf(cx) and not np.isnan(cx):
131
132         ### Calculate the PID value for motor speed differential ###
133         line_motor_diff = int(calcLinePID(line_sp, cx, old_line_err, LinePIDgains, dt))
```

```
134
135             ### Print values to the terminal ###
136             print("cx:"+str(cx)+" PID:"+str(line_motor_diff)+"\n")
137
138             ### Set the left and right motor speed ###
139             leftMotor.setSpeed(obs_motor_speed-line_motor_diff)
140             rightMotor.setSpeed(obs_motor_speed+line_motor_diff)
141
142         else:
143             ### Set the left and right motor speed ###
144             leftMotor.setSpeed(0)
145             rightMotor.setSpeed(0)
146
147         ### Calculate the running error ###
148         old_err = old_err + (obs_sp - percent_makeup)
149         old_line_err = old_line_err + (line_sp - cx)
150
151         ### Write data to file ###
152         f.write(str(TIMECOUNT)+","+str(obs_sp -
            percent_makeup)+","+str(line_sp-cx)+","+str(obs_motor_speed)+","+str(obs_motor_speed-
            line_motor_diff)+","+str(obs_motor_speed+line_motor_diff)+"\n")
153
154         ### Update TIMECOUNT ###
155         TIMECOUNT = TIMECOUNT + dt
156
157         ### Sleep for one frame ###
158         time.sleep(dt)
159
```