Codes For Line Following Using Camera:

#Basic logic used for line following using camera

#Codes:

#importing required libraries

import cv2

import numpy as np

import RPi.GPIO as g

g.setmode(g.BOARD)#define the numbering system to be used for the pin(This way is the simple way)

#set up pins for giving apropriate outputs to move the bot

#right side's motor

gpio.setup(31, gpio.OUT)#motor1 pwm

gpio.setup(5, gpio.OUT)#motor1 dir

#left side's motor

gpio.setup(33, gpio.OUT)#motor2pwm

gpio.setup(11, gpio.OUT)#motor2dir

#make PWM objects to use PWM methods(object-oriented programming: objects are variables of class type used to access the functions of the class)

motor1 = gpio.PWM(31, 100)#for right-side's motor

motor2 = gpio.PWM(33, 100)#for left-side's motor

#initiate the objects with duty-cycle of 100

motor1.start(100)

motor2.start(100)

#for direction pins; last I checked; if 0 is given as output, rotation is counter-clockwise and if 1 is given as output, rotation is clockwise

#capture video from camera (index : -1=>any random cam is selected; 0 => system's cam is selected; 1,2,3,...=>specify the external cams connected)

cap=cv2.VideoCapture(1)

try:

while True:

\_,frame=cap.read()#keep taking the video frome by frame

blur=cv2.GaussianBlur(frame,(15,15),0) # blur is applied to remove noice from the frames

gray=cv2.cvtColor(blur,cv2.COLOR\_BGR2GRAY) # convert the blured image to grayscale to simplify the image

#Thresholding the feed so that all the pixel values are either 100 or 0. Simplifies the image further(Extreamly, infact)

ret,otsu=cv2.threshold(gray,0,100,cv2.THRESH\_BINARY\_INV+cv2.THRESH\_OTSU)

#finding the contours in the thresholded feed

contours,\_=cv2.findContours(otsu.copy(),1,cv2.CHAIN\_APPROX\_NONE)

if len(contours)>0:#if find contours

#taking maximum region contour which will be our line to be followed

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

M=cv2.moments(c) #Taking moments of the maximum contour in the feed

#How moments work:

#The moments actually are the measure of spatial distribution of points

#if points represent a value; zeroth moment is the total; 1 st moment devided by zeroth moment(i.e. total) gives the centre for the data;

#center coordinates

cx=int(M['m10']/M['m00'])#centre co-ordinates for x

cy=int(M['m01']/M['m00'])#centre co-ordinates for y

#Keep the centre of contours at centre of the screen

if cx>155: #value for this part (let) a= centre of the window in x + 10

#Turn to the right if centre of the contour gets out of the range

#Change duty cycle of motors to control the speed or rotation

motor2.ChangeDutyCycle(50)

motor1.ChangeDutyCycle(50)

#to turn the bot to right; motor1 moves in clockwise direction and motor2 in counter clock direction

gpio.output(5, 1)

gpio.output(11, 0)

elif cx<145:#value for this part (let) b= centre of window in x - 10

#Turn to left if contour gets out of range

motor2.ChangeDutyCycle(50)

motor1.ChangeDutyCycle(50)

#to turn the bot to left; motor2 moves in clockwise direction and motor1 in counter clock direction

gpio.output(5, 0)

gpio.output(11, 1)

elif 140<cx<160: #Here the values are a>cx>b

#Keep moving straight if the centre is in range

motor2.ChangeDutyCycle(50)

motor1.ChangeDutyCycle(50)

#to turn the bot to right; motor1 and motor2 move in counter clockwise direction

gpio.output(5, 0)

gpio.output(11, 0)

if cv2.waitKey(1) & 0xFF == ord('q'):#stop capturing if 'q' is pressed

break

except KeyboardInterrupt :

cap.release()#release the camera

g.cleanup()#clear all the outputs given to the board pins