CODE

import cv2

import numpy as np

import os

def Gamma(img, gamma): #gamma correction function

#convert to hls image and operate on intensity function

#Inew=(I/255)^(gamma)\*255

hls=cv2.cvtColor(img,cv2.COLOR\_BGR2HLS)

hls[:,:,1]=pow(hls[:,:,1]/255,gamma)\*255

img = cv2.cvtColor(hls, cv2.COLOR\_HLS2BGR)

print(gamma)

return img

def getContours(img,imgBlankc):

contours,hierarchy = cv2.findContours(img,cv2.RETR\_EXTERNAL,cv2.CHAIN\_APPROX\_NONE)

#find contours

for cnt in contours:

area=cv2.contourArea(cnt)

if area>1000: #remove noise

approx=cv2.approxPolyDP(cnt,0.09\*cv2.arcLength(cnt,True),True) #extract points from contour

n=approx.ravel()

i=0

for pt in n:

if(i%2==1): #if no points of contour in lower part of image

#it could be a bridge or the sky or part of building

y=pt

if y>=height\*17//20:

cv2.fillPoly(imgBlankc, pts=[cnt], color=(255, 255, 255)) #fill contour

break

i=i+1

def addToblank(h,w,imgBlanksum,ver): #take a certain position and image

#and pick all points with simillar intensity to it

#and add to imgBlanksum

#ver(versions)-explained later

val = hls[h, w, 1]

#vas = hls[h, w, 2]

imgBlank = np.zeros\_like(imgGrey)

num=10

if ver!=0:

num=8

imgBlank[:, :] = 255

chk = (hls[:, :, 1] > val + num)

imgBlank[chk] = 0

chk1 = (hls[:, :, 1] < val - num)

imgBlank[chk1] = 0

mask = np.zeros\_like(imgGrey) #mask for ver=1 or ver=2

if ver==1:

cv2.rectangle(mask, (0, height//2), (width//2, height), 255, -1)

elif ver == 2:

cv2.rectangle(mask, (width//2, height//2), (width, height), 255, -1)

if ver > 0:

imgBlank=cv2.bitwise\_and(imgBlank,mask)#mask image

imgBlanksum = cv2.bitwise\_or(imgBlanksum, imgBlank)#add new pixels to imgBlanksum

return imgBlanksum

mainFolder='Resources'

imgfolder=os.listdir('Resources')

i=0

for imgn in imgfolder:#iterate through all images

path = mainFolder +'/' + imgn

images=[]

#myList=os.listdir('path')

img=cv2.imread(path)

img=Gamma(img,1/3.5) #added as it slightly improved output

height=img.shape[0]

width=img.shape[1]

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

hls=cv2.cvtColor(img,cv2.COLOR\_BGR2HLS)

imgBlanksum = np.zeros\_like(imgGrey)

imgBlanksum=addToblank(int(height\*0.75),int(width\*0.5),imgBlanksum,0)#pick points that are on the road

imgBlanksum=addToblank(int(height\*0.8),int(width\*0.15),imgBlanksum,1)

imgBlanksum=addToblank(int(height\*0.6),int(width\*0.5),imgBlanksum,0)

imgBlanksum=addToblank(int(height\*0.8),int(width\*0.85),imgBlanksum,2)

"""for j in range(height\*18//19,height-1):#

imgBlanksum[j,:,]=0"""

imgBlankc = np.zeros\_like(imgGrey)

kernel=np.ones((4,4),np.uint8)

imgBlanksum=cv2.erode(imgBlanksum,kernel,iterations=1)#erode image to divide the road if its connected thru few pixels

#slightly improved outputs

getContours(imgBlanksum,imgBlankc)#get final output and store in imgBlankc

#img=cv2.cvtColor(img,cv2.COLOR\_HSV2BGR)

cv2.imshow("image"+str(i), img)

cv2.imshow("imageF"+str(i), imgBlankc)

#cv2.imshow("imageB4F"+str(i), imgBlanksum)

i=i+1

cv2.imwrite("/Users/soumojitbhattacharya/PycharmProjects/road\_agv\_task/output3/"+str(imgn),imgBlankc)#store in folder

#print(width\*height)

cv2.waitKey(0)

I observed that some parts of the image are always on the road when considering dashboard views. So i picked these points (3\*h/4,w/2) and (3\*h/5,w/2) and picked all points with a similar intensity as those points. These points correspond to ver=0 in the addToBlank function

Another problem i faced was that the corners were not getting included

So i added two more points on the two corners and did the same thing on them only including the points in their respective corners using a mask.

ver=1 is for the left bottom corner

ver=2 is for the right bottom corner

Then add all these points to a image and take contours for that image and fill it

The contour part is to get a smoother final image