CODE

import cv2

import numpy as np

import os

from skimage.metrics import peak\_signal\_noise\_ratio as psnr

from skimage.metrics import structural\_similarity as ssim

w=0.95

mainFolder='Resources'

imgfolder=os.listdir('Resources')

a=0

pavg=savg=0

for imgn in imgfolder:

path = mainFolder +'/' + imgn

pathgt = "GroundTruth/"+imgn.replace("hazy","GT") #only for calculating psnir ssim values

images=[]

#print(path)

img=cv2.imread(path)

gt=cv2.imread(pathgt)

img1=img

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

b,g,r=cv2.split(img)

dark=np.minimum(r,g) #find dark channel

dark= np.minimum(dark, b)#-----------------

A=np.amax(dark) #atmospheric light= max of dark channel(used while changing sand v values in hsv)

kernel=np.ones((5,5),np.uint8)

dark=cv2.erode(dark,kernel) #use erode so output does not look pixelated

hsv=cv2.cvtColor(img,cv2.COLOR\_BGR2HSV)

hsv[:, :, 1] = hsv[:, :, 1] \* (1 + dark[:, :] / A \* 0.8) + 10 # increase saturation by more if pixel more hazy

hsv[:, :, 2] = hsv[:, :, 2] \* (1 - dark[:, :] / A \* 0.25) - 20 # increase value by more if pixel smore hazy

filter = hsv[:, :, 1] > 180 #decrease saturation if its too high

hsv[filter, 1] = 90+hsv[filter, 1]//2

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

img2=cv2.hconcat((img,img1))

a=a+1

cv2.imshow("img"+str(a),img2)

cv2.imwrite("/Users/soumojitbhattacharya/PycharmProjects/dehaze/output/" + str(imgn), img)

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

psnr\_value1 = psnr(gt, img) #calculate psnir and ssim

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

ssim\_val\_in = ssim(gray, img\_gray, channel\_axis=None)

print(psnr\_value1,ssim\_val\_in)

print(a,"----",imgn, "---------------------------------------------")

pavg=pavg+psnr\_value1

savg=savg+ssim\_val\_in

print(pavg/16,savg/16)

cv2.waitKey(0)

I took the dark channel values of each pixel

Taking haziness to be proportional to dark channel value i then changed the saturation and brightness based on the dark channel decreasing the saturation levels of highly saturated pixels

I had a different approach before:- using the hue values to segment the image and dehaze each part separately but the processing time was long and outputs weird so had to use a different approach

Slightly updated code on 3rd july(line 32 and 33) changed the // sign to \* sign and + to - :)

Same change for the video dehazing code