from tkinter import messagebox

from tkinter import \*

from tkinter import simpledialog

import tkinter

from tkinter import filedialog

import numpy as np

from tkinter.filedialog import askopenfilename

import numpy as np

from CannyEdgeDetector import \*

import skimage

import matplotlib.image as mpimg

import os

import scipy.misc as sm

import cv2

import matplotlib.pyplot as plt

main = tkinter.Tk()

main.title("Density Based Smart Traffic Control System")

main.geometry("1300x1200")

global filename

global refrence\_pixels

global sample\_pixels

def rgb2gray(rgb):

r, g, b = rgb[:,:,0], rgb[:,:,1], rgb[:,:,2]

gray = 0.2989 \* r + 0.5870 \* g + 0.1140 \* b

return gray

def uploadTrafficImage():

global filename

filename = filedialog.askopenfilename(initialdir="images")

pathlabel.config(text=filename)

def visualize(imgs, format=None, gray=False):

j = 0

plt.figure(figsize=(20, 40))

for i, img in enumerate(imgs):

if img.shape[0] == 3:

img = img.transpose(1,2,0)

plt\_idx = i+1

plt.subplot(2, 2, plt\_idx)

if j == 0:

plt.title('Sample Image')

plt.imshow(img, format)

j = j + 1

elif j > 0:

plt.title('Reference Image')

plt.imshow(img, format)

plt.show()

def applyCanny():

imgs = []

img = mpimg.imread(filename)

img = rgb2gray(img)

imgs.append(img)

edge = CannyEdgeDetector(imgs, sigma=1.4, kernel\_size=5, lowthreshold=0.09, highthreshold=0.20, weak\_pixel=100)

imgs = edge.detect()

for i, img in enumerate(imgs):

if img.shape[0] == 3:

img = img.transpose(1,2,0)

cv2.imwrite("gray/test.png",img)

temp = []

img1 = mpimg.imread('gray/test.png')

img2 = mpimg.imread('gray/refrence.png')

temp.append(img1)

temp.append(img2)

visualize(temp)

def pixelcount():

global refrence\_pixels

global sample\_pixels

img = cv2.imread('gray/test.png', cv2.IMREAD\_GRAYSCALE)

sample\_pixels = np.sum(img == 255)

img = cv2.imread('gray/refrence.png', cv2.IMREAD\_GRAYSCALE)

refrence\_pixels = np.sum(img == 255)

messagebox.showinfo("Pixel Counts", "Total Refrence White Pixels Count : "+str(sample\_pixels)+"\nTotal Sample White Pixels Count : "+str(refrence\_pixels))

def timeAllocation():

avg = (sample\_pixels/refrence\_pixels) \*100

if avg >= 90:

messagebox.showinfo("Green Signal Allocation Time","Traffic is very high allocation green signal time : 60 secs")

if avg > 85 and avg < 90:

messagebox.showinfo("Green Signal Allocation Time","Traffic is high allocation green signal time : 50 secs")

if avg > 75 and avg <= 85:

messagebox.showinfo("Green Signal Allocation Time","Traffic is moderate green signal time : 40 secs")

if avg > 50 and avg <= 75:

messagebox.showinfo("Green Signal Allocation Time","Traffic is low allocation green signal time : 30 secs")

if avg <= 50:

messagebox.showinfo("Green Signal Allocation Time","Traffic is very low allocation green signal time : 20 secs")

def exit():

main.destroy()

font = ('times', 16, 'bold')

title = Label(main, text='Density Based Smart Traffic Control System Using Canny Edge Detection Algorithm for Congregating Traffic Information',anchor=W, justify=CENTER)

title.config(bg='yellow4', fg='white')

title.config(font=font)

title.config(height=5, width=120)

title.place(x=0,y=30)

font1 = ('times', 14, 'bold')

upload = Button(main, text="Upload Traffic Image", command=uploadTrafficImage)

upload.place(x=50,y=100)

upload.config(font=font1)

pathlabel = Label(main)

pathlabel.config(bg='yellow4', fg='white')

pathlabel.config(font=font1)

pathlabel.place(x=50,y=150)

process = Button(main, text="Image Preprocessing Using Canny Edge Detection & CNN", command=applyCanny)

process.place(x=50,y=200)

process.config(font=font1)

count = Button(main, text="White Pixel Count", command=pixelcount)

count.place(x=50,y=350)

count.config(font=font1)

count = Button(main, text="Calculate Green Signal Time Allocation", command=timeAllocation)

count.place(x=50,y=400)

count.config(font=font1)

exitButton = Button(main, text="Exit", command=exit)

exitButton.place(x=50,y=450)

exitButton.config(font=font1)

main.config(bg='magenta3')

main.mainloop()







