import PIL.Image as Image  
import skimage.io as io  
import numpy as np  
import time  
from gf import guided\_filter  
from numba import jit  
import matplotlib.pyplot as plt  
  
  
class HazeRemoval(object):  
 def \_\_init\_\_(self, omega=0.95, t0=0.1, radius=7, r=20, eps=0.001):  
 pass  
  
 def open\_image(self, img\_path):  
 img = Image.open(img\_path)  
 self.src = np.array(img).astype(np.double) / 255.  
 # self.gray = np.array(img.convert('L'))  
 self.rows, self.cols, \_ = self.src.shape  
 self.dark = np.zeros((self.rows, self.cols), dtype=np.double)  
 self.Alight = np.zeros((3), dtype=np.double)  
 self.tran = np.zeros((self.rows, self.cols), dtype=np.double)  
 self.dst = np.zeros\_like(self.src, dtype=np.double)  
  
 @jit  
 def get\_dark\_channel(self, radius=7):  
 print("Starting to compute dark channel prior...")  
 start = time.time()  
 tmp = self.src.min(axis=2)  
 for i in range(self.rows):  
 for j in range(self.cols):  
 rmin = max(0, i - radius)  
 rmax = min(i + radius, self.rows - 1)  
 cmin = max(0, j - radius)  
 cmax = min(j + radius, self.cols - 1)  
 self.dark[i, j] = tmp[rmin:rmax + 1, cmin:cmax + 1].min()  
 print("time:", time.time() - start)  
  
 def get\_air\_light(self):  
 print("Starting to compute air light prior...")  
 start = time.time()  
 flat = self.dark.flatten()  
 flat.sort()  
 num = int(self.rows \* self.cols \* 0.001)  
 threshold = flat[-num]  
 tmp = self.src[self.dark >= threshold]  
 tmp.sort(axis=0)  
 self.Alight = tmp[-num:, :].mean(axis=0)  
 # print(self.Alight)  
 print("time:", time.time() - start)  
  
 @jit  
 def get\_transmission(self, radius=7, omega=0.95):  
 print("Starting to compute transmission...")  
 start = time.time()  
 for i in range(self.rows):  
 for j in range(self.cols):  
 rmin = max(0, i - radius)  
 rmax = min(i + radius, self.rows - 1)  
 cmin = max(0, j - radius)  
 cmax = min(j + radius, self.cols - 1)  
 pixel = (self.src[rmin:rmax + 1, cmin:cmax + 1] / self.Alight).min()  
 self.tran[i, j] = 1. - omega \* pixel  
 print("time:", time.time() - start)  
  
 def guided\_filter(self, r=60, eps=0.001):  
 print("Starting to compute guided filter trainsmission...")  
 start = time.time()  
 self.gtran = guided\_filter(self.src, self.tran, r, eps)  
 print("time:", time.time() - start)  
  
 def recover(self, t0=0.1):  
 print("Starting recovering...")  
 start = time.time()  
 self.gtran[self.gtran < t0] = t0  
 t = self.gtran.reshape(\*self.gtran.shape, 1).repeat(3, axis=2)  
 # import ipdb; ipdb.set\_trace()  
 self.dst = (self.src.astype(np.double) - self.Alight) / t + self.Alight  
 self.dst \*= 255  
 self.dst[self.dst > 255] = 255  
 self.dst[self.dst < 0] = 0  
 self.dst = self.dst.astype(np.uint8)  
 print("time:", time.time() - start)  
  
 def show(self):  
 import cv2  
 cv2.imwrite("img/src.jpg", (self.src \* 255).astype(np.uint8)[:, :, (2, 1, 0)])  
 cv2.imwrite("img/dark.jpg", (self.dark \* 255).astype(np.uint8))  
 cv2.imwrite("img/tran.jpg", (self.tran \* 255).astype(np.uint8))  
 cv2.imwrite("img/gtran.jpg", (self.gtran \* 255).astype(np.uint8))  
 cv2.imwrite("img/dst.jpg", self.dst[:, :, (2, 1, 0)])  
  
 io.imsave("result.jpg", self.dst)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 import sys  
  
 hr = HazeRemoval()  
 hr.open\_image(sys.argv[1])  
 hr.get\_dark\_channel()  
 hr.get\_air\_light()  
 hr.get\_transmission()  
 hr.guided\_filter()  
 hr.recover()  
 hr.show()