import cv2

import numpy as np

import argparse

from sklearn.cluster import KMeans

def redimage(rgb\_image):

rgb\_image = cv2.resize(rgb\_image, (600, 600))

red\_image = rgb\_image.copy()

red\_image[:, :, 0] = 0

red\_image[:, :, 1] = 0

return red\_image

def grey\_img(green\_img):

img = cv2.cvtColor(green\_img, cv2.COLOR\_BGR2GRAY)

new\_image = cv2.convertScaleAbs(img, alpha=2.5, beta=10)

return new\_image

class disk:

def \_\_init\_\_(self):

pass

def morphing(self, grey):

kernal = np.ones((6, 6), np.uint8)

mask = cv2.erode(grey, kernal, iterations=5)

mask = cv2.dilate(mask, kernal, iterations=5)

mask = cv2.GaussianBlur(mask, (11, 11), 0)

return mask

def kemins(self, img, grey\_imgg):

def veins(img):

kernel = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE, (3, 2))

erosion = cv2.morphologyEx(img, cv2.MORPH\_OPEN, kernel, iterations=1)

erosion = cv2.erode(erosion, kernel, iterations=1)

veins(img)

img2 = grey\_imgg # .cvtColor(img, cv2.COLOR\_BGR2GRAY)

x = img2.reshape(-1, 1)

# croped image

minVal, maxVal, minLoc, maxLoc = cv2.minMaxLoc(img2)

crop\_mask = np.zeros(img2.shape[:2], dtype="uint8")

cv2.circle(crop\_mask, maxLoc, 120, (255, 255, 0), -1)

masked\_crooop = cv2.bitwise\_and(img2, img2, mask=crop\_mask)

# k-meins algorithum

kmeans = KMeans(n\_clusters=5, n\_init=10)

kmeans.fit(x)

segim = kmeans.cluster\_centers\_[kmeans.labels\_]

segim = segim.reshape(masked\_crooop.shape)

segim = segim / 255

segim\_2d = (segim \* 255).astype(np.uint8)

segim\_2d\_reshaped = segim\_2d.reshape((600, 600, 1))

blur = cv2.GaussianBlur(grey, (11, 11), 0)

mask = cv2.equalizeHist(blur)

edge = cv2.Canny(mask, 10, 1)

(minVal, maxVal, minLoc, maxLoc) = cv2.minMaxLoc(segim\_2d\_reshaped)

cv2.circle(segim\_2d\_reshaped, maxLoc, 3, (0, 255, 0), 2)

masked\_img = cv2.inRange(segim\_2d\_reshaped, maxVal, 255)

masked\_crooop = cv2.bitwise\_and(masked\_img, masked\_img, mask=crop\_mask)

return masked\_crooop

def centroid\_calc(self, morph\_img):

# centroid calculation

imageMoments = cv2.moments(morph\_img)

cx = int(imageMoments['m10'] / imageMoments['m00'])

cy = int(imageMoments['m01'] / imageMoments['m00'])

bgrImage = cv2.cvtColor(morph\_img, cv2.COLOR\_GRAY2BGR)

return cx, cy

def vertical\_diamter\_cal(self, morph\_img, cx, cy):

x1 = cx

y1 = cy

x2 = cx

y2 = cx

while True:

temp\_image\_val = morph\_img[y1, x1]

if temp\_image\_val < 30:

break

y1 += 1

while True:

temp\_image\_val = morph\_img[y2, x2]

if temp\_image\_val < 30:

break

y2 -= 1

diamter = y1 - y2

radius = int(diamter / 2)

morph\_img = cv2.cvtColor(morph\_img, cv2.COLOR\_GRAY2BGR)

cv2.circle(morph\_img, (cx, cy), radius, (255, 255, 0), 1)

return radius

class cup:

def \_\_init\_\_(self):

pass

def green(self,rgb\_image):

rgb\_image = cv2.resize(rgb\_image, (600, 600))

green\_image = rgb\_image.copy()

green\_image[:, :, 2] = 0

green\_image[:, :, 0] = 0

return green\_image

def region\_calc(self,red\_img,grey,cx,cy,disk\_radius):

crop\_mask = np.zeros(grey.shape[:2], dtype="uint8")

cv2.circle(crop\_mask, (cx,cy), disk\_radius, (255, 255, 0), -1)

masked\_crooop = cv2.bitwise\_and(red\_img, red\_img, mask=crop\_mask)

grey\_masked\_crooop = cv2.bitwise\_and(grey, grey, mask=crop\_mask)

return masked\_crooop,grey\_masked\_crooop

def cup\_kemin(self,img,grey\_img):

def veins(img):

kernel = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE, (3, 2))

erosion = cv2.morphologyEx(img, cv2.MORPH\_OPEN, kernel, iterations=1)

erosion = cv2.erode(erosion, kernel, iterations=1)

veins(img)

img2 = grey\_img # .cvtColor(img, cv2.COLOR\_BGR2GRAY)

x = img2.reshape(-1, 1)

# k-meins algorithum

kmeans = KMeans(n\_clusters=6, n\_init=10)

kmeans.fit(x)

segim = kmeans.cluster\_centers\_[kmeans.labels\_]

segim = segim.reshape(img2.shape)

segim = segim / 255

segim\_2d = (segim \* 255).astype(np.uint8)

segim\_2d\_reshaped = segim\_2d.reshape((600, 600, 1))

blur = cv2.GaussianBlur(grey, (11, 11), 0)

mask = cv2.equalizeHist(blur)

(minVal, maxVal, minLoc, maxLoc) = cv2.minMaxLoc(segim\_2d\_reshaped)

cv2.circle(segim\_2d\_reshaped, maxLoc, 3, (0, 255, 0), 2)

crop\_mask = np.zeros(img2.shape[:2], dtype="uint8")

masked\_img = cv2.inRange(segim\_2d\_reshaped, maxVal, 255)

cv2.circle(crop\_mask, maxLoc, 120, (255, 255, 0), -1)

masked\_crooop = cv2.bitwise\_and(masked\_img, masked\_img, mask=crop\_mask)

return masked\_crooop

def cup\_morphing(self,cup\_keimen\_image):

kernal = np.ones((6, 6), np.uint8)

mask = cv2.erode(cup\_keimen\_image, kernal, iterations=1)

mask = cv2.dilate(mask, kernal, iterations=3)

mask = cv2.GaussianBlur(mask, (11, 11), 0)

return mask

def cup\_centroid\_calc(self,cup\_morph\_img):

imageMoments = cv2.moments(cup\_morph\_img)

cx = int(imageMoments['m10'] / imageMoments['m00'])

cy = int(imageMoments['m01'] / imageMoments['m00'])

bgrImage = cv2.cvtColor(cup\_morph\_img, cv2.COLOR\_GRAY2BGR)

return cx, cy

def cup\_vertical\_diameter(self, cup\_morph\_img, cx, cy):

x1 = cx

y1 = cy

x2 = cx

y2 = cx

while True:

temp\_image\_val = cup\_morph\_img[y1, x1]

if temp\_image\_val < 20:

break

y1 += 1

while True:

temp\_image\_val = cup\_morph\_img[y2, x2]

if temp\_image\_val < 20:

break

y2 -= 1

diamter = y1 - y2

radius = int(diamter / 2)

morph\_img = cv2.cvtColor(cup\_morph\_img, cv2.COLOR\_GRAY2BGR)

cv2.circle(morph\_img, (cx, cy), radius, (255, 255, 0), 1)

return radius

rgb\_image = cv2.imread("C:\\Users\\maqwi\\Desktop\\Intership\\Test\\Images\\glaucoma\\5.png")

rgb\_image = cv2.resize(rgb\_image, (600, 600))

red\_img = redimage(rgb\_image)

grey = grey\_img(red\_img)

# disk calculation

d = disk()

kemin\_img = d.kemins(red\_img, grey)

morph\_img = d.morphing(kemin\_img)

cx, cy = d.centroid\_calc(morph\_img)

disk\_radius = d.vertical\_diamter\_cal(morph\_img, cx, cy)

# cup calculation

c = cup()

green\_img=c.green(rgb\_image)

region\_img,grey\_cup\_kemin\_img=c.region\_calc(green\_img,grey,cx,cy,disk\_radius)

cup\_kemin\_img=c.cup\_kemin(region\_img,grey\_cup\_kemin\_img)

cup\_morph\_img=c.cup\_morphing(cup\_kemin\_img)

c2x,c2y=c.cup\_centroid\_calc(cup\_morph\_img)

cup\_radius=c.cup\_vertical\_diameter(cup\_morph\_img,c2x,c2y)

cv2.circle(rgb\_image, (cx, cy), disk\_radius, (255, 255, 0), 1)

cv2.circle(rgb\_image, (c2x, c2y), cup\_radius, (0, 255, 5), 1)

cv2.imshow("original img",rgb\_image)

cv2.imshow("kemins img",kemin\_img)

cv2.imshow("morphimg",morph\_img)

cv2.imshow("rgb image",rgb\_image)

cv2.imshow("cup morphing",cup\_morph\_img)

print("DISK VERTICAL DIAMETER : ",disk\_radius\*2)

print("CUP VERTICAL DIAMETER : ",cup\_radius\*2)

print("Disk/CUP RATIO :",((cup\_radius\*2)/(disk\_radius\*2)))

cv2.waitKey(0)