The commented out part is my attempt on stitching without using the built in function

I was able to write the code till the feature matching and homography section but then got confused and ran out of time

import cv2

import numpy as np

img2=cv2.imread("Resources/1B.png")

img1=cv2.imread("Resources/1A.png")

h1=img1.shape[0]

h2=img2.shape[0]

w1=img1.shape[1]

w2=img2.shape[1]

stitcher = cv2.Stitcher.create()

(dummy, output) = stitcher.stitch([img1,img2])

cv2.imshow('final result', output)

"""mpts=[]

dist=10

def distance(st,m,c):

x=lx[st]

y=ly[st]

d=pow((y+m\*x+c),2)/(1+m\*m)

return d

def ransac(n):

rc = 0

bcount=0

for i in range(1,n):

for j in range(0,i):

y1=ly[i]

x1=lx[i]

y2=ly[j]

x2=lx[j]

if x1==x2:

continue

m=(y1-y2)/(x2-x1)

c=(y1\*x2-y2\*x1)/(x1-x2)

count=0

rc=rc+1

for k in range(0,n):

if distance(k,m,c)<dist:

count=count+1

#print("count=",count,"p1=",x1,y1,"p2=",x2,y2)

if count>bcount:

bcount=count

mpts.clear()

for k in range(0, n):

if distance(k, m, c) < dist:

count = count + 1

mpts.append(k)

#print(k)"""

"""MIN\_MATCH\_COUNT = 4

## Create ORB object and BF object(using HAMMING)

orb = cv2.ORB\_create()

gray2 = cv2.cvtColor(img2, cv2.COLOR\_BGR2GRAY)

gray1 = cv2.cvtColor(img1, cv2.COLOR\_BGR2GRAY)

## Find the keypoints and descriptors with ORB

kpts1, descs1 = orb.detectAndCompute(gray1,None)

kpts2, descs2 = orb.detectAndCompute(gray2,None)

## match descriptors and sort them in the order of their distance

bf = cv2.BFMatcher(cv2.NORM\_HAMMING, crossCheck=True)

matches = bf.match(descs1, descs2)

dmatches = sorted(matches, key = lambda x:x.distance)

## extract the matched keypoints

src\_pts = np.float32([kpts1[m.queryIdx].pt for m in dmatches]).reshape(-1,1,2)

dst\_pts = np.float32([kpts2[m.trainIdx].pt for m in dmatches]).reshape(-1,1,2)

lx=[]

ly=[]

for i in range(0,len(src\_pts)):

lx.append(src\_pts[i][0][0]-dst\_pts[i][0][0])

ly.append(src\_pts[i][0][1]-dst\_pts[i][0][1])

## find homography matrix and do perspective transform

M, mask = cv2.findHomography(src\_pts, dst\_pts, cv2.RANSAC,5.0)

h,w = img1.shape[:2]

pts = np.float32([ [0,0],[0,h-1],[w-1,h-1],[w-1,0] ]).reshape(-1,1,2)

dst = cv2.perspectiveTransform(pts,M)

print(M)

#i=cv2.warpPerspective(dst,(500,600))

## draw found regions

#img2 = cv2.polylines(img2, [np.int32(dst)], True, (0,0,255), 1, cv2.LINE\_AA)

print(dst)

#img2=cv2.bitwise\_and(img2,img2,mask=)

"""

#for j in range (0,img2.shape[0]):

# for i in range(0,img2.shape[1]):

# if (img2[j, i, 0] == 0):

# break

# img2[j,i,0]=0

# img2[j, i, 1] = 0

# img2[j, i, 2] = 0

"""cv2.imshow("found", img2)

# draw match lines

res = cv2.drawMatches(img1, kpts1, img2, kpts2, dmatches[:20],None,flags=2)

print(res.shape)

print(h1,w1,h2,w2)

print(h,w)

cv2.imshow("orb\_match", res)"""

cv2.destroyAllWindows()

cv2.waitKey(0)