CODE

import cv2

import numpy as np

for a in range(1,6):

img2=cv2.imread("Resources/"+str(a)+"B.png")

img1=cv2.imread("Resources/"+str(a)+"A.png")

h1=img1.shape[0]

h2=img2.shape[0]

w1=img1.shape[1]

w2=img2.shape[1]

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

orb = cv2.ORB\_create()

## Find the keypoints and descriptors with ORB

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

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

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 matches]).reshape(-1,1,2)

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

lx=[]

ly=[]

## find homography matrix and do perspective transform

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

pts = np.float32([ [0,0],[0,h1-1],[w1-1,h1-1],[w1-1,0] ]).reshape(-1,1,2)

#dst = cv2.perspectiveTransform(pts,M)

dst=cv2.warpPerspective(img1,M,(w1+w2,h1+h2))

#cv2.imshow("foundq", dst)

#dst[0:w2,0:h2,:]=img2[0:w2,0:h2,:]

print(w2,img2.shape[:])

dst[0:h2,0:w2,:]=img2[0:h2,0:w2,:]

for i in range(0,w1+w1):

if dst[:,w1+w2-1-i,:].any()!=0:

break

dstf=dst[:,0:w1+w2-1-i,:]

for i in range(0,h1+h1):

if dst[h1+h2-1-i,:,:].any()!=0:

break

dstf=dstf[0:h1+h2-1-i,:,:]

cv2.imshow("found"+str(a), dstf)

cv2.waitKey(0)