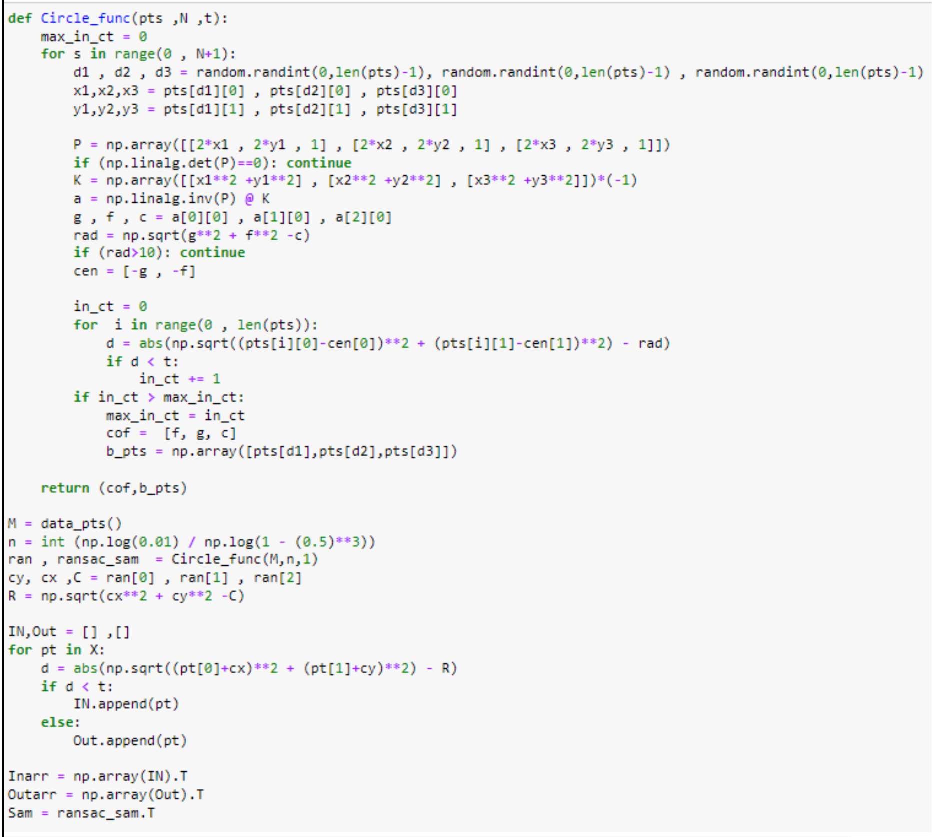
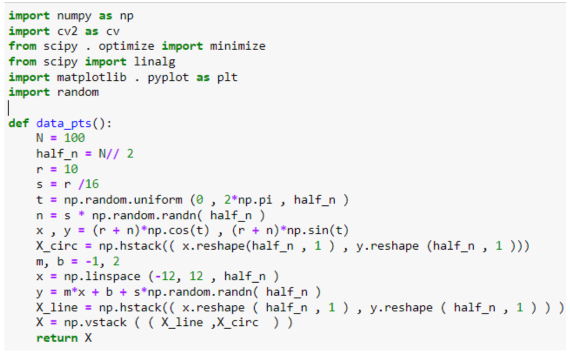
Gajaanan S.
190185D

Question 1

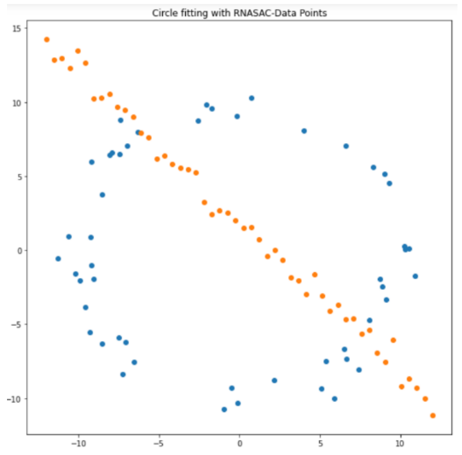
- **Estimate the circle using the RNASAC algorithm**

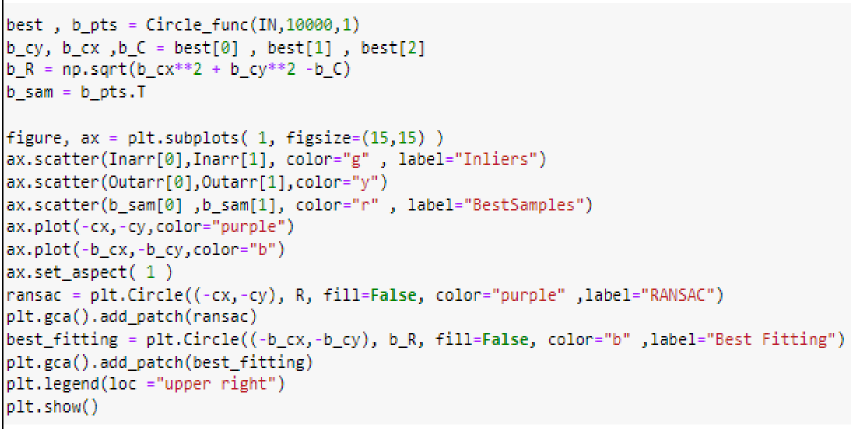


```
def Circle_func(pts ,N ,t):
    max_in_ct = 0
    for s in range(0 , N+1):
        d1 , d2 , d3 = random.randint(0,len(pts)-1), random.randint(0,len(pts)-1) , random.randint(0,len(pts)-1)
        x1,x2,x3 = pts[d1][0] , pts[d2][0] , pts[d3][0]
        y1,y2,y3 = pts[d1][1] , pts[d2][1] , pts[d3][1]

        P = np.array([[2*x1 , 2*y1 , 1] , [2*x2 , 2*y2 , 1] , [2*x3 , 2*y3 , 1]])
        if (np.linalg.det(P)==0): continue
        K = np.array([[x1**2 +y1**2] , [x2**2 +y2**2] , [x3**2 +y3**2]])*(-1)
        a = np.linalg.inv(P) @ K
        g , f , c = a[0][0] , a[1][0] , a[2][0]
        rad = np.sqrt(g**2 + f**2 -c)
        if (rad>10): continue
        cen = [-g , -f]

        in_ct = 0
        for  i in range(0 , len(pts)):
            d = abs(np.sqrt((pts[i][0]-cen[0])**2 + (pts[i][1]-cen[1])**2) - rad)
            if d < t:
                in_ct += 1
        if in_ct > max_in_ct:
            max_in_ct = in_ct
            cof =  [f, g, c]
            b_pts = np.array([pts[d1],pts[d2],pts[d3]])

    return (cof,b_pts)

M = data_pts()
n = int (np.log(0.01) / np.log(1 - (0.5)**3))
ran , ransac_sam  = Circle_func(M,n,1)
cy, cx ,C = ran[0] , ran[1] , ran[2]
R = np.sqrt(cx**2 + cy**2 -C)

IN,Out = [] ,[]
for pt in X:
    d = abs(np.sqrt((pt[0]+cx)**2 + (pt[1]+cy)**2) - R)
    if d < t:
        IN.append(pt)
    else:
        Out.append(pt)

Inarr = np.array(IN).T
Outarr = np.array(Out).T
Sam = ransac_sam.T
```

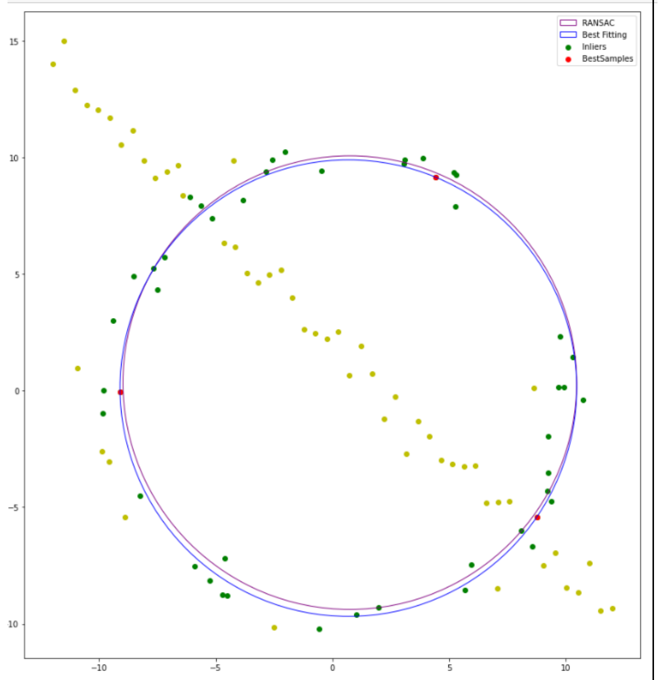


```
import numpy as np
import cv2 as cv
from scipy . optimize import minimize
from scipy import linalg
import matplotlib . pyplot as plt
import random

def data_pts():
    N = 100
    half_n = N// 2
    r = 10
    s = r /16
    t = np.random.uniform (0 , 2*np.pi , half_n )
    n = s * np.random.randn( half_n )
    x , y = (r + n)*np.cos(t) , (r + n)*np.sin(t)
    X_circ = np.hstack(( x.reshape(half_n , 1 ) , y.reshape (half_n , 1 )))
    m, b = -1, 2
    x = np.linspace (-12, 12 , half_n )
    y = m*x + b + s*np.random.randn( half_n )
    X_line = np.hstack(( x.reshape ( half_n , 1 ) , y.reshape ( half_n , 1 ) ) )
    X = np.vstack ( ( X_line ,X_circ  ) )
    return X
```





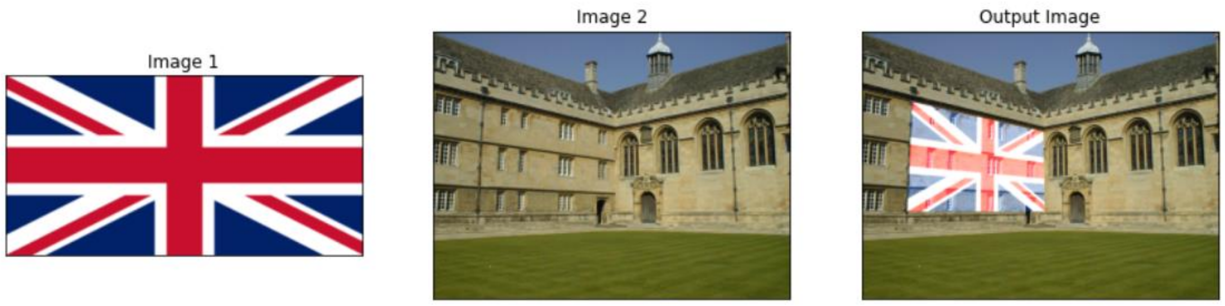
```
best , b_pts = Circle_func(IN,10000,1)
b_cy, b_cx ,b_C = best[0] , best[1] , best[2]
b_R = np.sqrt(b_cx**2 + b_cy**2 -b_C)
b_sam = b_pts.T

figure, ax = plt.subplots( 1, figsize=(15,15) )
ax.scatter(Inarr[0],Inarr[1], color="g" , label="Inliers")
ax.scatter(Outarr[0],Outarr[1],color="y")
ax.scatter(b_sam[0] ,b_sam[1], color="r" , label="BestSamples")
ax.plot(-cx,-cy,color="purple")
ax.plot(-b_cx,-b_cy,color="b")
ax.set_aspect( 1 )
ransac = plt.Circle((-cx,-cy), R, fill=False, color="purple" ,label="RANSAC")
plt.gca().add_patch(ransac)
best_fitting = plt.Circle((-b_cx,-b_cy), b_R, fill=False, color="b" ,label="Best Fitting")
plt.gca().add_patch(best_fitting)
plt.legend(loc ="upper right")
plt.show()
```



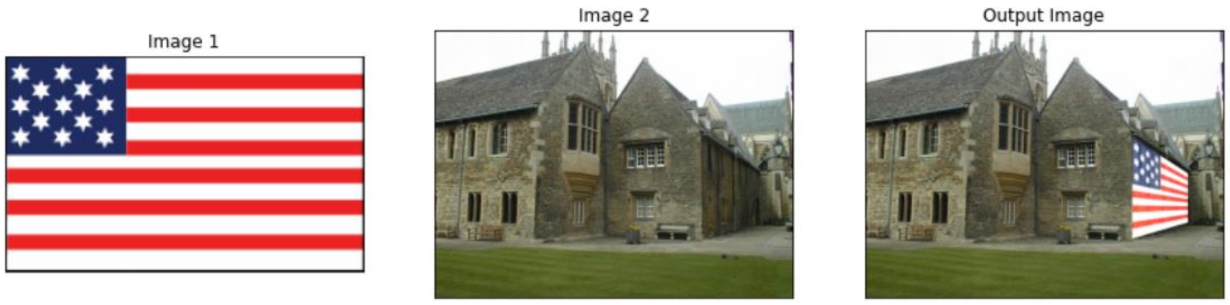
Gajaanan S.
190185D

Question 2

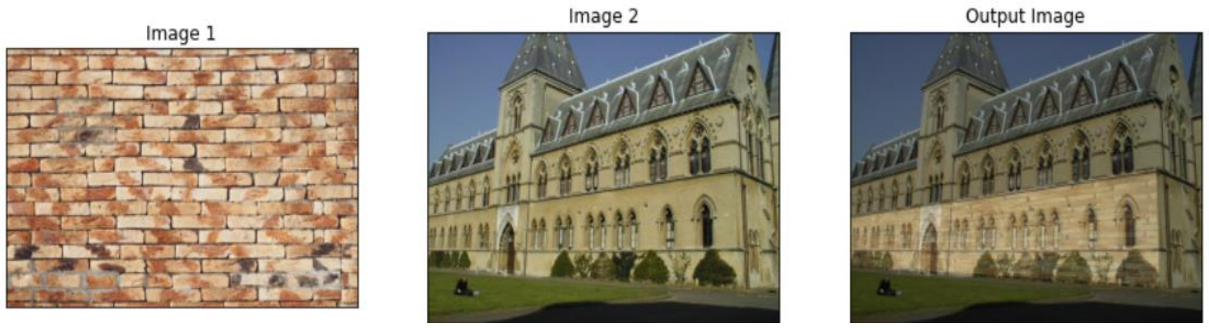
```
#Q2
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
im1 = cv.imread('flag.png')
im1 = cv.cvtColor(im1,cv.COLOR_BGR2RGB)
im1cor = np.array([[0,0],[383,0],[383,192],[0,192]])
im2 = cv.imread('001.jpg')
im2 = cv.cvtColor(im2,cv.COLOR_BGR2RGB)
im2cor = np.array([[144,205],[519,290],[523,518],[130,519]])
h, status = cv.findHomography(im1cor, im2cor)
im_out = cv2.warpPerspective(im1, h, (im2.shape[1],im2.shape[0]))
im3=cv.add(im2,im_out)
#im3=cv.addWeighted(im2,0.7,im_out,0.3,0)
plt.subplots(figsize=(15, 8))
plt.subplot(131),plt.imshow(im1),plt.title('Image 1'),plt.xticks([]), plt.yticks([])
plt.subplot(132),plt.imshow(im2),plt.title('Image 2'),plt.xticks([]), plt.yticks([])
plt.subplot(133),plt.imshow(im3),plt.title('Output Image'),plt.xticks([]), plt.yticks([])
plt.show()
```



- **Wadham College image with the British flag superimposed**.



- **Merton College image with the United States flag superimposed.**



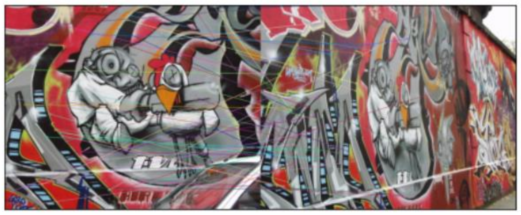
- **University library image with brick image superimposed.**

Gajaanan S.
190185D

Question 3

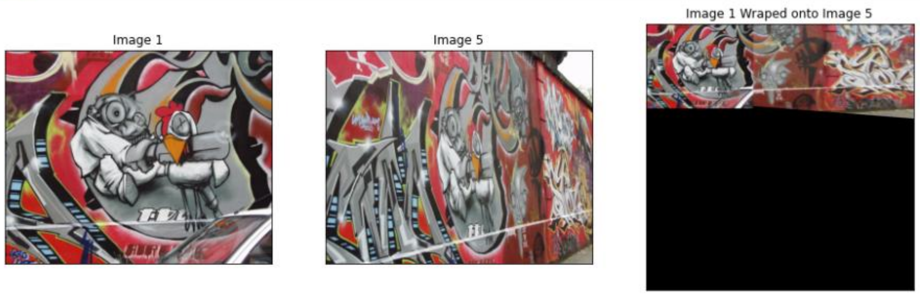
- **Computation and Matching of SIFT Features Between the Two Images (Img1 and Img5).**

```python
#Q3
import cv2
import matplotlib.pyplot as plt
img1=cv2.imread('img1.ppm')
img2=cv2.imread('img5.ppm')
img1 = cv2.cvtColor(img1, cv2.COLOR_BGR2RGB)
img2 = cv2.cvtColor(img2, cv2.COLOR_BGR2RGB)
sift = cv2.SIFT_create()
keypoints_1, descriptors_1 = sift.detectAndCompute(img1,None)
keypoints_2, descriptors_2 = sift.detectAndCompute(img2,None)
bf = cv2.BFMatcher(cv2.NORM_L1, crossCheck=True)
matches = bf.match(descriptors_1,descriptors_2)
matches = sorted(matches, key = lambda x:x.distance)
img3 = cv2.drawMatches(img1, keypoints_1, img2, keypoints_2, matches[:50], img2, flags=2)
plt.figure(figsize=(8,8))
plt.imshow(img3)
plt.xticks([]), plt.yticks([])
plt.show()
```



- **Computation of homography using relevant OpenCV function and stitching of img1 onto img5**

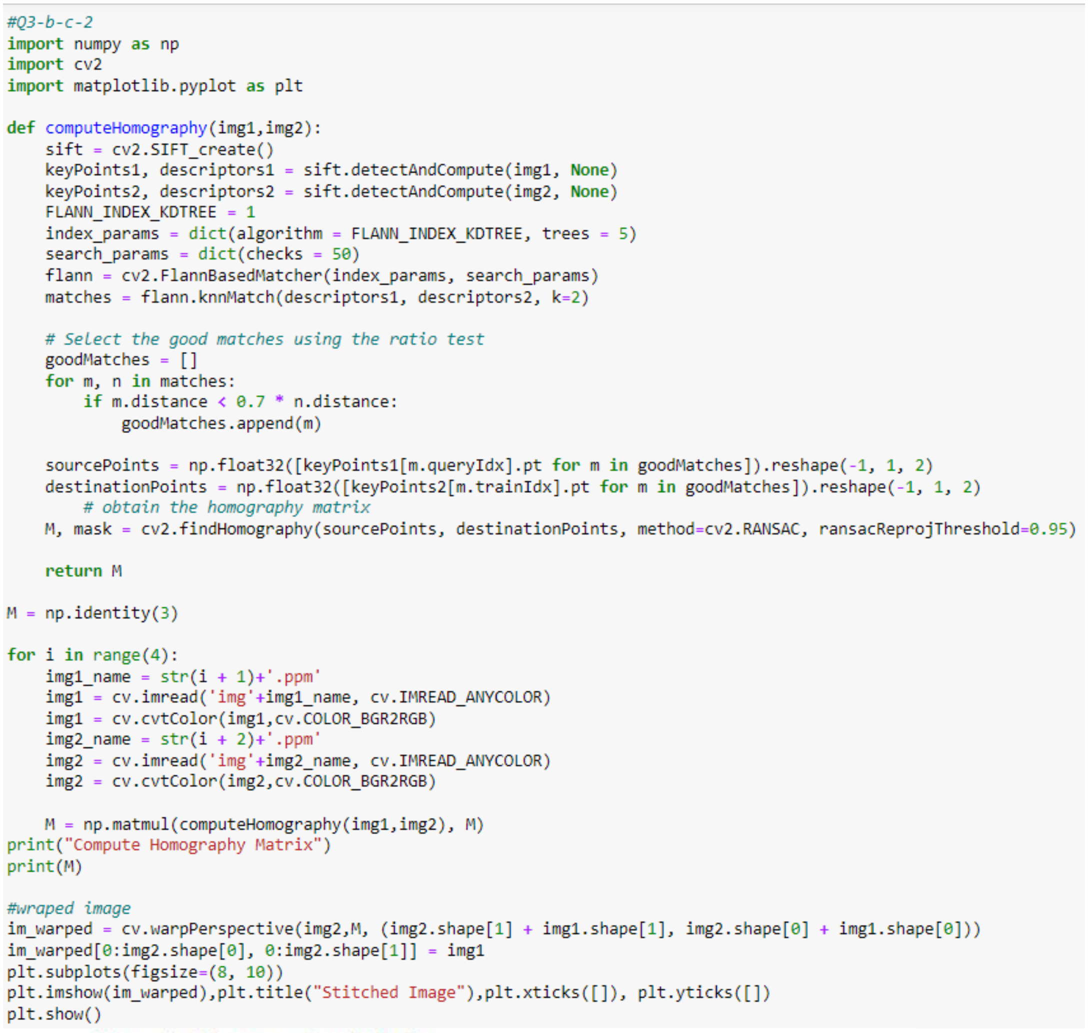
```python
#Q3-b-c-1
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
im1 = cv.imread('img1.ppm',cv.IMREAD_ANYCOLOR)
im1 = cv.cvtColor(im1,cv.COLOR_BGR2RGB)
im5 = cv.imread('img5.ppm',cv.IMREAD_ANYCOLOR)
im5 = cv.cvtColor(im5,cv.COLOR_BGR2RGB)
H = []
with open(r'H1to5p') as f:
 H = np.array([[float(h) for h in line.split()] for line in f])

im1to5 = cv.warpPerspective(im5,np.linalg.inv(H),(2000,2000))
im1to5[0:im1.shape[0],0:im1.shape[1]] = im1
fig, axes = plt.subplots(1,3, figsize=(16,16))
axes[0].imshow(im1,cmap='gray')
axes[0].set_title('Image 1')
axes[1].imshow(im5,cmap='gray')
axes[1].set_title('Image 5')
axes[2].imshow(im1to5,cmap='gray')
axes[2].set_title('Image 1 Wraped onto Image 5')
for i in range(3):
    axes[i].set_xticks([]), axes[i].set_yticks([])
plt.show()
print("Homography Matrix from dataset")
print(H)
```



```
Homography Matrix from dataset
[[ 6.2544644e-01  5.7759174e-02  2.2201217e+02]
 [ 2.2240536e-01  1.1652147e+00 -2.5605611e+01]
 [ 4.9212545e-04 -3.6542424e-05  1.0000000e+00]]
```

Gajaanan S.
190185D

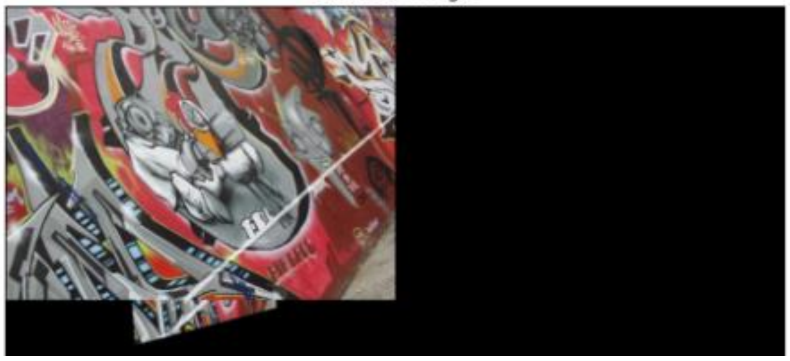
- **Computation of homography with RANSAC and stitching of img1 onto img5**



```python
#Q3-b-c-2
import numpy as np
import cv2
import matplotlib.pyplot as plt

def computeHomography(img1,img2):
    sift = cv2.SIFT_create()
    keyPoints1, descriptors1 = sift.detectAndCompute(img1, None)
    keyPoints2, descriptors2 = sift.detectAndCompute(img2, None)
    FLANN_INDEX_KDTREE = 1
    index_params = dict(algorithm = FLANN_INDEX_KDTREE, trees = 5)
    search_params = dict(checks = 50)
    flann = cv2.FlannBasedMatcher(index_params, search_params)
    matches = flann.knnMatch(descriptors1, descriptors2, k=2)

    # Select the good matches using the ratio test
    goodMatches = []
    for m, n in matches:
        if m.distance < 0.7 * n.distance:
            goodMatches.append(m)

    sourcePoints = np.float32([keyPoints1[m.queryIdx].pt for m in goodMatches]).reshape(-1, 1, 2)
    destinationPoints = np.float32([keyPoints2[m.trainIdx].pt for m in goodMatches]).reshape(-1, 1, 2)
        # obtain the homography matrix
    M, mask = cv2.findHomography(sourcePoints, destinationPoints, method=cv2.RANSAC, ransacReprojThreshold=0.95)

    return M

M = np.identity(3)

for i in range(4):
    img1_name = str(i + 1)+'.ppm'
    img1 = cv.imread('img'+img1_name, cv.IMREAD_ANYCOLOR)
    img1 = cv.cvtColor(img1,cv.COLOR_BGR2RGB)
    img2_name = str(i + 2)+'.ppm'
    img2 = cv.imread('img'+img2_name, cv.IMREAD_ANYCOLOR)
    img2 = cv.cvtColor(img2,cv.COLOR_BGR2RGB)

    M = np.matmul(computeHomography(img1,img2), M)
print("Compute Homography Matrix")
print(M)

#wraped image
im_warped = cv.warpPerspective(img2,M, (img2.shape[1] + img1.shape[1], img2.shape[0] + img1.shape[0]))
im_warped[0:img2.shape[0], 0:img2.shape[1]] = img1
plt.subplots(figsize=(8, 10))
plt.imshow(im_warped),plt.title("Stitched Image"),plt.xticks([]), plt.yticks([])
plt.show()
```

```
Compute Homography Matrix
[[ 6.10990234e-01  5.02467084e-02  2.21576840e+02]
 [ 2.14682120e-01  1.13214205e+00 -2.00428872e+01]
 [ 4.75429445e-04 -5.92812087e-05  9.92295100e-01]]
```

Stitched Image



GitHub Profile: https://github.com/Gajaan08/FUNDAMENTALS-OF-IMAGE-PROCESSING.git