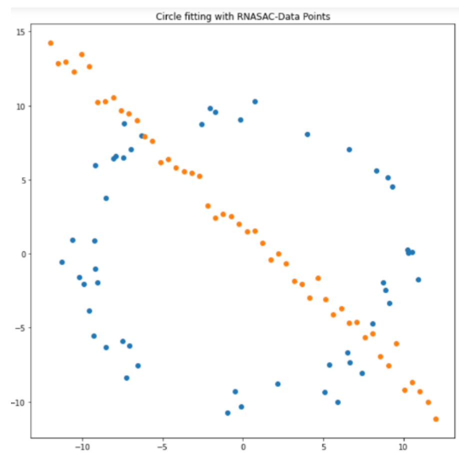
Gajaanan S.
190185D

Question 1

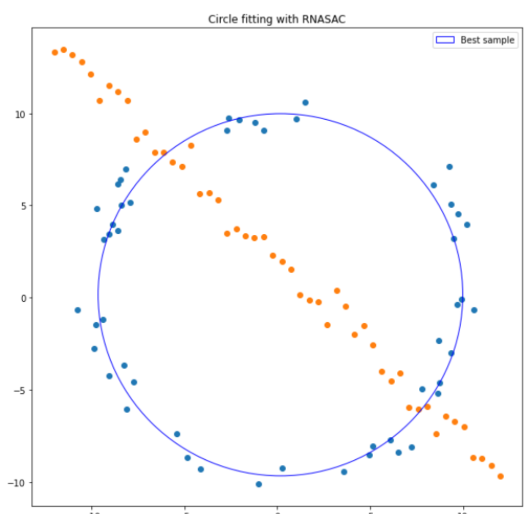
- **Estimate the circle using the RNASAC algorithm**

```python
class RANSAC:
    def __init__(self, x_data, y_data, n):
        self.x_data = x_data
        self.y_data = y_data
        self.n = n
        self.d_min = 99999
        self.best_model = None

    def random_sampling(self):
        sam = []
        _ran = []
        ct = 0

        # get three points from data
        while True:
            ran = np.random.randint(len(self.x_data))

            if ran not in _ran:
                sam.append((self.x_data[ran], self.y_data[ran]))
                _ran.append(ran)
                ct += 1

                if ct == 3:
                    break
        return sam

    def make_model(self, sam):
        p1 = sam[0]
        p2 = sam[1]
        p3 = sam[2]
        A = np.array([[p2[0] - p1[0], p2[1] - p1[1]], [p3[0] - p2[0], p3[1] - p2[1]]])
        B = np.array([[p2[0]**2 - p1[0]**2 + p2[1]**2 - p1[1]**2], [p3[0]**2 - p2[0]**2 + p3[1]**2 - p2[1]**2]])
        inv_A = inv(A)

        cx, cy = np.dot(inv_A, B) / 2
        cx, cy = cx[0], cy[0]
        r = np.sqrt((cx - p1[0])**2 + (cy - p1[1])**2)
        return cx, cy, r
```

```python
import numpy as np
import matplotlib.pyplot as plt
from numpy.linalg import inv

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)))
    x_data = x
    y_data = y
    plt.figure(figsize=(10,10))
    plt.scatter(x,y)
    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_circ,X_line))
    plt.scatter(x,y)
    return x_data, y_data
```




```python
    def eval_model(self, model):
        d = 0
        cx, cy, r = model

        for i in range(len(self.x_data)):
            dis = np.sqrt((self.x_data[i]-cx)**2 + (self.y_data[i]-cy)**2)
            if dis >= r:
                d += dis - r
            else:
                d += r - dis
        return d

    def execute_ransac(self):
        # find best model
        for i in range(self.n):
            model = self.make_model(self.random_sampling())
            d_temp = self.eval_model(model)
            if self.d_min > d_temp:
                self.best_model = model
                self.d_min = d_temp

if __name__ == '__main__':

    x_data, y_data = data_pts()

    ransac = RANSAC(x_data, y_data, 50)

    # executing ransac algorithm
    ransac.execute_ransac()

    # find the best model
    a, b, r = ransac.best_model[0], ransac.best_model[1], ransac.best_model[2]
    circle = plt.Circle((a, b), radius=r, color='b',label='Best sample', fc='y', fill=False)
    plt.gca().add_patch(circle)
    plt.title('Circle fitting with RNASAC')
    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).**

```
#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**

```
#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