EN2550 Assignment 2 on Fitting and Alignment

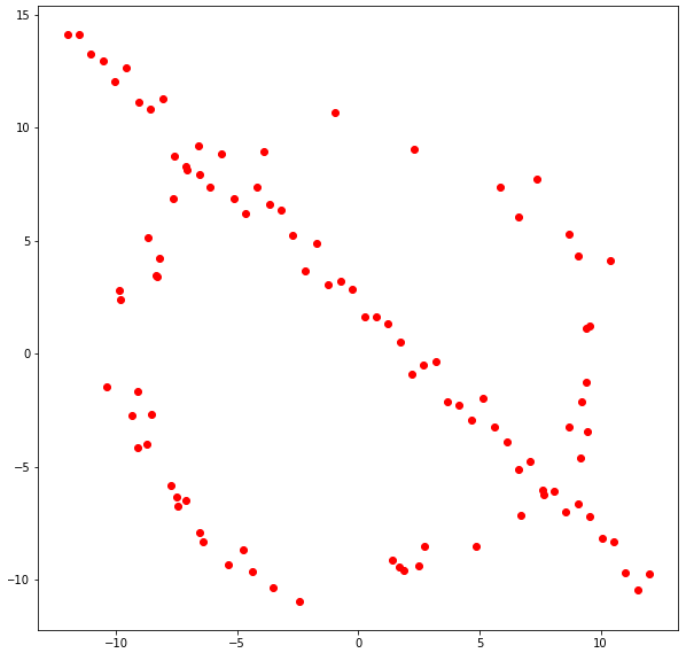
Name : R.M.K.L.Rathnayake

Index : 190520D

GitHub: https://github.com/rmklakshan/Assignment-02.git

Question 01

The RANSAC (RANdom SAmple Consensus) algorithm advances the linear regression algorithm by removing outliers from the training dataset. In here 50% percent outliers are in given dataset. To draw a circle we have to adjust following variables in this RANSAC algorithm.



S – Minimun number of points to estimate the shape. In order to estimate a circle, three points are required. Therefore S is equal to 3.

d – The threshold of Gaussian distribution. A circle's set of points has been tainted by mean-zero variance one Gaussian noise. As a result, a threshold of 1.96 gives a 0.95 chance of capturing all outliers. the value of d is 1.96.

e – Outlier probability. The dataset is made up of 100 points, 50 of which are outliers and the other 50 are inliers. Hence e is 0.5.

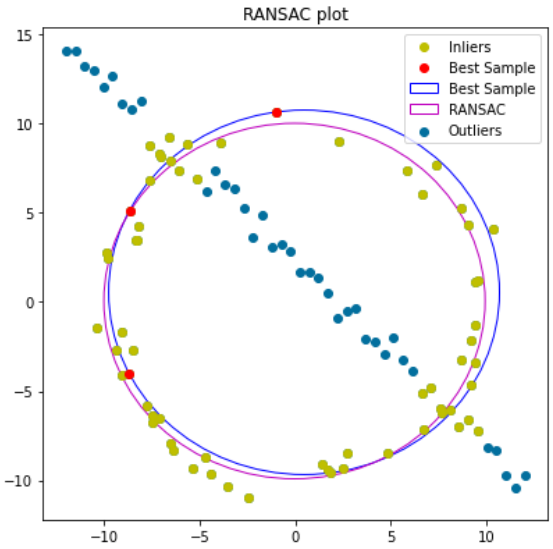
N – Number of samples. This is calculated according to the given values of e(outlier probability) and the probability of at least one random sample is free from outlier(in here its taken as 0.99).The equation is $N = \frac{\log(1-p)}{\log(1-(1-e)s)}$ .Hence the value of N is 35.

- RANSAC algorithm code:

```
def RANSAC(X):
    S = 3
    d = 1.96 # Gaussian threshold for 95%
    e = 0.50
    p = 0.99
    N = int(np.ceil(np.log(1-p)/np.log(1-(1-e)**S)))

    best_fit_circle = None
    best_fit_x = None
    best_inlier_count = 0
```

```
    for _ in range(N):
        x = []

        for i in range(S):
            hold = X[np.random.randint(0, 100), :]

            if len(x) == 0:
                x.append(hold)
            elif np.array_equal(hold, x[-1]):
                while np.array_equal(hold, x[-1]):
                    hold = X[np.random.randint(0, 100), :]

                x.append(hold)
            else:
                x.append(hold)

        a, b, r = circ_through_points(x[0], x[1], x[2])

        if a == None:
            continue

        count, inliers = get_inlier(a, b, r, X, d)

        if count > best_inlier_count:
            best_fit_circle = plt.Circle((a, b), r, color='b', fill=False, label="Best Sample")
            best_fit_x = x
            best_fit_inliers = inliers
            best_inlier_count = count

    if best_inlier_count < e:
        print("The RANSAC algorithm did not find a suitable model")
        return None, None, None, None

    xc,yc,r,_ = cf.least_squares_circle(best_fit_inliers)

    ransac_circle = plt.Circle((xc, yc), r, color='m', fill=False, label="RANSAC")

    return ransac_circle, best_fit_circle, best_fit_x, best_fit_inliers
```

- Obtained circle for the given data set.

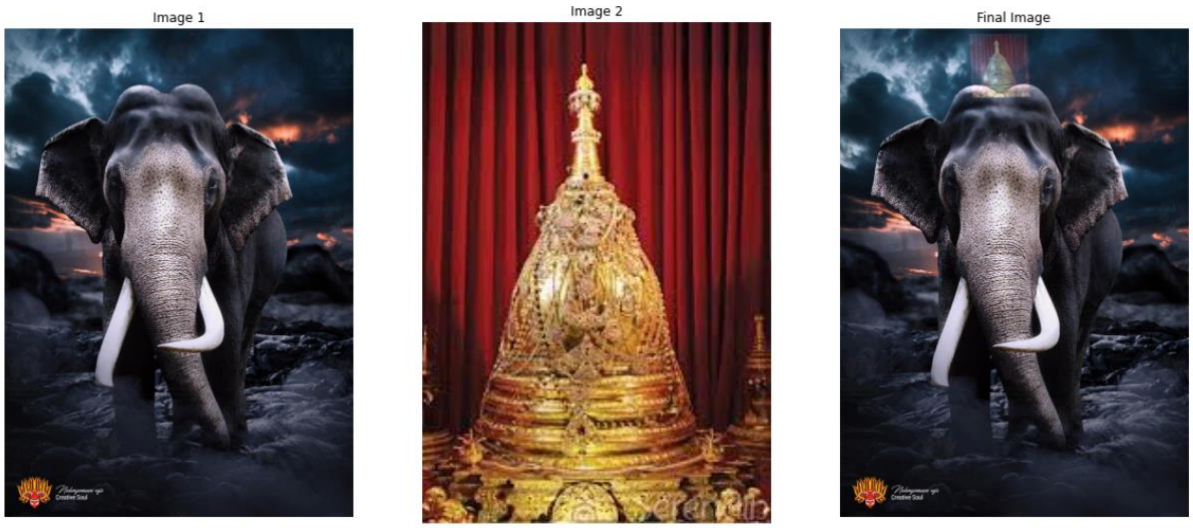


Question 2

- Maliyadeva collage with school flag.

This is the main building of Maliyadeva collage and other image is the collage flag. I chose these two images because this is my alma mater.

- Nadungamuwa Raja with a crown



This is a picture of Nadungamuwa Raja. He was the main casket bearer in the Esala procession for more than a decade.

Question 3

a) SIFT features between img1 and img5.

b) Homography computation

The key features of two images are first found and matched using SIFT matching. We can see in the above images that some feature matching is incorrect. Those are removed using RANSAC function.

```
def homography(pts1, pts2):
    mean1, mean2 = np.mean(pts1, axis=0), np.mean(pts2, axis=0)
    s1, s2 = len(pts1)*np.sqrt(2)/np.sum(np.sqrt(np.sum((pts1-mean1)**2, axis=1))), len(pts1)*np.sqrt(2)/np.sum(np.sqrt(np.sum((pts2-mean2)**2, a
    tx1, ty1, tx2, ty2 = -s1*mean1[0], -s1*mean1[1], -s2*mean2[0], -s2*mean2[1]
    T1, T2 = np.array(((s1, 0, tx1), (0, s1, ty1), (0, 0, 1))), np.array(((s2, 0, tx2), (0, s2, ty2), (0, 0, 1)))
    A = []

    for i in range(len(pts1)):
        X11, X21 = T1 @ np.concatenate((pts1[i], [1])).reshape(3, 1), T2 @ np.concatenate((pts2[i], [1])).reshape(3, 1)
        A.append((-X11[0][0], -X11[1][0], -1, 0, 0, 0, X21[0][0]*X11[0][0], X21[0][0]*X11[1][0], X21[0][0]))
        A.append((0, 0, 0, -X11[0][0], -X11[1][0], -1, X21[1][0]*X11[0][0], X21[1][0]*X11[1][0], X21[1][0]))

    A = np.array(A)
    U, S, V = np.linalg.svd(A, full_matrices=True)
    h = np.reshape(V[-1], (3, 3))
    H = linalg.inv(T2) @ h @ T1
    H = (1 / H.item(8)) * H
    return H

def dist(P1, P2, H):
    p1 = np.array([P1[0], P1[1], 1])
    p2 = np.array([P2[0], P2[1], 1])

    p2_estimate = np.dot(H, p1.T)
    p2_estimate = (1 / p2_estimate[2]) * p2_estimate
    return np.linalg.norm(p2.T - p2_estimate)

def RANSAC_homography(points1, points2):
    inlier_count, selected_inliers = 0, None
    points = np.hstack((points1, points2))
    num_iterations = int(np.log(1 - 0.95)/np.log(1 - (1 - 0.5)**4))

    for _ in range(num_iterations):
        np.random.shuffle(points)
        pts1, pts1_rem, pts2, pts2_rem = points[:4, :2],  points[4:, :2], points[:4, 2:], points[4:, 2:]
        H = homography(pts1, pts2)
        inliers = [(pts1_rem[i], pts2_rem[i]) for i in range(len(pts1_rem)) if dist(pts1_rem[i], pts2_rem[i], H) < 100]
        if len(inliers) > inlier_count:
            inlier_count = len(inliers)
            selected_inliers = np.array(inliers)
```

c) Stitch img1.ppm onto img5.ppm.