EN2550 - Assignment 2

Fitting and Alignment

Name: B.S.V.W. Munasinghe

Index Number: 190397E

Github Repo:https://github.com/vidurawarna/EN2550_CV/tree/main/Assignments/Assignment%202

Question 1

(a) Parameter selection for RANSAC algorithm

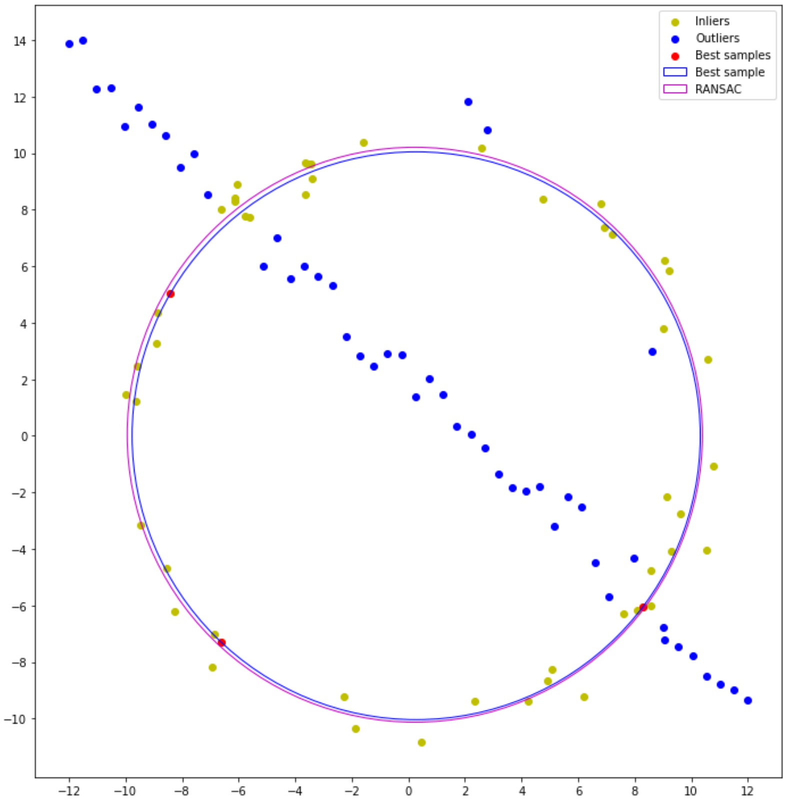
- The initial number of points: $s = 3$
- Distance threshold: $t = \frac{10}{16} \times 1.96 = 1.225$
  (Since the noise for radius is gaussian, 95% probability capturing is considered)
- Consensus set size: $d = 50$
  (Half of the points belong to the circle)
- The probability that at least one sample is outside of outliers: $p = 0.97$ (Our sample has 3 points out of 100. Therefore, $p = 0.97$)
- outlier ratio: $e = 0.5$ (50% of the total number of points)
- The number of samples:

$$N = \frac{\log(1-p)}{\log(1-(1-e)^s)} = \frac{\log(1-0.97)}{\log(1-(1-0.5)^3)} = 26$$

❖ Inside the RANSAC algorithm we have to estimate the bestfit circle to obtain the final result. Following function is used to determine the bestfit circle using all the inlier points.

```
def bestfitCircle(inlier_dots):
    x,y = inlier_dots[:,0],inlier_dots[:,1]
    N = len(x)
    x_bar,y_bar = np.mean(x),np.mean(y)
    u,v = (x - x_bar),(y - y_bar)

    a11 = sum(u**2)
    a12 = a21 = sum(u*v)
    a22 = sum(v**2)

    b1 = 0.5*(sum(u*(u**2 + v**2)))
    b2 = 0.5*(sum(v*(u**2 + v**2)))

    A = np.array([[a11,a12],[a21,a22]])
    B = np.array([[b1],[b2]])
    u_c, v_c = np.linalg.inv(A) @ B

    circ_x = u_c + x_bar
    circ_y = v_c + y_bar

    radius = np.sqrt(u_c**2 + v_c**2 + (sum(u**2 + v**2))/(N))

    return circ_x, circ_y, radius
```

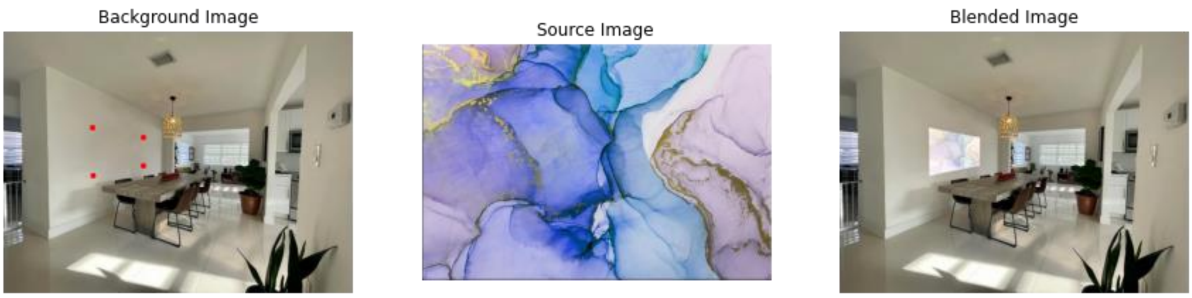
(b) Output of the RANSAC algorithm to detect the best fit circle to a given set of points.



❖ RANSAC algorithm was able to detect the circle by considering only inlier points to that circle and ignore the outlier points completely. Therefore the algorithm gives the result we expected.

Question 2

1. Testing a wall design before putting the design physically.



2. Advertise a softdrink brand on a truck box.

3. A billboard display for a Marvel super hero movie.

Background Image

Source Image

Blended Image

Question 3

(a) The similarity features between two images are found using the SIFT feature mapping function in opencv.

SIFT Matches

(b)

- Following functions are mainly used to calculate the homography between img1.ppm and img5.ppm. The match points are generated using SIFT feature matching. Then those points are further filtered by considering geometric distance between points.
- Rather than calculating the homography ($H_{1,5}$) for img1.ppm and img5.ppm directly, ($H_{1,2}$), ($H_{2,3}$), ($H_{3,4}$), ($H_{4,5}$) homographies are calculated. Then using matrix properties, $H_{1,5}$ is obtained by $(H_{1,5}) = (H_{4,5})\,(H_{3,4})\,(H_{2,3})\,(H_{1,2})$
- Here ($H_{i,j}$) means homgraphy between i^{th} image and j^{th} image.

```
def solve_H(points,des_points):
# This function solves a homography using given four data points
    A=[]
    for i in range(len(points)):
        A1 = np.concatenate((np.zeros(3),-
points[i],des_points[i][1]*points[i]),axis=None)
        A2 = np.concatenate((-
points[i],np.zeros(3),des_points[i][0]*points[i]),axis=None)
        A.append(A2)
        A.append(A1)
    A = np.array(A)
    W,V = np.linalg.eig(A.T @ A)
    # Take the vector corresponds to the minimum eigen value of A.T @ A
    # Sloution H is that vector
    H = np.reshape(np.array(V[:,np.argmin(W)]),(3,3))
    return H/H[2][2]
```

```python
def H_Ransac(points):
# This function is a RANSAC algorithm for homography calculation
    (ptsLeft,ptsRight) = points
    N = 200
    inlier_counts=[]
    H_set=[]
    for _ in range(N):
        left_4 = []
        right_4 = []
        inliers=[]
        #pick random 4 points to calculate Homography
        for i in range(4):
            j = np.random.randint(0,len(ptsLeft))
            left_4.append(ptsLeft[j])
            right_4.append(ptsRight[j])
        # Find a temporary homography
        H = solve_H(left_4,right_4)
        ssd_sum = 0
        # Filter out the inlier points according to the homography
        for j in range(len(ptsLeft)):
            p = ptsLeft[j]
            p_new = (H @ np.reshape(p,(3,1)))
            p_new = p_new/p_new[2:]
            ssd= np.sqrt(np.sum(np.square(np.reshape(p_new,(1,3))-ptsRight[j])))
            if ssd<1:
                inliers.append(p)
        inlier_counts.append(len(inliers))
        H_set.append(H)
    # select the homography with most inliers
    s = H_set[inlier_counts.index(max(inlier_counts))]
    return s/s[2][2]
```

Homography which is calculated using written RANSAC code:

$$
\begin{matrix}
6.25170214 \times 10^{-1} & 6.30718258 \times 10^{-2} & 2.20377906 \times 10^{2} \\
2.19019399 \times 10^{-1} & 1.16637164 \times 10^{0} & -2.45215250 \times 10^{1} \\
4.88113336 \times 10^{-4} & -3.34116504 \times 10^{-5} & 1 \times 10^{0}
\end{matrix}
$$

Given Homography:

$$
\begin{matrix}
6.2544644 \times 10^{-1} & 5.7759174 \times 10^{-2} & 2.2201217 \times 10^{2} \\
2.2240536 \times 10^{-1} & 1.1652147 \times 10^{0} & -2.5605611 \times 10^{1} \\
4.9212545 \times 10^{-4} & -3.6542424 \times 10^{-5} & 1 \times 10^{0}
\end{matrix}
$$

- ❖ There are some slight differences of the values in the matrices. But they are not significant. Therefore, it can be concluded that this method can be used to calculate the homography between two images successfully.

(c) This is the stitched image that is obtained using the above calculated homography.