CSE 402 Project 2
Rajaditya Shrikishan Bajaj

Q1.) Write a program that inputs a 3 × 3 matrix consisting of orientation field values and determines if the matrix corresponds to a singular point or not. If it corresponds to a singular point, determine if it is a core (loop) or a delta point. Input the following matrices into your program and report the output.

Ans.

The whole code for Q1 is given in Appendix A. The matrices and their corresponding points are:

1.)

10	15	-10
12	0	15
13	12	-5

This matrix corresponds to a **non-singular point**.

2.)

45	90	-50
50	0	-45
5	0	-5

This matrix corresponds to a **delta singularity point**.

3.)

50	0	-50
75	0	-70
85	90	-85

This matrix corresponds to a **loop singularity point**.

4.)

45	2	-50
90	0	90
-50	2	50

This matrix corresponds to a **whorl singularity point**.

Q2.) The ridge pattern in a local area of a finger can be approximated by a cosine wave:

$$w(x, y) = A\cos[2\pi f0\,(x\cos\theta + y\sin\theta)]\ .$$

Here, w (x, y) denotes the pixel intensity at location (x, y). Generate and display ridge patterns, each of size 600 × 600, at the following orientation (θ) values: 0◦, 45◦, 90◦, 135◦. You may set A = 80 and f0 = 0.01. Now repeat the exercise, with A = 160 and f0 = 0.01; A = 80 and f0 = 1; and A = 80 and f0 = 10. What are your observations?

Ans. The whole code for this question is given in Appendix B.

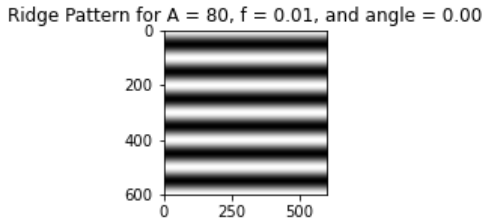
The given angle values were 0^o, 45^o, 90^o, 135^o

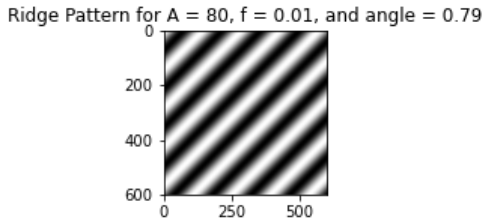
$$PI = \frac{1}{\pi}\sum_{i=0}^{7}\delta(O[(i+1)_{\text{mod }8}] - O[i]),$$

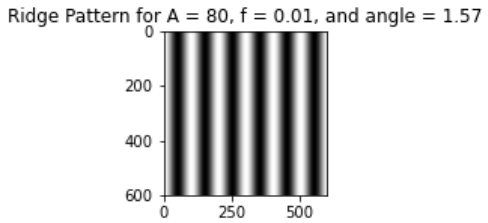
where

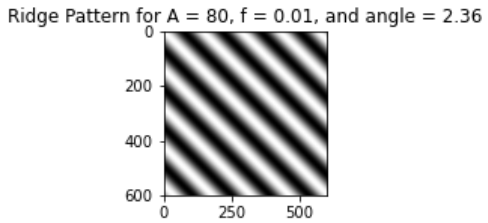
$$\delta(\theta) = \begin{cases} \theta - \pi, & \text{if } \theta > \pi/2 \\ \theta, \text{if } -\pi/2 \le \theta \le \pi/2 \\ \theta + \pi, & \text{if } \theta < -\pi/2 \end{cases}$$

1.) For A = 80 and f = 0.001 -

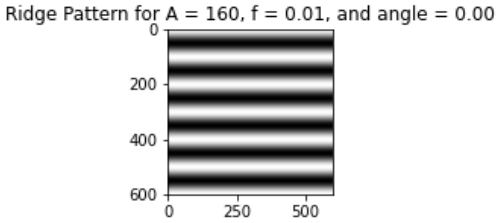


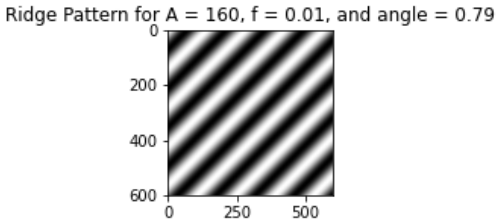


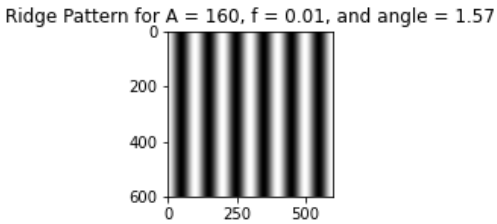


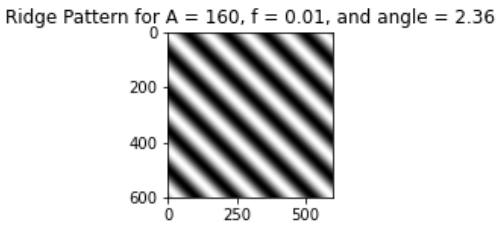


2.) For A = 160 and f = 0.01

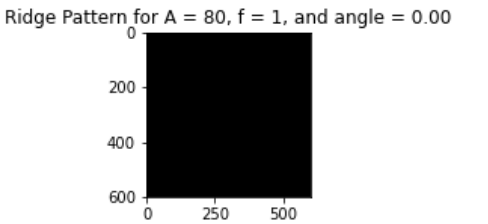


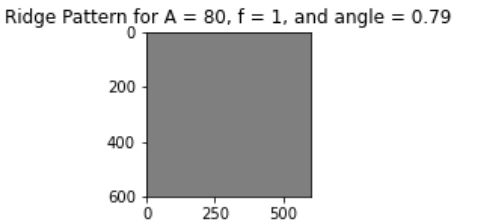


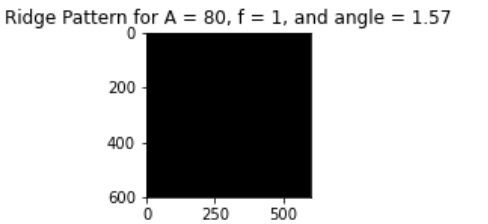


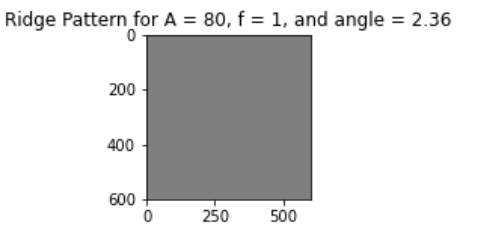


3.) For A = 80 and f = 1

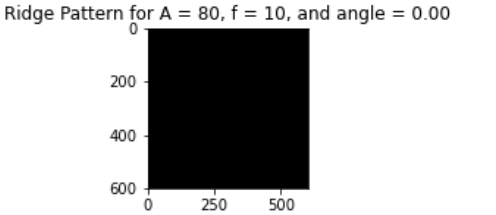


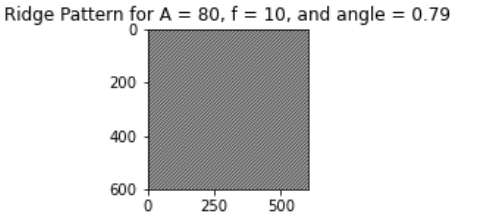


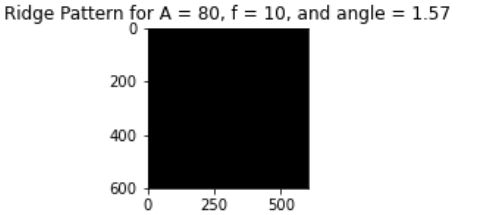


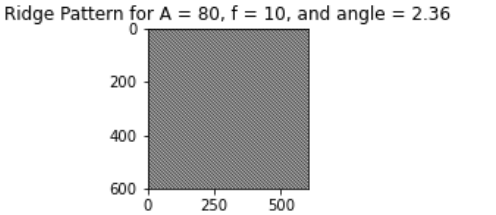


4.) For A = 180 and f = 10









As it can be seen that as the frequency increases, the distance between the ridges becomes more concentrated, i.e., the distance decreases. Another observation was that as amplitude increases, all other things same, there is no difference in the ridge graph because amplitude will increase the wavelength, but it can be seen in a 2d graph. For f = 1 and f = 10, all graphs are mostly black or gray with no pattern being shown. The reason behind this is that as frequency increased from 0.01 to 1 and 10, the distance between ridges decreases and it becomes dense which causes the graph to have a dense frequency.

Q3.) Using the gradient estimation method discussed in class (based on edge filters), write a program to compute the orientation field of a fingerprint image. The orientation should be computed for each pixel location. Use the Sobel Operator to compute the x and y gradient values at each pixel location. Use a window size of 9 × 9 when computing the orientation field value associated with a pixel location (so the value of k is 4).

Ans. For this process, Sobel Filters were used as edge filters. The gradients were computed in the horizontal and vertical directions. Then a convolution was done with the edge filters to obtain gradient images, Gx and Gy as in appendix C1. Then, the orientation field was calculated using the following formula:

$$O(x,y) = \frac{1}{2}\tan^{-1}\left[\frac{\sum_{i=-k}^{k}\sum_{j=-k}^{k} 2Gx(x+i,y+j)G_y(x+i,y+j)}{\sum_{i=-k}^{k}\sum_{j=-k}^{k} G_x^{\,2}(x+i,y+j) - G_y^{\,2}(x+i,y+j)}\right]$$

Sobel Filters:

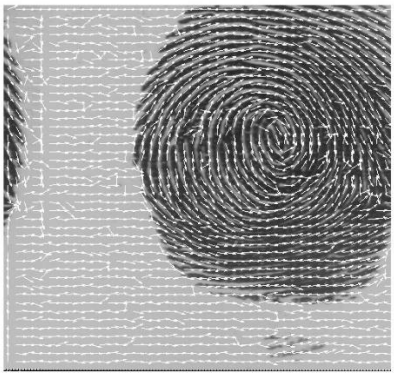
$$S_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$ ← This will highlight horizontal edges

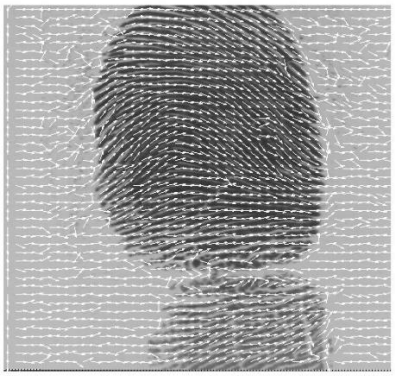
$$S_y = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$ ← This will highlight vertical edges

This was done in Appendix C2.

Then, after getting the orientation point matrix in Python, it was converted to a MATLAB matrix. Using MATLAB, the orientation direction was imposed on the original image as shown in Appendix C4. The whole code is in Appendix C.

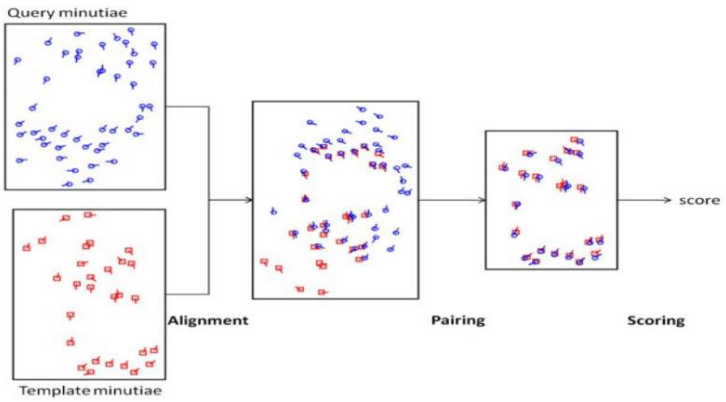
The output images were:



Q4.) Recall that minutiae set, M, is a set of 3-tuples values M = {(xi , yi,θi)}, i = 1, 2 . . .NM , where (xi , yi) is the location of minutiae i, θi is its orientation, and NM is the total number of minutiae in M. Implement the minutiae matching method discussed in class (RANSAC method) that compares two minutiae sets M1 and M2 , and outputs the transformation parameters t x , t y and tθ relating M2 with M1 , along with the number of matching minutiae pairs (you can use a tolerance value of 10 when determining matching minutiae pairs).

Ans. For this problem, the RANSAC (Random Sample Consensus) method was used. This is also known as the Brute Force Method. The simple method in visualization is:



The first ten lines of output are given below for reference, the remaining lines are in the code. The code is in Appendix 4.

first file name	second file name	tx	ty	tr	matching pairs
user001_1.minpoints	user001_2.minpoints	-65	-6	0.052360	19
user001_1.minpoints	user002_1.minpoints	-140	38	-0.069813	11
user001_1.minpoints	user002_2.minpoints	-127	27	6.143559	10
user001_1.minpoints	user003_1.minpoints	-96	117	-2.897247	14
user001_1.minpoints	user003_2.minpoints	-126	-205	4.153884	11
user001_1.minpoints	user004_1.minpoints	-241	-63	4.136430	12
user001_1.minpoints	user004_2.minpoints	131	45	1.623156	13
user001_1.minpoints	user005_1.minpoints	-189	111	-2.809980	9
user001_1.minpoints	user005_2.minpoints	-91	21	0.122173	12
user001_2.minpoints	user001_1.minpoints	65	6	-0.052360	19
user001_2.minpoints	user002_1.minpoints	-184	243	2.757620	13
user001_2.minpoints	user002_2.minpoints	0	-143	2.757620	12
user001_2.minpoints	user003_1.minpoints	26	124	5.270894	13
user001_2.minpoints	user003_2.minpoints	7	144	-1.151917	12
user001_2.minpoints	user004_1.minpoints	-198	107	1.448623	12
user001_2.minpoints	user004_2.minpoints	-22	40	-5.550147	14
user001_2.minpoints	user005_1.minpoints	-85	-305	3.089233	12
user001_2.minpoints	user005_2.minpoints	-57	-77	4.118977	11

APPENDIX

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as img
import math
import cv2
from PIL import Image
import itertools
import scipy.io
from scipy import signal
```

Appendix A (The type of singular point, Question 1)

Appendix A1 – Calculating the Poincare Index

```
def delta(theta):
    '''
    Function to find the theta in Poincare Index equation
    '''
    if (theta > 90):
        return theta - 180
    elif (theta >= -90) and (theta <= 90):
        return theta
    elif (theta < -90):
        return theta + 180
```

```
def poincare_index(matrix):
    sigma = 0
    for i in range(len(matrix)):
       # print(matrix[(i+1)%8] - matrix[i])
        #print(delta(matrix[(i+1)%8] - matrix[i]))
        sigma += delta(matrix[(i+1)%8] - matrix[i])
    pi = sigma / 180
    return int(pi)
```

Appendix A2 – Calculating the type of singularity point for the given arrays

```
def singular_points(matrix):
    pi = poincare_index(matrix)
    if (pi == 0):
        print("The matrix corresponds to a non-singular point.")
    elif (pi == 1):
        print("The matrix corresponds to a loop singularity point.")
    elif (pi == -1):
        print("The matrix corresponds to a delta singularity point.")
    elif (pi == 2):
        print("The matrix corresponds to a whorl singularity point.")
```

```
Q1_a = [15, -10, 15, 10, 12, 13, 12, -5]
singular_points(Q1_a)
```

```
The matrix corresponds to a non-singular point.
```

```
Q1_b = [-45, -50, 90, 45, 50, 5, 0, -5]
singular_points(Q1_b)
```

```
The matrix corresponds to a delta singularity point.
```

```
Q1_c = [-70, -50, 0, 50, 75, 85, 90, -85]
singular_points(Q1_c)
```

```
The matrix corresponds to a loop singularity point.
```

```
Q1_d = [90, -50, 2, 45, 90, -50, 2, 50]
singular_points(Q1_d)
```

```
The matrix corresponds to a whorl singularity point.
```

Appendix B (Graphing the Ridge Pattern, Question 2)

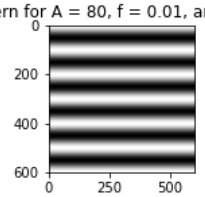
Appendix B1 – Graphing the Ridge Pattern

```
def ridge_pattern(A, f, theta):
    x = np.arange(0, 601, 1)
    X = x[:,np.newaxis]
    Y = x[np.newaxis,:]
    theta = theta*(math.pi/180)
    nu = A*np.cos(2*np.pi*f*(X*np.cos(theta) + Y*np.sin(theta)))
    title = f'Ridge Pattern for A = {A}, f = {f}, and angle = {theta:.2f}'
    plt.figure(figsize=(2,2))
    plt.title(title)
    plt.imshow(nu, "gray")
    plt.show()
```

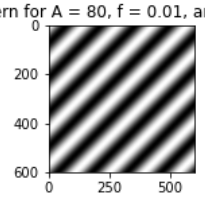
Appendix B2 – Graph for the given values

```
tetha_list = [0, 45, 90, 135]
for i in range(len(tetha_list)):
    ridge_pattern(80, 0.01, tetha_list[i])
plt.show()
```

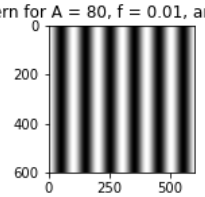
Ridge Pattern for A = 80, f = 0.01, and angle = 0.00



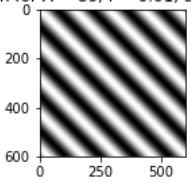
Ridge Pattern for A = 80, f = 0.01, and angle = 0.79



Ridge Pattern for A = 80, f = 0.01, and angle = 1.57

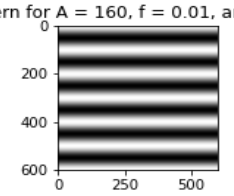


Ridge Pattern for A = 80, f = 0.01, and angle = 2.36

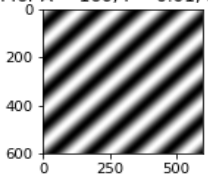



```
tetha_list = [0, 45, 90, 135]
for i in tetha_list:
    ridge_pattern(160, 0.01, i)
```

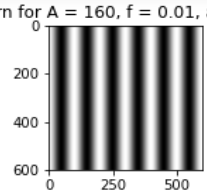
Ridge Pattern for A = 160, f = 0.01, and angle = 0.00



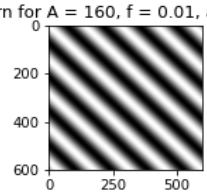
Ridge Pattern for A = 160, f = 0.01, and angle = 0.79



Ridge Pattern for A = 160, f = 0.01, and angle = 1.57

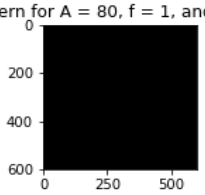


Ridge Pattern for A = 160, f = 0.01, and angle = 2.36

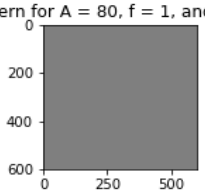



```
tetha_list = [0, 45, 90, 135]
for i in tetha_list:
    ridge_pattern(80, 1, i)
```

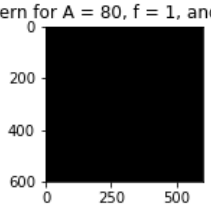
Ridge Pattern for A = 80, f = 1, and angle = 0.00



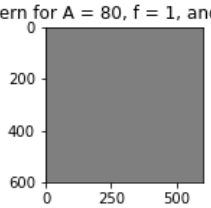
Ridge Pattern for A = 80, f = 1, and angle = 0.79



Ridge Pattern for A = 80, f = 1, and angle = 1.57

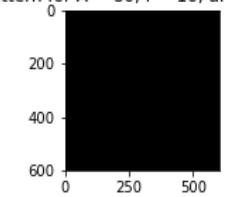


Ridge Pattern for A = 80, f = 1, and angle = 2.36

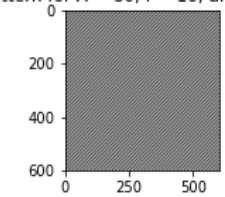


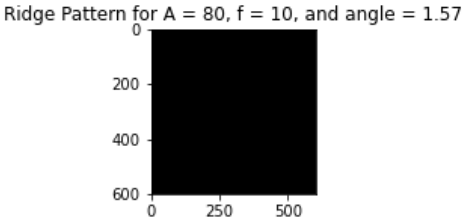

```
tetha_list = [0, 45, 90, 135]
for i in tetha_list:
    ridge_pattern(80, 10, i)
```

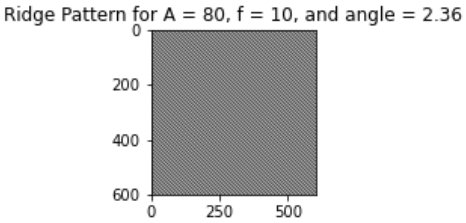
Ridge Pattern for A = 80, f = 10, and angle = 0.00



Ridge Pattern for A = 80, f = 10, and angle = 0.79







Appendix C (Orientation field, Question 3)

Appendix C1 – Convolution of an image with Sobel Filter

```
def read_img(path):
    img = Image.open(path)
    return np.array(img)
```

```
def sobel_filters(image):
    Sx = np.matrix([[-1, -2, -1],[0, 0, 0],[1, 2, 1]])
    Sy = np.matrix([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]])

    Gx = signal.convolve(image, Sx)
    Gy = signal.convolve(image, Sy)

    convolved_img = np.zeros(Gx.shape)
    for i in range(4,len(Gx)-4):
        for j in range(4,len(Gx[0])-4):
            num = 0
            den = 0
            for k in range(-4,5):
                for l in range(-4,5):
                    num += 2 * Gx[i+k,j+l] * Gy[i+k,j+l]
                    den += Gx[i+k,j+l]**2 - Gy[i+k,j+l]**2
            convolved_img[i,j] = .5*math.atan2(num,den)+np.pi/2

    return convolved_img
```

Appendix C2 – Converting to MATLAB array

```
images = ['proj02_q1_fingerprint_images/user001_1.gif', 'proj02_q1_fingerprint_images/user002_1.gif',
          'proj02_q1_fingerprint_images/user003_1.gif', 'proj02_q1_fingerprint_images/user004_1.gif',
          'proj02_q1_fingerprint_images/user005_1.gif', 'proj02_q1_fingerprint_images/user006_1.gif',
          'proj02_q1_fingerprint_images/user007_1.gif', 'proj02_q1_fingerprint_images/user008_1.gif',
          'proj02_q1_fingerprint_images/user009_1.gif', 'proj02_q1_fingerprint_images/user010_1.gif']
for i in images:
    img = read_img(i)
    a = sobel_filters(img)
    file = i[29:38] + '.mat'
    np.savetxt(file, a)
    print(i)
# img = read_img(images[0])
# a = sobel_filters(img)
# np.savetxt(i[28:38].mat', a)
```

```
proj02_q1_fingerprint_images/user001_1.gif
proj02_q1_fingerprint_images/user002_1.gif
proj02_q1_fingerprint_images/user003_1.gif
proj02_q1_fingerprint_images/user004_1.gif
proj02_q1_fingerprint_images/user005_1.gif
proj02_q1_fingerprint_images/user006_1.gif
proj02_q1_fingerprint_images/user007_1.gif
proj02_q1_fingerprint_images/user008_1.gif
proj02_q1_fingerprint_images/user009_1.gif
proj02_q1_fingerprint_images/user010_1.gif
```

Appendix C3 – Using MATLAB to impose orientation on fingerprint image

```
input_image = double(imread('user010_1.gif'));
filename = 'user010_1.mat';
M = double(load(filename, '-ASCII'));
drawOrientation(input_image, M)

function drawOrientation(img, ofield, varargin)
if (nargin==2)
    blksz = 11;
else
    blksz = varargin{1};
end

hblksz = round(blksz/2);
r = hblksz;

[nr nc] = size(ofield);
u_ofield = r*cos(ofield);
v_ofield = r*sin(ofield);

[X, Y] = meshgrid(hblksz:blksz:nr-hblksz, hblksz:blksz:nc-hblksz);
X = X(:);
Y = Y(:);
for i=1:size(X)
    U(i) = u_ofield(X(i), Y(i));
    V(i) = v_ofield(X(i), Y(i));
end
figure;
imshow(img,[]);
hold on;
h=quiver(Y, X, V', U');
set(h,'Color',[1 1 1]);
end
```

Appendix D (RANSAC Algorithm, Question 4)

Appendix D1 – Loading the data and calling the RANSAC Function

```
user_data = {}
files = ['proj02_q2_minpoints/user001_1.minpoints', 'proj02_q2_minpoints/user001_2.minpoints',
         'proj02_q2_minpoints/user002_1.minpoints', 'proj02_q2_minpoints/user002_2.minpoints',
         'proj02_q2_minpoints/user003_1.minpoints', 'proj02_q2_minpoints/user003_2.minpoints',
         'proj02_q2_minpoints/user004_1.minpoints', 'proj02_q2_minpoints/user004_2.minpoints',
         'proj02_q2_minpoints/user005_1.minpoints', 'proj02_q2_minpoints/user005_2.minpoints']
count = 0
for i in files:
    with open(i, 'r') as f:
        temp_list = []
        for x in f.read().splitlines():
            x = x.split('\t',)
            temp_list.append(x)
        user_data[i] = temp_list

        for x in range(len(user_data[i])):
            temp_list = []
            for a in user_data[i][x]:
                a = int(a)
                temp_list.append(a)
            user_data[i][x] = temp_list
```

```
tolerance = 10
print('{:^25} {:^25} {:^4} {:^4} {:^12} {:^10}'.format('first file name', 'second file name', 'tx', 'ty', 'tr', 'matching pa
for i in range(len(files)):
    for j in range(len(files)):
        if (i == j):
            continue
        else:
            c, tx, ty, tr = ransac_matching_algorithm(user_data[files[i]], user_data[files[j]], tolerance)
            print('{:25s} {:25s} {:4d} {:4d} {:12f} {:4d}'.format(files[i][20:], files[j][20:], tx, ty, tr, c))
```

Appendix D2 – RANSAC Algorithm implementation

```python
def ransac_matching_algorithm(P, Q, tolerance):
    C = []
    transformation_points = []

    final = []
    for i in P:
        for j in Q:
            tx = (j[0] - i[0])
            ty = (j[1] - i[1])
            tr = (j[2] - i[2]) * math.pi/180

            P_new = []

            for k in P:
                xk = (k[0] - i[0])*math.cos(tr) + (k[1] - i[1])*math.sin(tr) + i[0] + tx
                yk = -(k[0] - i[0])*math.sin(tr) + (k[1] - i[1])*math.cos(tr) + i[1] + ty
                P_new.append([xk, yk, k[2]])

            matching_score = 0
            for x1, y1, z1 in Q:
                for x2, y2, z2 in P_new:
                    d = math.sqrt(((x2-x1)**2) + ((y2-y1)**2))
                    if d <= tolerance:
                        matching_score += 1

            C.append(matching_score)
            transformation_points.append([tx, ty, tr])
    if (len(C) == 0):
        return 0, 0, 0, 0
    else:
        maximum_value = max(C, default=0)
        maximum_index = C.index(maximum_value)
        transformation = transformation_points[maximum_index]
        tx, ty, tr = transformation[0], transformation[1], transformation[2]
        return maximum_value, tx, ty, tr
```

Appendix D3 – Output of the implementation

first file name	second file name	tx	ty	tr	matching pairs
user001_1.minpoints	user001_2.minpoints	-65	-6	0.052360	19
user001_1.minpoints	user002_1.minpoints	-140	38	-0.069813	11
user001_1.minpoints	user002_2.minpoints	-127	27	6.143559	10
user001_1.minpoints	user003_1.minpoints	-96	117	-2.897247	14
user001_1.minpoints	user003_2.minpoints	-126	-205	4.153884	11
user001_1.minpoints	user004_1.minpoints	-241	-63	4.136430	12
user001_1.minpoints	user004_2.minpoints	131	45	1.623156	13
user001_1.minpoints	user005_1.minpoints	-189	111	-2.809980	9
user001_1.minpoints	user005_2.minpoints	-91	21	0.122173	12
user001_2.minpoints	user001_1.minpoints	65	6	-0.052360	19
user001_2.minpoints	user002_1.minpoints	-184	243	2.757620	13
user001_2.minpoints	user002_2.minpoints	0	-143	2.757620	12
user001_2.minpoints	user003_1.minpoints	26	124	5.270894	13
user001_2.minpoints	user003_2.minpoints	7	144	-1.151917	12
user001_2.minpoints	user004_1.minpoints	-198	107	1.448623	12
user001_2.minpoints	user004_2.minpoints	-22	40	-5.550147	14
user001_2.minpoints	user005_1.minpoints	-85	-305	3.089233	12
user001_2.minpoints	user005_2.minpoints	-57	-77	4.118977	11
user002_1.minpoints	user001_1.minpoints	140	-38	0.069813	11
user002_1.minpoints	user001_2.minpoints	86	196	-5.323254	13
user002_1.minpoints	user002_2.minpoints	15	-2	0.000000	40
user002_1.minpoints	user003_1.minpoints	15	-84	-0.244346	17
user002_1.minpoints	user003_2.minpoints	-9	33	1.239184	16
user002_1.minpoints	user004_1.minpoints	52	61	-4.380776	13
user002_1.minpoints	user004_2.minpoints	-69	-28	1.082104	17
user002_1.minpoints	user005_1.minpoints	56	124	-1.204277	15
user002_1.minpoints	user005_2.minpoints	36	30	-1.204277	14
user002_2.minpoints	user001_1.minpoints	127	-27	-6.143559	10
user002_2.minpoints	user001_2.minpoints	0	143	-2.757620	12
user002_2.minpoints	user002_1.minpoints	-15	2	0.000000	40
user002_2.minpoints	user003_1.minpoints	121	-3	1.762783	17
user002_2.minpoints	user003_2.minpoints	6	-94	0.087266	18
user002_2.minpoints	user004_1.minpoints	12	26	-4.293510	13
user002_2.minpoints	user004_2.minpoints	34	17	-0.680678	19
user002_2.minpoints	user005_1.minpoints	116	-62	-4.049164	14
user002_2.minpoints	user005_2.minpoints	43	196	-5.113815	12

user003_1.minpoints	user001_1.minpoints	96	-117	2.897247	14
user003_1.minpoints	user001_2.minpoints	-26	-124	-5.270894	13
user003_1.minpoints	user002_1.minpoints	-15	84	0.244346	17
user003_1.minpoints	user002_2.minpoints	-121	3	-1.762783	17
user003_1.minpoints	user003_2.minpoints	-1	17	0.069813	26
user003_1.minpoints	user004_1.minpoints	38	104	-5.986479	16
user003_1.minpoints	user004_2.minpoints	232	-6	2.932153	19
user003_1.minpoints	user005_1.minpoints	35	-59	-0.139626	16
user003_1.minpoints	user005_2.minpoints	31	86	0.558505	17
user003_2.minpoints	user001_1.minpoints	126	205	-4.153884	11
user003_2.minpoints	user001_2.minpoints	-7	-144	1.151917	12
user003_2.minpoints	user002_1.minpoints	9	-33	-1.239184	16
user003_2.minpoints	user002_2.minpoints	-6	94	-0.087266	18
user003_2.minpoints	user003_1.minpoints	1	-17	-0.069813	26
user003_2.minpoints	user004_1.minpoints	153	219	-4.450590	14
user003_2.minpoints	user004_2.minpoints	-80	-142	2.251475	18
user003_2.minpoints	user005_1.minpoints	-21	75	-1.762783	17
user003_2.minpoints	user005_2.minpoints	-79	49	1.012291	13
user004_1.minpoints	user001_1.minpoints	241	63	-4.136430	12
user004_1.minpoints	user001_2.minpoints	198	-107	-1.448623	12
user004_1.minpoints	user002_1.minpoints	-16	-18	0.855211	13
user004_1.minpoints	user002_2.minpoints	-12	-26	4.293510	13
user004_1.minpoints	user003_1.minpoints	-104	146	-5.148721	16
user004_1.minpoints	user003_2.minpoints	-162	161	1.378810	14
user004_1.minpoints	user004_2.minpoints	60	-39	-0.052360	30
user004_1.minpoints	user005_1.minpoints	264	86	-2.914700	13
user004_1.minpoints	user005_2.minpoints	-49	179	-3.630285	13
user004_2.minpoints	user001_1.minpoints	-134	-91	2.460914	13
user004_2.minpoints	user001_2.minpoints	-28	38	0.436332	14
user004_2.minpoints	user002_1.minpoints	69	28	-1.082104	17
user004_2.minpoints	user002_2.minpoints	-34	-17	0.680678	19
user004_2.minpoints	user003_1.minpoints	-232	6	-2.932153	19
user004_2.minpoints	user003_2.minpoints	80	142	-2.251475	18
user004_2.minpoints	user004_1.minpoints	-60	39	0.052360	30
user004_2.minpoints	user005_1.minpoints	-49	163	-4.310963	16
user004_2.minpoints	user005_2.minpoints	-211	-19	1.745329	14
user005_1.minpoints	user001_1.minpoints	110	143	-3.543018	9
user005_1.minpoints	user001_2.minpoints	85	305	-3.089233	12
user005_1.minpoints	user002_1.minpoints	-56	-124	1.204277	15
user005_1.minpoints	user002_2.minpoints	-116	62	4.049164	14
user005_1.minpoints	user003_1.minpoints	-35	59	0.139626	16
user005_1.minpoints	user003_2.minpoints	21	-75	1.762783	16
user005_1.minpoints	user004_1.minpoints	-264	-86	2.914700	13
user005_1.minpoints	user004_2.minpoints	-223	-50	2.984513	16
user005_1.minpoints	user005_2.minpoints	-75	97	0.000000	26
user005_2.minpoints	user001_1.minpoints	91	-21	-0.122173	12
user005_2.minpoints	user001_2.minpoints	27	-30	0.139626	11
user005_2.minpoints	user002_1.minpoints	-36	-30	1.204277	14
user005_2.minpoints	user002_2.minpoints	-49	95	-2.356194	12
user005_2.minpoints	user003_1.minpoints	-31	-86	-0.558505	17
user005_2.minpoints	user003_2.minpoints	79	-49	-1.012291	13
user005_2.minpoints	user004_1.minpoints	49	-179	3.630285	13
user005_2.minpoints	user004_2.minpoints	211	19	-1.745329	14
user005_2.minpoints	user005_1.minpoints	75	-97	0.000000	26