General Sir John Kotelawala Defence University

Faculty of Engineering

Department of Electrical, Electronic and Telecommunication Engineering

Image Processing and Machine Vision

Assignment 02

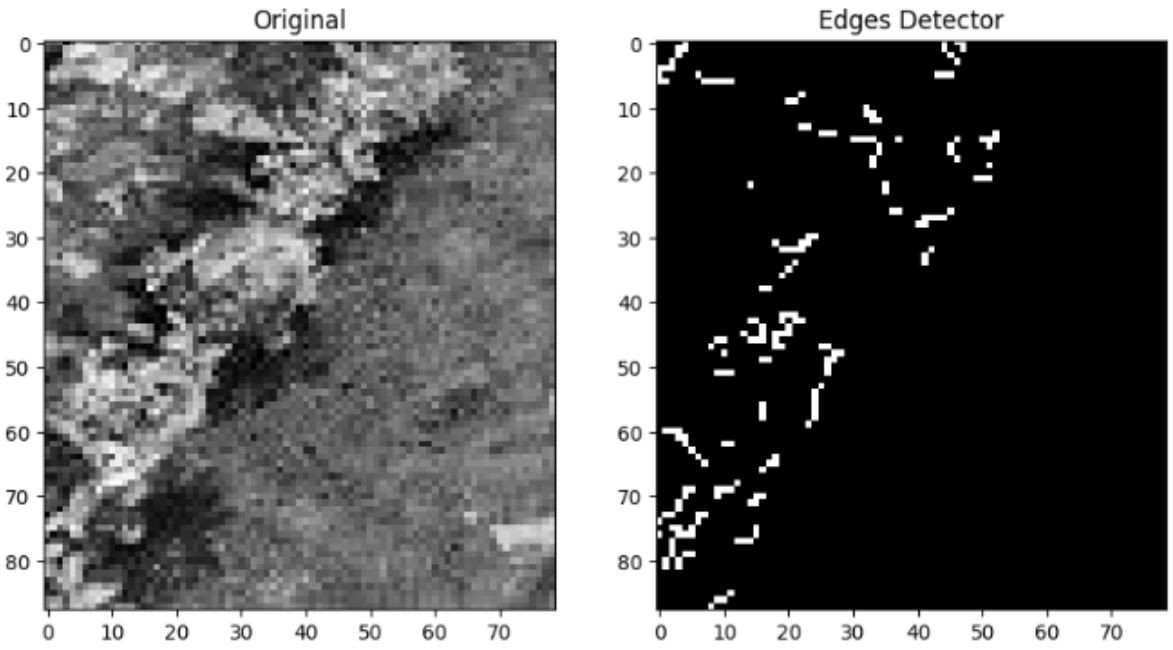
Fitting

Name: MPSM Pathirana

Reg No: D/ENG/22/0061/EE

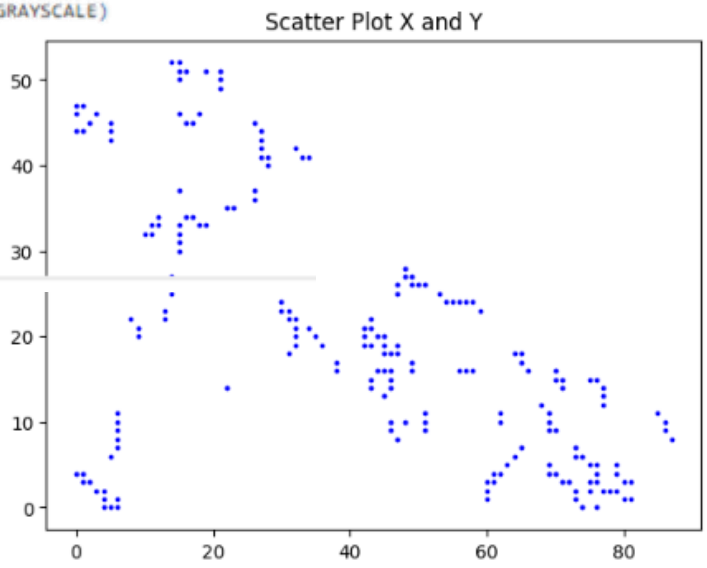
Question 01

```
import cv2 as cv
import numpy as np
import matplotlib.pylab as plt

img = cv.imread('Crop_field_cropped.jpg', cv.IMREAD_GRAYSCALE)

edges = cv.Canny(img,550,690)
indices = np.where(edges != [0])
x = indices[1]
y = indices[0]

fig, ax = plt.subplots(1, 2, figsize=(10, 20))
ax[0].imshow(img, cmap="gray")
ax[0].set_title("Original")
ax[1].imshow(edges, cmap="gray")
ax[1].set_title("Edges Detector")
plt.show()
```



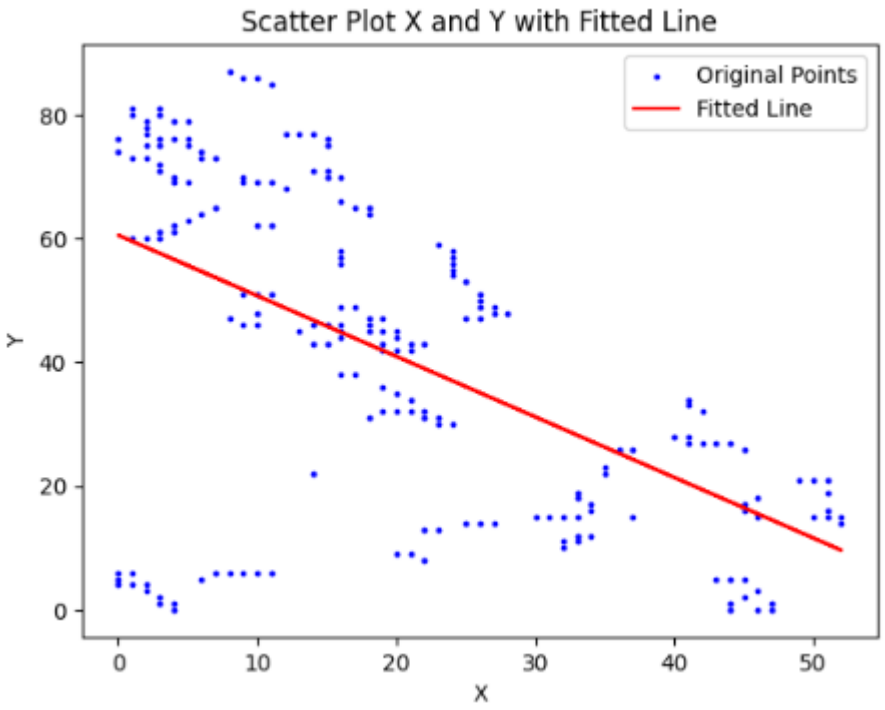
Question 02

```
import cv2 as cv
import numpy as np
import matplotlib.pylab as plt

img = cv.imread('Crop_field_cropped.jpg', cv.IMREAD_GRAYSCALE)

edges = cv.Canny(img,550,690)
indices = np.where(edges != [0])
x = indices[0]
y = indices[1]

plt.scatter(x,y, c='blue',s=3)
plt.title('Scatter Plot X and Y')
plt.show()
```



Question 03

```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

img = cv.imread('Crop_field_cropped.jpg', cv.IMREAD_GRAYSCALE)

edges = cv.Canny(img, 550, 690)
indices = np.where(edges != [0])
x = indices[1]
y = indices[0]

m = np.polyfit(x, y, 1)
fitted_line = np.poly1d(m)

fitted_y = fitted_line(x)

plt.scatter(x, y, c='blue', s=3, label='Original Points')
plt.plot(x, fitted_y, c='red', label='Fitted Line')
plt.title('Scatter Plot X and Y with Fitted Line')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.show()
```



Question 04

```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

img = cv.imread('Crop_field_cropped.jpg', cv.IMREAD_GRAYSCALE)

edges = cv.Canny(img, 550, 690)
indices = np.where(edges != [0])
x = indices[1]
y = indices[0]

m = np.polyfit(x, y, 1)

fitted_line = np.polyfit(x, y, 1)
angle = np.arctan(fitted_line[0]) * 180 / np.pi

print(f"Estimated angle of the crop field: {angle:.2f} degrees")
```

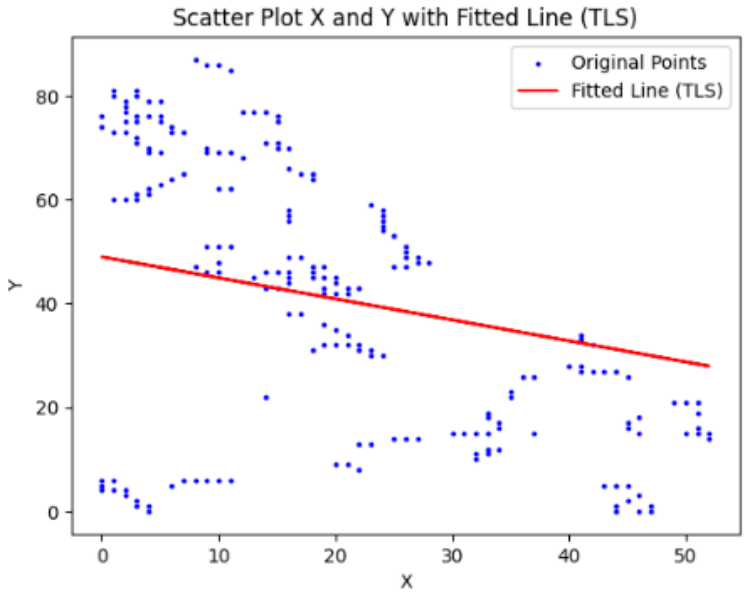
```
Estimated angle of the crop field: -44.39 degrees
```

Question 05

No. If the scatter plot points are not distributed symmetrically and the line is not fitted correctly, it can significantly impact the accuracy of the estimated crop angle using the least-squares-fit line method.

Question 06

```
data = np.vstack([x, y]).T
centroid = np.mean(data, axis=0)
data_centered = data - centroid
U, S, Vt = np.linalg.svd(data_centered, full_matrices=False)
slope = -Vt[0, 0] / Vt[1, 0]
intercept = centroid[1] - slope * centroid[0]

fitted_y_tls = slope * x + intercept

plt.scatter(x, y, c='blue', s=3, label='Original Points')
plt.plot(x, fitted_y_tls, c='red', label='Fitted Line (TLS)')
plt.title('Scatter Plot X and Y with Fitted Line (TLS)')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.show()
```



Question 07

```
import cv2 as cv
import numpy as np

img = cv.imread('Crop_field_cropped.jpg', cv.IMREAD_GRAYSCALE)

edges = cv.Canny(img, 550, 690)
indices = np.where(edges != [0])
x = indices[1]
y = indices[0]

data = np.vstack([x, y]).T
centroid = np.mean(data, axis=0)
data_centered = data - centroid
U, S, Vt = np.linalg.svd(data_centered, full_matrices=False)
slope = -Vt[0, 0] / Vt[1, 0]

angle_radians = np.arctan(slope)
angle_degrees = np.degrees(angle_radians)
print("Angle of the TLS fitted line:", angle_degrees)
```

```
Angle of the TLS fitted line: -22.026565186321758
```

Question 08

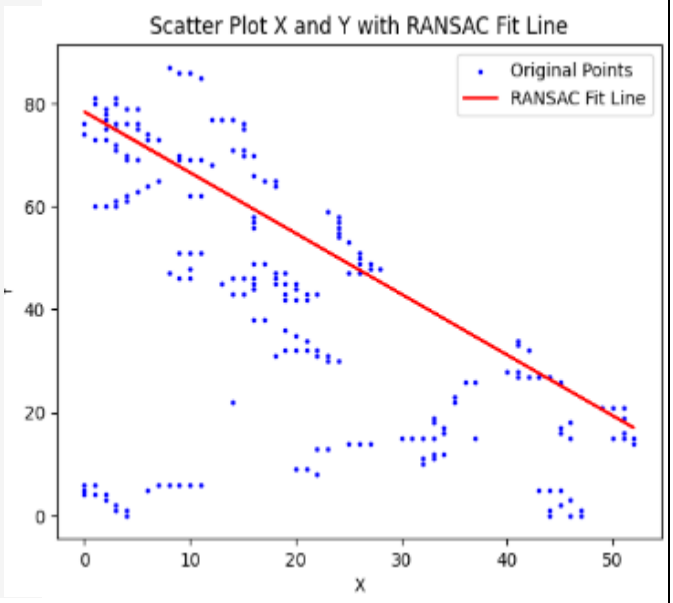
If the scatter plot points representing elevation data across the crop field are not symmetrically distributed and the line fitted using the total least-squares-fit method is not aligned correctly, it can significantly impact the accuracy of the estimated crop angle.

Question 09

RANSAC Algorithm

Question 10

```
iterations = 1000
sample_size = 2
inlier_threshold = 5

best_model = None
best_inliers = []

for _ in range(iterations):
    sample_indices = np.random.choice(len(points), sample_size, replace=False)
    sample_points = points[sample_indices]

    x1, y1 = sample_points[0]
    x2, y2 = sample_points[1]
    if x2 - x1 != 0:
        model_slope = (y2 - y1) / (x2 - x1)
        model_intercept = y1 - model_slope * x1

        distances = np.abs(points[:, 1] - model_slope * points[:, 0] - model_intercept)

        inliers = points[distances < inlier_threshold]
        if len(inliers) > len(best_inliers):
            best_inliers = inliers
            best_model = (model_slope, model_intercept)

x_inliers = best_inliers[:, 0]
y_inliers = best_inliers[:, 1]
m, c = np.polyfit(x_inliers, y_inliers, 1)
fitted_y = m * x + c
```



Question 11

```
iterations = 1000
sample_size = 2
inlier_threshold = 5

best_model = None
best_inliers = []

for _ in range(iterations):
    sample_indices = np.random.choice(len(points), sample_size, replace=False)
    sample_points = points[sample_indices]

    x1, y1 = sample_points[0]
    x2, y2 = sample_points[1]
    if x2 - x1 != 0:
        model_slope = (y2 - y1) / (x2 - x1)
        model_intercept = y1 - model_slope * x1

        distances = np.abs(points[:, 1] - model_slope * points[:, 0] - model_intercept)

        inliers = points[distances < inlier_threshold]
        if len(inliers) > len(best_inliers):
            best_inliers = inliers
            best_model = (model_slope, model_intercept)

x_inliers = best_inliers[:, 0]
y_inliers = best_inliers[:, 1]
m, c = np.polyfit(x_inliers, y_inliers, 1)

angle_radians = np.arctan(m)
angle_degrees = np.degrees(angle_radians)
fitted_y = m * x + c

print (f'Estimated angle of the crop field with RANSEC fitting: {angle_degrees:.2f} degrees')
```

```
Estimated angle of the crop field with RANSEC fitting: -50.70 degrees
```

Question 12

The RANSAC method is recommended over standard least-squares-fit procedures because it is more resilient to outliers and noise in the data. It accomplishes this by iteratively fitting models to subsets of the data, which allows it to remove outliers while providing accurate parameter estimation. RANSAC is adaptable in model fitting, accommodating models other than linear ones. It adjusts to various data properties and is appropriate for use in real-world scenarios. However, it can be computationally costly and does not guarantee finding the globally best solution.

Git Hub Link : https://github.com/Sandeepa0/Image-Processing.git