EN3160 Assignment 2- Fitting

Instructed by Dr. Ranga Rodrigo Done by: Rajapaksha I.P.D.D.- 210503H

# GitHub : <https://github.com/DinethraDivanjana2001/EN3166-Image-processing-and-machine-vision/tree/main/A02%20-%20Fitting>

## Question 01-Blob Detection in a Sunflower Field Image

First, I converted the sunflower field image to grayscale to simplify blob detection. The algorithm applies Gaussian blur followed by a Laplacian filter to detect circular shapes at multiple scales.

* **Gaussian Blur:** Smooths the image, reducing noise.
* **Laplacian Filter:** Identifies circular edges by detecting intensity changes.

I varied the scale values (σ) between 1.0 and 4.0 in 8 equal steps**,** as larger scales can capture larger structures, while smaller scales highlight finer details.

The range of **σ values** used was from 1.0 to 4.0

A screen shot of a computer program

Description automatically generatedA field of flowers with yellow circles

Description automatically generated

Figure 2 - Gaussian-Laplacian Transformation

=

A computer screen with text

Description automatically generated

Figure 4 - detects blobs by applying the Gaussian-Laplacian transformation across scales and extracting circles from contours

Figure -Detected Circles

**A black screen with white text

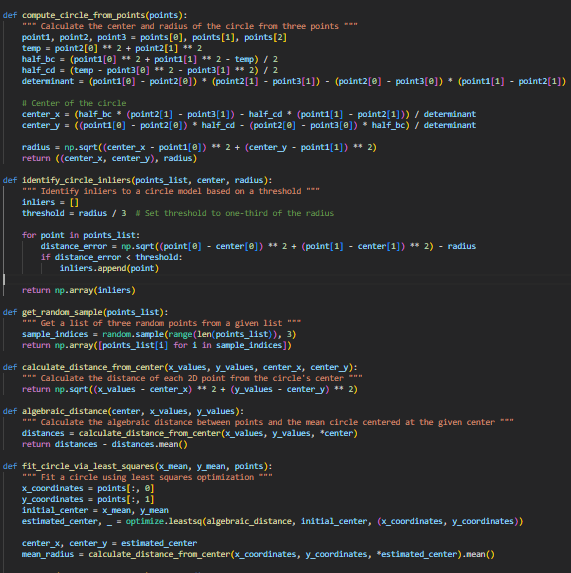
Description automatically generated**

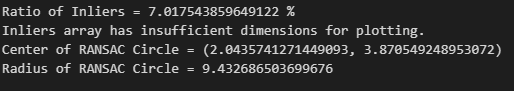
Figure - parameters of the largest circle

## 2.0. Question - 02

Implemented the RANSAC algorithm to estimate a line and a circle from a noisy set of points.

For line estimation, I constrained the line parameters [a,b][a, b][a,b]. I selected the error as the perpendicular distance from points to the line and defined a threshold for consensus, ensuring only points within this distance were considered in the fitting process. After obtaining the best-fit line, I subtracted the consensus points to analyse the remaining points for circle fitting. Using RANSAC again, I calculated the radial error for circle fitting and set a threshold to determine consensus points for the circle. Finally, I visualized the results by plotting the point set, the estimated line and circle, their inliers, and the sample points used for their estimation, presenting a comprehensive view of the fitting results.

* The calculate\_line\_equation function computes normalized line coefficients from two points; total\_least\_squares\_error calculates the sum of squared errors for a line model; constraint\_function enforces the normalization constraint a2+b2=1a^2 + b^2 = 1a2+b2=1; and identify\_inliers determines inliers by comparing the absolute error to a threshold.(figure 5)
* The compute\_circle\_from\_points function calculates the center and radius of a circle given three points; identify\_circle\_inliers identifies inliers based on the radial error being below one-third of the circle's radius; get\_random\_sample selects three random points from a list for circle fitting; calculate\_distance\_from\_center computes the distance of points from the circle’s center; algebraic\_distance determines the deviation of point distances from the mean radius; and fit\_circle\_via\_least\_squares fits a circle using least-squares optimization to estimate the center and radius.( figure 6)



A computer screen shot of a program code

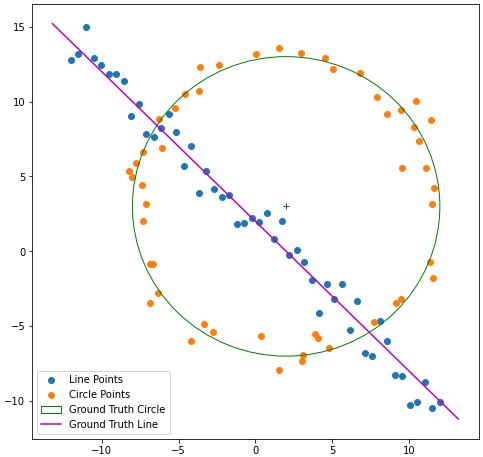
Description automatically generated

Figure 6

Figure 5

A diagram of a circle with blue dots

Description automatically generatedA diagram of a line with colored dots

Description automatically generated with medium confidence

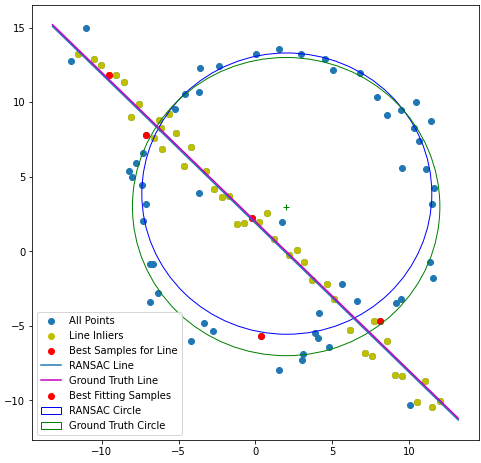
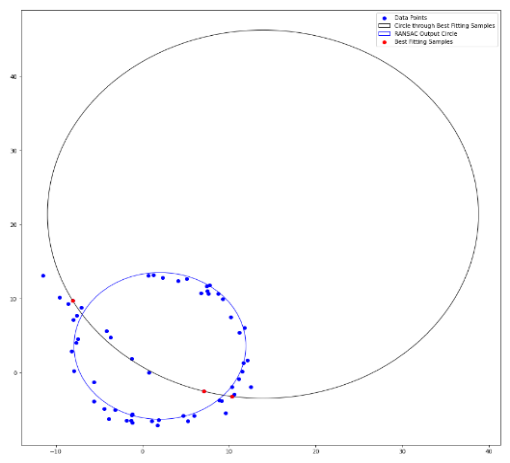


Figure 10- RANSAC Circle Fitting Function

Figure 11 - Revised Plotting Code

Figure 9 - Detected Outliers and Fitted Circle Visualization

Figure 8- RANSAC Line Fitting

Figure 7 - Original image

* Fitting a circle first on predominantly linear data can lead to poor results because the circle will try to capture all points, even those that follow a line, resulting in a large radius and a poorly positioned center. Many line points might be misclassified as outliers, lowering the inlier ratio and weakening RANSAC performance. Additionally, since a circle requires more parameters than a line, fitting it first may lead to overfitting, reducing RANSAC's effectiveness.

## 3.0. Question 03

This task overlays a flag onto an architectural image using homography by first loading both images and selecting four points on the surface for placement. These points are used to compute a homography matrix that aligns the flag's corners with them. The flag is then warped to match the architectural perspective, and the images are blended using an `alpha` parameter for transparency, resulting in a seamless integration of the flag into the scene.

# 4.0. Question – 04

Figure 13

Figure 12

Figure 11

The implementation stitches img1.ppm onto img5.ppm by first detecting and matching keypoints using the SIFT (Scale-Invariant Feature Transform) algorithm, applying Lowe's ratio test to filter good matches. Next, it employs a custom RANSAC (Random Sample Consensus) method to compute the best homography matrix based on the matched points, ensuring robust estimation. Finally, the homography is used to warp img1.ppm and seamlessly stitch it onto img5.ppm. This workflow effectively combines feature matching, homography computation, and image warping to create a cohesive stitched image.

A close-up of a colorful wall

Description automatically generated

A colorful string in a wall

Description automatically generated with medium confidence

Figure 16- Output

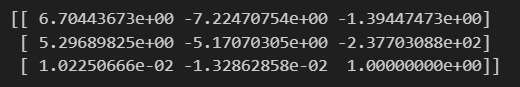
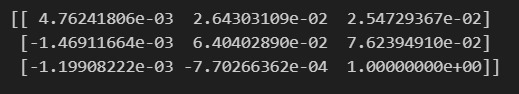


Figure 14- Homography Matrix from RANSAC

Figure 15 Homography Matrix from RANSAC:

