EN3160 – Image Processing and Machine Vision

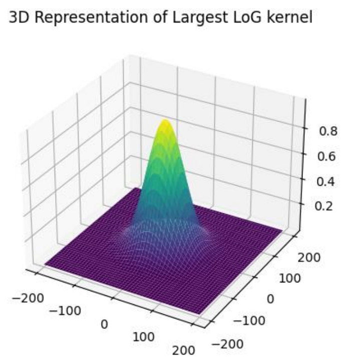
Assignment 2 - Fitting and Alignment

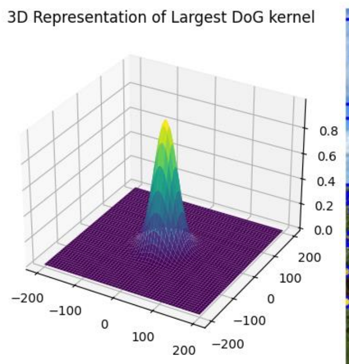
Name: M.M.H.H.B Gallella
Index No.: 210174X
Submission Date: 23rd of October 2024
GitHub: https://github.com/HasithaGallella
Assignment Codes on: Google Colab

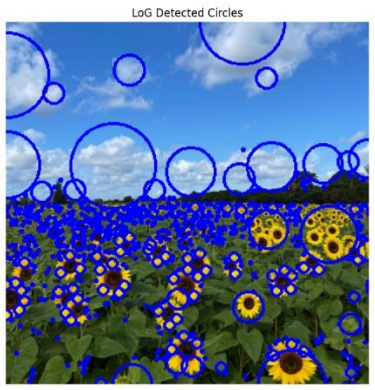
Question 1 – Sunflower field Blob Detection

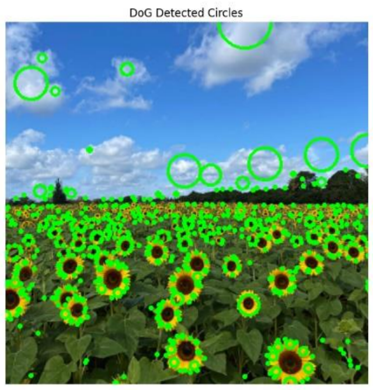
In this task, we compared the **performance** of **Laplacian of Gaussian (LoG)** with **Difference of Gaussians (DoG)** for detecting white and dark blobs at different r values. The larger blobs were successfully detected with r values of 9 and 10, whereas smaller blobs were detected with r values of 1, 2, and 3.

1.1 Light Blob detection: LoG and DoG kernels were configured to detect local maxima, effectively identifying brighter regions like sunflower petals instead sunflower centers.

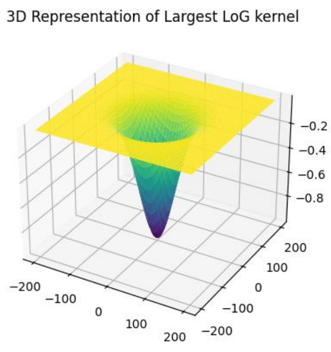


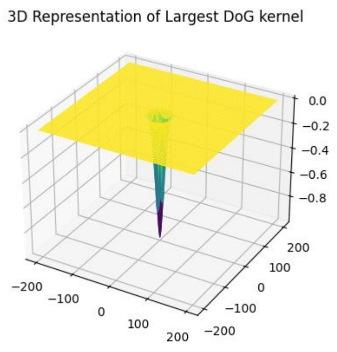


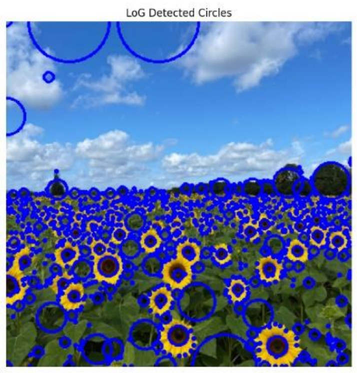


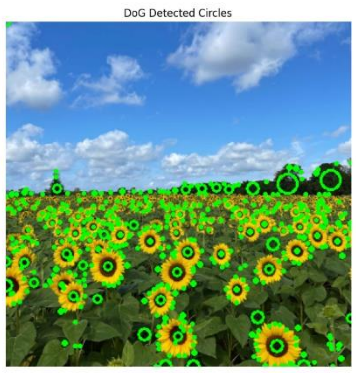


1.2 Dark Blob Detection: LoG and DoG kernels were adapted to detect local minima, focusing on darker areas such as the sunflower centers.





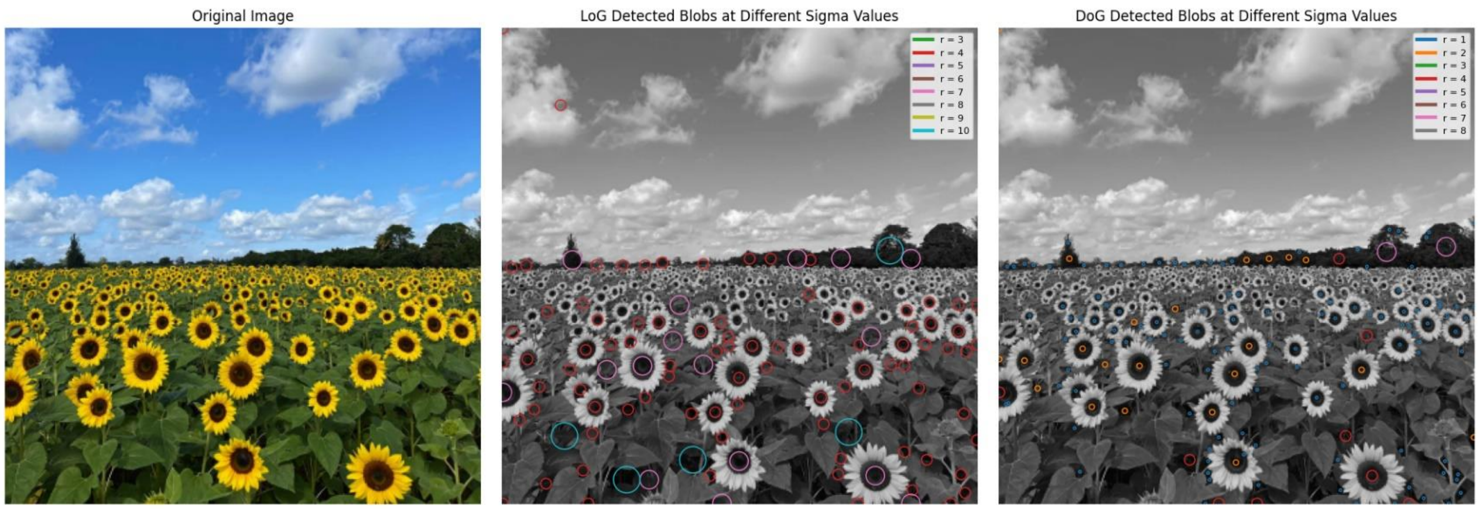




```
# Find the largest circle for LoG
if len(blobs_log) > 0:
   largest_blob_log = max(blobs_log, key=lambda b: b[2])
   x_largest_log, y_largest_log, radius_largest_log = largest_blob_log
   print(f"Largest circle using LoG: Center=({x_largest_log:.2f},
{y_largest_log:.2f}), Radius={radius_largest_log:.2f}")
else:
   print("No circles detected using LoG.")

# Find the largest circle for DoG
if len(blobs_dog) > 0:
   largest_blob_dog = max(blobs_dog, key=lambda b: b[2])
   x_largest_dog, y_largest_dog, radius_largest_dog = largest_blob_dog
   print(f"Largest circle using DoG: Center=({x_largest_dog:.2f},
{y_largest_dog:.2f}), Radius={radius_largest_dog:.2f}")
else:
   print("No circles detected using DoG.")
```

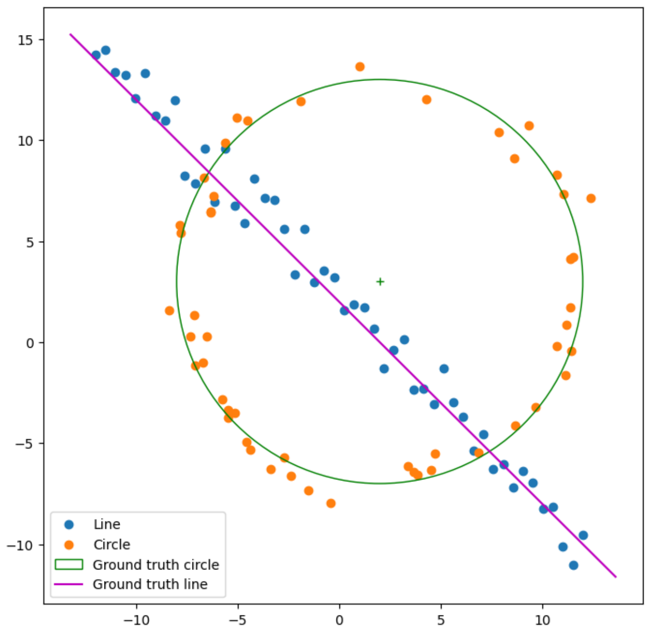
```
# Output:
Largest circle using LoG: Center=(0.00, 65.00), Radius=42.43
Largest circle using DoG: Center=(165.00, 338.00), Radius=10.49
```

Discussion: We used LoG and DoG to detect bright petals and dark centers in the sunflower image. LoG performed better for larger blobs, while DoG excelled at capturing smaller details, showing their complementary strengths in feature detection.

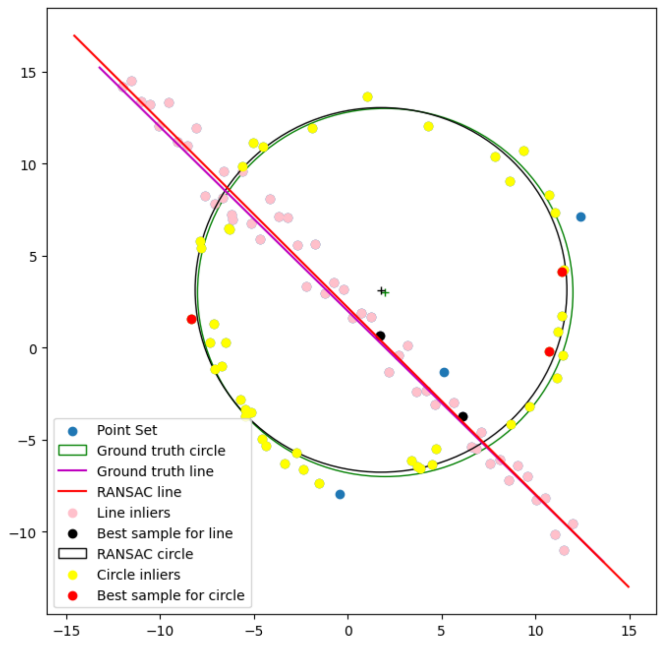


Question 2 – Line and Circle Fitting using RANSAC

```
# RANSAC algorithm for line fitting (similar logic applies for circle fitting)
def ransac_line_fitting(point_set, max_iterations, min_samples, threshold,
consensus_size):
   best_model = None
   min_error = np.inf

   for iteration in range(max_iterations):
      # Randomly select points to estimate the model
      sample_indices = np.random.choice(np.arange(len(point_set)),
size=min_samples, replace=False)
      sample_points = point_set[sample_indices]

      # Calculate initial line parameters
      model_params = compute_line_parameters(sample_points[0],
sample_points[1])

      # Determine consensus set (inliers) based on error threshold
      inliers = find_line_consensus(model_params, point_set, threshold)

      if len(inliers) >= consensus_size:
         # Refine the model using inliers
         result = fit_line_least_squares(inliers, model_params, point_set)

         # Update the best model if current model has lower error
         if result.fun < min_error:
            min_error = result.fun
            best_model = result.x

   return best_model

# Function to find consensus set (inliers) for a given line model
```



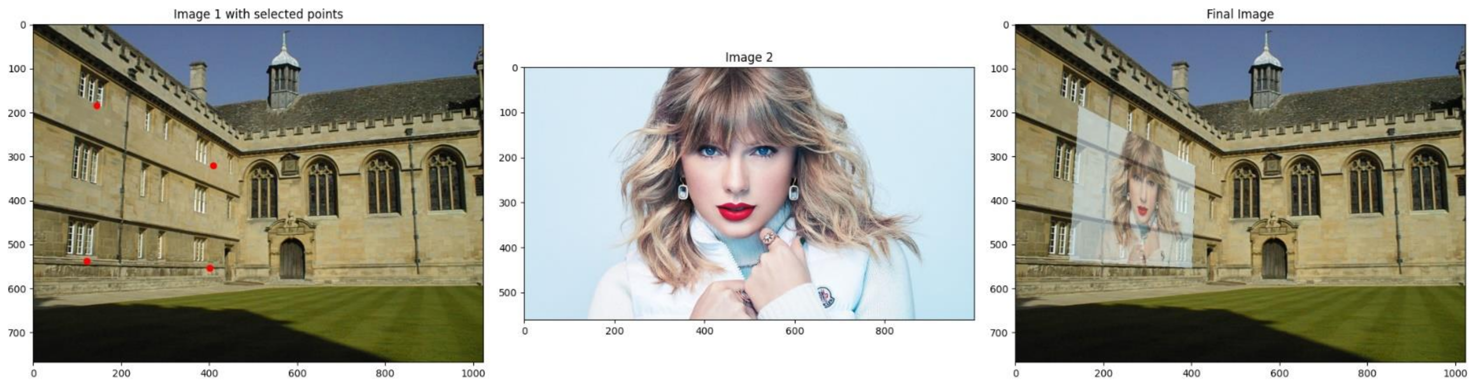
- **s = 2 for line, s = 3 for circle**
  These are the minimum number of points needed to define each shape.
- **Error threshold (t): t = 0.9 for line, t = 1.3 for circle**
  Since the noise properties were unknown, ttt was optimized through trial and error.
- **Consensus size (d = 40)**
  The original set contained 50 points each for the line and circle. Setting d=40d = 40d=40 allowed a sufficient number of inliers to be included.



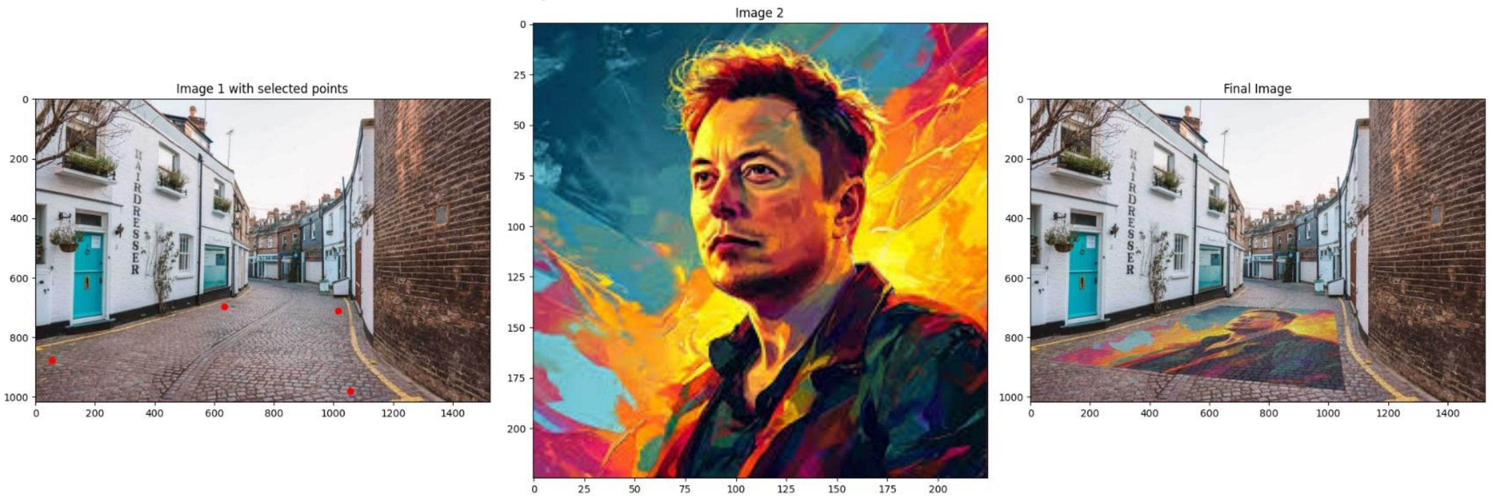
Question 3 – Superimposing an Image on Another

The goal of this task is to superimpose an image onto another image by estimating a homography between defined points, followed by blending them with weighted addition.

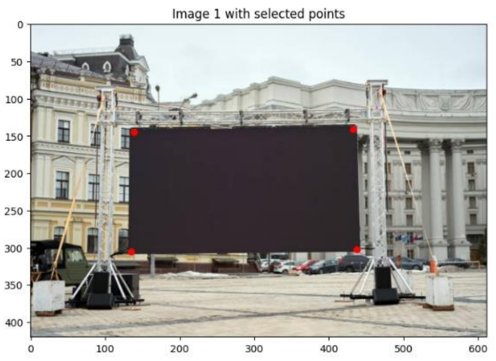
3.1 Taylor Swift superimposing on a wall

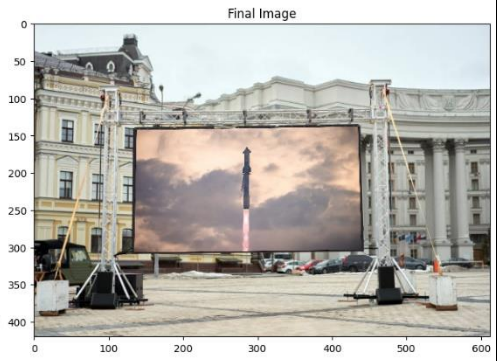


3.2 Elon Musk superimposing as a street art on a road

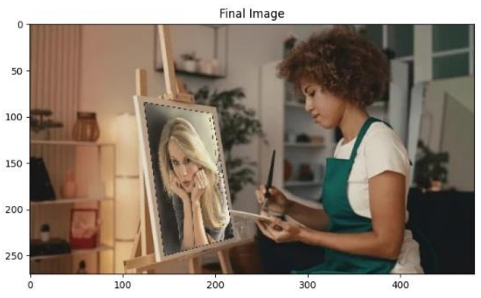


3.3 Space X - Starship launch superimposing on a screen



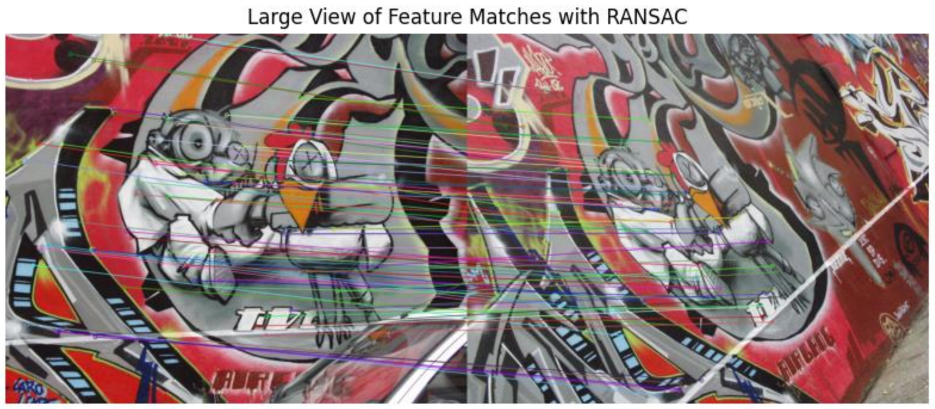


3.4 Taylor Swift stitching on a art painting



Question 4 – Image Stitching

Images 1 and 5 lacked sufficient feature matches for direct stitching. Instead, sequential homographies were computed between adjacent images, mapping image 1 stepwise to image 5 for successful alignment.



```
Computed Homography Matrix:
 [[ 6.52761162e-01  6.76929173e-01 -2.87891167e+01]
 [-1.49769351e-01  9.65158152e-01  1.49392999e+02]
 [ 4.07986271e-04 -1.63555855e-05  1.00000000e+00]]

Actual Homography Matrix from Dataset:
 [[ 6.6378505e-01  6.8003334e-01 -3.1230335e+01]
 [-1.4495500e-01  9.7128304e-01  1.4877420e+02]
 [ 4.2518504e-04 -1.3930359e-05  1.0000000e+00]]

Error between Computed and Actual Homography:
 2.5184619751388344
```