**Final Report: Image Transformations and Filtering**

**Introduction**

This project explores various image processing techniques, including geometric transformations, noise addition, filtering, edge detection and performance comparison . The images used include industrial objects, natural scenes, and geometric shapes.

**Step 1: Geometric Transformations**

Geometric transformations modify the arrangement of pixels in the image while keeping the intensity values unchanged.

**Rotation and Scaling**

The industrial object image was rotated by 45 degrees and scaled by a factor of 1.2.

rotated\_scaled\_industrial = rotate\_and\_scale(industrial\_gray, 45, 1.2)

plt.imshow(rotated\_scaled\_industrial, cmap='gray')

plt.title("Rotated and Scaled Industrial Objects")

plt.show()

**Result:**

*Image showing the industrial objects rotated and scaled.*

A collage of gears and screws

Description automatically generated

**Step 2: Noise Addition**

To simulate real-world image imperfections, Gaussian noise and Salt & Pepper noise were added to the images.

**Gaussian Noise**

Gaussian noise was added to the natural scene, modeled as a random variable with a mean of 0 and a standard deviation of 15.

noisy\_natural = add\_gaussian\_noise(natural\_gray)

plt.imshow(noisy\_natural, cmap='gray')

plt.title("Natural Scene with Gaussian Noise")

plt.show()

**Result:**

*Image of the natural scene with Gaussian noise added.*

A tall tree in front of a lake

Description automatically generated

**Salt & Pepper Noise**

Salt & Pepper noise was added to the geometric shapes image. This noise model introduces random black and white pixels.

salt\_pepper\_geometric = add\_salt\_pepper\_noise(geometric\_gray, 0.05)

plt.imshow(salt\_pepper\_geometric, cmap='gray')

plt.title("Geometric Shapes with Salt & Pepper Noise")

plt.show()

**Result:**

*Image of geometric shapes with Salt & Pepper noise.* A close-up of different shapes

Description automatically generated

**Step 3: Filtering Techniques**

After introducing noise, filtering techniques were applied to remove the noise while preserving important image features such as edges.

**Gaussian Filter (Smoothing)**

The Gaussian filter was applied to the noisy natural scene to reduce the Gaussian noise.

smoothed\_natural = apply\_gaussian\_filter(noisy\_natural)

plt.imshow(smoothed\_natural, cmap='gray')

plt.title("Smoothed Natural Scene with Gaussian Filter")

plt.show()

**Result:**

*Image showing the natural scene after applying the Gaussian filter.*

A black and white photo of a lake

Description automatically generated

**Median Filter**

A median filter was used to remove Salt & Pepper noise from the geometric shapes.

filtered\_geometric = apply\_median\_filter(salt\_pepper\_geometric)

plt.imshow(filtered\_geometric, cmap='gray')

plt.title("Filtered Geometric Shapes with Median Filter")

plt.show()

**Result:**

*Image showing the geometric shapes after applying the median filter.*

A group of different shapes

Description automatically generated

**Step 4: Edge Detection**

Edge detection is a crucial step in image analysis as it highlights the boundaries in an image.

**Sobel Edge Detection**

The Sobel operator was applied to the rotated and scaled industrial objects image to detect edges both with horizontal and vertical axes.

sobel\_industrial = sobel\_edge\_detection(rotated\_scaled\_industrial)

plt.imshow(sobel\_industrial, cmap='gray')

plt.title("Sobel Edge Detection on Industrial Objects")

plt.show()

**Result:**

*Sobel edge detection on the rotated and scaled industrial objects.*

A close-up of a graph

Description automatically generated

**Laplacian Edge Detection**

Laplacian edge detection was applied to the smoothed natural scene to capture the second derivative of the image.

laplacian\_natural = laplacian\_edge\_detection(smoothed\_natural)

plt.imshow(laplacian\_natural, cmap='gray')

plt.title("Laplacian Edge Detection on Natural Scene")

plt.show()

**Result:**

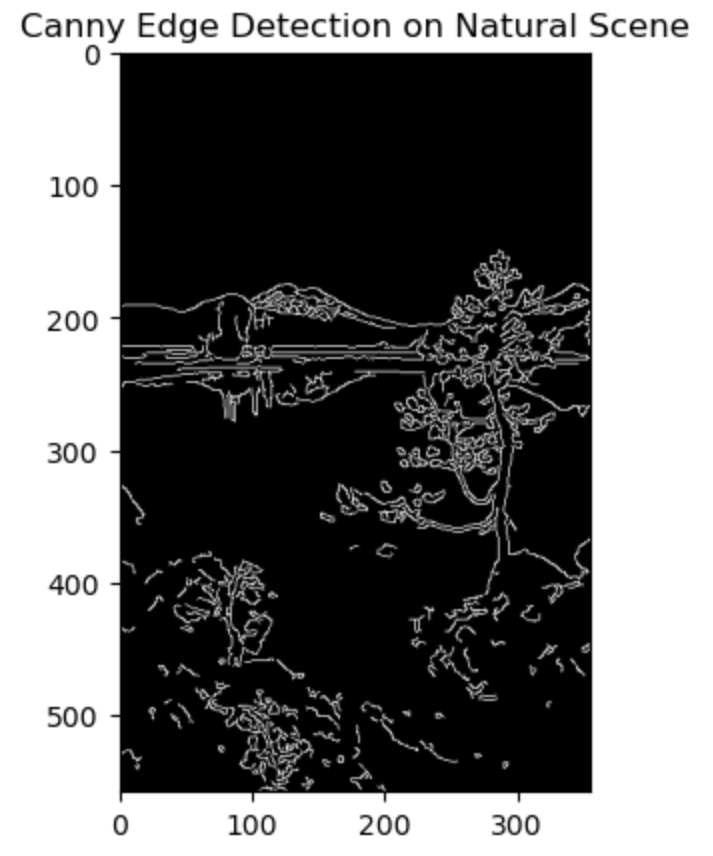
*Laplacian edge detection on the smoothed natural scene.*

A graph showing a tree and a lake

Description automatically generated

**Canny Edge Detection**  
The Canny edge detection method is known for its ability to detect both strong and weak edges in an image. It first applies Gaussian filtering, computes the gradients, and then applies non-maximum suppression. The final result is a well-refined edge-detected image.

Result: The Canny edge detection on the smoothed natural scene highlights edges more accurately, especially compared to Sobel and Laplacian methods.

****

**Step 5: Performance Comparison**  
In this section, we compare the performance of the Sobel, Laplacian, and Canny edge detection.

**Observations:**

* Sobel: Produces clear edges but may miss finer details. It’s relatively fast and performs well in low-noise environments.
* Laplacian: Captures more edges but is highly sensitive to noise, leading to additional inaccuracies in the final image.
* Canny: The best method, providing a good balance between edge sharpness and noise suppression. It’s efficient compared to Laplacian but slower than Sobel.

**Challenges**

During the project, one of the main challenges was handling the noise in the images when applying the Laplacian method, which resulted in a lot of false edges. To overcome this, I applied Gaussian filtering before edge detection to smooth the images and reduce the effect of noise. Additionally, tuning the thresholds for Canny edge detection took time, but Canny provided the best overall results.

**Conclusion**

In this project, we applied geometric transformations, noise addition, and filtering techniques to images of industrial objects, natural scenes, and geometric shapes. Geometric transformations included rotation and scaling, while noise addition simulated real-world imperfections. Filtering techniques such as Gaussian and median filters effectively reduced the noise. Finally, edge detection using Sobel and Laplacian methods highlighted important features and boundaries in the images.