Document Scanner and Perspective Correction
ECE501 - Digital Image Processing
Group-1

Members:

Name	Enrolment No.
Dhruvik Shah	AU2340113
Ayush Makwana	AU2340138
Divyam Thakkar	AU2340227
Jainik Patel	AU2340042

1. Objective for the Week (5/10/25 - 10/10/25)

Understand contour detection techniques for identifying document boundaries.

---

2. Tasks Completed:

- Installed and set up OpenCV in Python.
- Collected 4-6 sample document images with different lighting conditions.

Python

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from google.colab import files
import io


uploaded = files.upload()


for filename in uploaded.keys():
    # Read the image from the uploaded bytes
    img_bytes = uploaded[filename]
    img = cv2.imdecode(np.frombuffer(img_bytes, np.uint8),
cv2.IMREAD_COLOR)


    if img is None:
        print(f"Could not read image '{filename}'. Skipping.")
        continue


    scale_width = 800
```

```
    scale_height = int(img.shape[0] * scale_width / img.shape[1])

    img_resized = cv2.resize(img, (scale_width, scale_height))


    gray_img = cv2.cvtColor(img_resized, cv2.COLOR_BGR2GRAY)


    smooth_img = cv2.GaussianBlur(gray_img, (5, 5), 0)


    edge_map = cv2.Canny(smooth_img, 75, 200)


    _, binary_img = cv2.threshold(gray_img, 120, 255,
cv2.THRESH_BINARY)


    adaptive_img = cv2.adaptiveThreshold(

        gray_img, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
cv2.THRESH_BINARY, 11, 2

    )


    labels = ['Original Image', 'Grayscale', 'Gaussian Blur', 'Canny
Edges', 'Binary Threshold', 'Adaptive Threshold']

    outputs = [img_resized, gray_img, smooth_img, edge_map,
binary_img, adaptive_img]


    plt.figure(figsize=(15, 8))

    for idx in range(6):

        plt.subplot(2, 3, idx + 1)
```

```
        if len(outputs[idx].shape) == 2:
            plt.imshow(outputs[idx], cmap='gray')
        else:
            plt.imshow(cv2.cvtColor(outputs[idx],
cv2.COLOR_BGR2RGB))
        plt.title(labels[idx])
        plt.axis('off')

    plt.tight_layout()
    plt.show()

    base_filename = filename.split('.')[0]
    cv2.imwrite(f"{base_filename}_output_gray.jpg", gray_img)
    cv2.imwrite(f"{base_filename}_output_blurred.jpg", smooth_img)
    cv2.imwrite(f"{base_filename}_output_edges.jpg", edge_map)

    print(f"Image preprocessing done successfully for '{filename}'.
Edge map saved.")
```

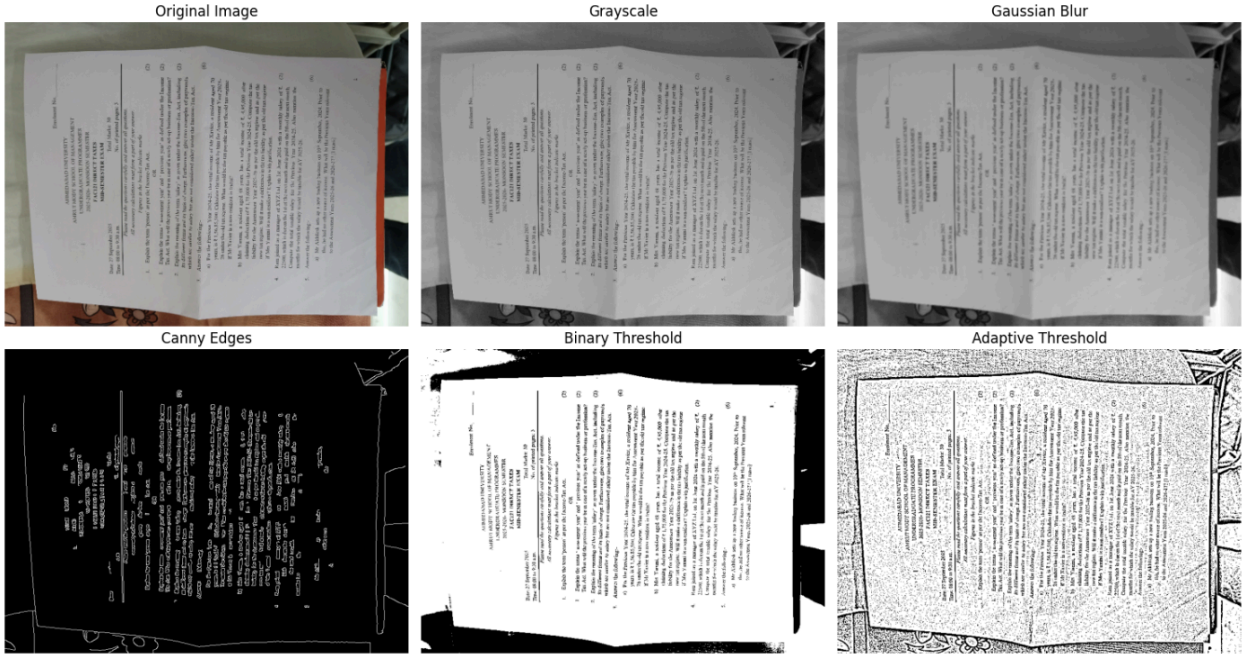
- PPT preparation for Mid-term evaluation

---

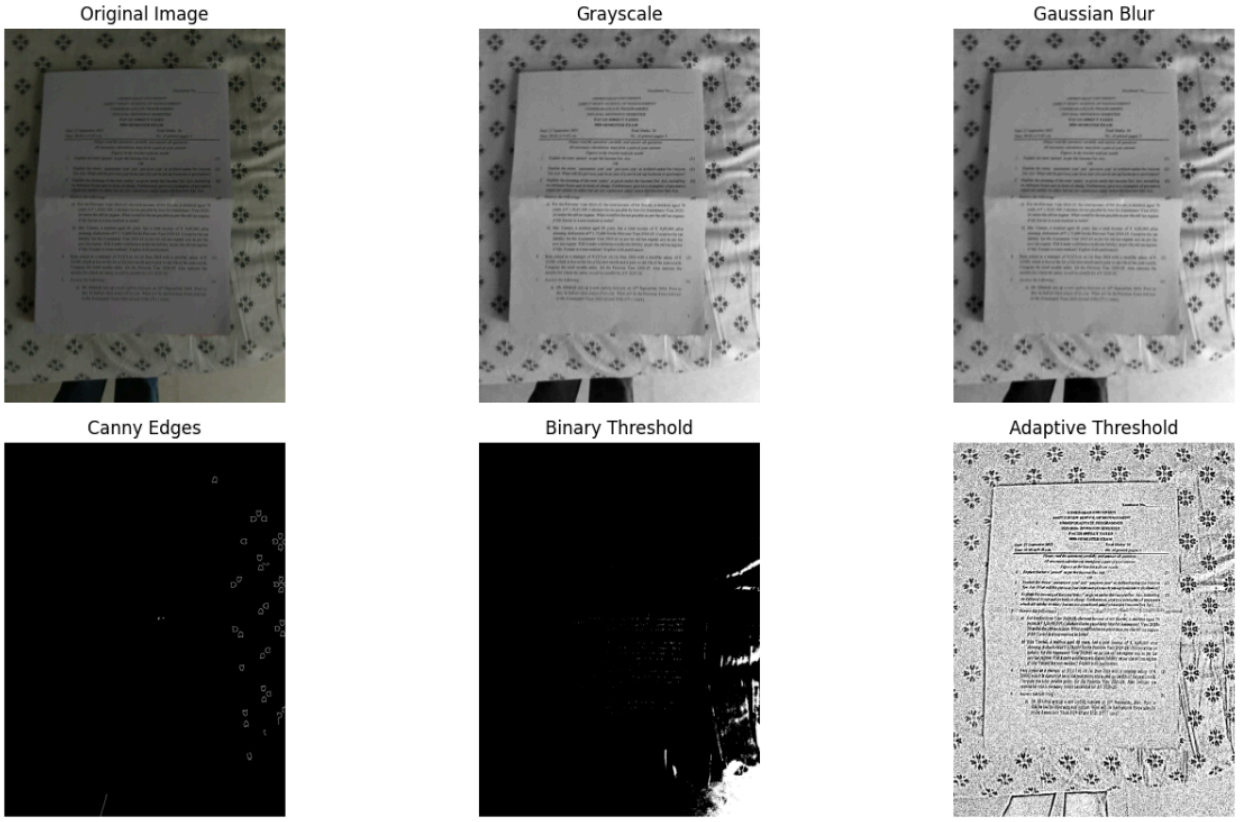
3. Observations and Results

- Successfully detected document outlines in about 70% of images as seen below:

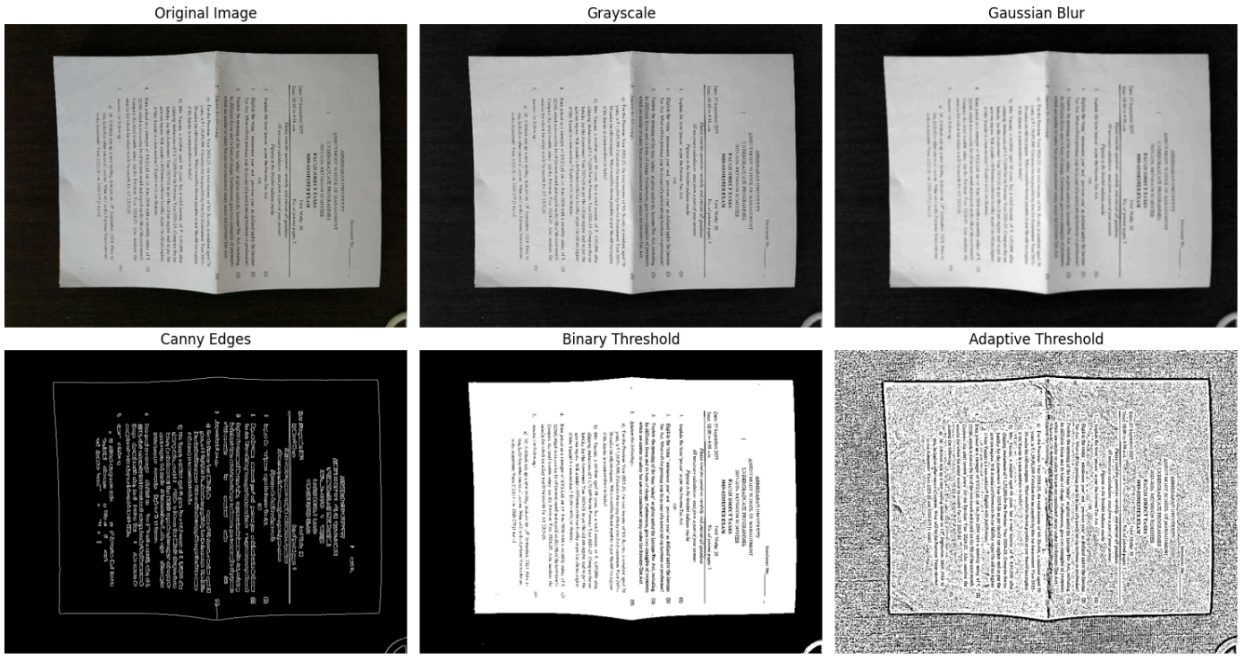
- Normal Lighting



- Low lighting



- Well lit



- Faced difficulty in detecting edges for low-contrast images in low light.

---

4. Next Week's Plan

- Implement perspective correction.
- Using adaptive thresholding for clean scanning effect .

---

5. References

- OpenCV Documentation: https://docs.opencv.org/
- OpenCV-Python Tutorials: Youtube and other sources.
- Youtube: https://youtu.be/q-gWjCgigTg?si=vk71qO7aJkUDmj7O