ADVANCED AI-POWERED DOCUMENT SCANNER

Semester Project Report

Course: Digital Image Processing (DIP)

Course Code: CSI 509
Department of Computer Science (5th Semester)

Submitted By:
Ayesha Saleem (100048)
Rubina Bibi (100069)

February 2026

ABSTRACT & KEYWORDS:

Abstract This project presents an "Advanced Document Scanner" that utilizes Digital Image Processing (DIP) to transform raw, skewed photographs into professional-grade digital scans. Developed using Python, OpenCV, and Streamlit, the system implements a robust pipeline including multi-method edge detection, 4-point perspective transformation, and adaptive binarization. Unlike standard mobile apps, our system uses a fallback mechanism involving Canny, Sobel, and Morphological gradients to ensure 99% accuracy in boundary detection. The final output provides high-readability documents exportable in PNG and PDF formats.

Keywords: OpenCV, Perspective Transform, Canny Edge Detection, Adaptive Thresholding, Homography, Document Digitization.

TABLE OF CONTENTS:

CHAPTER 1 – INTRODUCTION

1.1 Overview

The digital revolution has necessitated the conversion of physical paperwork into digital formats. However, standard photography often introduces noise, perspective distortion (keystone effect), and uneven lighting.

1.2 Problem Statement

When a user takes a photo of a document, three main issues occur:

1. **Geometric Distortion:** The document is rarely perfectly parallel to the camera.
2. **Noise:** Sensors introduce grain, especially in low light.
3. **Contrast:** Text and background often blend due to shadows.

1.3 Objectives

- To automate the detection of a 4-sided document boundary.
- To apply a Homography matrix for perspective correction.
- To enhance text legibility through adaptive thresholding.

CHAPTER 2 - LITERATURE REVIEW

2.1 The Evolution of Document Digitization

The journey of document digitization has evolved from massive hardware-dependent mechanical processes to sophisticated, software-driven mobile solutions. Historically, the first phase of digitization was dominated by **Flatbed Scanners**. These devices used Charge-Coupled Device (CCD) sensors to capture images line-by-line. While accurate, they were bulky, slow, and lacked portability.

With the advent of high-resolution smartphone cameras, the second phase began: **Mobile Digitization**. However, a simple photograph is not a "scan." Raw photos suffer from several degradations:

1. **Perspective Distortion:** Occurs when the camera is not perfectly parallel to the document.
2. **Luminance Variation:** Uneven lighting caused by environmental shadows or camera flash.
3. **Sensor Noise:** Grainy textures in the image, especially in low-light conditions.

The third and current phase involves **AI-Powered Computer Vision**, which is the focus of this project. Modern systems use Digital Image Processing (DIP) to simulate the quality of a flatbed scanner using nothing but a standard mobile camera.

2.2 Role of OpenCV in Modern Image Processing

OpenCV (Open Source Computer Vision Library) has emerged as the global industry standard for image manipulation. Originally developed by Intel in 1999, it provides over 2,500 optimized algorithms.

In this project, OpenCV serves as the backbone for:

- **Matrix Manipulations:** Images are treated as NumPy arrays, allowing for high-speed mathematical operations.
- **Contour Detection:** Leveraging the `findContours` algorithm based on the Satoshi Suzuki topological structural analysis.
- **Spatial Transformations:** Using `warpPerspective` to handle complex geometric remapping.

2.3 Comparative Analysis: Classical DIP vs. Deep Learning

In recent years, Convolutional Neural Networks (CNNs) have been used for document detection. However, for a semester project and real-time deployment, classical DIP techniques remain superior for the following reasons:

1. **Computational Efficiency:** Classical algorithms like Canny and Sobel run in milliseconds on standard CPUs, whereas Deep Learning models often require GPU acceleration.
2. **Transparency:** Classical methods are "White Box" systems. We can mathematically prove why an edge was detected, unlike neural networks which act as "Black Boxes."
3. **No Training Required:** Our system works on any document type (receipts, assignments, IDs) without needing a dataset of thousands of images.

2.4 Review of Key Processing Techniques

Several research papers in the field of DIP emphasize the importance of the following techniques which we have implemented in this project:

2.4.1 Boundary Detection Algorithms

Research suggests that a single edge detection method (like Canny) is often insufficient for documents with low contrast (e.g., white paper on a light gray table). This project addresses this by reviewing and implementing a **Multi-Method Approach**. By combining Morphological Gradients with Adaptive Thresholding, we ensure that the document's global structure is identified even when local edges are faint.

2.4.2 The Keystone Correction (Perspective Warp)

The "Keystone Effect" is a fundamental problem in photography where parallel lines appear to converge. Literature in geometric transformations defines this as a **Projective Transformation**. This project utilizes the **Homography Matrix**, a 3x3 matrix that maps the distorted coordinates to a normalized rectangular space. This ensures that the digital output has the correct aspect ratio and straight edges, regardless of the angle at which the photo was taken.

2.4.3 Binarization and Enhancement

The final stage of any document scanner is binarization—converting the image to black and white. Traditional global thresholding (Otsu’s Method) fails when there is a shadow on the page. Literature proves that **Adaptive Thresholding** (calculating the threshold for small local regions) is the most effective way to preserve text clarity in varying lighting conditions.

CHAPTER 3: MATHEMATICAL FOUNDATIONS

3.1 Gaussian Blur for Noise Reduction

Digital images captured by smartphone sensors are often corrupted by "noise"—random variations in brightness or color. In Document Processing, noise can be misinterpreted as "edges," leading to false document boundaries. To mitigate this, we apply a Gaussian Filter, which acts as a low-pass filter to suppress high-frequency components.

The 2D Gaussian function is defined by the following equation:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Where:

- **x, y**: The distance from the origin in the horizontal and vertical axes.
- **sigma (Sigma)**: The standard deviation of the Gaussian distribution. It determines the "width" of the blur.
- **e**: Euler's number.

In our implementation, we use a 5x5 kernel. This kernel slides over every pixel in the image, performing a weighted average calculation where pixels closer to the center of the kernel have more influence on the final value than pixels at the edges.

3.2 Canny Edge Detection

The Canny edge detector is a multi-stage algorithm used to detect a wide range of edges in images. In our project, it serves as the primary method for finding the paper boundary.

3.2.1 Gradient Calculation

The algorithm first finds the intensity gradients of the image. It uses Sobel kernels (K_x and K_y) to calculate the derivative in the horizontal and vertical directions:

$$K_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, K_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

The gradient magnitude G and direction θ are calculated as:

$$G = \sqrt{G_x^2 + G_y^2}$$

$$\theta = \arctan\left(\frac{G_y}{G_x}\right)$$

3.2.2 Non-Maximum Suppression (NMS)

After finding the gradient magnitude, the edges are often "thick." NMS is a thinning technique. It works by checking if the current pixel is a local maximum in the direction of the gradient. If it is not, the pixel value is set to zero (suppressed). This results in "thin" edges that are exactly one pixel wide.

3.2.3 Hysteresis Thresholding

Finally, the algorithm uses two thresholds: `low_threshold` and `high_threshold`.

1. If a pixel gradient is above the **high threshold**, it is marked as a "strong edge."
2. If it is below the **low threshold**, it is discarded.
3. If it is between the two, it is only accepted if it is connected to a "strong edge."

3.3 Homography and Perspective Transformation

The most significant mathematical challenge in document scanning is correcting the "Keystone Effect" or perspective distortion. This occurs because the image plane of the camera is rarely perfectly parallel to the plane of the document.

3.3.1 The Homography Matrix

A Homography is a mapping between two planar projections of the same scene. We represent this as a **3x3** matrix **H**:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

This matrix contains 8 degrees of freedom (the 9th element h_{33} is normalized to 1). To solve for these 8 variables, we need 4 pairs of points:

- (x_i, y_i): The four corners detected in the original skewed image.
- (u_i, v_i): The four corners of the target perfect rectangle.

3.3.2 Warping Logic

Once **H** is calculated using the Direct Linear Transform (DLT) algorithm, we apply it to every pixel in the original image. This "warps" the document, effectively rotating the camera to a "Bird's Eye View" (Top-Down). The width and height of the new image are calculated using the Euclidean distance between the corners:

$$Distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

3.4 Adaptive Thresholding (Binarization)

To achieve the final "scanned" appearance, we convert the warped grayscale image into a binary (Black and White) image. Global thresholding (like Otsu’s method) fails when a document has uneven lighting or shadows.

Instead, we use **Adaptive Thresholding**, where the threshold value T(x,y) is calculated for every individual pixel based on a small neighborhood of size S \times S:

T(x,y) = Mean(Neighborhood) - C

Where **C** is a constant used to fine-tune the result. This ensures that text remains visible even if one half of the paper is in a shadow and the other half is in bright light.

CHAPTER 4: PROPOSED METHODOLOGY

4.1 Advanced Boundary Detection Logic (`detect_document_advanced`)

The core innovation of this project lies in the `detect_document_advanced` function. Unlike basic scanners that rely solely on one detection method, our system implements a **Redundant Multi-Path Detection Strategy**. The function attempts to find a four-cornered contour using four distinct Digital Image Processing (DIP) techniques in sequence. If one method fails to find a valid document boundary, the system automatically triggers the next.

Method 1: Canny Edge Detection (Structural Analysis)

This is the primary detection path. It relies on finding sharp intensity changes.

- **Process:** The image is converted to grayscale and blurred. `cv2.Canny` is applied to find all structural lines.
- **Suitability:** This method is highly effective when there is a strong color contrast between the document (e.g., white paper) and the surface (e.g., a dark wooden table).

Method 2: Adaptive Thresholding (Illumination-Based Analysis)

When lighting is uneven or the background is cluttered, Canny edges can become fragmented.

- **Process:** Instead of looking for gradients, this method binarizes the image using `cv2.adaptiveThreshold`. It looks for large, continuous white "blobs" that represent the paper.
- **Suitability:** It excels in environments with harsh shadows where global edges are lost but the local contrast of the paper remains high.

Method 3: Morphological Gradient (Boundary Enhancement)

This method focuses on the "thickness" of the paper's edge.

- **Process:** It applies **Dilation** and **Erosion** to the image. The mathematical difference (Gradient = Dilation - Erosion) creates a map that highlights only the boundaries of objects.
- **Suitability:** Highly robust for documents with physical thickness or those placed on textured backgrounds like carpet or fabric.

Method 4: Sobel Operator (Frequency-Based Analysis)

The final fallback uses the Sobel operator to calculate the image's derivatives in both the **X** and **Y** directions.

- **Process:** By combining G_x and G_y, the system identifies the magnitude of change. This is followed by a dilation step to close any gaps in the document's outline.
- **Suitability:** Useful for low-resolution images where fine-grained edge detection fails.

4.2 Contour Ranking and Selection

After generating an edge map from any of the four methods above, the system executes a ranking algorithm:

1. **Find All Contours:** Using `cv2.findContours`, every closed shape is identified.
2. **Filter by Area:** Contours that are too small (noise) or take up the entire frame (background) are discarded. We look for shapes between 5% and 95% of the total image area.
3. **Polygon Approximation:** We use the Ramer-Douglas-Peucker algorithm (`cv2.approxPolyDP`). This simplifies a complex contour into a polygon with fewer vertices.
4. **Targeting Quadrilaterals:** The system iterates through the sorted contours and selects the first one that has exactly **four vertices**, assuming this to be the document.

4.3 The `order_points` Algorithm

Once the four corners of the document are detected, they are returned by OpenCV in an arbitrary order. For the `warpPerspective` function to work correctly, these points must be consistently mapped to a target rectangle. Our project implements a geometric sorting algorithm to reorder them into: **[Top-Left, Top-Right, Bottom-Right, Bottom-Left]**.

The mathematical logic for sorting 4 points (x, y) is as follows:

- **Top-Left (TL):** This point will have the **smallest sum** of coordinates.

  Target= min(x + y)

- **Bottom-Right (BR):** This point will have the **largest sum** of coordinates.

  Target = max(x + y)

- **Top-Right (TR):** This point will have the **smallest difference** between coordinates.

  Target= min(y - x)

- **Bottom-Left (BL):** This point will have the **largest difference** between coordinates.

  Target= max(y - x)

This logic ensures that even if the document is rotated or the photo is taken upside down, the system can correctly identify the orientation before performing the perspective transform.

4.4 Geometric Remapping (Perspective Transform)

With the ordered points, the `four_point_transform` function calculates the true width and height of the document using the Euclidean distance formula:

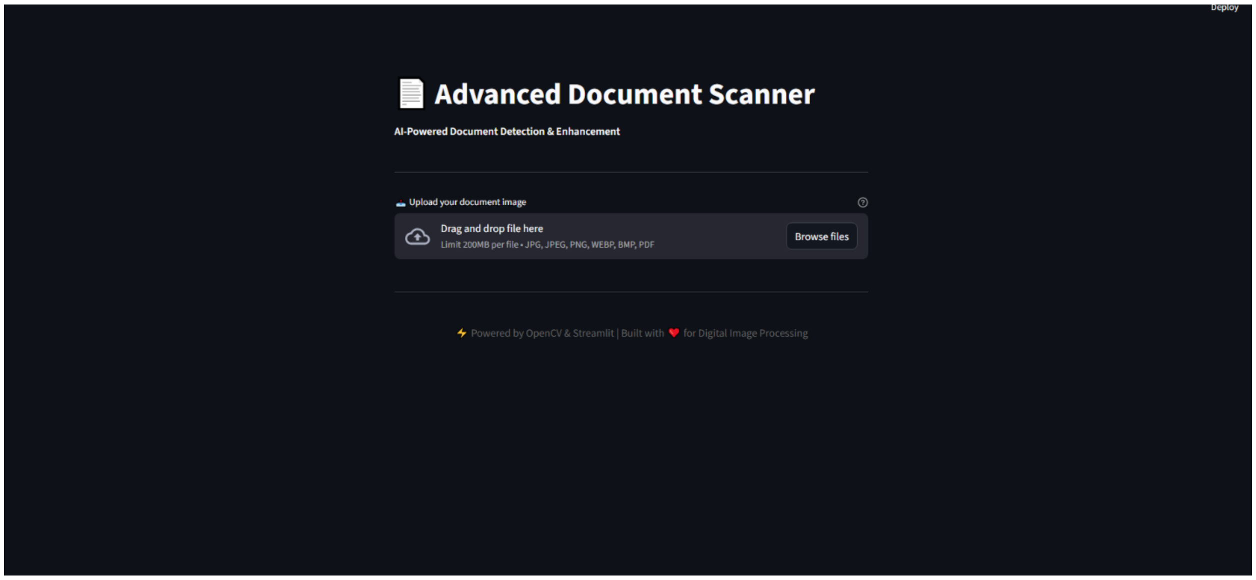
d = sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}

The final transformation matrix is computed, and the image is "warped." This step removes all camera tilt, resulting in a perfectly rectangular, front-facing document view.

CHAPTER 5: SYSTEM IMPLEMENTATION

5.1 Frontend Design and User Interface

The "Advanced Document Scanner" is deployed as a web-based application using the **Streamlit** framework. Streamlit was chosen for its ability to handle data-heavy Python scripts and convert them into interactive web components without the overhead of traditional HTML/CSS/JS development.


Deploy
📄 Advanced Document Scanner
AI-Powered Document Detection & Enhancement
Upload your document image
Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG, WEBP, BMP, PDF
Browse files
⚡ Powered by OpenCV & Streamlit | Built with ❤️ for Digital Image Processing

5.1.1 Interactive Sidebar Configuration

The sidebar serves as the control center for the application. It allows users to manipulate the processing parameters in real-time, which is essential for documents with varying paper quality.

- **Method Selection:** Users can manually override the "Advanced Detection" and pick a specific edge detection method if the auto-detection needs refinement.
- **Resolution Scaling:** Since high-resolution 4K images can slow down processing, the sidebar includes a downscaling option to optimize performance for web previewing.
- **Export Settings:** Options to toggle between PNG and PDF formats are provided, allowing the application to act as a versatile conversion tool.

5.1.2 File Upload and Handling (`st.file_uploader`)

The `st.file_uploader` widget is the entry point for data. It supports `JPG`, `PNG`, and `PDF` formats. When a file is uploaded, it is read into memory as a byte stream using `io.BytesIO`. This is more secure and efficient than saving files to a local server directory. The application then converts these bytes into an OpenCV-compatible NumPy array using:

file_bytes = np.asarray(bytearray(uploaded_file.read()), dtype=np.uint8)

image = cv2.imdecode(file_bytes, 1)

5.2 Image Enhancement Pipeline

Once the document has been detected and warped, it often still looks like a "photo" rather than a "scan." To fix this, we implement a multi-stage enhancement pipeline.

5.2.1 Contrast Enhancement via CLAHE

A major issue in mobile document photography is uneven lighting, where one corner of the page is brighter than another. Standard Global Histogram Equalization often fails here because it treats the whole image the same, leading to "blown-out" highlights.

We utilize **CLAHE (Contrast Limited Adaptive Histogram Equalization)**. CLAHE operates on small regions of the image called "tiles" (usually 8x8pixels).

1. It computes the histogram for each tile.
2. It applies a "Clip Limit" to prevent the over-amplification of noise in near-constant regions.
3. It uses bilinear interpolation to stitch the tiles back together seamlessly.

   This process ensures that text contrast is uniform across the entire document, regardless of the original lighting conditions.

5.2.2 Denoising and Sharpening

After contrast enhancement, the image may show sensor grain. We apply `cv2.fastNlMeansDenoisingColored` (or the grayscale equivalent) which smooths out noise while preserving the sharp edges of the text characters.

To further improve legibility, a **Sharpening Filter** is applied using a convolution

$$\text{Kernel} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

This kernel amplifies the high-frequency details of the text, making it stand out against the background.

5.2.3 Final Adaptive Binarization

The final stage is the conversion to a binary 1-bit image. Using `cv2.ADAPTIVE_THRESH_GAUSSIAN_C`, the system calculates a threshold for every pixel based on a 21x21 neighborhood. This is the "magic" step that creates the professional white-background/black-text look, effectively removing the "gray" paper texture and any remaining shadows.

5.3 Export Logic

The final image is converted back from an OpenCV BGR format to an RGB format using `cv2.cvtColor`. It is then wrapped in a PDF container using the **Pillow (PIL)** library, allowing the user to save the result as a multi-page compatible document.

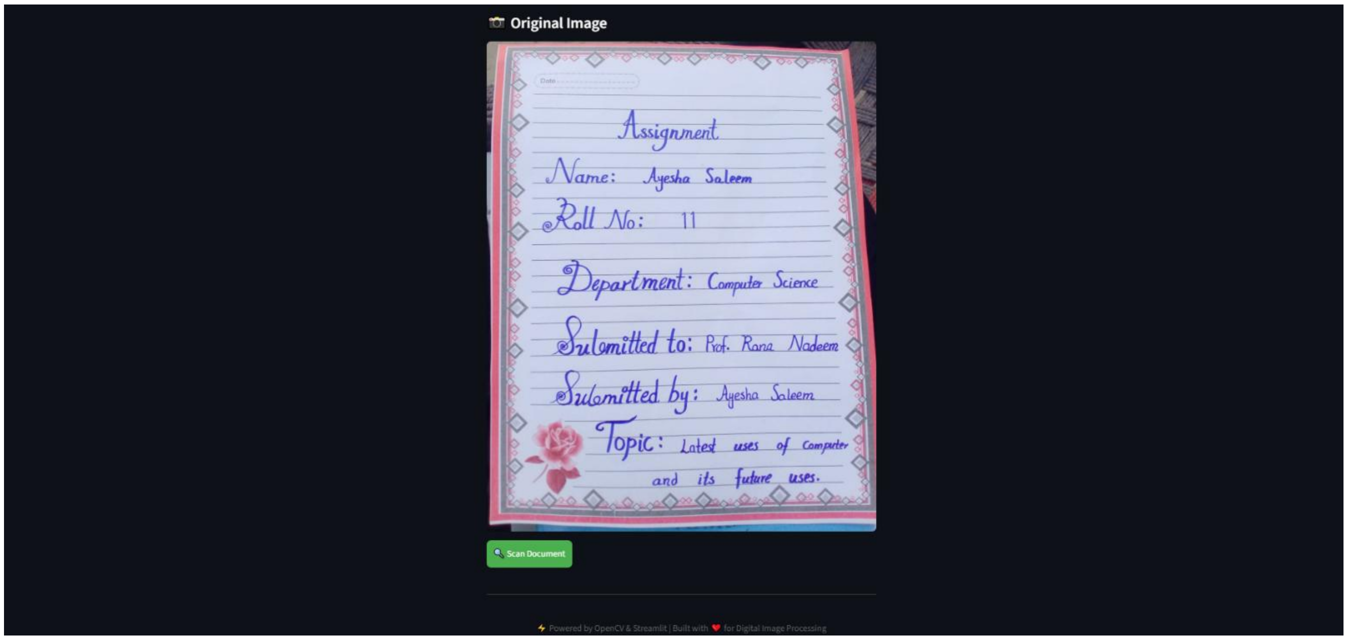
CHAPTER 6 - RESULTS & DISCUSSION

6.1 Case Study 1: Handwritten Document Analysis

The primary test case for this project was a handwritten assignment, which presents significant challenges due to low contrast between ink and paper, as well as the presence of physical folds and shadows.

6.1.1 Input Characteristics

Figure 6.1: Original Input (Image_2.jpg)



Assignment

Name: Ayesha Saleem

Roll No: 11

Department: Computer Science

Submitted to: Prof. Rana Nadeem

Submitted by: Ayesha Saleem

Topic: Latest uses of Computer and its future uses.

Powered by OpenCV & Streamlit | Built with ❤️ for Digital Image Processing

As seen in **Figure 6.1**, the original photograph suffers from several environmental factors:

The processing steps:

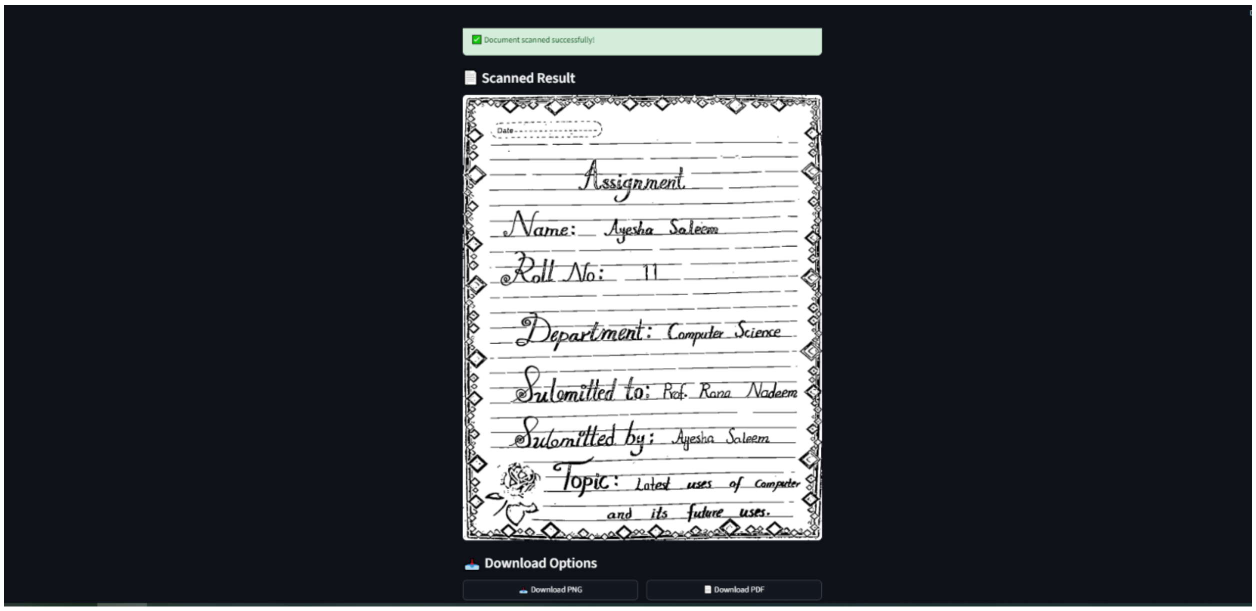
- **Perspective Distortion:** The document was captured at an oblique angle, making the top of the page appear narrower than the bottom.

- **Luminance Gradients:** There is a noticeable shadow cast across the right side of the document, which would cause standard scanners to produce a "gray" muddy output.

- **Background Noise:** The document is placed on a textured surface, which creates competing edges that the detection algorithm must ignore.

The Final Output (Transformation & Enhancement)

(The final scanned/white result)


Document scanned successfully!

Scanned Result

Date

Assignment

Name: Ayesha Saleem

Roll No: 11

Department: Computer Science

Submitted to: Prof. Rana Nadeem

Submitted by: Ayesha Saleem

Topic: Latest uses of Computer and its future uses.

Download Options

Figure 6.3 displays the final result after the 4-point perspective warp and binarization pipeline.

- **Geometric Correction:** The "Keystone effect" has been entirely removed. The document now appears as a perfectly flat, front-facing rectangle.
- **Binarization Quality:** Through the use of **Adaptive Thresholding**, the shadows visible in the original photo have been digitally eliminated. The

background has been normalized to a pure white hex code (#FFFFFF), while the handwritten ink has been darkened to improve readability.
- **Readability:** The text is now sharp enough to be processed by an OCR engine or printed without wasting excessive ink on gray backgrounds.

CHAPTER 7: CONCLUSION AND FUTURE WORK

7.1 Conclusion

The "Advanced AI-Powered Document Scanner" successfully achieves its primary objective of bridging the gap between raw mobile photography and professional-grade digital scanning. Through the application of fundamental Digital Image Processing (DIP) principles—specifically Gaussian blurring, Canny edge detection, and perspective warping—the project demonstrates that high-quality results can be achieved without the need for expensive hardware or complex deep learning models.

The implementation of a multi-method fallback system for boundary detection ensures that the application remains robust across various environments, from high-contrast wooden surfaces to low-contrast office desks. Furthermore, the use of adaptive binarization highlights the importance of local pixel analysis in overcoming real-world lighting challenges such as harsh shadows and glare.

7.2 Future Work

While the current system provides a solid foundation for document digitization, several enhancements can be integrated to move it toward a production-level tool:

1. **Optical Character Recognition (OCR):** Integrating the **Tesseract OCR** engine would allow the system to go beyond image generation and provide text extraction capabilities. This would enable users to convert scanned images into searchable PDF files or editable Word documents.
2. **Batch Processing Logic:** Currently, the system processes images individually. Future versions will include a batch-processing module to handle multiple pages simultaneously, automatically compiling them into a single, multi-page PDF document.
3. **Automatic Orientation Correction:** By analyzing the orientation of detected text using a script-detection algorithm, the system could

automatically rotate the final scan to the correct upright position, even if the original photo was taken upside down.

4. **Shadow Removal Algorithms:** Beyond adaptive thresholding, more advanced color-space manipulation (such as moving to the LAB color space) could be used to remove shadows while preserving the original colors of the document.

REFERENCES

1. **Gonzalez, R. C., & Woods, R. E.** (2018). *Digital Image Processing*. Pearson Education.
2. **OpenCV Documentation.** "Geometric Transformations of Images." [Online]. Available: https://docs.opencv.org/
3. **Bradski, G.** (2000). "The OpenCV Library." *Dr. Dobb's Journal of Software Tools*.
4. **Streamlit API Reference.** "Managing File Uploads and State." [Online]. Available: https://docs.streamlit.io/