BBM 418 - Computer Vision Laboratory

Assignment 1 Report

Melike Akkaya
2210356124

Introduction

In the age of digital documentation, the capture of documents on paper using mobile devices has become commonplace. However, these images often suffer from perspective distortions, folds, and other geometric inconsistencies that hinder automated processing and readability. This project focuses on addressing these issues by implementing a perspective correction system using edge detection, line fitting, and geometric transformations.

The goal of this assignment is to detect and correct distortions in document images by identifying their structural boundaries and transforming them into a front-facing view. To achieve this, two core techniques are used: Hough Transform for detecting line structures in edge-detected images, and RANSAC for robust line fitting in the presence of noise and outliers. Once the quadrilateral shape of the document is estimated, geometric transformations were applied to correct its perspective.

The effectiveness of my method is evaluated using the Structural Similarity Index, which quantitatively compares my corrected outputs with their corresponding ground truth images.

Method

Hough Transform

The **Hough Transform** is a voting-based technique used to identify straight lines in an image. It converts each edge pixel from the Cartesian image space (x, y) into a parameter space (ρ, θ), where:

$$\rho = x \cdot \cos(\theta) + y \cdot \sin(\theta) \tag{1}$$

where:

- ρ is the perpendicular distance from the origin to the line.
- θ is the angle perpendicular to the origin of the line.

To define the parameter space:

- The **maximum possible distance** is calculated as:

  $$diagonal = \sqrt{H^2 + W^2} \tag{2}$$

  where H and W are the height and width of the image, respectively.

- The range for ρ is set as:

  $$\rho \in [-diagonal, diagonal] \tag{3}$$

  ensuring that all possible line distances in the image are covered.

- The range for θ is:

  $$\theta \in [0°, 180°] \tag{4}$$

  which is converted to radians.

Each edge pixel **votes** for all possible values of (ρ, θ) using the above equation. These votes are accumulated in a 2D matrix called the **accumulator**. The peaks in this matrix correspond to potential straight lines in the original image. This peaks are determined with the help of a threshold value.

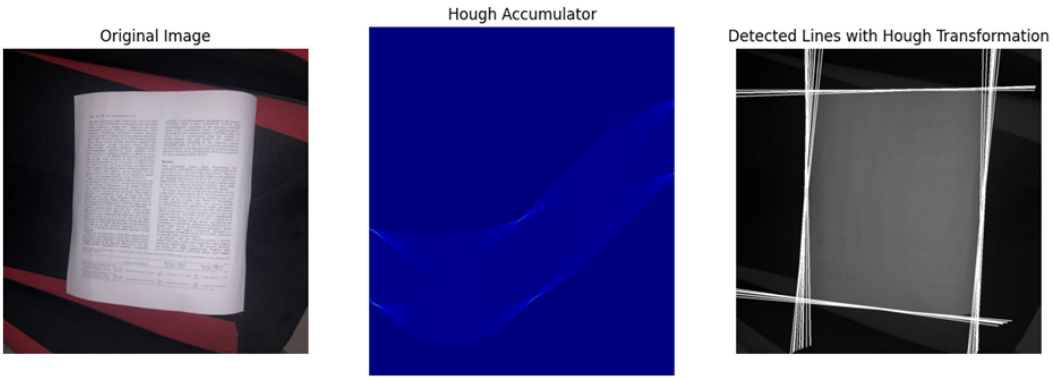

Figure 1: Example Application of Hough Transform in Line Detection

Figure 1 shows the use of the Hough Transform for line detection in a document image. The first panel shows a distorted image of a page. The middle panel visualizes the parameter space where each point represents a potential line in the image; brighter points indicate higher votes for potential line configurations. These are created by transforming edge points in the image space into sinusoidal curves in the parameter space. The last panel shows the result of the Hough Transform where the most prominent lines are detected and superimposed on the original image.

RANSAC

The Hough Transform may produce multiple noisy or redundant lines. To address this issue, a variant of **RANSAC** is used, which operates as follows:

1. Randomly select a candidate line as a model.
2. Identify other similar lines, within the defined thresholds for ρ and θ.
3. If a model has enough **inliers**, retain it and discard the inliers from future iterations.
4. Repeat the process until a set number of lines are detected or the pool of candidates is exhausted.

This method clusters similar lines and filters out spurious detections, producing a cleaner and more consistent set of document edge lines.

Figure 2: Example Application of RANSAC after Hough Transform

Figure 2 shows using RANSAC and Hough Transform together for line detection The first panel displays the original image of the distorted page. The second panel illustrates the initial results using the Hough Transform. The third panel shows the output after applying RANSAC on the result of Hough Transform.

Geometric Transformations

Finding the Intersections

The filtered lines from the RANSAC process are first converted from their **polar representation** (θ, ρ) into **Cartesian line equations** in general form:

$$Ax + By + C = 0 \tag{1}$$

where:

- $A = \cos(\theta)$,
- $B = \sin(\theta)$,
- $C = -\rho$.

Each line is represented by the coefficients (A, B, C), defining the orientation and position of the line in the 2D image plane.

To find the intersection point of two lines, $L_1 : A_1x + B_1y + C_1 = 0$ and $L_2 : A_2x + B_2y + C_2 = 0$, the system of linear equations is solved:

$$\begin{cases} A_1x + B_1y + C_1 = 0 \\ A_2x + B_2y + C_2 = 0 \end{cases} \tag{2}$$

Using the determinant method, solution for (x, y) is calculated as:

$$\text{denominator} = A_1B_2 - A_2B_1 \tag{3}$$

If the denominator is close to zero, the lines are parallel or overlapping, indicating that the system of equations does not have a unique solution. This means the lines have the same or proportional direction vectors and thus do not intersect unless they are the same line.

If the lines are not parallel, the intersection point is:

$$x = \frac{B_1C_2 - B_2C_1}{A_1B_2 - A_2B_1}, \quad y = \frac{C_1A_2 - C_2A_1}{A_1B_2 - A_2B_1} \tag{4}$$

Controlling if Points Form a Convex Quadrilateral

This function checks whether a given set of four points forms a **convex quadrilateral**, which is crucial for applying a perspective transformation. The steps include:

1. **Centroid Calculation:** The centroid is calculated as the average point of the four corners to determine their relative positions from the center:

$$C_x = \frac{1}{4}\sum_{i=1}^{4} x_i, \quad C_y = \frac{1}{4}\sum_{i=1}^{4} y_i \tag{5}$$

2. **Angle Calculation:** The angle between the vector from the centroid to each point and the x-axis is calculated using:

$$\theta_i = \text{atan2}(y_i - C_y, x_i - C_x) \tag{6}$$

   This helps to sort the points in a consistent order.

3. **Convexity Check Using Cross Product:** For every triplet of consecutive points (O, A, B), the cross product is calculated to determine the shape's convexity:

$$\text{cross}(O, A, B) = (A_x - O_x)(B_y - O_y) - (A_y - O_y)(B_x - O_x) \tag{7}$$

   A consistent sign (all positive or all negative) across these products confirms that the quadrilateral is convex.

Ordering Points Clockwise

Before applying a perspective warp, it is essential to ensure the four corner points of a document are ordered consistently (clockwise or counterclockwise), to avoid warping in the wrong direction.

Finding the Best Quadrilateral

Although multiple quadrilaterals can be formed from selected points, choosing the largest area convex quadrilateral is typically the best practice.

Warping the Document

In this step, the goal is to transform the detected quadrilateral region into a front-facing rectangle. This process, known as perspective transformation or warping, helps eliminate distortions caused by the camera angle. This is done using a **homography matrix**, which defines a projective transformation between two planes. In the updated code, we manually compute the homography matrix using a system of linear equations, which is then used to apply the perspective transformation to the image.

The math behind it:

The transformation matrix M is a 3×3 matrix that maps input coordinates (x, y) to new coordinates (x', y'):

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

To get the actual pixel coordinates in the output image:

$$x_{\text{new}} = \frac{x'}{w'}, \quad y_{\text{new}} = \frac{y'}{w'}$$

The transformation matrix is computed using the a defined function that solves for the matrix that satisfies the equations that map the source points to the destination points. Once the homography matrix M is computed, another function applies the matrix to the original image. For each pixel in the output image, it maps the corresponding source pixel using the inverse of the homography matrix. This way, the image is warped correctly, preserving the alignment of the document's content.

Normalizing Colors

If the SSIM value is still low due to issues like lighting or poor contrast, color normalization is applied to enhance local contrast and balance brightness using the LAB color space.

Calculating SSIM Score

Finally, the quality of the corrected image is compared against the ground-truth using the Structural Similarity Index Measure.

Dataset

The dataset has the 2 folder. One contains the distorted images and other one contains digital version (ground-truth) of this images. This folder have following categories

- Curved
- Fold
- Incomplete
- Perspective
- Random
- Rotate

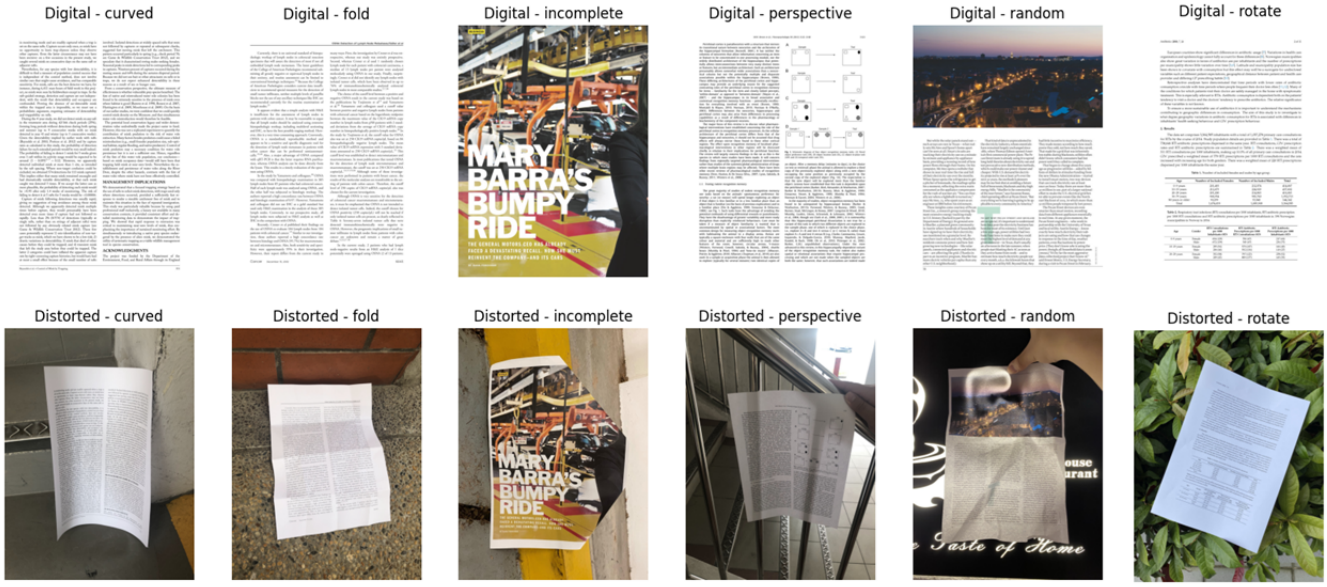

Figure 3: Example Images from all Categories

Preprocessing the Dataset

Before applying line detection and perspective correction, it is essential to preprocess the input images. This preprocessing enhances features, suppresses noise, and ensures the visibility and continuity of document edges, making later processing stages more effective and reliable.

Resize: Standardizes the size of images, which ensures uniformity in processing and helps reduce computational costs.

Increased Contrast: Enhances lighter regions such as white paper, while suppressing darker regions like shadows or background. This step is crucial for improving the visibility of the document edges.

Grayscale: Converts the image to grayscale, which simplifies the data by reducing the dimensionality from three color channels to a single intensity channel.

Gaussian Blur: Applies a Gaussian blur to smooth out the image. This reduces noise and helps in minimizing false edge detections in the subsequent stages.

Dilate: Uses dilation to bridge gaps between broken edges, which is particularly beneficial for processing folded documents as it enhances edge connectivity.

Canny Edges: Employs the Canny edge detector to extract sharp edges from the image, providing a clear boundary representation necessary for effective line detection and perspective transformations.

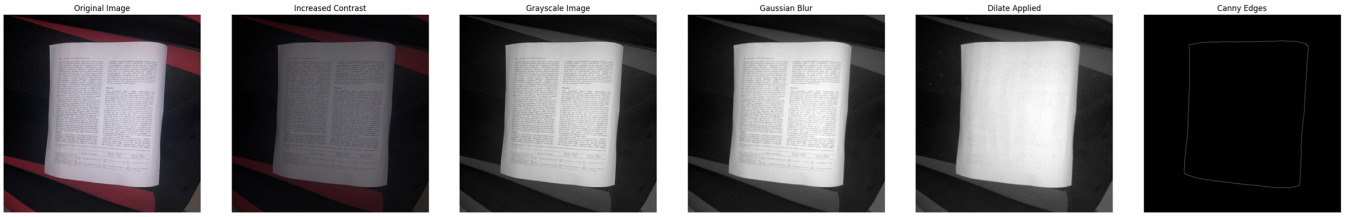

Figure 4: Example Preprocess

Virtual Results

Curved

First Experiment

In Figure 4, there is an example of the effect of preprocessing in a curved image. Also in Figure 1 and 2, the line detecting process is shown for the same image.

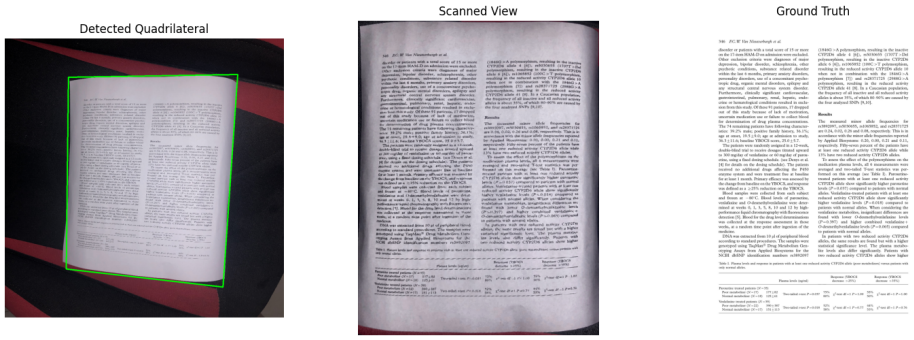

Figure 5: Example of Detected Quadrilateral and Scanned Image

Second Experiment

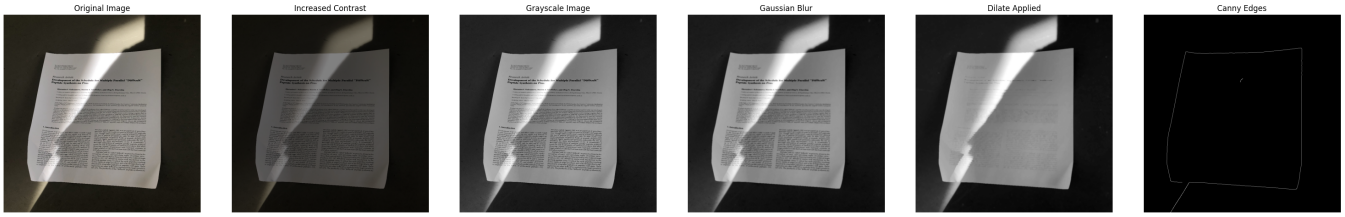

Figure 6: Example Preprocess

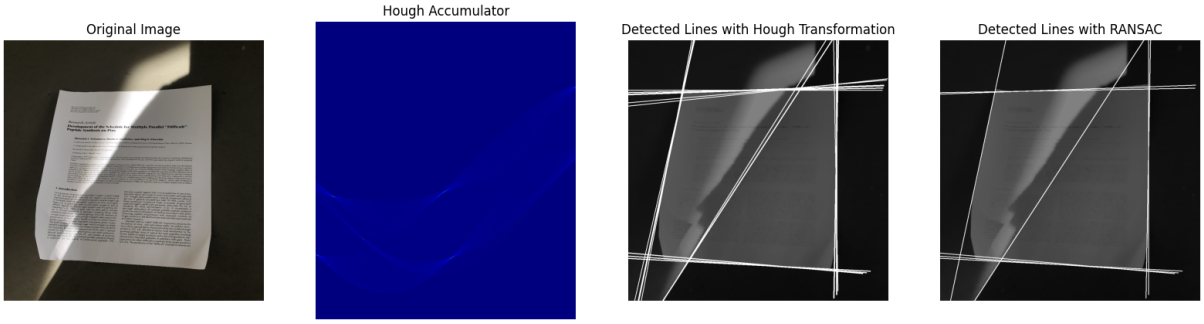

Figure 7: Example Application of RANSAC and Hough Transformation in Line Detection

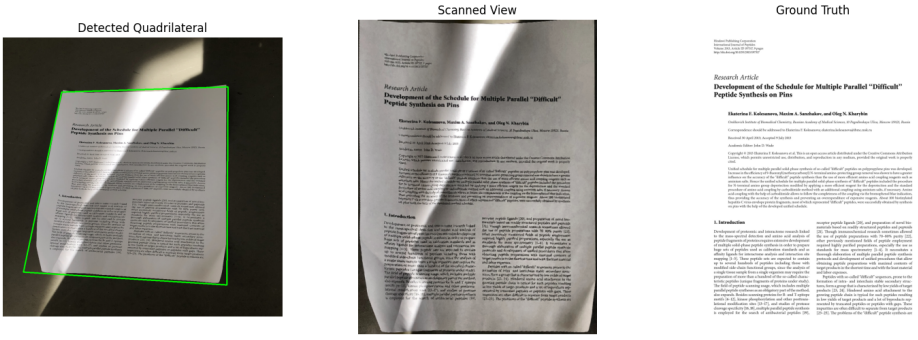

Figure 8: Example of Detected Quadrilateral and Scanned Image

Third Experiment

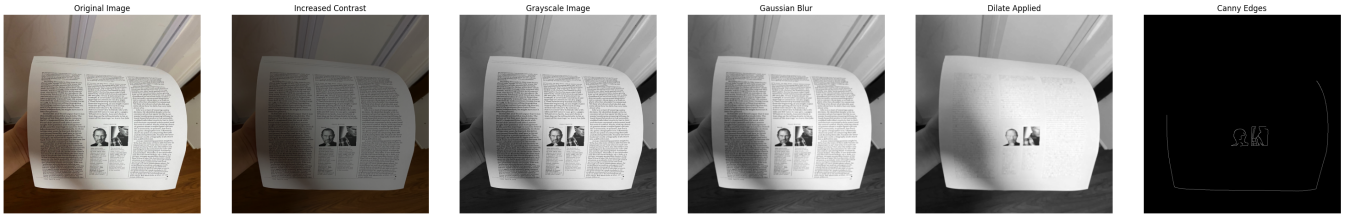

Figure 9: Example Preprocess

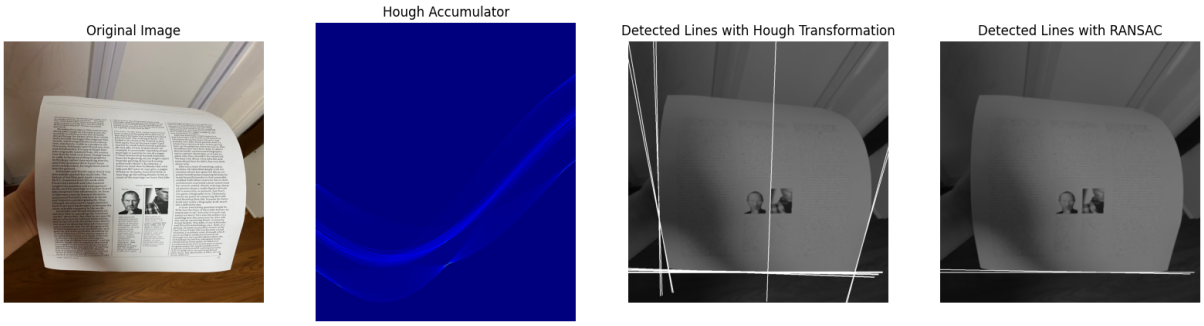

Figure 10: Example Application of RANSAC and Hough Transformation in Line Detection

In this experiment, the borders of the paper cannot be highlighted with this preprocess (Figure 9). Therefore the borders cannot be detected (Figure 10) with RANSAC and Hough Transform. During geometric transformation process, there cannot be detected a quadrilateral and cannot produce a scanned paper.

Fold

First Experiment

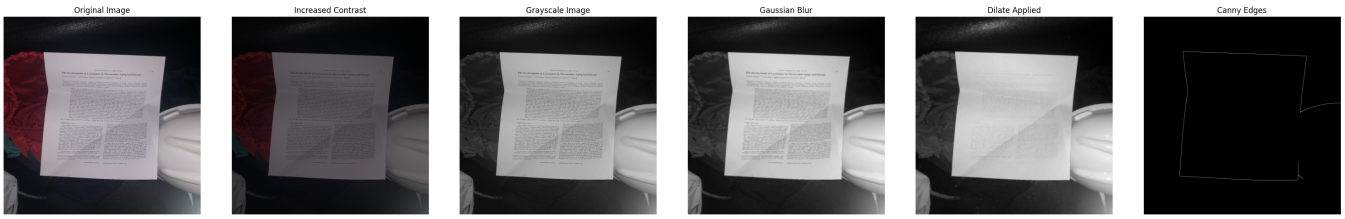

Figure 11: Example Preprocess

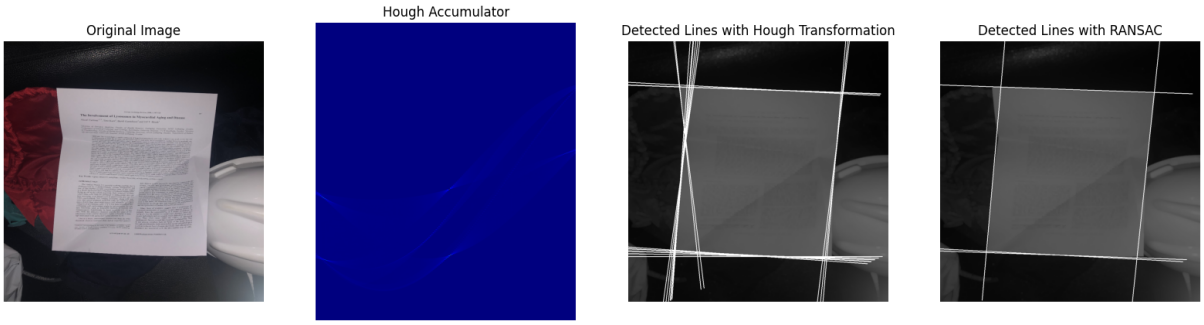

Figure 12: Example Application of RANSAC and Hough Transformation in Line Detection

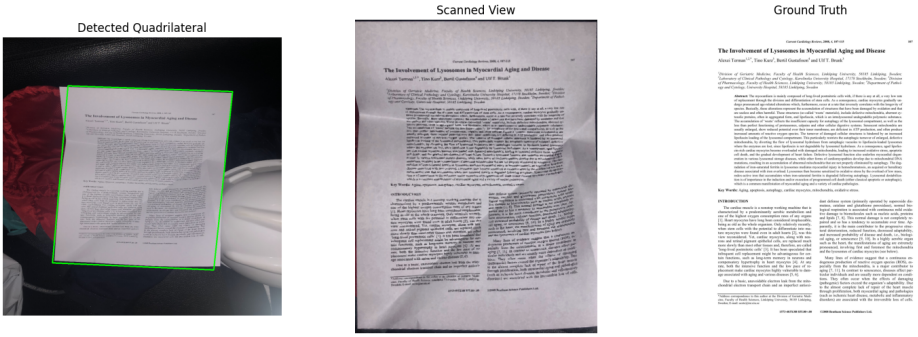

Figure 13: Example of Detected Quadrilateral and Scanned Image

Second Experiment

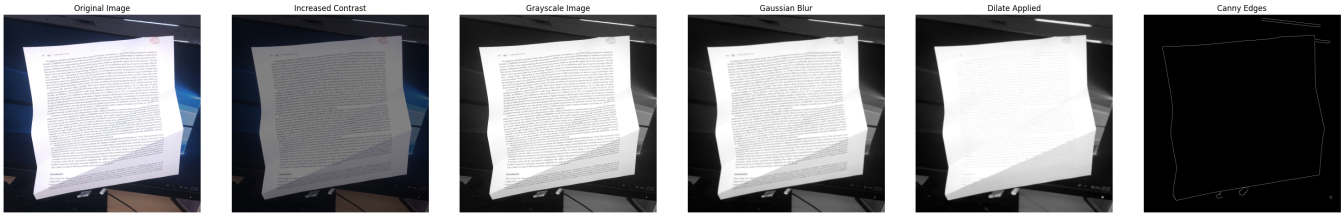

Figure 14: Example Preprocess

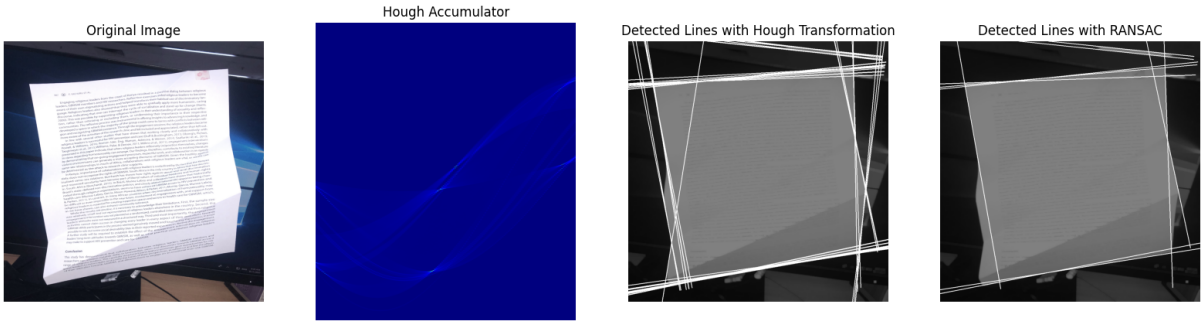

Figure 15: Example Application of RANSAC and Hough Transformation in Line Detection

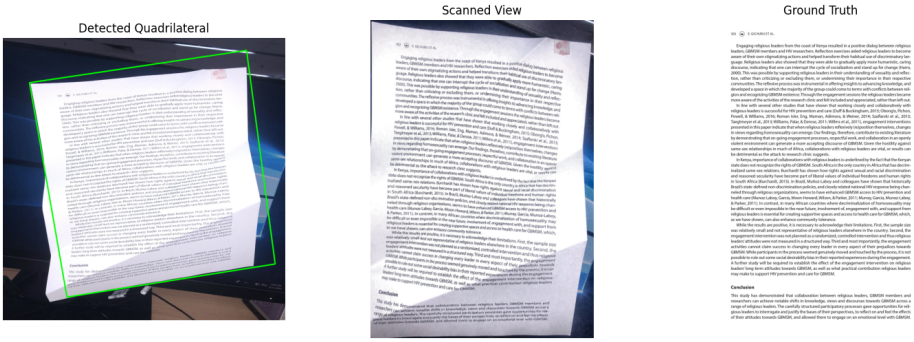

Figure 16: Example of Detected Quadrilateral and Scanned Image

Third Experiment

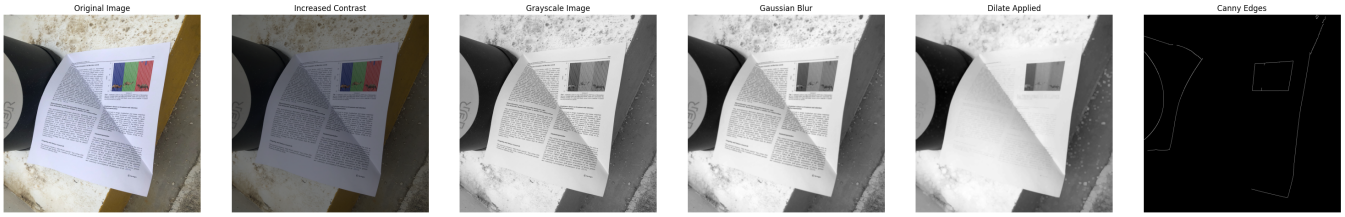

Figure 17: Example Preprocess

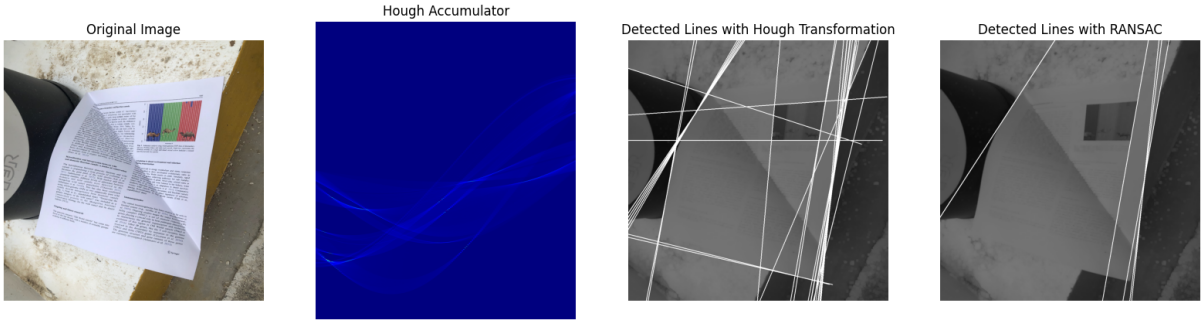

Figure 18: Example Application of RANSAC and Hough Transformation in Line Detection

In this experiment, the borders of the paper cannot be highlighted with this preprocess (Figure 17). Therefore the borders cannot be detected (Figure 18) with RANSAC and Hough Transform. During geometric transformation process, there cannot be detected a quadrilateral and cannot produce a scanned paper.

Incomplete

First Experiment

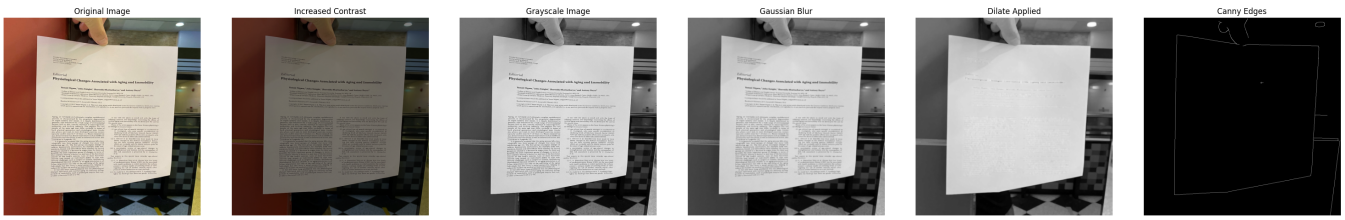

Figure 19: Example Preprocess

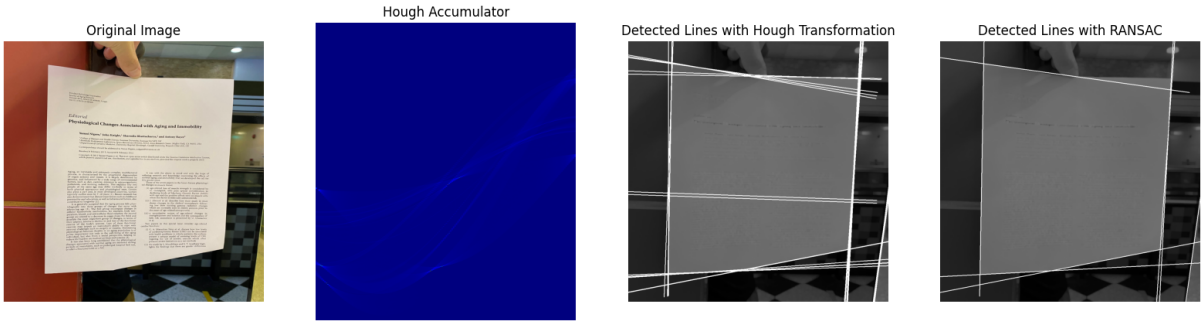

Figure 20: Example Application of RANSAC and Hough Transformation in Line Detection

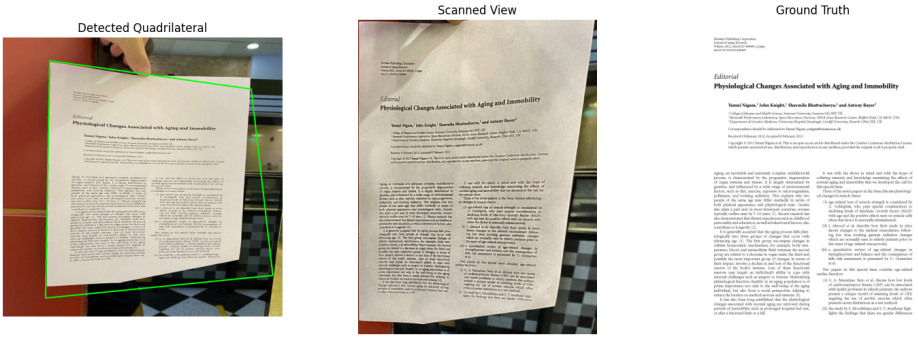

Figure 21: Example of Detected Quadrilateral and Scanned Image

Second Experiment

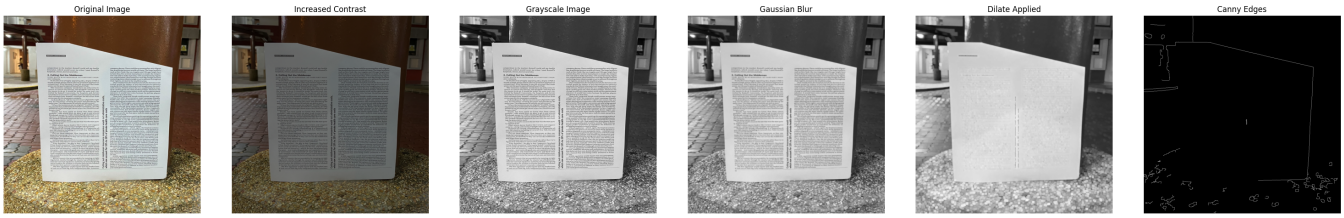

Figure 22: Example Preprocess

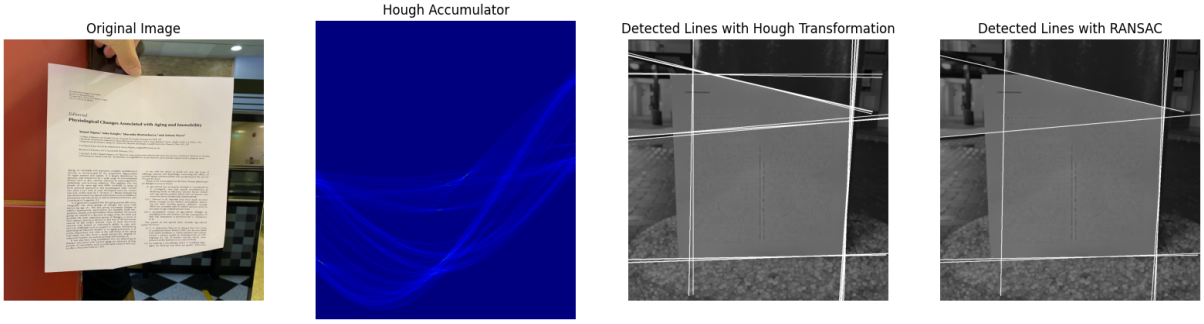

Figure 23: Example Application of RANSAC and Hough Transformation in Line Detection

Figure 24: Example of Detected Quadrilateral and Scanned Image

Third Experiment

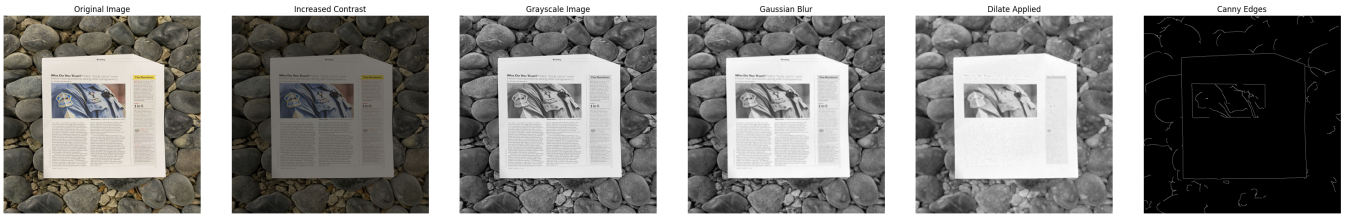

Figure 25: Example Preprocess

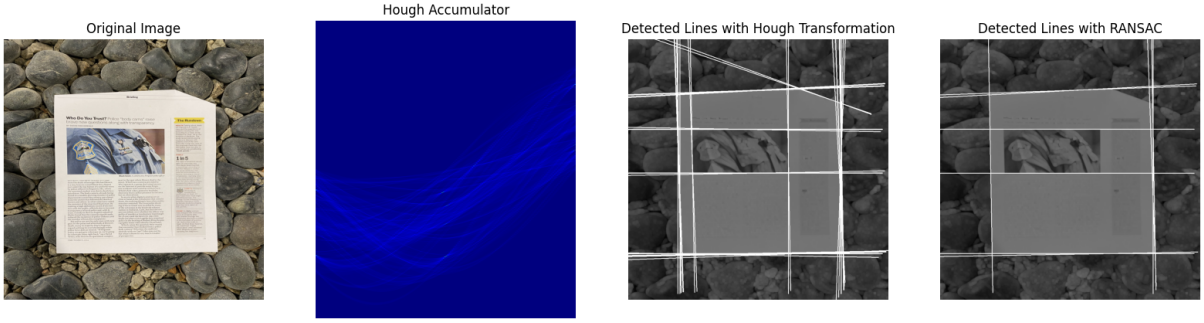

Figure 26: Example Application of RANSAC and Hough Transformation in Line Detection

Figure 27: Example of Detected Quadrilateral and Scanned Image

Perspective

First Experiment

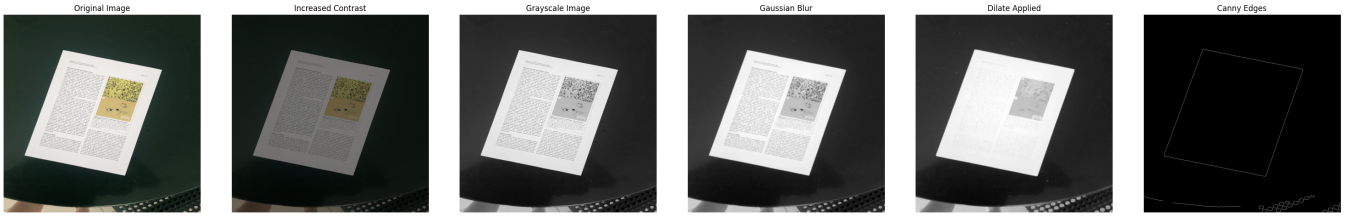

Figure 28: Example Preprocess

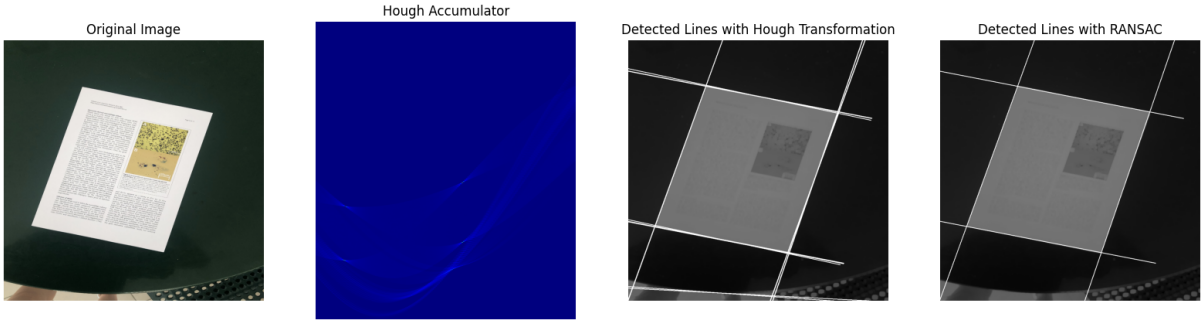

Figure 29: Example Application of RANSAC and Hough Transformation in Line Detection

Figure 30: Example of Detected Quadrilateral and Scanned Image

Second Experiment

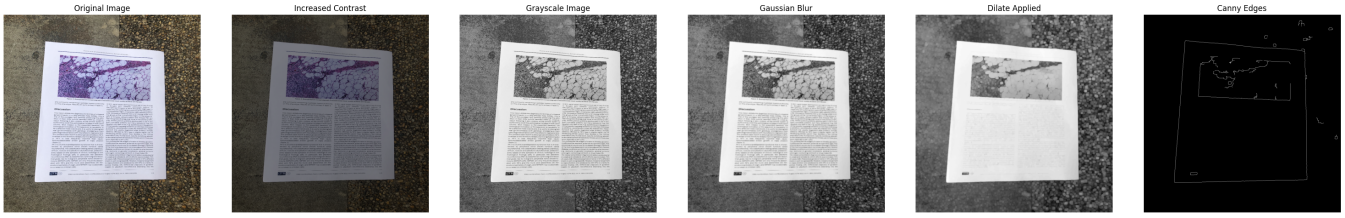

Figure 31: Example Preprocess

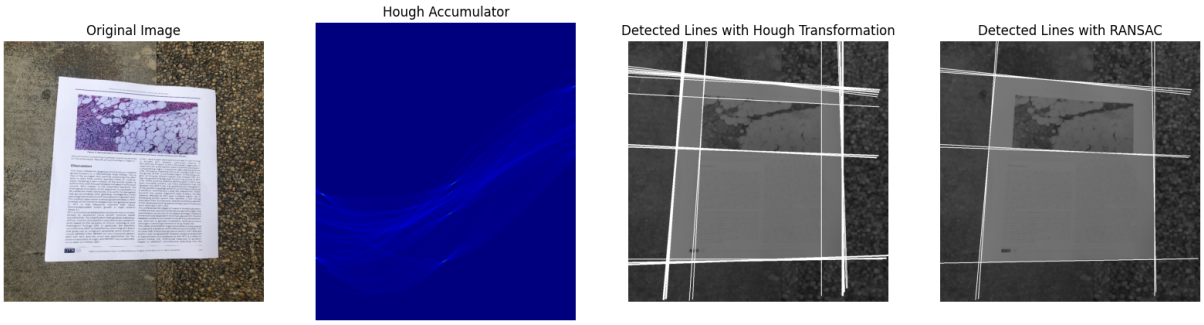

Figure 32: Example Application of RANSAC and Hough Transformation in Line Detection

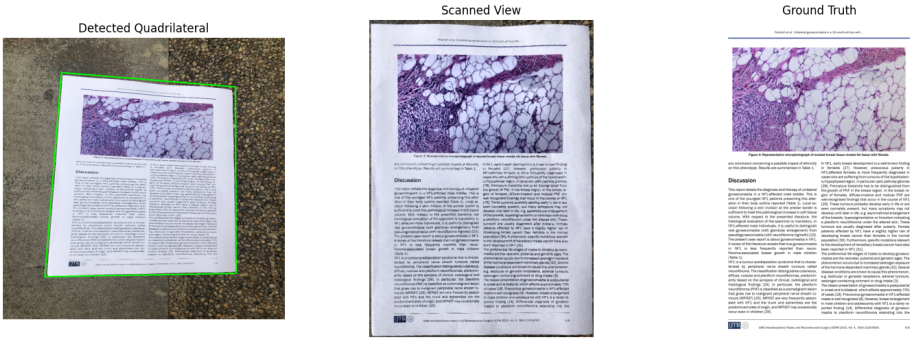

Figure 33: Example of Detected Quadrilateral and Scanned Image

Third Experiment

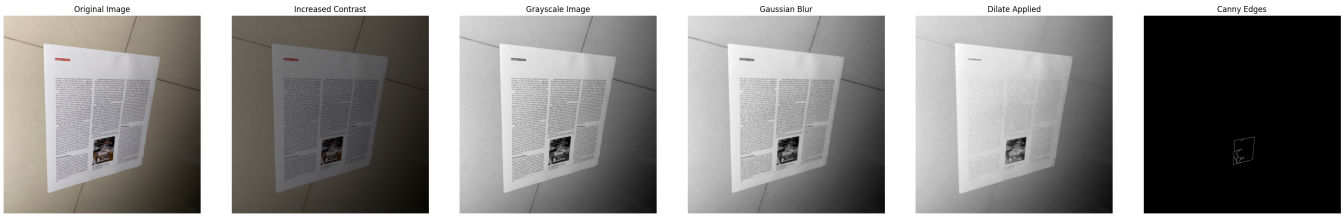

Figure 34: Example Preprocess

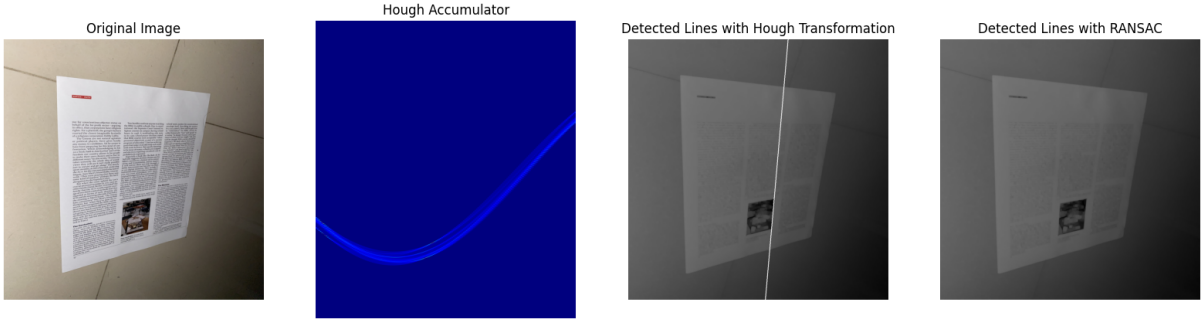

Figure 35: Example Application of RANSAC and Hough Transformation in Line Detection

In this experiment, the borders of the paper cannot be highlighted with this preprocess (Figure 34). Therefore the borders cannot be detected (Figure 35) with RANSAC and Hough Transform. During geometric transformation process, there cannot be detected a quadrilateral and cannot produce a scanned paper.

Random

First Experiment

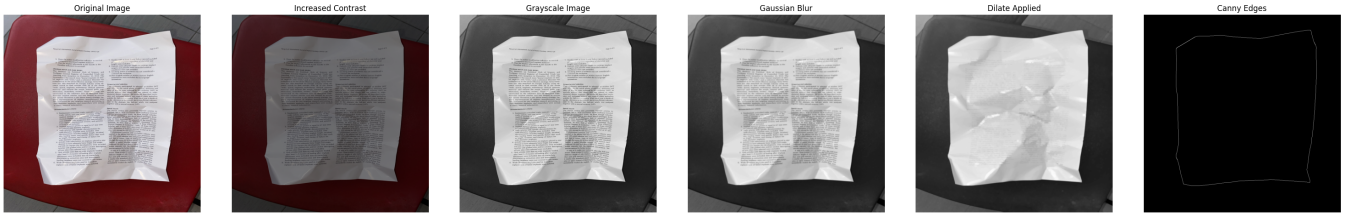

Figure 36: Example Preprocess

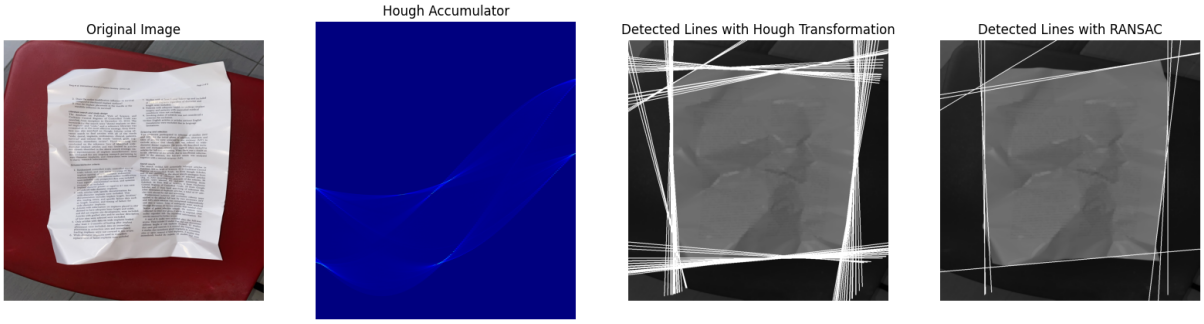

Figure 37: Example Application of RANSAC and Hough Transformation in Line Detection

Figure 38: Example of Detected Quadrilateral and Scanned Image

Second Experiment

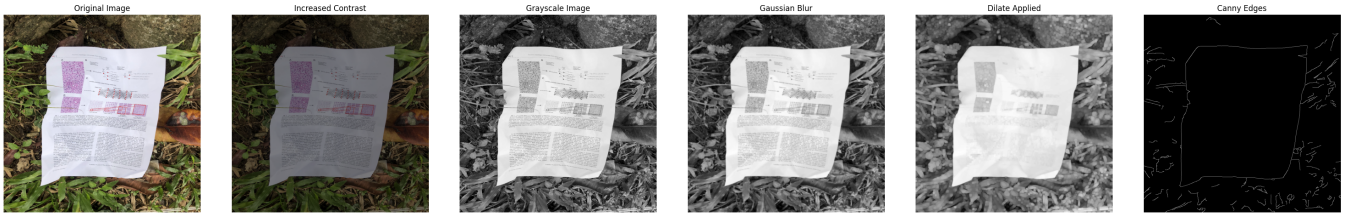

Figure 39: Example Preprocess

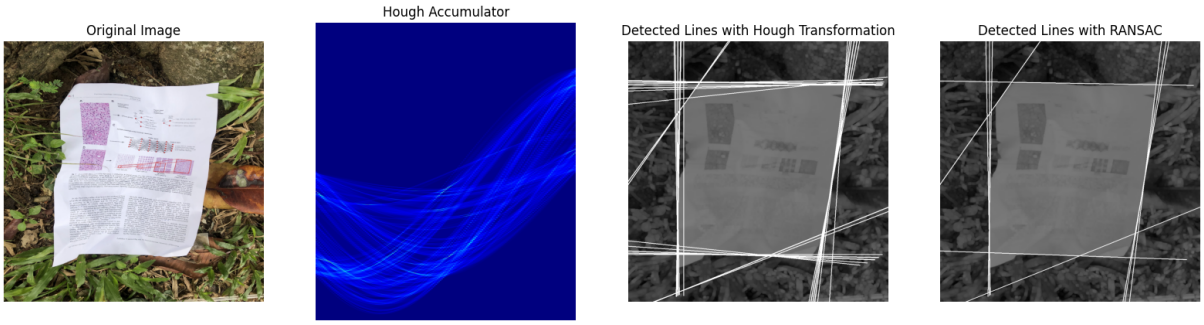

Figure 40: Example Application of RANSAC and Hough Transformation in Line Detection

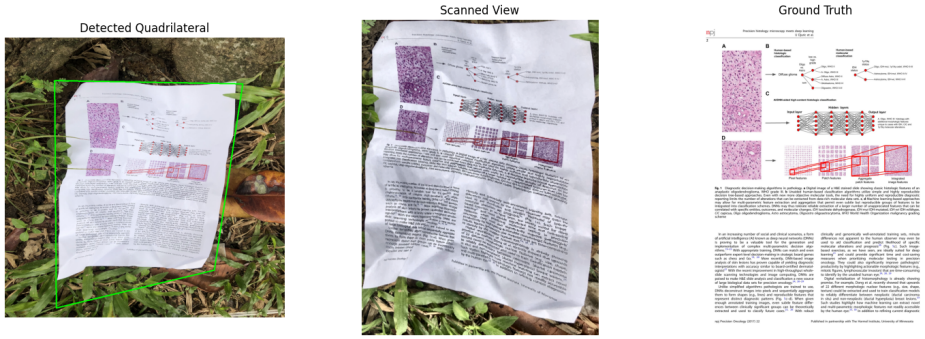

Figure 41: Example of Detected Quadrilateral and Scanned Image

Third Experiment

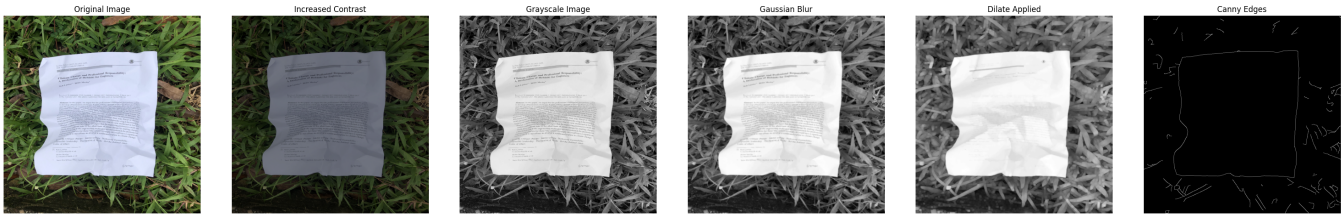

Figure 42: Example Preprocess

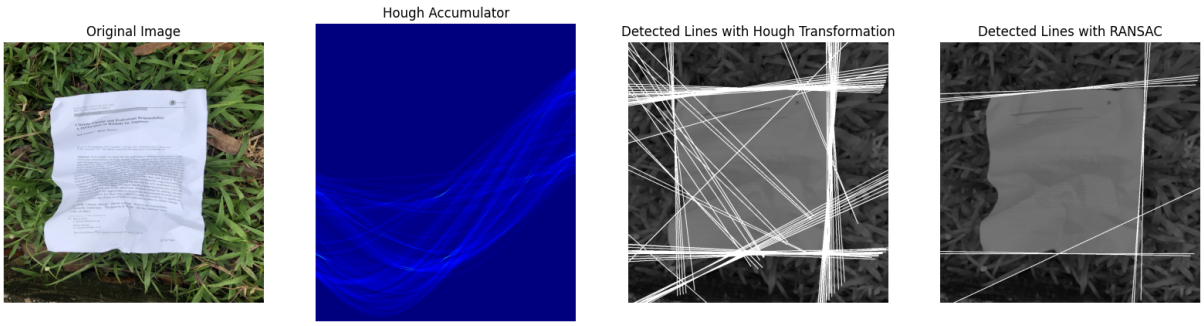

Figure 43: Example Application of RANSAC and Hough Transformation in Line Detection

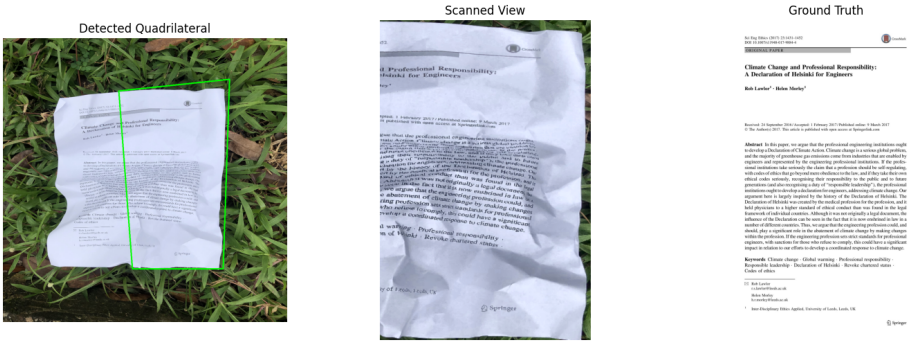

Figure 44: Example of Detected Quadrilateral and Scanned Image

Rotate

First Experiment

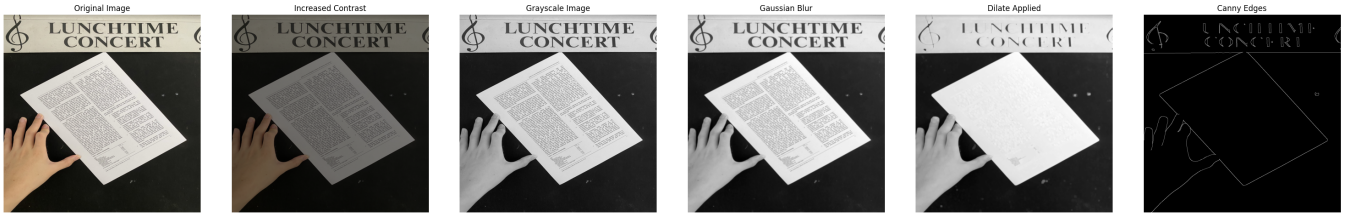

Figure 45: Example Preprocess

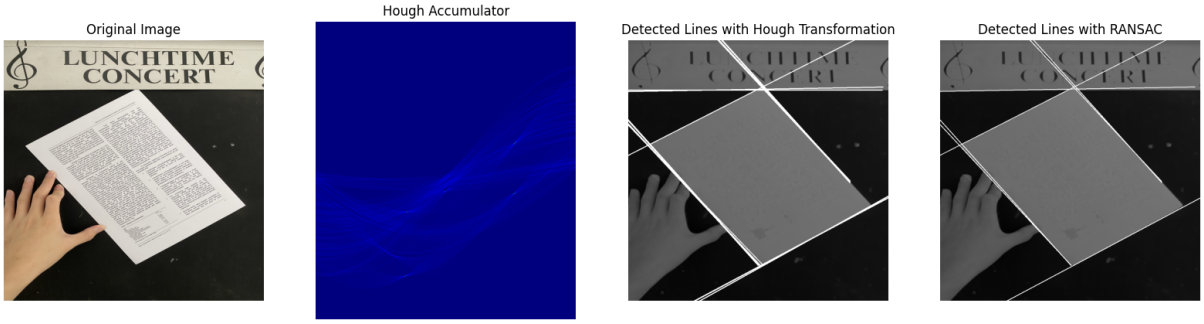

Figure 46: Example Application of RANSAC and Hough Transformation in Line Detection

Figure 47: Example of Detected Quadrilateral and Scanned Image

Second Experiment

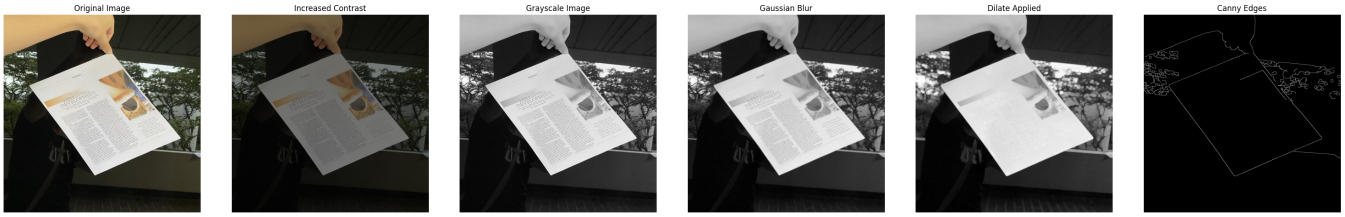

Figure 48: Example Preprocess

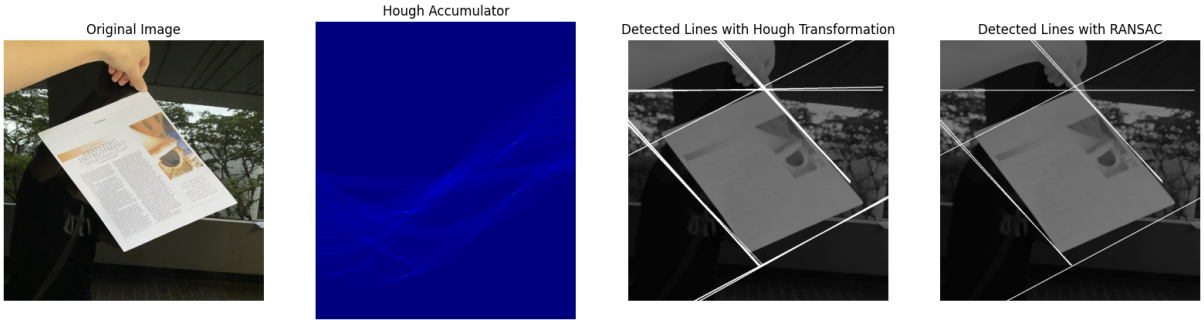

Figure 49: Example Application of RANSAC and Hough Transformation in Line Detection

Figure 50: Example of Detected Quadrilateral and Scanned Image

Third Experiment

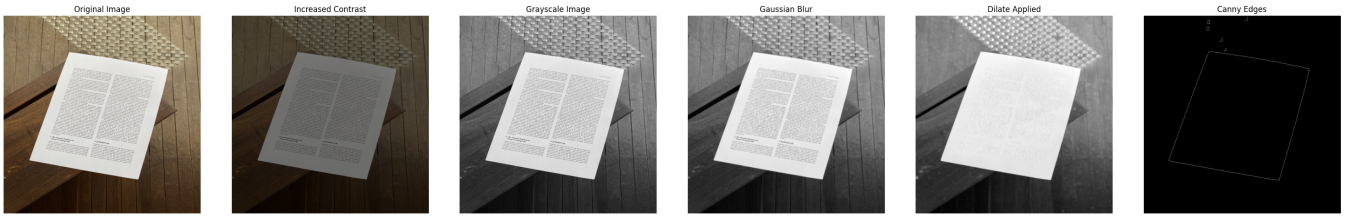

Figure 51: Example Preprocess

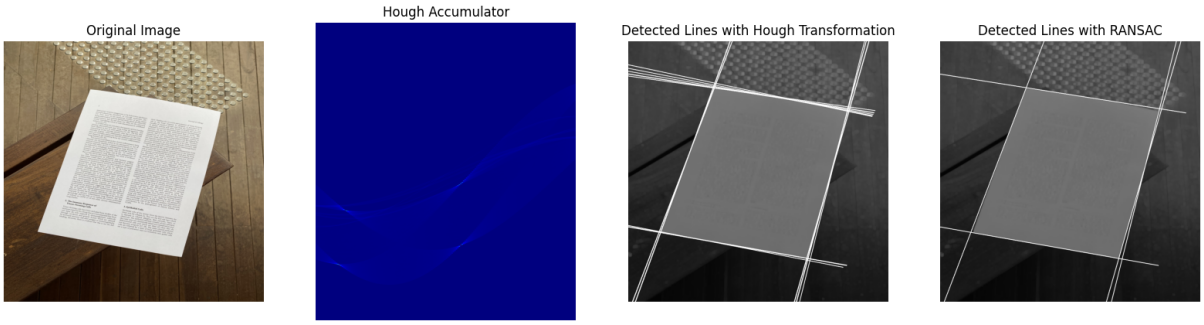

Figure 52: Example Application of RANSAC and Hough Transformation in Line Detection

Figure 53: Example of Detected Quadrilateral and Scanned Image

Conclusion

In the **curved** images, the method struggles to accurately detect document borders in some cases (see Figures 9–10), leading to incomplete or misaligned corrections. Similarly, in the **fold** category, two main issues are observed: the detected quadrilateral sometimes includes noisy background regions outside the actual document area (Figure 16), and in other instances, borders cannot be reliably identified at all (Figures 17–18).

In the case of **incomplete** images, the algorithm often misinterprets the surrounding noise as part of the document. This results in incorrect boundary estimation, as seen in Figures 21, 24, 27, and 30. These examples highlight the limitations of the model when edge information is insufficient or misleading.

On the other hand, the algorithm performs significantly better on **perspective** distortions, where the document is fully visible but geometrically skewed. Most perspective images are well-corrected (Figures 30–33), with only rare failures to detect borders (Figures 34–35). This is expected, as the method is primarily designed to address geometric distortions like skew and perspective changes.

In the **random** category, border detection is generally successful. However, due to structural inconsistencies in the documents, noisy outer areas are frequently included in at least one side of the detected quadrilateral (Figures 38, 41, and 44).

Lastly, the model also shows strong performance on **rotate** images. Despite the challenges rotation poses for some detection algorithms, the method consistently corrects these images well in most cases (Figures 47–53).

Statistical Results

A function was written to evaluate the approach for perspective correction for all data. Although the SSIM metric may be misleading in some cases, since there is no better alternative, it will be made on the SSIM value. The accepted coefficients were found by trial and error during the assignment and were accepted as the most optimal. There is a coefficient called *threshold*. This value is used to eliminate lines that received fewer votes than the threshold and is affected by the structure of the images. It will be given according to the results in the *Virtual Results* section.

- Curved Images → *threshold* = 80,
- Fold Images → *threshold* = 80,
- Incomplete Images → *threshold* = 80,
- Perspective Images → *threshold* = 80,
- Random Images → *threshold* = 60,
- Rotate Images → *threshold* = 100

Table 1: SSIM Evaluation Results

Category	Total Processed	Total Skipped	Average SSIM	Median SSIM	Std. Deviation	Range SSIM
Curved	73	27	0.4571	0.4647	0.1329	0.0971 - 0.9009
Fold	69	31	0.4483	0.4669	0.1121	0.1815 - 0.6339
Incomplete	69	31	0.3724	0.3727	0.1038	0.1407 - 0.6339
Perspective	77	23	0.4861	0.4802	0.1161	0.2117 - 0.7557
Random	72	28	0.4522	0.4564	0.1060	0.1742 - 0.7735
Rotate	54	46	0.3903	0.3960	0.1157	0.1001 - 0.6923

The table above presents the performance of the implemented perspective correction method across six categories of distorted document images. Each row corresponds to one distortion type.

- **Curved Images:** With an average SSIM of 0.4571 and a median of 0.4647, the method performs reasonably well on curved documents. The standard deviation (0.1329) suggests moderate variability in the results. The model was able to handle these deformations fairly well. In most of the images, a quadrilateral can be found, only 27% of the images were skipped because the detected lines could not form a quadrilateral.

- **Fold Images:** The average SSIM is slightly lower at 0.4483, with a standard deviation of 0.1121, indicating moderate variability in the results. Folded documents introduce abrupt changes in structure, which likely made it more difficult for the model to detect reliable lines and contours. The detected lines could not form a quadrilateral in more cases (31%) compared to curved images.

- **Incomplete Images:** This category shows a lower performance, with an average SSIM of 0.3724. The median SSIM (0.3727) further reflects this weakness. Since there are missing parts in the documents and I am detecting quadrilaterals, when a quadrilateral is detected, it is often expected to show some outside area in at least one corner of the detected area (see Figures 21-24 and 27). These detected noisy areas significantly reduced the SSIM values. Additionally, this category has a high rate of skipped images (31%), where the detected lines could not form a valid quadrilateral.

- **Perspective Images:** This is the best-performing category, with the highest average SSIM of 0.4861 and relatively low variability (std. dev. 0.1161). Since the algorithm is designed to correct perspective distortions using Hough Transform and RANSAC, it makes sense that it excels in this case. As shown in the Virtual Results section, it is challenging to find optimal constant coefficients that provide high scores across all possible images. These coefficients are highly affected by the noisy parts of the images. In this category, there are 23 test cases that could not create a quadrilateral. Additionally, there is one case that resulted in a scanned image with a relatively low SSIM (0.21). If custom parameters had been applied to these images, the results could have been improved further.

- **Random Images:** These images likely contain a variety of distortions, but the model still achieved a respectable average SSIM of 0.4522. Since in these images the linearity of borders is more disrupted, a relatively lower threshold (60) was needed. This lower threshold helped retain more line candidates, improving the robustness of correction. This category is the best in terms of detecting lines that could create quadrilaterals, with a rate of 72%, and also has the lowest standard deviation. By looking at Figures 38-41 and 44, it can be seen that it is nearly impossible to eliminate all outsider areas due to the structure of the borders. These noisy areas significantly affect the SSIM value. If there were a way to compare the similarity between images without the effect of these noisy parts, the results could be further improved.

- **Rotate Images:** This category had a moderate average SSIM of 0.3903 but suffered from a high skip rate of 46%. The main reason for this is that the threshold value for this category is much larger than the other categories. Although this threshold value was determined by testing with different images from the category, it is difficult to find a constant that would work effectively for all the images.

Curved

Figure 54 shows the histogram of SSIM scores for the curved image category. Most SSIM values are concentrated between **0.35 and 0.55**, indicating that the algorithm performed moderately well on the majority of curved documents. The peak of the distribution is around **0.45**, which aligns with both the average (0.4571) and median (0.4647) SSIM reported in the statistical summary.

There are a few images that achieved significantly higher SSIM scores, even approaching **0.9**, but these are outliers and not representative of the general trend. Moreover, these images can be unreliable because if the detected quadrilateral is too poor and shows only white areas, the SSIM result could be misleadingly high. Similarly, a smaller portion of the results fall below 0.3, representing cases where the algorithm likely failed to properly correct the curvature. This could be due to noisy areas that fall into the detected region.

The spread of the data, along with the visible skew toward the middle-high range, reflects that while the approach is not perfect, it is consistently able to provide a reasonable correction for most curved documents. The standard deviation of **0.1329** supports this conclusion.

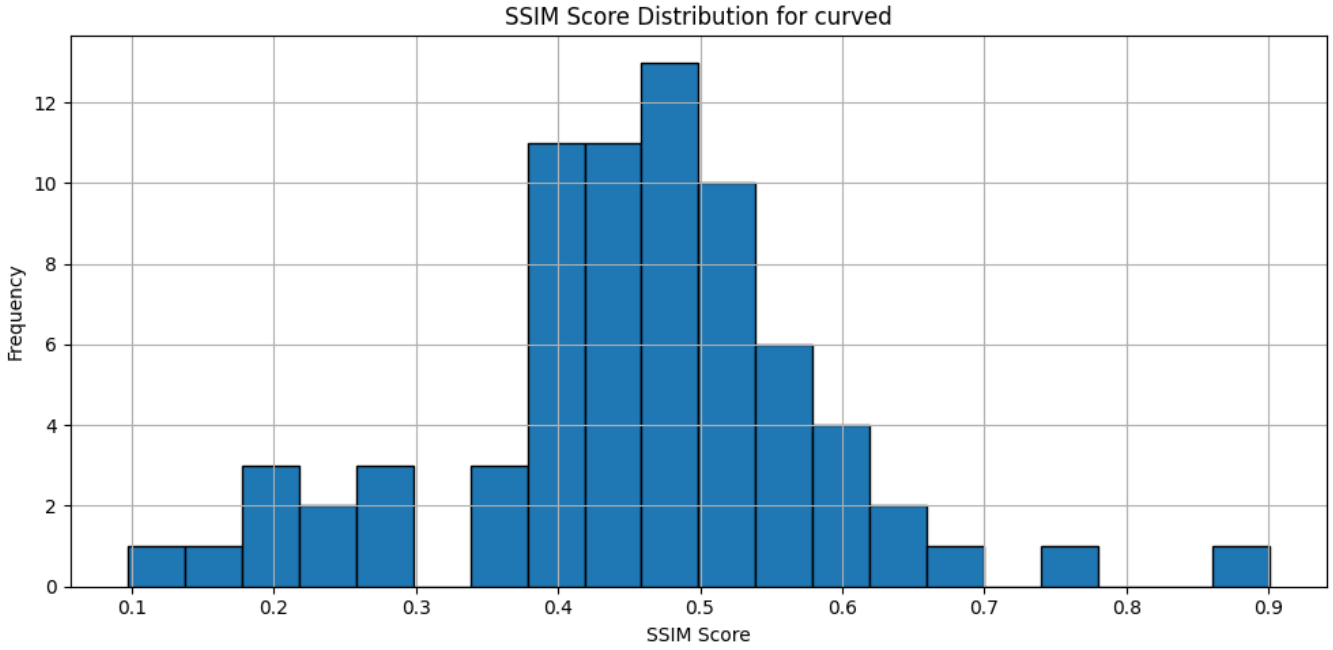

Figure 54: SSIM Score Distribution

Fold

Figure 55 shows the histogram of SSIM scores for the fold image category. The majority of SSIM values are spread between **0.3 and 0.6**, with the most common score falling around **0.48**, which aligns with the peak in the histogram. This is consistent with the statistical table, where the median SSIM was reported as **0.4669**.

The histogram reveals a fairly wide spread in performance. Although many results cluster around the mid-range (0.4–0.6), there is a notable number of outliers with SSIM values below 0.2. These low scores suggest that the algorithm had difficulty handling certain fold distortions, potentially due to sharp creases or broken line continuity that disrupted the detection of quadrilateral structures.

The presence of these low-end outliers contributes to the higher standard deviation observed in this category (**0.1121**), indicating inconsistent performance across different fold scenarios. Nevertheless, the algorithm was still able to produce acceptable results for a large portion of the dataset.

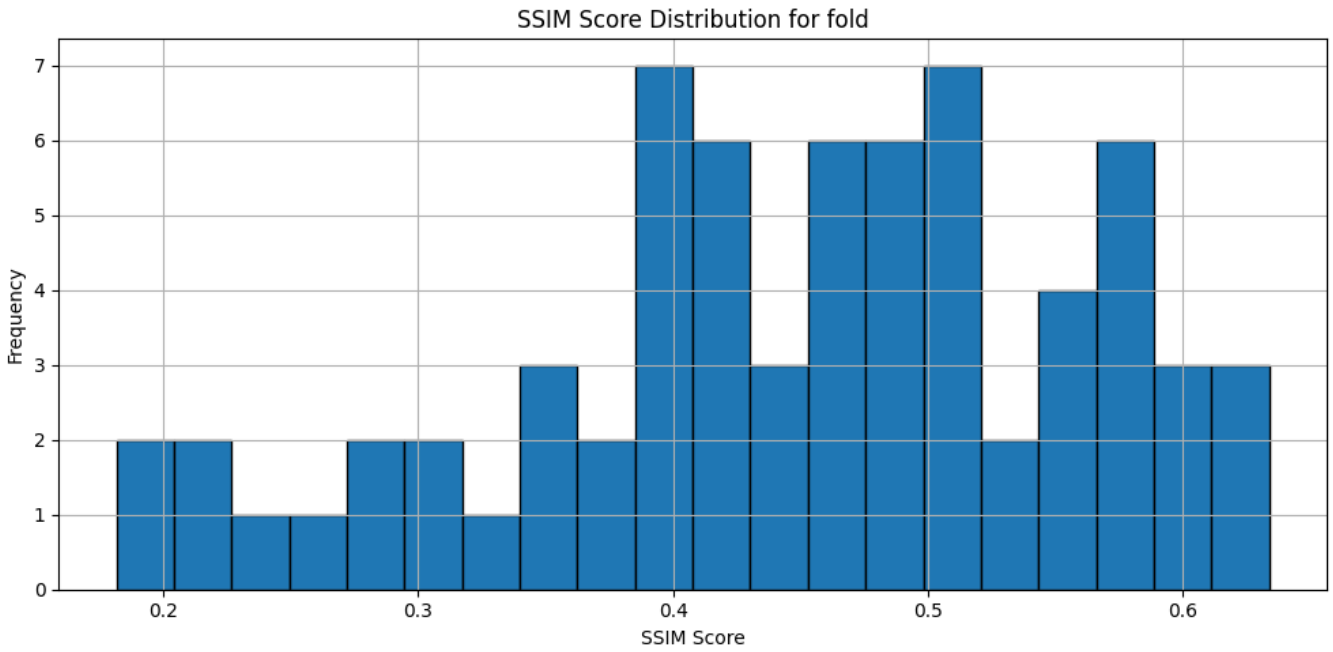

Figure 55: SSIM Score Distribution

Incomplete

Figure 56 shows the histogram of SSIM scores for the incomplete category. Unlike the previous categories, the histogram here is slightly more skewed toward the lower end, with the most common SSIM values falling between **0.35 and 0.45**. This aligns closely with the reported average SSIM of **0.3724** and median of **0.3727**.

The tail on the left side of the histogram, including values as low as **0.14**, suggests that the algorithm struggled with certain images in this category. This is expected, as missing or occluded document regions can confuse line detection and quadrilateral estimation, leading to misalignments or partial corrections.

While the distribution does include a few higher scores (above 0.6), these are relatively rare. The presence of both low and high outliers contributes to a standard deviation of **0.1038**, indicating some variability, though slightly lower than that of the Fold category.

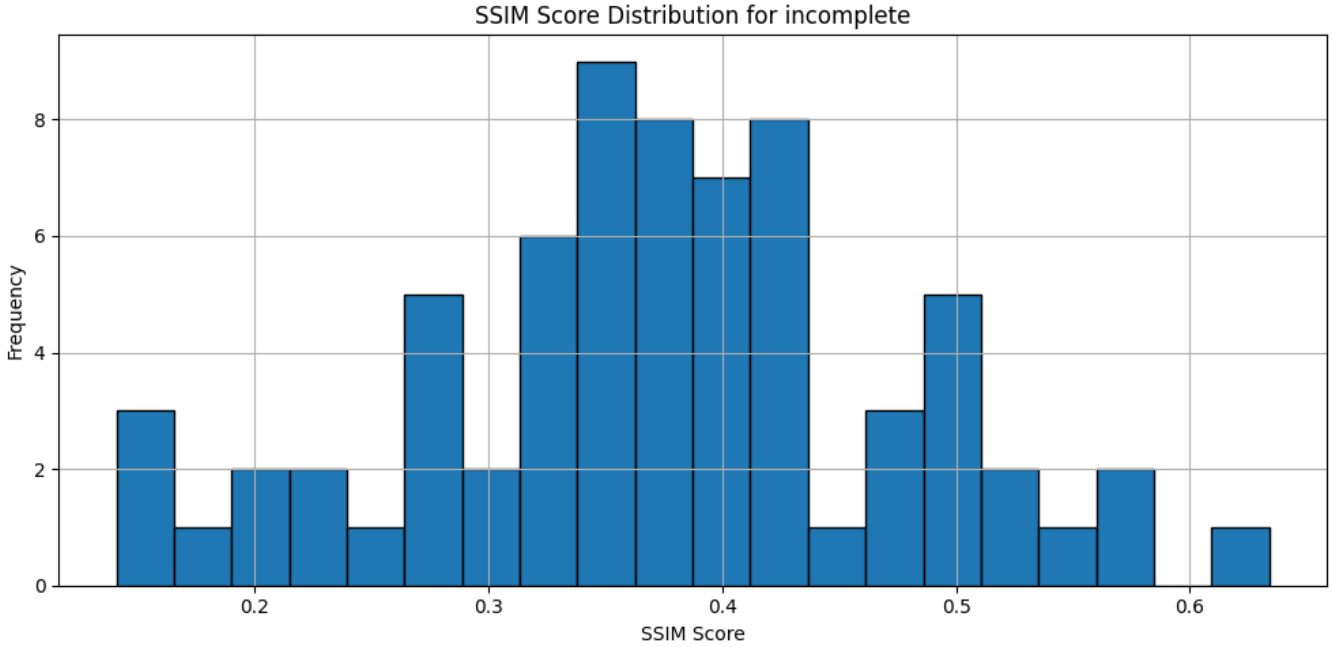

Figure 56: SSIM Score Distribution

Perspective

Figure 57 shows the histogram of SSIM scores for the perspective category. The scores are generally high and centered between **0.4 and 0.6**, with the most frequent values appearing around **0.45 to 0.55**. This is consistent with the reported average SSIM of **0.4861** and median of **0.4802**, confirming that the algorithm performed best on this category.

Compared to other distortion types, the histogram here is more symmetric and compact, indicating more consistent results. The standard deviation of **0.1161** supports this observation.

There are still a few outliers at both ends, including very low scores (below 0.2) and higher scores (above 0.7). These results can be interpreted similarly to the explanations in other categories. Overall, the distribution demonstrates that the method is well-suited for handling perspective distortions, and the core design of the algorithm specifically targets this type of geometric issue.

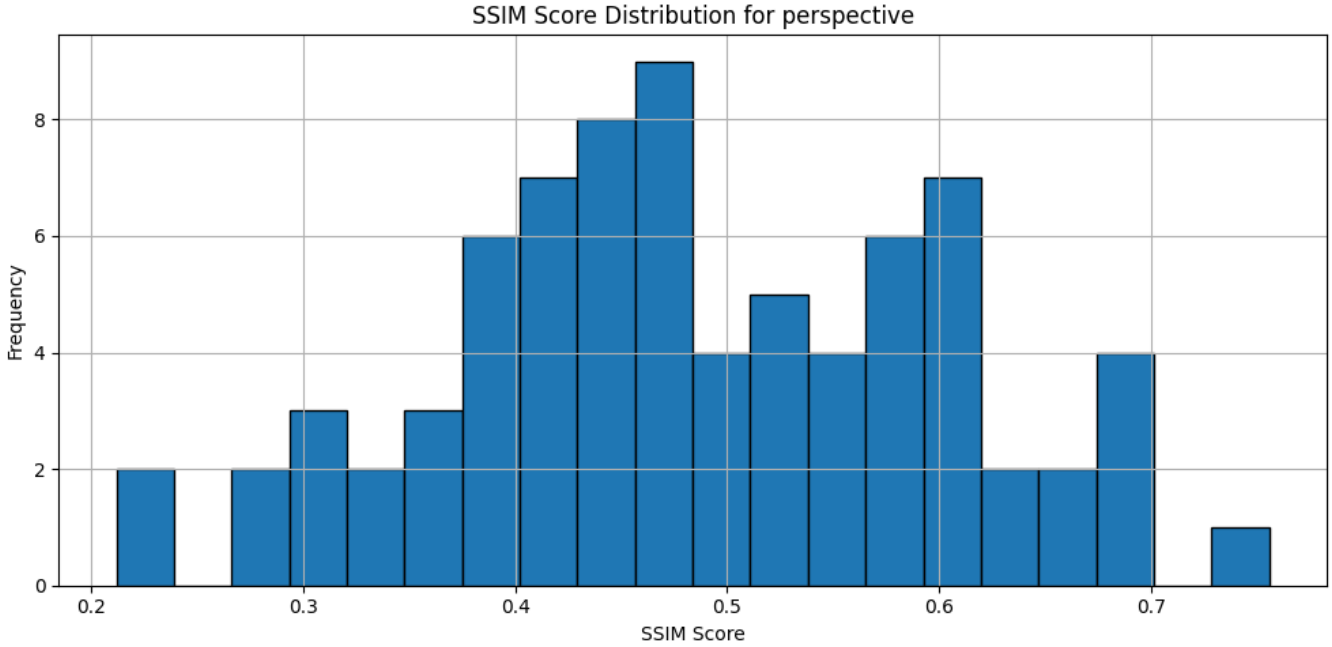

Figure 57: SSIM Score Distribution

Random

Figure 58 shows the histogram of SSIM scores for the random category. The scores are generally concentrated between **0.4 and 0.55**, with a clear peak around **0.45**, indicating stable and relatively strong performance. These values align well with the average SSIM of **0.4522** and median of **0.4564** reported earlier.

While there are a few low and high outliers (scores below 0.3 and above 0.7), they are infrequent. The distribution's shape suggests that, overall, the method performs consistently well on random distortions, with a higher concentration of results in the mid-range. This is consistent with the standard deviation of **0.1060**, indicating relatively low variability.

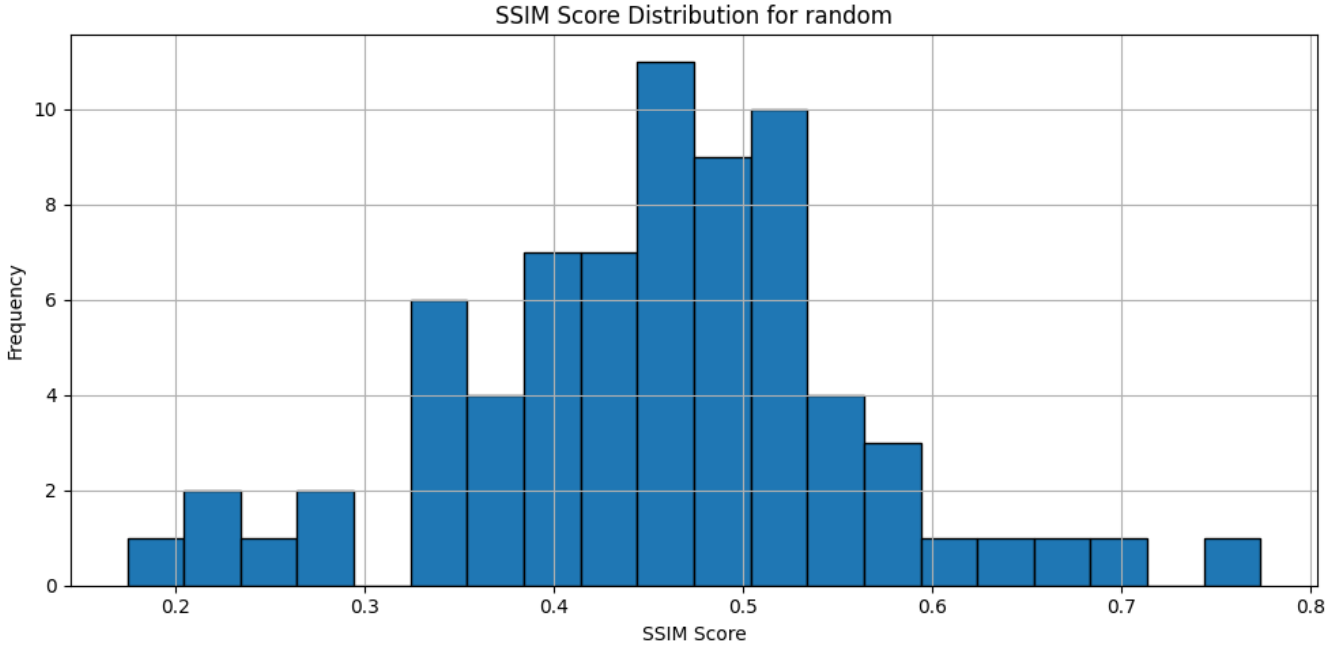

Figure 58: SSIM Score Distribution

Rotate

Figure 59 shows the histogram of SSIM scores for the rotate category. The results are spread widely across the full range, from as low as **0.1** to as high as **0.7**. The scores are most frequently concentrated between **0.3 and 0.5**, with a noticeable peak around **0.4 to 0.45**, which aligns with the reported average SSIM of **0.3903** and median of **0.3960**.

The broader spread in scores and visible presence of both low and high outliers reflect the inconsistent performance of the algorithm on rotated images. This inconsistency is also supported by the relatively high standard deviation of **0.1157**, the highest among all categories.

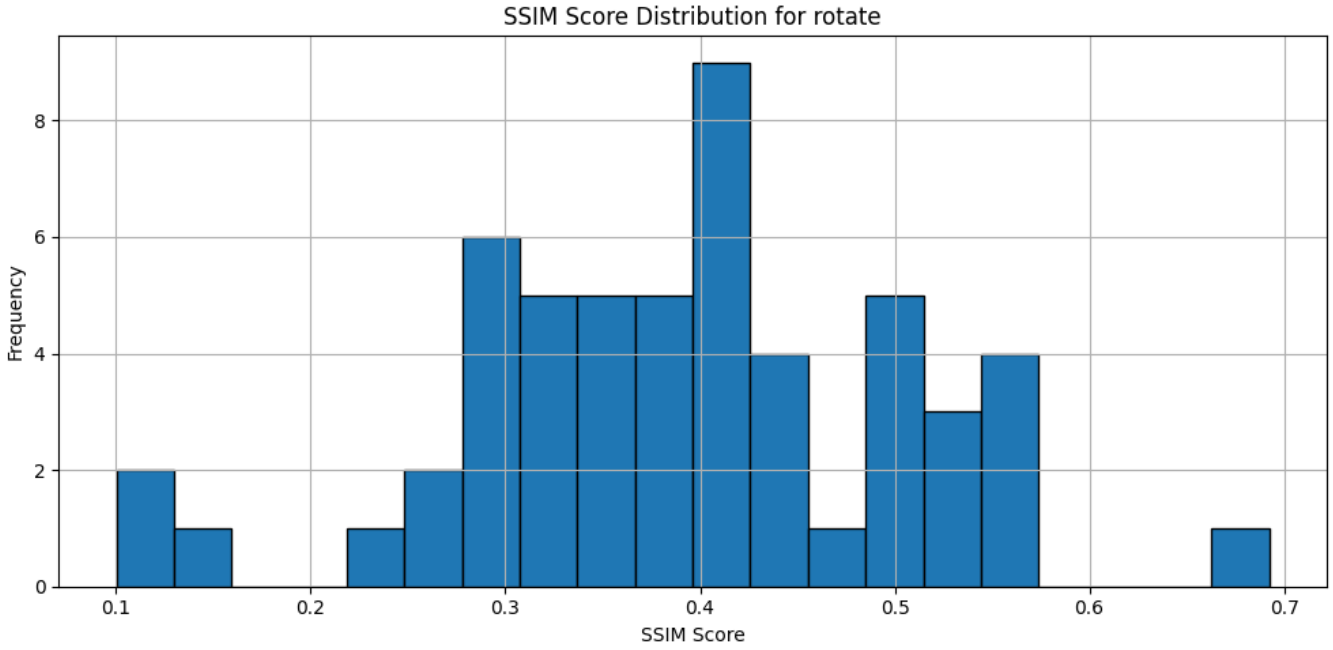

Figure 59: SSIM Score Distribution

Conclusion

Overall, the method shows the strongest performance on images with clean perspective distortions, while more complex issues such as folds, missing areas, or rotation reduce its effectiveness. Most SSIM scores fall in the 0.4–0.48 range, indicating that while the results are fair, there is still room for improvement in both detection consistency and geometric transformation accuracy.