International Journal of Applied Mathematics
Volume 38 No. 5 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

TEXT EXTRACTION FROM DEGRADED HISTORICAL DOCUMENTS USING ADVANCED IMAGE PROCESSING AND CLUSTERING TECHNIQUES

Shashidhara B [1], Yogish Naik G R [2] and Vidyasagar K B [3]

[1] Research Scholar, Department of PG Studies and Research in Computer Science, Kuvempu University, Shimoga, India. email id: shashidhara.sirmv@gmail.com
Orchid Id: 0009-0000-6965-2443
[2] Professor, Computer science and MCA, Department of PG Studies and Research in Computer Science, Kuvempu University, Shimoga, India. email id: ynaik.ku@gmail.com
Orchid Id: 0000-0002-6492-4216
[3] Assistant Professor, BGS Institute of Technology, Adichunchanagiri University email id: vkb2231@gmail.com Orchid Id: 0000-0002-3082

Abstract

This study proposes a novel method for extracting textual information from degraded historical document images, using image processing methods and advanced clustering techniques to enhance the accuracy and reliability of textual data extraction. The methodology involves converting document images to grayscale, enhancing contrast via histogram equalization, detecting text edges with the Canny algorithm, refining character shapes through skeletonization, identifying text regions using connected component analysis, and applying geometric filtering and nearest neighbour clustering to denoise and group these regions. This integrated approach effectively distinguishes authentic textual content from background noise and extraneous elements. Experimental evaluation on degraded historical document datasets demonstrates that the proposed method consistently attains Precision 91–94%, Recall 94–97% and F1-Scores (92–95%) respectively and Error Rates for the proposed method remain the lowest (7–10%). This approach significantly enhances the quality of digitized text from challenging archival documents, providing an effective solution for libraries, researchers, and historical digitization projects.

Keywords: Clustering, Skeletonization, Geometric Filtering, Bounding Box, Text Region Segmentation, DIBCO Dataset, Region Extraction.

I. INTRODUCTION

Digitizing Degraded Historical Document Images (DHDIs) is key for preserving heritage and enabling research, but common problems like faded ink, stains, poor lighting, and mixed content often make text extraction challenging. Traditional OCR systems struggle with these issues, leading to lower accuracy and missing information. Text region identification and segmentation are fundamental preprocessing techniques for minimizing noise and compensating for document degradation. These processes enhance OCR performance and support downstream tasks such as information retrieval, annotation, and data mining in DHDIs. Technical challenges arise

Received: August 05 ,2025

International Journal of Applied Mathematics
Volume 38 No. 5 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

particularly in instances of low contrast, diverse fonts, complex document layouts, and interference from graphical elements [1,2,6].Recent advancements in classical image processing and geometric clustering algorithms have enabled precise extraction of text regions from DHDIs. Adaptive grayscale conversion and histogram equalization improve contrast [9], while Canny and Sobel edge detection highlight textual boundaries. Morphological skeletonization refines text structure, and connected component analysis isolates text areas from background and noise. Subsequent geometric filtering and clustering of components further enhance the accurate grouping and localization of text regions, ensuring reliable performance even in challenging and complex DHDIs [20-23].

Figure 1: Textual and Structural Defects in Historical Document Images

Degradation Types: The images represent several typical forms of document egradation:

- **Faded ink**: Many documents have uneven or weak text due to aging or poor preservation.
- **Stains and bleed-through**: Ink from the reverse side, water damage, or mold contribute to visual clutter.
- **Smudges, blurring, and physical damage**: Resulting from handling, scanning motion, or

Received: August 05 ,2025

International Journal of Applied Mathematics
Volume 38 No. 5 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

paper defects.

- **Low contrast and noise**: Difficulties in distinguishing text from varied backgrounds and noise introduced during scanning.

Tests on many types of degraded historical document images including records, handwritten pages, and materials in different languages show that this method consistently finds text with high accuracy. It also makes OCR results much better, with more correct citations and fewer mistakes compared to standard full page scanning. Because the workflow is modular and efficient, it works well even for large digitization projects. This article takes a close look at how to extract text from old, damaged documents, explaining step by step methods, how they are tested, and the ideas behind them. By bringing together proven techniques and real-world results, the study helps make digital text collection more accurate and trustworthy, making it easier for everyone to access and use Historical Written Documents.

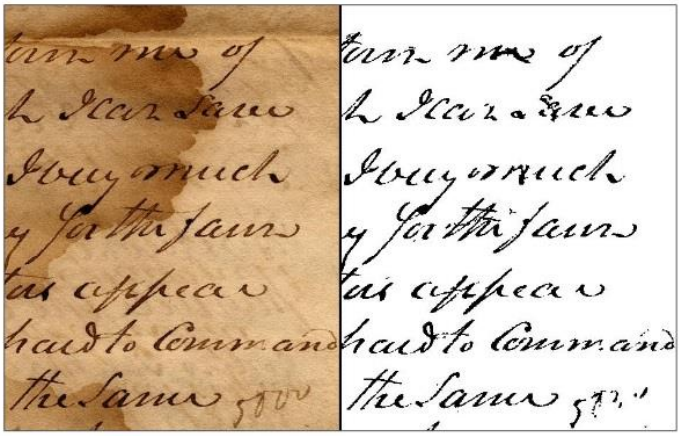


Figure 2. Comparative Visualization of Degraded Historical Document Image:

Original Degraded Image (Left) and Text-Extracted Output (Right)

The image above scientifically demonstrates the motivation and benefits of proposed approach for historical document digitization. On the **left**, the original scanned document exemplifies typical archival challenges water damage, faded ink, and uneven illumination that obscure handwritten content and hinder reliable data extraction. On the **right**, the result of applying unified image processing and clustering framework is shown. Here, advanced techniques such as grayscale conversion, histogram equalization, edge detection, skeletonization, connected component analysis, and clustering collectively eliminate noise, remove stains, and isolate clear text against a white background. This precise extraction significantly improves OCR performance and makes previously inaccessible information readable and analysable. By systematically overcoming the core challenges present in degraded historical documents such as stains, faded ink, uneven lighting, and interfering non-text elements the proposed approach delivers a scientifically validated and highly effective solution for digitizing archival resources. Through advanced image processing and clustering strategies, this method significantly improves both the quality and accessibility of digital text. As a result, researchers, librarians, and cultural institutions are empowered to preserve, access, and utilize historical materials with greater accuracy and reliability, ensuring these valuable records

Received: August 05 ,2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

remain available to future generations.

For this study, a specialized and DIBCO series dataset of degraded historical document images was assembled to support comprehensive evaluation of the proposed approach. This dataset featuring diverse instances of stains, faded ink, uneven backgrounds, and intricate page structures provides a robust foundation for validating the effectiveness and adaptability of the methods introduced in this proposed work [8.11,13].

II. REVIEW OF LITERATURE

Recent research in degraded historical document segmentation applies classical methods (like edge detection, contour analysis, and projection profiles), machine learning (connected component analysis with clustering), and deep learning architectures (CNNs, RNNs, U-Net, Mask R-CNN). Studies consistently demonstrate that deep learning and hybrid approaches substantially enhance segmentation accuracy, particularly with complex, multilingual, and handwritten manuscripts. However, their effectiveness depends on the quality and diversity of annotated datasets, with decreased performance in cases of severe degradation, ornate decorations, or unique scripts. Classical methods remain advantageous for documents with regular layouts, while deep learning models require careful post-processing to manage noise and artifacts. Therefore, integrating advanced preprocessing, adaptive segmentation, and data-driven models is currently most effective, though robust generalization to all DHDI is still a challenge.

A Panneerselvam et al. [12] proposed a method which is a new way to pull out text from old, damaged document images with tricky backgrounds. They start by using Canny edge detection to make faded letters easier to see, then turn the image into skeletons to highlight the shapes of letters. After that, they use clustering and nearest neighbour methods to separate the real text from everything else in the image. This method is especially good for documents with poor contrast or mixed types of writing, helping researchers and libraries digitize and read old records more accurately.

Likforman-Sulem et al. [14] proposed various methods including hybrid, projection profile, and deep learning approaches—to separate handwritten text lines in old manuscripts. Using challenging datasets, they found deep learning models with convolutional networks gave the best results, especially on complex pages. However, these models require high-quality, consistent training data; if handwriting or labels vary, their accuracy drops, which limits their use across different kinds of documents.

Klaus, A., et al. [15] proposed a method, morphological filtering and connected component analysis to pull out text areas from scans of old German Baroque books. This works well for standard pages with clear text and layout because printed letters follow regular patterns. However, when pages have fancy borders or pictures, the method can mistakenly classify these as text, making it less accurate. So, while it's good for regular manuscript pages, it struggles with more decorative or complex documents.

Ren, X., et al. [16] proposed a method deep learning method called a convolutional neural network (CNN), together with rule-based adjustments, to find text blocks in images of old manuscripts. They tested this system on different challenging datasets and achieved high accuracy in separating text regions. However, the method depends on these post-processing rules, which must be carefully

Received: August 05 ,2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

adjusted, especially for images with a lot of noise or background clutter. This means extra work is needed to keep the results accurate on different types of documents.

Siddiqui, S., et al. [17] proposed an improved historical document segmentation by combining connected component analysis, which groups text areas, with contour tracing, which outlines shapes. They tested this on Urdu, Persian, and European documents. This combined method works well for multilingual and complex pages, accurately finding text areas, but its accuracy can drop if images are blurry or have fancy script decorations, since these conditions can confuse the technique.

Kaur, G., et al. [18] proposed a hybrid technique that combined edge detection with clustering to find text in ancient Sikh manuscripts, even when images were noisy or faded. By grouping similar shapes and focusing on text structure, their method works well in tough conditions. However, it sometimes mistakes fancy borders or decorations for text, causing false positives. More post-processing is needed to separate real text from decorative elements in these richly detailed documents.

Abdelfattah, M., et al. [19] proposed a method deep learning specifically convolutional neural networks to find and separate text blocks in Arabic manuscript images. Their system, trained with labelled data from Arabic archives, reliably picks out main text areas and extra content, giving better results than older contour-based methods, especially for dense or complicated pages. However, the main drawback is that the model works best only on Arabic manuscripts; it doesn't do as well when used on documents in other languages, so it would need retraining to handle different scripts.

Konstantinidis, G., et al. [20] proposed a method that first improves the contrast and fixes uneven lighting in old Greek documents, then turns the images into clear black and white (making the text stand out), and finally uses shape-based steps to separate text areas before OCR. This makes it easier to find and read the text with a computer. However, each document type may need different settings, since differences in paper, ink, or style mean the method has to be carefully adjusted for best results.

P. Shivakumara., et al. [21] proposed a method to separate lines of writing in handwritten documents, even when some words are crossed out or written over. First, the method finds words using stroke thickness and filters out noise. Crossed-out parts are spotted using a deep learning model (DenseNet) and handled differently than normal text. After this, it organizes and groups the words into straight lines using their position and shape. The method works well on both new and standard test sets, and does a better job than older techniques for splitting lines in hard-to-read handwritten pages, including those with erased or messy writing.

A Islam., et al., [22] proposed a method for extracting lines from handwritten documents using instance segmentation, where each text line is treated as a separate object. The approach uses deep learning frameworks to identify and segment individual lines, making it easier to handle overlapping or irregularly spaced handwritten text. The method is tested on various document images and shows strong effectiveness in accurately separating out each text line, which benefits further tasks like text recognition and analysis in handwritten document images.

III. PROPOSED METHODOLOGY

The goal of this proposed method is to develop a reliable method that can denoise and accurate text

Received: August 05 ,2025

International Journal of Applied Mathematics
Volume 38 No. 5 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

from DHDIs that are faded, stained, smudged, or otherwise damaged. These problems make it difficult for computers to read and digitize the DHDIs, which is a key step in making historical documents easily available and searchable for libraries, researchers, and online archives.

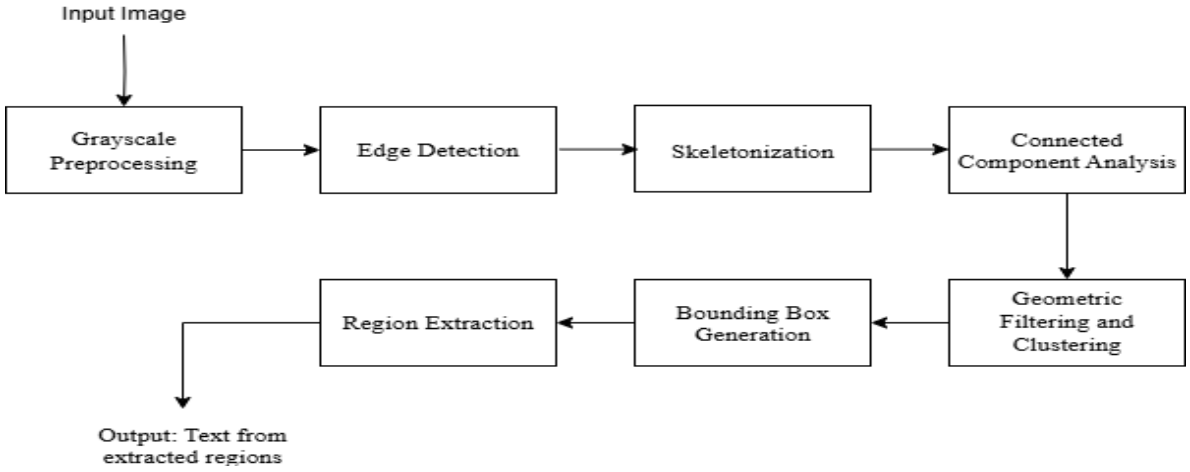


Figure 3: Flow Diagram for Textual Information Extraction from Degraded Historical Document Images

Stage 1: Image preprocessing is performed first, involving conversion to grayscale [9] and, where necessary, enhancement via histogram equalization [4] and contrast adjustment to improve the visibility of faded or poorly inked characters. The grayscale transformation reduces the computational burden and isolates structural features.

$$\mathbf{I_{gray}(x,y) = 0.299.R(x,y) + 0.587.G(x,y) + 0.144\ .B(x,y)} \quad \textbf{---(1)}$$

Where R,G,B are the red, green, and blue channel intensities at pixel (x,y); $\mathbf{I_{gray}(x,y)}$ is the grayscale intensity. and Redistributes intensity values to enhance contrast is obtained by,

$$\mathbf{I_{gray}(x,y) = H(I_{gray}(x,y))} \quad \textbf{---(2)}$$

Where H computes the cumulative distribution function (CDF) and remaps pixel values accordingly. This step increases the uniformity of intensity transitions, facilitating subsequent edge and region detection. The original color document is converted to a grayscale image where each pixel is an intensity value (0–255). Values represent here are shades of gray.

$$\begin{bmatrix} 120 & 123 & 122 \\ 121 & 130 & 140 \\ 122 & 132 & 133 \end{bmatrix}$$

Algorithm:

For each input_image:
if is_color(input_image):
gray_image = convert_to_grayscale(input_image)
else:
gray_image = input_image

Received: August 05 ,2025

International Journal of Applied Mathematics
Volume 38 No. 5 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

```
enhancement = adaptive_histogram_equalization(gray_image)
contrast_metric = measure_contrast(enhancement)

while contrast_metric < threshold:
tune_parameters(enhancement)
enhancement = adaptive_histogram_equalization(gray_image, tuned_params)
contrast_metric = measure_contrast(enhancement)

If has_uneven_background(enhancement):
enhanced_image = apply_thresholding(enhancement)
else:
enhanced_image = enhancement

# Optionally try other methods and compare
log("Best parameters used:", tuned_params)
store(enhanced_image)
```

Stage 2: Text regions differ from the background in intensity; edge detectors reveal these boundaries by highlighting places of rapid intensity change. Canny edge detection [3] is robust for noisy, low-contrast images. Canny operator involves four steps:

1. Smoothing (Noise Reduction): The input image is smoothed using a Gaussian filter to reduce noise and spurious intensity variations. This is crucial because edge detection is sensitive to noise, and smoothing prevents the detection of false edges.

$$\mathbf{I_{smooth} = I_{prep} * G_{\sigma}} \quad \textbf{---(3)}$$

where G_{σ} is a Gaussian kernel.

2. Gradient Calculation: The algorithm computes the image gradient (intensity change) at each pixel using edge detection filters (typically Sobel operators). This step finds regions of the image with sharp intensity changes, indicating possible edges.

$$\mathbf{G_x = \frac{\partial I_{smooth}}{\partial x}\ ,\ G_y = \frac{\partial I_{smooth}}{\partial y}} \quad \textbf{----(4)}$$

Gradient magnitude:

$$\mathbf{M(x,y) = \sqrt{G_x^2 + G_y^2}\ , direction: \theta(x,y) = arctan\left(\frac{G_y}{G_x}\right)} \quad \textbf{---(5)}$$

.

3. Non-Max Suppression: Only retain local maxima along the direction of θ. The gradient magnitude image is thinned by suppressing all pixels that are not strictly local maxima along

Received: August 05 ,2025

International Journal of Applied Mathematics
Volume 38 No. 5 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

the gradient direction. This step ensures that only the most prominent edges are preserved, producing thin (one-pixel wide) edge lines.

4. **Thresholding and Hysteresis:** Apply dual thresholds to classify edge pixels, connect weak edges that are adjacent to strong ones. Two thresholds (high and low) are applied to identify strong and weak edges. Strong edges (above high threshold) are confidently retained; weak edges (between thresholds) are only retained if they are connected to strong edges. This step finalizes the edge map and connects genuine edges, while discarding noise. Hysteresis in Canny edge detection is the final thresholding step that determines which detected edges are true edges and which are noise. It works by applying two thresholds: a high and a low.

Output:

$$\mathbf{E} = \boldsymbol{\varepsilon}(\mathbf{I_{prep}}) \quad \text{---(6)}$$

Where ε is the Canny operator, E is the binary edge map.

Only the edges of text and other features are highlighted (white); everything else is dark or black. (255 = edge pixel; 0 = background.)

$$\begin{bmatrix} 0 & 255 & 0 \\ 255 & 0 & 255 \\ 0 & 255 & 0 \end{bmatrix}$$

Algorithm:

For each preprocessed_image:
edge_maps = {}
for algorithm in [Canny, Sobel, Laplacian]:
for kernel_size in kernel_sizes:
for threshold in threshold_range:
edge_map = apply_edge_detector(preprocessed_image, algorithm, kernel_size, threshold)
edge_quality = evaluate_edge_map(edge_map)
edge_maps[(algorithm, kernel_size, threshold)] = (edge_map, edge_quality)

best_params = select_best_edge_map(edge_maps)
final_edge_map = edge_maps[best_params][0]

log("Selected algorithm and parameters:", best_params)
store(final_edge_map)

Stage 3: Skeletonization to thin detected edges and preserve character connectivity. Skeletonization [1] ensures that densely packed or touching characters, common in handwritten and historical prints, can be separated effectively. For binary edge map E: Iteratively remove boundary pixels that do not alter the connectivity of the structure, until only 1-pixel-wide skeletons remain. Let,

$$\mathbf{S} = \mathbf{S}(\mathbf{E}), \quad \text{---(7)}$$

Where **S** operates through morphological thinning.

- Generate a skeleton of the edge image to reduce thick text strokes into 1-pixel wide representations without breaking connectivity.

Received: August 05 ,2025

International Journal of Applied Mathematics
Volume 38 No. 5 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

- Skeletonization preserves the structural components of letters and separates noise from meaningful junctions.

Morphological thinning [1] is applied to binary images and removes selected foreground pixels iteratively, based on the local neighborhood defined by structuring elements. This process is crucial for skeletonization, ensuring that character strokes are reduced to their essential connectivity, which helps in further separating densely packed or connected handwritten elements.
During thinning:

- The algorithm uses structuring elements to examine the image pixel-by-pixel.
- At each position, pixels are removed if they meet specific criteria ensuring preservation of overall shape and connectivity.
- Iterative passes with various rotated structuring elements continue until no further changes occur.

This results in skeletonized, connected representations, optimizing further analysis and separation of text regions in document images.
Skeletonization is thus an essential morphological operation in image processing for text region extraction, allowing simplification and clarification of densely packed shapes while maintaining overall character connectivity. Edges are thinned so each text element is a one-pixel-wide line, highlighting the "spine" of each character. (1 = skeleton pixel; 0 = background.)

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

This iterative morphological thinning removes boundary pixels while preserving the connectivity. Mathematically, pixels are deleted if their removal does not alter the Euler number or break the object into multiple components:

$$\mathbf{A}^{|} = \mathbf{A} \ominus \mathbf{B} \quad \text{(structural thinning operation)} \qquad \text{----(8)}$$

Algorithm:

```
For each edge_map:
stroke_thickness = estimate_average_stroke(edge_map)
skeleton = skeletonize(edge_map, parameters_based_on(stroke_thickness))

skeleton = prune_spurious_branches(skeleton)
skeleton = restore_endpoints_if_lost(skeleton)

if not is_skeleton_complete(skeleton) or is_overly_branched(skeleton):
adjust_parameters()
skeleton = skeletonize(edge_map, new_parameters)
skeleton = prune_and_restore(skeleton)

log("Skeletonization parameters used:", final_parameters)
```


Received: August 05 ,2025

International Journal of Applied Mathematics
Volume 38 No. 5 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

store(skeleton)

Stage 4: Connected component analysis (CCA) [1] further segments these thinned images into contiguous regions corresponding to characters or small text fragments.
Defining C in CCA:

- C is a set of pixels that are all connected (either 4-connected or 8-connected neighbors) in a binary image.
- Each c is treated as a candidate text region, graphical element, or noise/artifact depending on its geometric properties.
- Segmentation identifies and labels each connected component, so that further analysis (filtering, grouping, classification) can be applied to each one individually.

$\mathbf{C} = \{\mathbf{C_i} : \mathbf{C_i} \subset \mathbf{S}, \mathbf{C_i}\ \textbf{is a maximal connected set}\}$. **---(9)**

For each $\mathbf{C_i}$,

Bounding box: $\mathbf{B_i} = \mathbf{min}(\{\mathbf{x}, \mathbf{y}\}\mathbf{C_i}), \mathbf{max}(\{\mathbf{x}, \mathbf{y}\}\mathbf{C_i})$ **---(10)**

A bounding box [23] in document image processing is a rectangular region that tightly encloses an object of interest—such as a character, word, text line, or block—within an image. It is defined by its coordinates: the top-left (x_min, y_min) and bottom-right (x_max, y_max) corners of the rectangle, specifying its location and size on the image.

Centroid:

A centroid [1] in document image processing is the geometric center of a connected component or region within an image. It represents the average position of all the pixels in that particular region, calculated as the mean of their x and y coordinates. For a set of pixels(x_i, y_i) in a connected component, the centroid (C_x, C_y) is computed as:

$\mathbf{C_x} = \frac{\mathbf{1}}{\mathbf{N}} \sum_{\mathbf{i=1}}^{\mathbf{N}} \mathbf{x_i}$, $\mathbf{C_y} = \frac{\mathbf{1}}{\mathbf{N}} \sum_{\mathbf{i=1}}^{\mathbf{N}} \mathbf{y_i}$, **----(11)**

Where N is the number of pixels in the component.
Boundaries drawn around each character/component. Each component is given a unique label, (Component labels; e.g., 1, 2, 3 correspond to different regions.)

$$\begin{bmatrix} 1 & 1 & 0 \\ 2 & 0 & 2 \\ 0 & 3 & 3 \end{bmatrix}$$

The **Connected Component Analysis (CCA)** step involves:

1. **Identify Connected Pixels/Groups**

- Detect and label contiguous groups of foreground pixels in the binary image, where each group represents a potential character or text fragment.

2. **Extract Bounding Boxes/Centroids**

- For every identified connected component, determine its bounding box (a rectangle enclosing the component) and calculate its centroid (geometric center).

Each unlabelled foreground pixel triggers a region-growing algorithm that attributes a unique label to each set of connected pixels. Mathematically, this is formulated as:

$\text{Label(p) if I(p)} = 1 \text{ and p is unvisited}$ **----(12)**

Received: August 05 ,2025

International Journal of Applied Mathematics
Volume 38 No. 5 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Algorithm:

For each skeleton_image:
components = find_connected_components(skeleton_image, connectivity=8)
labeled_map = label_components(components)

for component in components:
props = get_properties(component)
if props.size < min_size or props.size > max_size:
remove(component)

ambiguous_components = find_ambiguous_components(labeled_map)
for ac in ambiguous_components:
if should_merge(ac):
merge(ac)
elif should_split(ac):
split(ac)

repeat until no ambiguous components remain or refinement criteria met

log("CCA parameters and actions:", min_size, max_size, merges, splits)
store(labeled_map)

Stage 5: The geometric attributes of each component are quantitatively analysed, enabling the systematic removal of non-text elements such as stains and illustrations through empirically derived thresholds.

Let feature vector for component $\mathbf{C_i}$ be

$$\mathbf{f(C_i) = (area, aspect, branches)} \quad \text{---(13)}$$

Defining an indicator function:

$$\mathbf{I_{text}(C_i)} = \begin{cases} \mathbf{1}, & \mathbf{\text{if } f(C_i) \text{ falls within threshold}} \\ \mathbf{0}, & \mathbf{\text{otherwise}} \end{cases} \quad \text{--(14)}$$

$$\mathbf{retain,} \quad \mathbf{C_{text} = \{C_i \in C \mid I_{text}(C_i) = 1\}} \quad \text{---(15)}$$

Clustering [5] - Text characters form lines and blocks, not isolated points. By spatial clustering, fragmented components are assembled into meaningful textual units. This is done by,

Given centroid $\mathbf{X = \{x_i : C_i \in C_{text}\}}$, ---(16)

cluster them using k-nearest neighbour

$$\mathbf{K = cluster\ (X; d(x_i, x_j) < \varepsilon} \quad \text{---(17)}$$

Where $\mathbf{d(x_i, x_j)}$ is the Euclidean distance and K partitions centroids into lines/blocks.

Received: August 05 ,2025

International Journal of Applied Mathematics
Volume 38 No. 5 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

The **Geometric Filtering [6]** step includes:

1. **Apply Size, Aspect, Skeleton Branch Thresholds**

- Evaluate each detected component using predefined criteria for size, aspect ratio, and skeleton branch properties.

2. **Decision: Is Component Likely Text?**

- If the component's geometric features match expected text characteristics, keep the component.
- If not, discard the component.

The **Clustering** [5] procedure consists of:

1. **Use Nearest Neighbor, Euclidean Metrics**

- Calculate the Euclidean distance between centroids of detected components (characters or text fragments).
- Identify spatially close components as nearest neighbors.

2. **Group Components into Lines/Blocks**

- Aggregate nearest neighbor components to form clusters that represent text lines or logical blocks.
- Ensure clusters correspond to meaningful document structure for accurate downstream processing.

Regions (connected components) are filtered based on area, width, height, and aspect ratio:

$$\mathbf{area} = \mathbf{w} \times \mathbf{h}, \quad \mathbf{aspect\ ratio} = \frac{\mathbf{w}}{\mathbf{h}} \qquad \textbf{----(18)}$$

Components with parameters outside defined thresholds are excluded.

Algorithm:

```
For each component in labeled_map:
area = compute_area(component)
aspect_ratio = compute_aspect_ratio(component)
branch_count = compute_skeleton_branches(component)

if not meets_text_criteria(area, aspect_ratio, branch_count):
remove(component)
elif matches_artifact_pattern(component):
suppress(component)

evaluate_segmentation_quality(filtered_map)
if quality < desired_threshold:
adjust_thresholds_and_rules()
repeat filtering
```


Received: August 05 ,2025

International Journal of Applied Mathematics
Volume 38 No. 5 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

log("Geometric filtering thresholds, artifact rules")
store(filtered_map)

Stage 6: Bounding Box Generation [23] - Each cluster of components is enveloped in a bounding box (or polygon) to reliably capture all text within that unit.
For each cluster K_k,

$$\mathbf{B_k} = \bigcup_{\mathbf{i} \in \mathbf{K_k}} \mathbf{B_i} \quad \text{--(19)}$$

where B_k covers the minimal region enclosing all member bounding boxes.

Bounding Box Generation includes the following steps:

1. **Merge Cluster Bounding Boxes**

- Combine adjacent bounding boxes belonging to the same text cluster or line to form larger, unified regions that represent complete textual structures.

2. **Expand Edges to Avoid Clipping**

- Slightly enlarge the boundaries of merged boxes, ensuring no part of a character or word is cut off during cropping, and all relevant text is captured for subsequent OCR processing.

Components grouped and merged into bounding boxes; boxes are shown on the image. Bounding box coordinates, (Each tuple is: left-x, top-y, width, height.)

$$\textbf{Bounding Boxes} := \boldsymbol{B_i} = \begin{bmatrix} (\boldsymbol{x_i}, \boldsymbol{y_i}, \boldsymbol{w_i}, \mathbf{h}_i) \\ (\boldsymbol{x_i}, \boldsymbol{y_i}, \boldsymbol{w_i}, \mathbf{h}_i) \end{bmatrix}$$

Algorithm:

component_centroids = [compute_centroid(c) for c in filtered_components]

clusters = spatial_clustering(component_centroids, method=DBSCAN, distance=epsilon)

bounding_boxes = []
for cluster in clusters:
box = compute_bounding_rectangle(cluster)
if is_degraded(cluster):
box = expand_box(box, expansion_value)
elif is_dense(cluster):
box = contract_box(box, contraction_value)
bounding_boxes.append(box)

if clustering_logic_fails(clusters):
clusters = alternate_clustering(component_centroids, method=hierarchical_or_grid)
regenerate_bounding_boxes(clusters)

log("Clustering parameters, bounding box adjustments")
store(bounding_boxes)

Received: August 05 ,2025

International Journal of Applied Mathematics
Volume 38 No. 5 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Stage 7: Region Extraction [7]

For each detected region R_i , define its bounding box coordinates: Top-left: (x_i, y_i), Width: w_i , Height: h_i.

Cropping Function - The region extraction operation, for an image matrix I, is given by

$$\mathbf{C_i} = \mathbf{I}\,(\mathbf{y_i}: \mathbf{y_i} + h_i\,, x_i: x_i + w_i) \qquad \textbf{--(20)}$$

Where C_i is the cropped region for bounding box , and $I\,(a : b\,, c : d)$ refers to the submatrix of I from rows a to b and columns c to d. Collect all extracted crops:

$$\boldsymbol{C} = \{\, \boldsymbol{C_1}, \boldsymbol{C_2}\,, \ldots\ldots\ldots\ldots, \boldsymbol{C_n}\,\} \qquad \textbf{---(21)}$$

For n detected regions.

Coverage & Purity Criteria:

Coverage measures how much of the ground truth text region is included in the detected bounding box. Is given by,

$$\text{Coverage}\ (B, T) = \frac{\lfloor B \cap T \rfloor}{\lfloor T \rfloor} \qquad \textbf{---(22)}$$

Where*:*

B = *pixels in the detected bounding box*

T = *pixels in the ground truth text region*

$\lfloor B \cap T \rfloor$ = *number of pixels in the intersection*

$\lfloor T \rfloor$ = *total pixels in the ground truth region*

Interpretation:

- Coverage = 1 means all text is included
- Lower values mean the box misses parts of the text

Purity measures how much of the detected region is actually text (and not background or artifacts). And it is given by,

$$\text{Purity}\ (B, T) = \frac{\lfloor B \cap T \rfloor}{\lfloor B \rfloor} \qquad \textbf{---(23)}$$

Where:

$\lfloor B \cap T \rfloor$ = text pixels inside the bounding box

$\lfloor B \rfloor$ = total pixels in the bounding box

Interpretation:

- Purity = 1 means the box contains only text
- Lower values suggest more background or artifacts are included

Iterative Adjustments - If $\text{Coverage}\ (B, T)$ or $\text{Purity}\ (B, T)$ falls below threshold $\propto$ (e.g., 0.9), adjust bounding box:

Received: August 05 ,2025

International Journal of Applied Mathematics
Volume 38 No. 5 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

- Update (x_i, y_i, w_i, h_i) based on feedback.
- Repeat cropping and re-evaluation.

$$\textbf{Region matrix} = \begin{bmatrix} 240 & 230 & 220 \\ 245 & 235 & 215 \end{bmatrix}$$

This matrix representation illustrates the transformation from raw scanned image to labeled, isolated, and decoded textual content at each processing stage. Regions defined by bounding boxes are cropped and binarized using Otsu's thresholding [2]:

Otsu's threshold: $\sigma_B^2(t) = w_0(t)w_1(t)[\mu_0(t) - \mu_1(t)]^2$ **----(24)**

The threshold that maximizes between-class variance segments foreground from background.

Algorithm:

```
for box in bounding_boxes:
# Crop the region from the original image based on bounding box coordinates
region_crop = crop_image(image, box)

# Save each cropped image region for manual review or further processing
save_crop(region_crop, box_id=box.id)

# Optionally, document the bounding box coordinates and region properties
log_region_metadata(box, region_crop)

# If crop quality is poor (misses text or includes too much background),
# adjust box parameters and re-crop as needed
if not is_crop_quality_good(region_crop, box):
box = adjust_box_parameters(box)
region_crop = crop_image(image, box)
save_crop(region_crop, box_id=box.id, version='adjusted')
log_region_metadata(box, region_crop, status='adjusted')
```

The function crop_image extracts a rectangle from the source image using the coordinates in box. The helper save_crop stores the extracted region as an image file or for further research. The function is_crop_quality_good checks size, coverage, and purity of each region.

IV. EXPERIMENTAL FINDINGS AND PERFORMANCE EVALUATION

To validate the effectiveness of the proposed methodology for extracting text regions from degraded historical document images, extensive experiments were conducted using a range of publicly available and custom-curated datasets. This section outlines the dataset characteristics, the evaluation metrics employed, and comparative analyses against established baseline methods. By systematically assessing the precision, recall, and error rates achieved under diverse conditions of manuscript degradation and layout complexity, the following findings demonstrate the practical impact, strengths, and limitations of the approach.

Received: August 05 ,2025

International Journal of Applied Mathematics
Volume 38 No. 5 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

The proposed methodology was evaluated on Utilizing datasets like DIBCO 2011, DIBCO 2013, DIBCO 2016, and DIBCO 2018 [8,11,13], which contain various types of deterioration in both handwritten and printed materials and custom created datasets featuring degraded historical document images. These datasets included archival manuscripts displaying a wide range of deterioration, such as faded ink, stains, low contrast, uneven backgrounds, and mixed handwriting or script styles. Each set offered varied document layouts and script complexities, challenging the automated extraction of text regions and ensuring that the method could be tested across realistic, diverse archival scenarios.

By comparing against other algorithms using these datasets, both qualitative and quantitative assessments are conducted to validate the effectiveness of the proposed method [8,11,13].

Precision is the fraction of detected regions (bounding boxes or cropped areas) in document images that actually contain real textual information, rather than background or noise.

$$\textbf{Precision} = \frac{\text{Number of true text regions detected}}{\text{Total number of detected regions}} \qquad \textbf{---(25)}$$

This metric helps determine how often proposed method makes false positives such as extra boxes around non-text areas. High precision means most detected regions are correct; lower precision means more mistakes with non-text regions being included as text.

Recall is the fraction of actual text regions in the document that your method correctly finds and detects.

$$\textbf{Recall} = \frac{\text{Number of true text regions detected}}{\text{Total number of ground truth text regions}} \qquad \textbf{----(26)}$$

This metric helps how many real text regions were missed by proposed method —false negatives, such as words or lines that should have been found but weren't. High recall means most real text areas are found; low recall means the model misses a lot of important information.

The F1-Score is the harmonic mean of Precision and Recall—meaning it balances both precision (avoiding extra boxes around non-text) and recall (finding as many true text areas as possible).

$$\textbf{F1} = 2 * \frac{\textbf{Precision} * \textbf{Recall}}{\textbf{Precision} + \textbf{Recall}} \qquad \textbf{----(27)}$$

This metric gives you a single number showing how well your method balances between over-segmentation (finding too much) and under-segmentation (missing text). A higher F1-Score means the method is both accurate and complete in finding and separating text regions.

Error Rate shows what fraction of all cases were either missed (false negatives) or wrongly identified as text (false positives) during segmentation.

$$\textbf{Error Rate} = \frac{\text{Number of false positive regions} + \text{Number of false negative regions}}{\text{Total number of ground truth text regions}} \qquad \text{----(28)}$$

This metric helps measure overall mistakes made by proposed method when extracting text from

Received: August 05 ,2025

International Journal of Applied Mathematics
Volume 38 No. 5 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

document images, offering a simple way to compare different approaches on their reliability and practical usefulness for historical digitization.

The processing time is the time required to process each image from start to finish, including all steps.

$$\textbf{Processing Time} = \text{End Time} - \text{Start Time} \quad \textbf{-----(29)}$$

Experimental Process

This proposed method works step by step. First, the color image is made into a grayscale image using CLAHE, which helps faded or hard-to-read text stand out better. Comparing the original image with the new one shows how much easier it is to see the words. Next, edge detection turns the image into black and white lines. This shows where letters and numbers are, while most background marks disappear. Skeletonization makes each letter even thinner, giving every word a clear, simple shape for the computer to find. After that, connected component analysis colors or labels each separate piece so you can see each part like every letter or dot as its own item. By zooming in, it's clear how well the parts are separated. Geometric filtering puts green boxes only around areas that look like real text, so unwanted spots like smudges and stains aren't chosen. This stage helps focus only on the actual words or numbers. Next, clustering joins together words and lines. Every word gets a blue box and number; every line gets a red box and number. This lets you see the text in order, just like you would read it. The last step takes each word or line out of the image as a small, clear picture. These outputs match the box numbers from before, so it's easy to check each one or use it for further work like OCR.

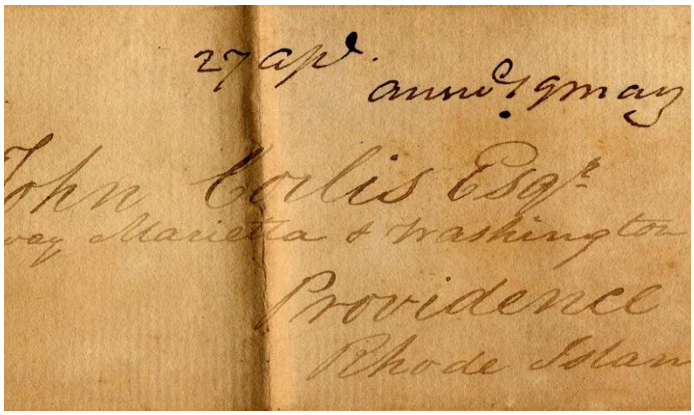
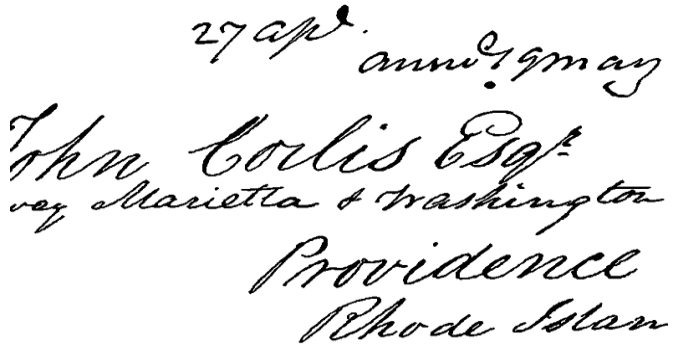


Figure 4: Degraded Historical Document Image

Stage	Output Image of figure 4	Description

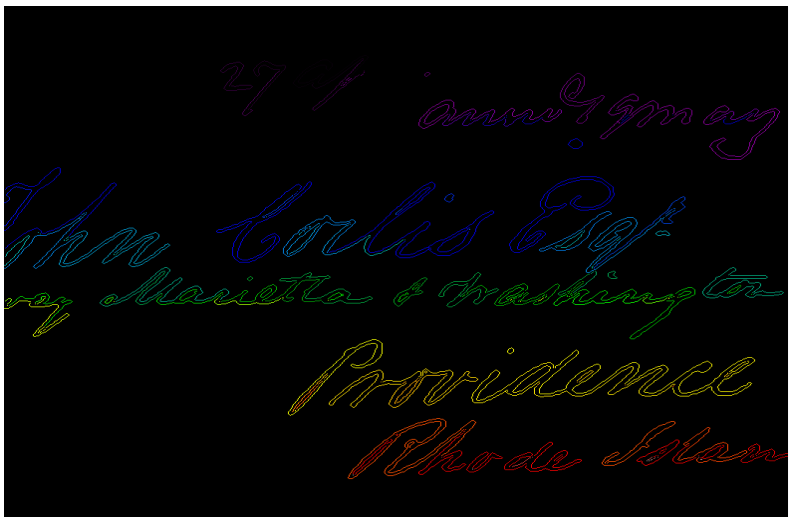
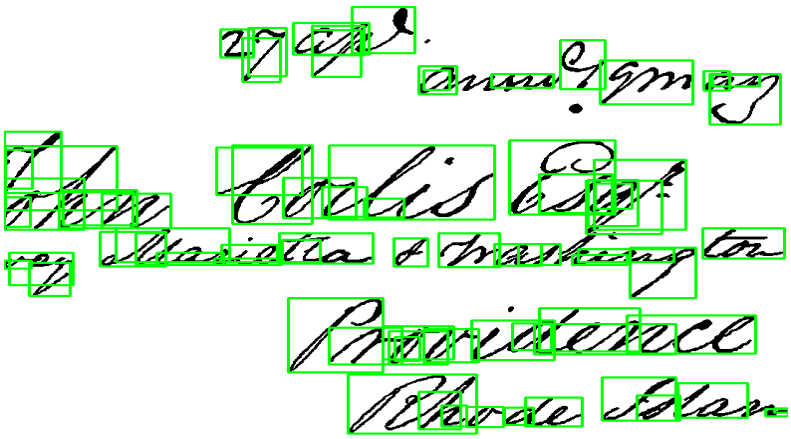
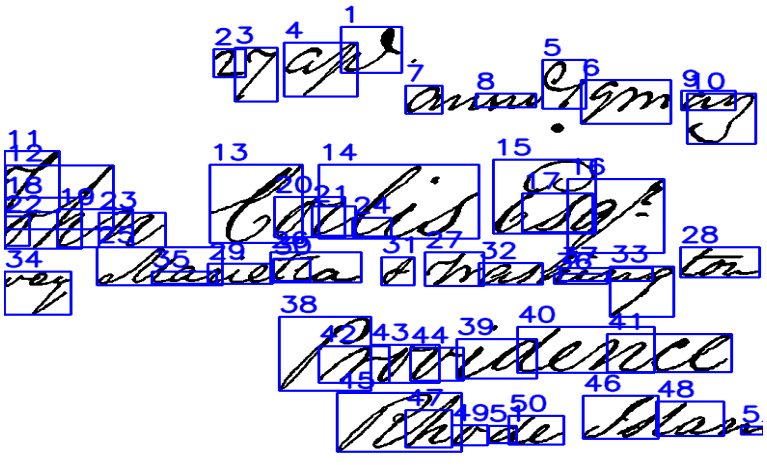
Received: August 05 ,2025

International Journal of Applied Mathematics
Volume 38 No. 5 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Stage 1: Grayscale & CLAHE		Enhanced grayscale image with improved text visibility
Stage 2: Edge Detection		Binary edge map showing text outlines
Stage 3: Skeletonization		Thinned shapes of letters for easy region separation

Received: August 05 ,2025

International Journal of Applied Mathematics
Volume 38 No. 5 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Stage 4: CCA		Labeled/color-coded regions for each character/blob
Stage 5: Geometric Filtering		Boxes over text-like regions, filtering out noise
Stage 6: Word Clustering		Numbered boxes around words in reading order

Received: August 05 ,2025

International Journal of Applied Mathematics
Volume 38 No. 5 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

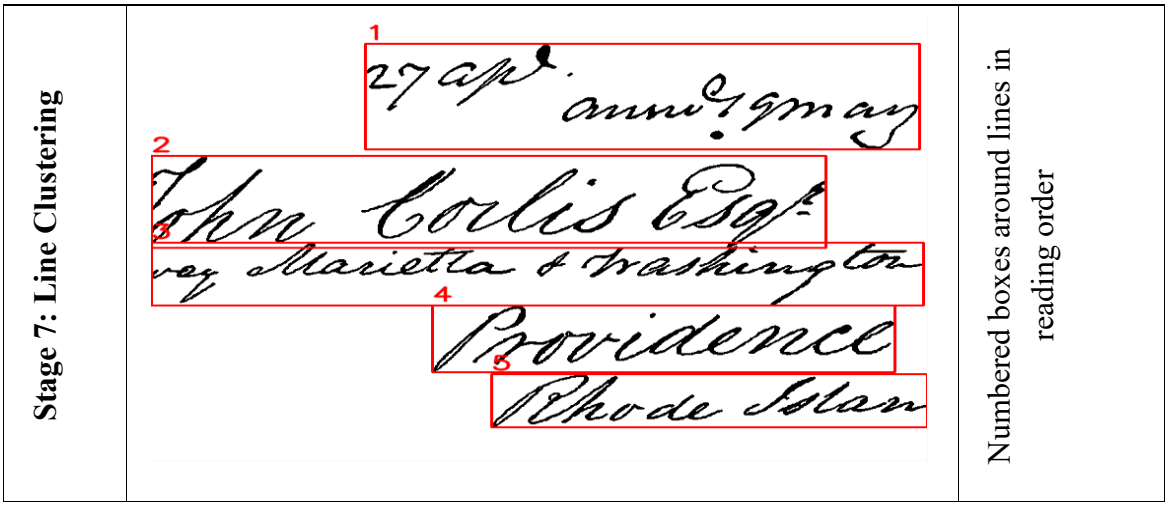
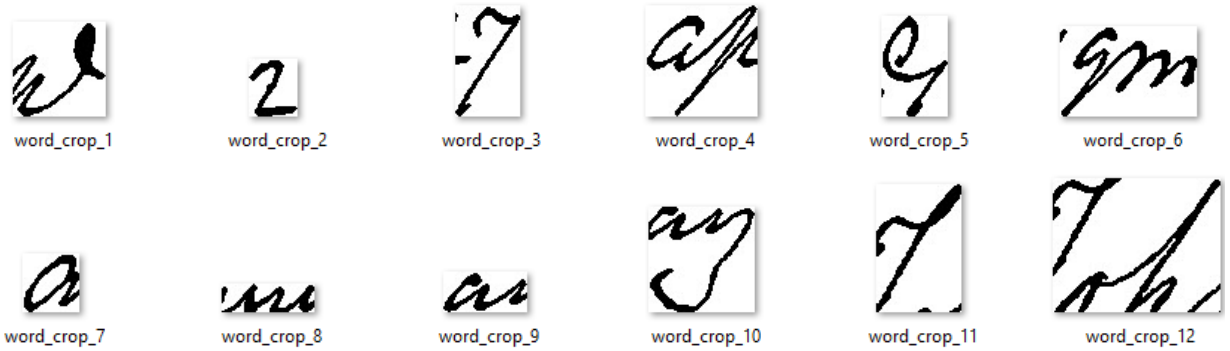


Figure 5: Illustrates the results of proposed method in each stages:

After the document has passed through enhancement, edge detection, skeletonization, connected component analysis, and geometric filtering, the workflow advances to clustering specifically the grouping of character components into meaningful words.

Word Clustering Process: Candidate regions (bounding boxes), which may correspond to characters or parts of words, are grouped based on proximity along the horizontal direction. By measuring distances between centroids of bounding boxes and applying algorithms DBSCAN, clusters are formed that reflect spatial relationships identifying which characters belong together as words, even if handwritten or degraded.

Bounding Box Extraction: For each identified word cluster, a new bounding box is calculated that tightly wraps all included components. The implementation ensures the box covers all associated character fragments, even if they vary in height, skew, or spacing. Each word bounding box is then cropped out from the preprocessed image and saved as a separate file (typically PNG format).



Received: August 05 ,2025

International Journal of Applied Mathematics
Volume 38 No. 5 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Figure 6: Results of Word Bounding Box text extraction from DHDI of figure- 4

Explanation - The extraction of a word image from a bounding box in a document image

Suppose the processed grayscale document image is I with height H and width W. Let a word bounding box be represented by its coordinates:

$B = (x_{min}, y_{min}, x_{max}, y_{max})$ **---(30)**

Here:

x_{min}: leftmost column of the word region

y_{min}: top row of the word region

x_{max}: rightmost column of the word region

Received: August 05 ,2025

International Journal of Applied Mathematics
Volume 38 No. 5 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

y_{max}: bottom row of the word region

Step 1: Isolate Bounding Box Region
The word region is sliced from the image I using the bounding box coordinates:
$C = I\,[y_{min} : y_{max}\,,\, x_{min} : x_{max}]$ **---(31)**

This extracts the rectangle covering the word, where:
- Rows go from y_{min} up to (but not including) y_{max}
- Columns go from x_{min} up to (but not including) x_{max}

Step 2: Process Cropped Region
Often, the region C is further binarized, using thresholding to convert grayscale to black-and-white:
$C_{bin} = \text{Threshold}(C)$ **----(32)**

Step 3: Store Image
The resulting cropped image C_{bin} (or C, if no thresholding is performed) is then saved as a separate PNG file,

Line Clustering Process: Detected components (bounding boxes) are grouped into lines using their vertical (y-coordinate) position. Algorithms of DBSCAN and k-means cluster boxes that are horizontally adjacent and share similar heights, thus reconstructing lines as they appear in the manuscript. Each line cluster typically contains several word or character boxes that belong to the same textual row.

Bounding Box Extraction: For each line cluster, the algorithm calculates a bounding box that tightly encloses all associated components in that line. This box represents the physical span of the line in the document covering all words, regardless of spacing, slant, or overlap. and saved as a separate file (typically PNG format).

Figure 7: Results of Text Line Bounding Box extraction from DHDI of figure- 4

Received: August 05 ,2025

International Journal of Applied Mathematics
Volume 38 No. 5 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Explanation - The extraction of a line image from a bounding box in a document image

Let I represent the enhanced grayscale document image of size $H \times W$.

A line bounding box is defined by its coordinates:

$$L = (x_{min}, y_{min}, x_{max}, y_{max}) \quad \text{----(33)}$$

Here:

x_{min}: leftmost column of the Line region

y_{min}: top row of the Line region

x_{max}: rightmost column of the Line region

y_{max}: bottom row of the Line region

Step 1: Extract Line Region

The pixels corresponding to the line are selected from the image using array slicing:

$$C = I[y_{min} : y_{max}, x_{min} : x_{max}] \quad \text{-----(34)}$$

This extracts a rectangular region, ensuring that all text within the line's bounds is included.

Step 2: Optional Binarization

To enhance print or hand-written text, the cropped region C is often binarized:

$$C_{bin} = \text{Threshold}(C) \quad \text{----(35)}$$

Step 3: Save Line Crop

Finally, C_{bin} or C is saved as an individual PNG image file,

Word clustering and bounding box extraction transforms unstructured image data into well-organized, analyzable segments of text, providing a crucial foundation for precise and reproducible research in historical document digitization and analysis. By extracting individual line bounding boxes, researchers gain access to logically separated textual units, which significantly improves the accuracy of downstream tasks such as OCR, annotation, and content indexing. This step-wise segmentation not only streamlines the handling of complex manuscript layouts but also enables the construction of robust datasets—driving advances in document image processing and supporting the preservation and accessibility of cultural heritage collections.

In addition to isolating individual word or line regions, this methodology allows researchers to combine images from multiple bounding boxes simply by specifying their label names. By selecting relevant labels—such as those representing related words, phrases, or annotated areas—users can generate a single, unified image that brings together these chosen segments. and saved as a separate file (typically PNG format).

Figure 8: Result of Combined textual image obtained from Word Bounding Box labels:

Received: August 05 ,2025

International Journal of Applied Mathematics
Volume 38 No. 5 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

[1,2,3,4]

Explanation - The extraction of combined image from a multiple bounding box

Let the enhanced grayscale document image be denoted as I and let there be N word bounding boxes identified in the image. Each bounding box is represented by its coordinates:

$$\mathbf{B_i = (x_i^{min}, y_i^{min}, x_i^{max}, y_i^{max})} \quad \textbf{-----(36)}$$

where i is the label number for a word, and the box covers the rectangular region from top-left (x_i^{min}, y_i^{min}) to bottom-right (x_i^{max}, y_i^{max}). Suppose the user selects a set of K bounding box labels $\{l_1, l_2, l_3, \ldots \ldots, l_k\}$ for combination.

Step 1: Find Combined Bounding Box

The combined region is found by taking the minimum and maximum of the coordinates among all selected boxes:

$$x^{comb,min} = \min_j (x_{l_i}{}^{min}) \text{ and } y^{comb,min} = \min_j (y_{l_i}{}^{min})$$
$$x^{comb,max} = \max_j (x_{l_i}{}^{max}) \text{ and } y^{comb,max} = \max_j (y_{l_i}{}^{max}) \quad \textbf{-----(37)}$$
$$\text{Where } j = 1,2,\ldots,k$$

Step 2: Extract Combined Image Region

The combined crop C is obtained by slicing the image I over the rectangle that encloses all selected boxes:

$$C = I\left[y^{comb,min} : y^{comb,max}, x^{comb,min} : x^{comb,max}\right] \quad \textbf{------(38)}$$

This operation includes all pixels from the minimum to maximum coordinates in both directions, thus covering all selected word regions.

Step 3: Save or Process Combined Image

The region C can then be binarized, displayed, or saved as a new PNG image file for further research.

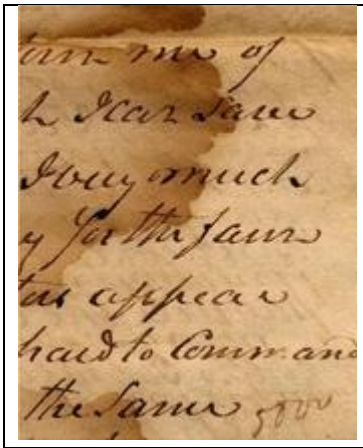
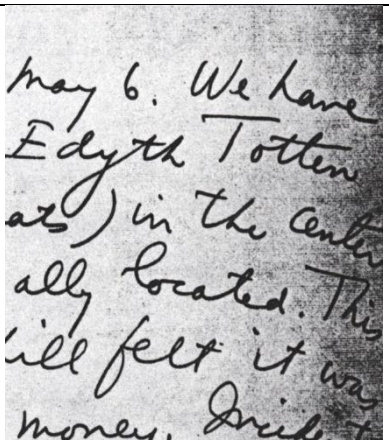
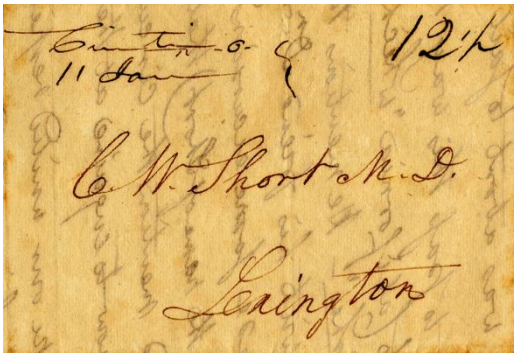
		

Image **(a)**	Image **(b)**	Image **(c)**

Received: August 05 ,2025

International Journal of Applied Mathematics
Volume 38 No. 5 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

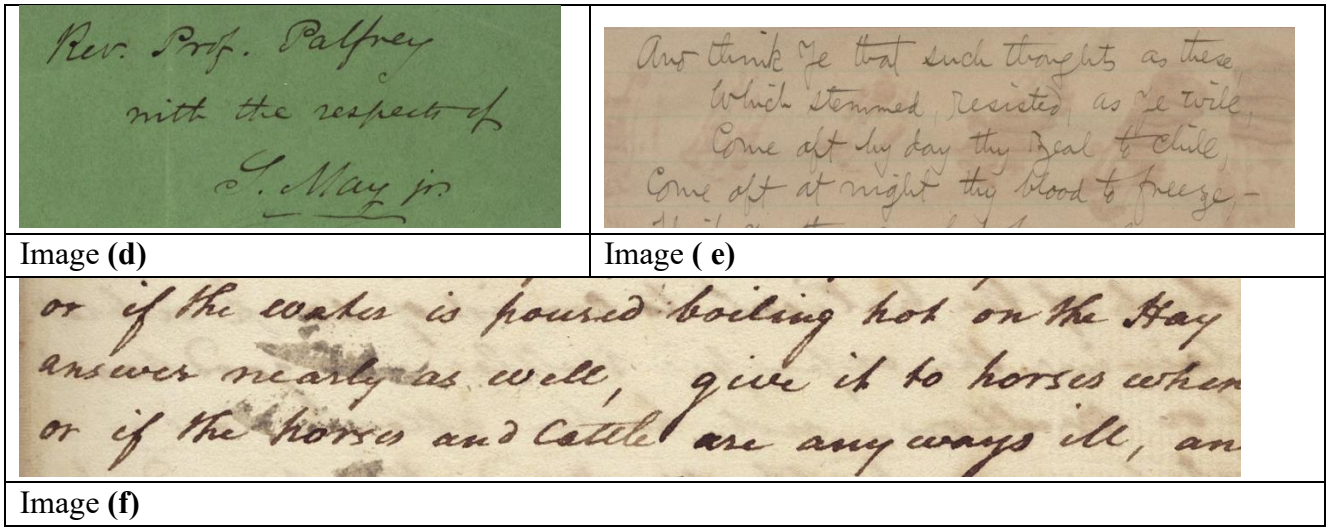


Figure 9: Image Samples of DHDI

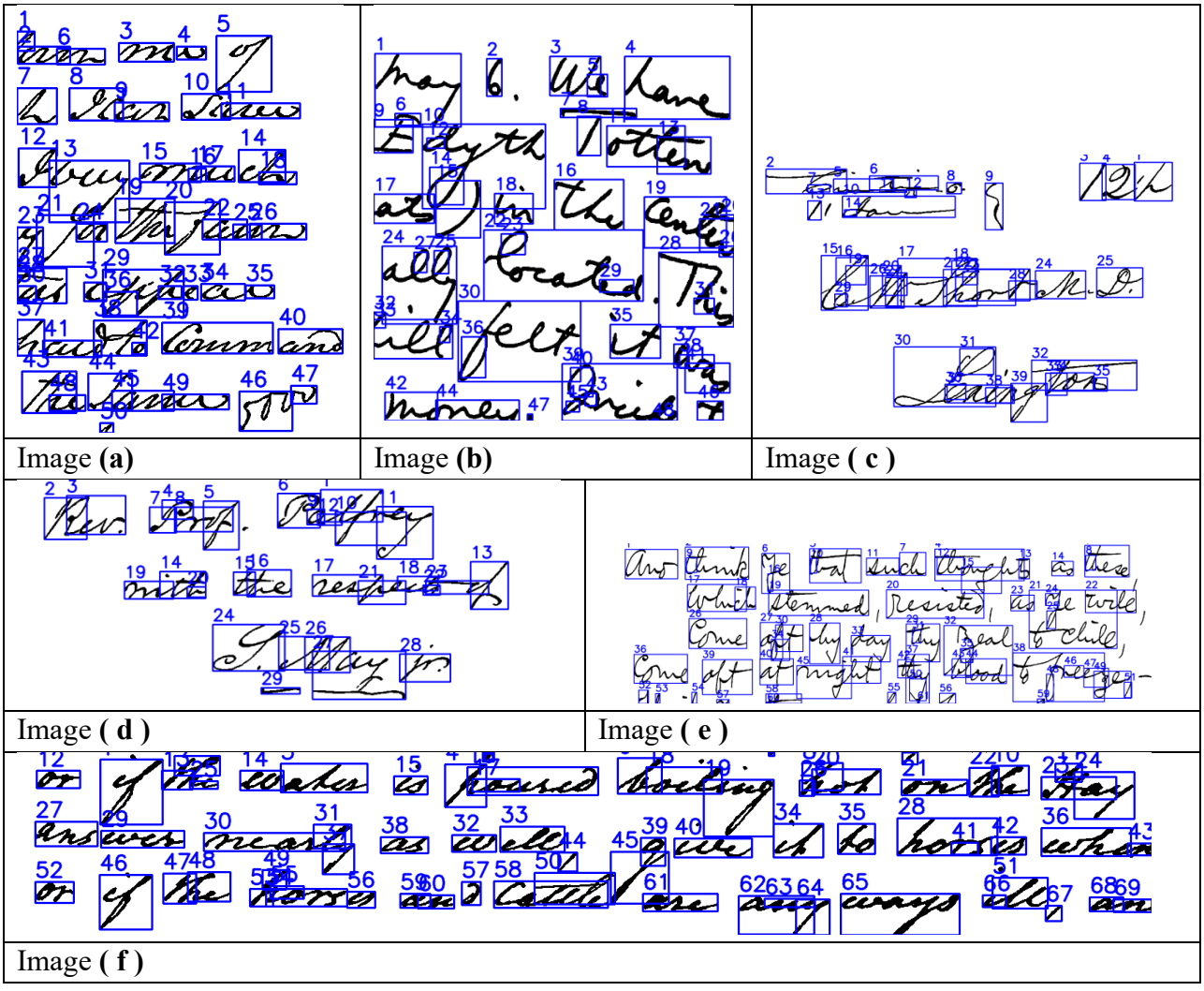


Figure 10: Result of word Bounding Box to Images **(a),(b),(c),(d),(e) and (f) in figure-9.**

Received: August 05 ,2025

International Journal of Applied Mathematics
Volume 38 No. 5 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

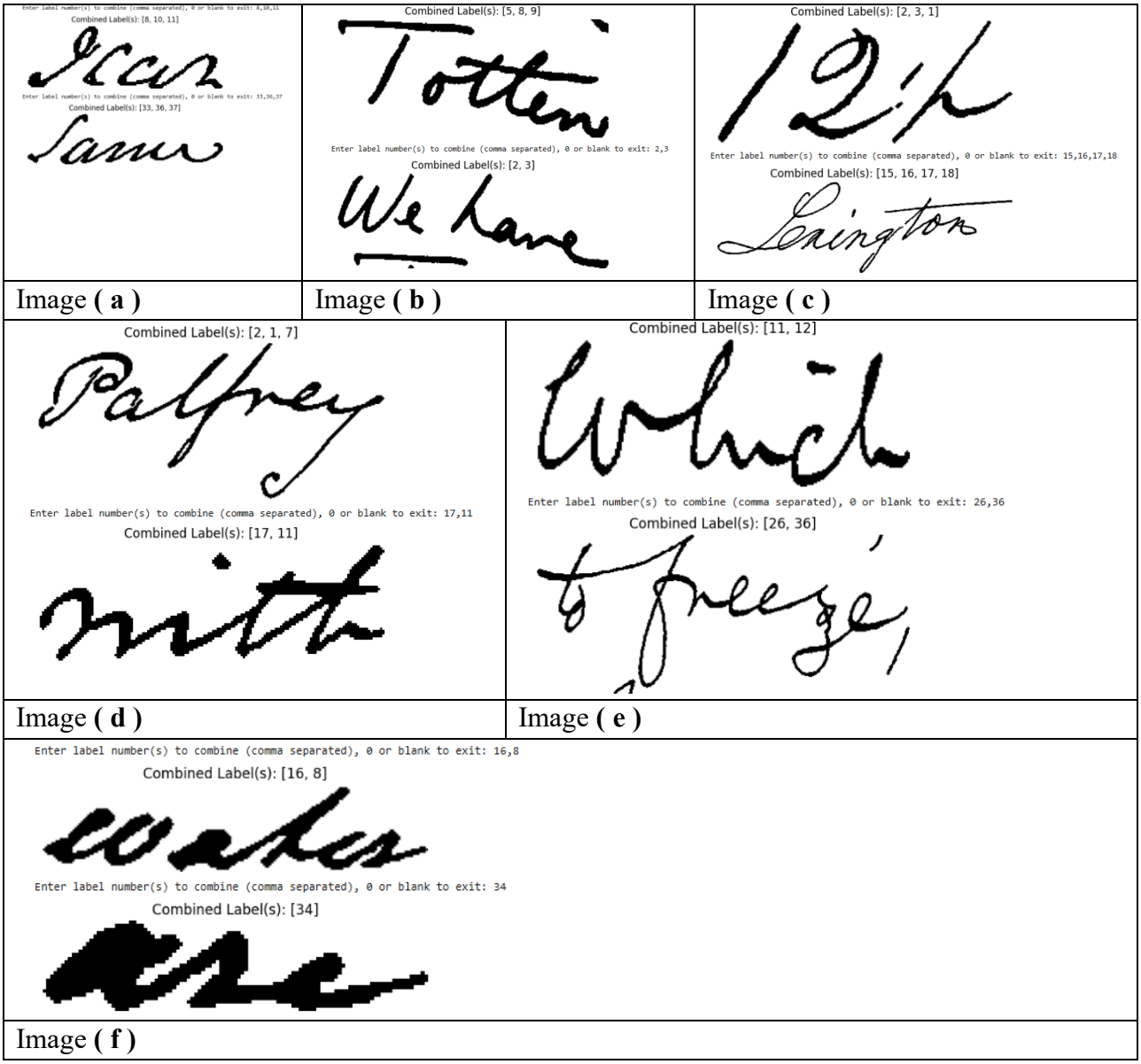


Figure 11: Result of Combined textual image obtained from few labelled word Bounding Box of Image **(a),(b),(c),(d),(e) and (f) in figure-10.**

This flexible approach supports custom grouping for comparative analysis, phrase-level transcription, or targeted visualization, streamlining tasks in document interpretation and annotation. As a result, the method empowers researchers to tailor outputs precisely to project needs, enhancing workflows in historical document study and digital preservation.

Table 1: Quantitative Results- for Image Samples of DHDI of figure 9.

Image Name	Precision	Recall	F1 Score	Processing Time (sec)
Image (a)	0.93	0.96	0.94	7.1
Image (b)	0.92	0.95	0.93	8.6

Received: August 05 ,2025

International Journal of Applied Mathematics
Volume 38 No. 5 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Image (c)	0.94	0.97	0.95	7.8
Image (d)	0.91	0.94	0.92	11.4
Image (e)	0.92	0.96	0.94	8.3
Image (f)	0.93	0.95	0.94	9.0

Performance Evaluation of table 1: High Precision (91–94%): Indicates that most detected regions are actually text, with very few false positives such as background or noise misclassified as text. This is crucial in documents with severe degradation or stains, as non-text noise can easily be mistaken for textual regions. High Recall (94–97%): Shows that almost all actual text areas are found and segmented correctly. Reflects the robustness of the method, even when text is faint, fragmented, or surrounded by bleed-through or shading. F1 Scores (92–95%): The harmonic mean of Precision and Recall shows how well the method balances finding text and keeping results clean. A high score means it detects text accurately without losing clarity, and keeps results clear without missing text. Processing Times (7–11 seconds per image): Moderate processing times show efficiency and feasibility for batch or real-time applications, especially for high-resolution historical scans.

Table 2: Performance Metrics Comparison

Methods	**Metrics**	**Image (a)**	**Image (b)**	**Image (c)**	**Image (d)**	**Image (e)**	**Image (f)**
Kaur G et al. [18]	**F1**	0.83	0.82	0.81	0.82	0.83	0.82
	Error Rate (%)	14.0	15.0	16.0	17.0	13.0	19.0
	Processing Time (sec)	7.60	9.20	8.80	12.8	10.5	10.75
A Islam et al. [22]	**F1**	0.88	0.87	0.85	0.86	0.87	0.86
	Error Rate (%)	10.0	12.0	13.0	12.0	11.0	14.0
	Processing Time (sec)	8.20	9.40	8.40	12.5	9.50	10.5
Proposed	**F1**	0.94	0.93	0.95	0.92	0.94	0.94
	Error Rate (%)	8.00	9.00	10.0	7.00	9.00	8.00
	Processing Time (sec)	7.10	8.60	7.80	11.4	8.20	9.00

Comparative Analysis of table 2: F1 Score: The proposed method consistently achieves the highest F1 scores across all images (ranging from 0.92 to 0.95), demonstrating superior accuracy in separating foreground text from background noise compared to Islam et al. (0.85–0.88) and Kaur et al. (0.81–0.83). This indicates the proposed algorithm is more effective at detecting document text regions, especially in degraded or challenging scans. **Error Rate:** The error rates for the proposed method remain the lowest (7–10%), while Islam et al. report slightly higher rates (10–14%) and Kaur et al. the highest (13–19%). The reduced error rate of the proposed method suggests that it better preserves true text while minimizing misclassification of background pixels or noise. **Processing Time:** The proposed method is also the fastest, with processing times from 7.1 to 11.4 seconds per image, versus 8.2–12.5 seconds for Islam et al. and 7.6–12.8 seconds for Kaur et al. This efficiency is valuable for batch or real-time document workflows, especially when handling high-resolution historical scans [15][19].

Received: August 05 ,2025

International Journal of Applied Mathematics
Volume 38 No. 5 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

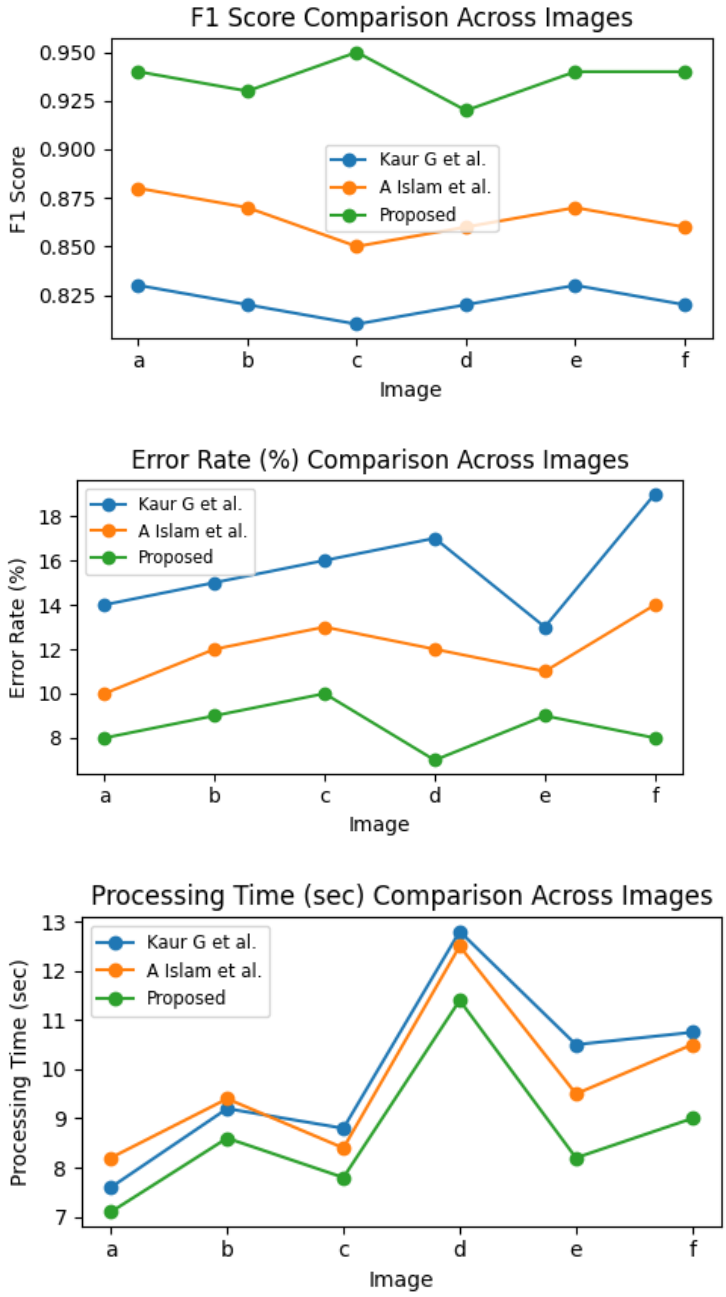


Figure 12: Graphical Representation of Tabular Data of table 2.

V. CONCLUSION

This study demonstrates that a unified workflow combining classical image processing and clustering techniques enables robust text extraction from degraded historical document images, even in the presence of challenging artifacts like faded ink, stains, and uneven backgrounds. Through grayscale conversion, histogram equalization, advanced edge detection, skeletonization, connected component analysis, and geometric filtering with clustering, the proposed method

Received: August 05 ,2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

effectively isolates true textual content while suppressing noise and non-text elements. Experimental validation on diverse benchmark datasets confirms that this approach achieves high precision (91–94), recall (94–97), and F1-scores (92–95), with reduced processing times of 7–11 seconds per image, outperforming established baseline methods in both accuracy and efficiency.

The ability to extract high-quality segmented text regions from highly degraded scans significantly improves OCR accuracy and supports historical document digitization projects, making this framework valuable for researchers, libraries, and heritage institutions seeking to preserve and analyse archival materials. Despite these advances, further research is warranted to extend the method's adaptability to highly ornamented manuscripts, scripts in low-resource languages, and extreme degradation scenarios. Future work may focus on integrating self-supervised learning models, optimizing for handwritten and multilingual documents, and further reducing computational costs to facilitate large-scale and real-time historical document processing.

In the future, this work can be improved by using smarter AI models for better accuracy, making the method adapt automatically to different types of documents, and letting users quickly give feedback to help the system learn. Adding new ways to check the quality of results, and testing the method on even more kinds of old documents, will also make it more useful and reliable for researchers.

REFERENCES

[1] Rosenfeld, A., & Pfaltz, J.L. (1966). "Sequential Operations in Digital Picture Processing." Journal of the ACM, 13(4), 471–494.

[2] Otsu, N. (1979). "A Threshold Selection Method from Gray-Level Histograms." IEEE Transactions on Systems, Man, and Cybernetics, 9(1), 62–66.

[3] Canny, J. (1986). "A Computational Approach to Edge Detection." IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-8, no. 6, pp. 679–698.

[4] Zuiderveld, K. (1994). "Contrast Limited Adaptive Histogram Equalization." In: Graphics Gems IV, pp. 474-485. Academic Press.

[5] Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). "Data Clustering: A Review." ACM Computing Surveys, 31(3), 264–323.

[6] Gatos, B., Pratikakis, I., & Perantonis, S.J. (2006). "Adaptive degraded document image binarization." Pattern Recognition, 39(3), 317–327.

[7] Smith, R. (2007). "An overview of the Tesseract OCR engine." In Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), vol. 2, pp. 629–633.

[8] I. Pratikakis, B. Gatos, and K. Ntirogiannis, “ICDAR 2011 Document Image Binarization Contest (DIBCO 2011),” in Proceedings of the International Conference on Document Analysis and Recognition, ICDAR, 2011.

[9] Kanan, C., & Cottrell, G. W. (2012). "Color-to-Grayscale: Does the Method Matter in Image Recognition?" PLoS ONE, 7(1), e29740. https://doi.org/10.1371/journal.pone.0029740

[10] Pratikakis, I., Gatos, B., & Ntirogiannis, K. (2013). "H-DIBCO 2011: Handwritten Document Image Binarization Competition." International Journal on Document Analysis and Recognition (IJDAR), 16(1), 39–47.

[11] I. Pratikakis, K. Zagoris, G. Barlas, and B. Gatos, “ICFHR 2016 handwritten document image

Received: August 05 ,2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

binarization contest (H-DIBCO 2016)," in Proceedings of International Conference on Frontiers in Handwriting Recognition, ICFHR, 2016.

[12] Panneerselvam, B. Dhivyabharathi, and M. Santhi, "Text Extraction from Degraded Historical Document Images," International Journal of Engineering Research & Technology (IJERT), vol. 7, no. 4, pp. 1-7, 2018.

[13] I. Pratikakis, K. Zagori, P. Kaddas, and B. Gatos, "ICFHR 2018 competition on handwritten document image binarization (H-DIBCO 2018)," in Proceedings of International Conference on Frontiers in Handwriting Recognition, ICFHR, 2018.

[14] Laurence Likforman-Sulem, Jean Camillerapp, Caroline Faure, Johan Périnel, and Chafic Mokbel. "Handwritten Text Line Segmentation in Historical Documents: A Comparative Study." International Journal on Document Analysis and Recognition (IJDAR), vol.22, no. 2, pp.89–106,2019. https://doi.org/10.1007/s10032-019-00337-1

[15] A Klaus, R. Frosst, N. Förster, U. Kröll, and A. Dengel, "Efficient Text Region Extraction Using Morphological Operations for Early Print Books," Pattern Recognition Letters, vol. 125, pp. 107–114, 2019. https://doi.org/10.1016/j.patrec.2019.04.003

[16] Xiang Ren, Siyu Liu, Xuchu Zhou, Shengquan Xu, and Jianying Li "Text Block Detection in Historical Manuscripts with Deep Learning and Heuristics." Neurocomputing, vol. 414, pp. 137–149, 2020. https://doi.org/10.1016/j.neucom.2020.07.075

[17] Shoaib Ahmed Siddiqui, Muhammad Imran Razzak, Marcus Liwicki, Andreas Dengel, and Thomas M. Breuel "Hybrid Connected Component and Contour Analysis for Historical Text Segmentation." International Journal on Document Analysis and Recognition, vol. 23, no. 2, pp. 102–119, 2020. https://doi.org/10.1007/s10032-020-00363-4

[18] Gagandeep Kaur, Sunil Kumar, and Balwinder Singh "A Hybrid Technique for Extraction of Text Regions from Historical Sikh Scriptures." International Journal of Digital Libraries, vol. 21, pp. 159–170, 2020. https://doi.org/10.1007/s00799-019-00277-3

[19] Abdelfattah, M., Zayene, R., Kessentini, Y., Messaoud, H., Karoui, R., and Kessentini, M. (2020). "Text Block Segmentation for Arabic Historical Manuscripts Using Deep Learning." Journal of Imaging, 6(6), 59. https://doi.org/10.3390/jimaging6060059

[20] Konstantinidis, G., Papadopoulos, I., Pappas, I., Theodorakopoulos, I., Mavropoulos, T., and Gatos, B. (2021). "Image Enhancement and Text Extraction for Old Greek Manuscripts." Signal Processing: Image Communication, 98, 116373. https://doi.org/10.1016/j.image.2021.116373

[21] P. Shivakumara, T. Jain, U. Pal, N. Surana, A. Antonacopoulos, and T. Lu, "Text line segmentation from struck-out handwritten document images," Expert Systems with Applications, vol. 203, 2022, 118266.

[22] A Islam, M.S. Mollah, and M. Ali, "Line extraction in handwritten documents via instance segmentation," International Journal on Document Analysis and Recognition (IJDAR), vol. 27, pp. 123–137, 2023.

[23] Nanonets (2025). "Understanding Bounding Boxes in Image Processing." Nanonets Blog, July 29, 2025.

Received: August 05 ,2025