AI Power Document Scanner with Optical Character Recognition (OCR)

Introduction

Document scanning and text digitization are fundamental tasks in the fields of document management, archival systems, and digital transformation. Conventional scanners rely on hardware mechanisms to capture document images and are limited by physical setup, cost, and lack of adaptability to diverse document types and conditions. With advancements in computer vision and machine learning, it is now possible to replicate and even improve these scanning processes through intelligent algorithms that can automatically detect, enhance, and interpret document images.

Motivation

In academic, corporate, and governmental sectors, the need to convert physical documents into searchable digital formats has significantly increased. However, uneven lighting, shadows, rotations, and low-quality captures often reduce readability and OCR performance. The motivation behind this project is to build a low-cost, software-based solution that overcomes these challenges using AI-driven image processing techniques to deliver clean and high-quality scanned documents.

Objectives

The main objectives of this project are:

1. To design a robust image-processing pipeline that automatically detects and enhances document images.
2. To implement contour detection and perspective correction for document alignment.
3. To integrate OCR (Optical Character Recognition) for converting scanned images into editable, searchable text.
4. To create an adaptive preprocessing system suitable for both printed and handwritten text.
5. To ensure the system operates efficiently on desktop environments without requiring specialized hardware.

Scope and Applications

The proposed AI-powered document scanner can be used in multiple scenarios, including:

- Digitizing academic notes, receipts, and official documents.
- Archiving handwritten or printed records.
- Automating OCR-based data entry tasks.
- Preprocessing documents for NLP (Natural Language Processing) applications

This project combines classical image processing methods with intelligent preprocessing strategies to produce high-quality digital copies and readable OCR outputs, bridging the gap between traditional scanning hardware and modern AI-based document digitization solutions.

Literature Review

Overview

Document scanning and Optical Character Recognition (OCR) have evolved from basic hardware-based systems to advanced software-driven solutions. Earlier scanners required flat surfaces and stable lighting, which made them less flexible. With the progress of image processing and artificial intelligence, it is now possible to automatically detect, enhance, and extract text from images captured using regular cameras.

Traditional Methods

Traditional document scanning mainly relied on fixed hardware devices that captured static images of pages. These methods worked well under controlled lighting but often failed when documents were tilted, wrinkled, or poorly illuminated. Early OCR systems could recognize printed text but struggled with low-quality or handwritten inputs. As a result, scanned outputs often contained unclear or missing text.

Image Processing Advancements

Libraries like OpenCV enabled flexible, software-based scanning using methods such as:

- **Adaptive Thresholding:** Handles uneven illumination.
- **Morphological Operations:** Reduces noise and improves text structure.
- **Edge Detection & Perspective Transform:** Finds document boundaries and corrects shape distortions.

OCR and AI Enhancements

Tesseract OCR, developed by Google, converts enhanced images into text. With AI, OCR systems can now handle multiple fonts, handwriting, and noisy inputs. Deep learning and NLP-based spell correction improve text accuracy and readability.

Research Gap and Contribution

Many existing scanners fail in poor lighting or non-flat documents. This project overcomes such issues by combining advanced preprocessing (CLAHE, denoising, morphology) with OCR and providing both **automatic and manual corner detection**. It delivers a lightweight, portable, and accurate desktop solution for document digitization.

Methodology & Implementation (Code)

This project follows a step-by-step process to convert a captured document image into a clean, enhanced, and readable digital version. The system performs several image processing and AI-based operations to detect, correct, and extract text from the document.

The main stages of the methodology are described below.

Step 0: Prerequisites

Install Python3 from https://www.python.org/downloads/

Create a file named `requirements.txt` with these lines:

```
blinker==1.9.0
click==8.3.0
fpdf==1.7.2
inexactsearch==1.0.2
itsdangerous==2.2.0
Jinja2==3.1.6
MarkupSafe==3.0.3
numpy==2.2.6
opencv-contrib-python==4.12.0.88
opencv-python==4.12.0.88
packaging==25.0
pillow==12.0.0
pyspellchecker==0.8.3
pytesseract==0.3.13
silpa-common==0.3
```

```
soundex==1.1.3
spellchecker==0.4
Werkzeug==3.1.3
```

And Install everything with below command:

```
pip install -r requirements.txt
```

Step 1: Image Acquisition

The process begins with loading an image of a document, either from a file or captured using a camera. The image is then read using the **OpenCV** library and prepared for further processing.

This step ensures the input image is properly loaded and converted into a suitable format for enhancement and analysis.

```
# libs and init
import os
import cv2
import numpy as np
import pytesseract
from spellchecker import SpellChecker
from PIL import Image
from matplotlib import pyplot as plt

# Image Path specifications
IMG = "/home/krushang/radhey/personal/college/Doc-Scanner-Project/test/printed_eng.png"
OUTPUT_DIR = "./out"
```

```
# helper function to see image output

def show(window_name, img, wait=True, max_width=1000, max_height=800):
    h, w = img.shape[:2]
    scale = min(max_width / w, max_height / h, 1.0)
    if scale < 1.0:
        new_w, new_h = int(w * scale), int(h * scale)
        img = cv2.resize(img, (new_w, new_h), interpolation=cv2.INTER_AREA)

    plt.figure(figsize=(10, 8))
    if len(img.shape) == 2:  # grayscale
```

```
    plt.imshow(img, cmap='gray')
  else:  # color
    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
  plt.title(window_name)
  plt.axis('off')
  plt.show()
```

Step 2: Image Preprocessing

The preprocessing stage improves the overall quality of the document image. It involves several sub-steps:

1. **Grayscale Conversion:**
   The image is converted from color (RGB) to grayscale to simplify processing and focus on text structure.
2. **Contrast Enhancement (CLAHE):**
   CLAHE (Contrast Limited Adaptive Histogram Equalization) is applied to improve contrast in low-light or uneven images.

3. **Noise Reduction:**
   Noise is removed using **Non-local Means Denoising** and **Bilateral Filtering** to make text clearer while preserving edges.
4. **Thresholding:**
   Adaptive or Otsu's thresholding is applied to create a binary (black and white) image, making text stand out from the background.
5. **Morphological Operations:**
   Techniques like opening and closing are used to refine text lines, remove small dots, and make characters more readable.
6. **Upscaling:**
   The image is enlarged to improve OCR accuracy, especially for handwritten or small-text documents.

Each step is visualized in the Jupyter Notebook using the custom `show()` function to display the output of every stage.

```
# preproceding pip line
def preprocess_for_handwriting(image, mode='normal', threshold_type='adaptive'):
    print("Pre Processing Pipline")

    # Convert to grayscale
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

```
print("Grayscale")
show("1. Grayscale", gray)

# Increase contrast using CLAHE
clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8, 8))
contrast_enhanced = clahe.apply(gray)
print("CLAHE Contrast Enhancement")
show("2. Contrast Enhanced", contrast_enhanced)

# Denoise while preserving edges (critical for handwriting)
denoised = cv2.fastNlMeansDenoising(
    contrast_enhanced, None, h=10, templateWindowSize=7, searchWindowSize=21
)
print("Denoising")
show("3. Denoised", denoised)

# Apply bilateral filter to smooth while keeping edges
bilateral = cv2.bilateralFilter(denoised, 9, 75, 75)
print("Bilateral Filter")
show("4. Bilateral Filter", bilateral)

# Apply chosen thresholding method
print(f"Applying {threshold_type.upper()} thresholding...")

adaptive_thresh = None
normal_thresh = None
thresh = None

if threshold_type == 'adaptive':
    adaptive_thresh = cv2.adaptiveThreshold(
        bilateral, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
        cv2.THRESH_BINARY, 21, 10
    )
    thresh = adaptive_thresh
    print("Adaptive Thresholding Applied")
    show("5. Adaptive Threshold", thresh)

elif threshold_type == 'normal':
    _, normal_thresh = cv2.threshold(
        bilateral, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU
    )
    thresh = normal_thresh
    print("Normal (Otsu) Thresholding Applied")
    show("5. Normal (Otsu) Threshold", thresh)

else:
```

```
        print("Invalid threshold_type; defaulting to Adaptive.")
        thresh = cv2.adaptiveThreshold(
            bilateral, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
            cv2.THRESH_BINARY, 21, 10
        )
        show("5. Adaptive Threshold (Default)", thresh)

    # Morphological operations for thin/thick character adjustment
    print("Morphological Operations for Thickness Adjustment")
    if mode == 'thin':
        kernel_open = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (1, 1))
        kernel_close = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (2, 2))
    elif mode == 'thick':
        kernel_open = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3, 3))
        kernel_close = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (4, 4))
    else:
        kernel_open = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (2, 2))
        kernel_close = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3, 3))

    morph_open = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel_open, iterations=1)
    morph_cleaned = cv2.morphologyEx(morph_open, cv2.MORPH_CLOSE, kernel_close,
iterations=1)
    show("6. Morphological Cleanup", morph_cleaned)

    # Invert if needed (text should be black on white)
    if np.mean(morph_cleaned) > 127:
        morph_cleaned = cv2.bitwise_not(morph_cleaned)

    # Upscale for better OCR (handwriting needs higher resolution)
    h, w = morph_cleaned.shape
    scale_factor = 2
    if h < 1500:
        scale_factor = 3

    upscaled = cv2.resize(
        morph_cleaned,
        None,
        fx=scale_factor,
        fy=scale_factor,
        interpolation=cv2.INTER_CUBIC,
    )
    print("Upscaling")
    show("7. Upscaled (Final)", upscaled)

    return image, bilateral, upscaled
```

```
def preprocessing(img_path, out_dir="./out", mode='normal', threshold_type='adaptive'):
    os.makedirs(out_dir, exist_ok=True)

    print(f"Loading IMAGE: {img_path}")
    image = cv2.imread(img_path)
    if image is None:
        raise RuntimeError(f"Cannot read image: {img_path}")

    # For handwriting, always use specialized preprocessing
    return preprocess_for_handwriting(image, mode, threshold_type)
```

Step 3: Document Boundary Detection

In this stage, the system locates the document area within the image.
Three options are available:

1. **Automatic Detection:** Using edge detection (Canny) and contour analysis to find the four corners of the document.
2. **Manual Selection:** The user can click on the four document corners if automatic detection fails.
3. **Full Image Mode:** Used if the document occupies the entire image.

Once detected, the corner points are ordered properly to define the document's shape.

```
# contour detection
def order_points(pts):
    rect = np.zeros((4, 2), dtype="float32")
    s = pts.sum(axis=1)
    rect[0] = pts[np.argmin(s)]
    rect[2] = pts[np.argmax(s)]
    diff = np.diff(pts, axis=1)
    rect[1] = pts[np.argmin(diff)]
    rect[3] = pts[np.argmax(diff)]
    return rect


def manual_corner_selection(image):
    points = []
    clone = image.copy()

    def click_event(event, x, y, flags, params):
        if event == cv2.EVENT_LBUTTONDOWN and len(points) < 4:
```

```
            points.append([x, y])
            cv2.circle(clone, (x, y), 10, (0, 0, 255), -1)
            cv2.putText(
                clone,
                str(len(points) - 1),
                (x, y),
                cv2.FONT_HERSHEY_SIMPLEX,
                1,
                (255, 0, 0),
                2,
            )
            if len(points) > 1:
                cv2.line(clone, tuple(points[-2]), tuple(points[-1]), (0, 255, 0), 2)
            if len(points) == 4:
                cv2.line(clone, tuple(points[-1]), tuple(points[0]), (0, 255, 0), 2)
            cv2.imshow("Select 4 Corners", clone)

    cv2.namedWindow("Select 4 Corners")
    cv2.setMouseCallback("Select 4 Corners", click_event)
    cv2.imshow("Select 4 Corners", clone)

    print("Click 4 corners in order: top-left, top-right, bottom-right, bottom-left")
    print("Press any key when done...")

    cv2.waitKey(0)
    cv2.destroyWindow("Select 4 Corners")

    if len(points) == 4:
        return np.array(points, dtype=np.float32)
    return None


def contour_detection(image, processed):
    h, w = image.shape[:2]

    print("\nContour Detection Options:")
    print("1. Auto-detect document edges")
    print("2. Use whole image")
    print("3. Manual corner selection")

    choice = input("Enter choice (1-3): ").strip()

    if choice == "2":
        print("Using whole image...")
        doc_contour = (
            np.array([[0, 0], [w - 1, 0], [w - 1, h - 1], [0, h - 1]], dtype=np.float32)
```

```
            .reshape(4, 1, 2)
            .astype(np.int32)
        )
        return doc_contour

    elif choice == "3":
        doc_contour = manual_corner_selection(image)
        if doc_contour is not None:
            doc_contour = order_points(doc_contour.reshape(4, 2))
            doc_contour = doc_contour.reshape(4, 1, 2).astype(np.int32)
            return doc_contour
        else:
            print("Manual selection failed. Using whole image.")
            doc_contour = (
                np.array(
                    [[0, 0], [w - 1, 0], [w - 1, h - 1], [0, h - 1]], dtype=np.float32
                )
                .reshape(4, 1, 2)
                .astype(np.int32)
            )
            return doc_contour

    # Auto-detect (option 1)
    edges = cv2.Canny(processed, 50, 150)
    kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5))
    edges = cv2.dilate(edges, kernel, iterations=2)

    contours, _ = cv2.findContours(edges, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)

    if contours:
        contours = sorted(contours, key=cv2.contourArea, reverse=True)[:10]

        for contour in contours:
            peri = cv2.arcLength(contour, True)
            approx = cv2.approxPolyDP(contour, 0.02 * peri, True)

            if len(approx) == 4:
                doc_contour = order_points(approx.reshape(4, 2))
                doc_contour = doc_contour.reshape(4, 1, 2).astype(np.int32)

                # Show detected contour
                contour_img = image.copy()
                cv2.drawContours(contour_img, [doc_contour], -1, (0, 255, 0), 3)
                show("Auto-detected Contour", contour_img)

                return doc_contour
```

```
    print("Auto-detection failed. Using whole image.")
    doc_contour = (
        np.array([[0, 0], [w - 1, 0], [w - 1, h - 1], [0, h - 1]], dtype=np.float32)
        .reshape(4, 1, 2)
        .astype(np.int32)
    )
    return doc_contour
```

Step 4: Perspective Transformation

After identifying the document boundaries, a **perspective transform** is applied to "flatten" the document.
This step corrects any tilt or angle distortions, giving the appearance of a properly scanned page. The output is a top-down view of the document with straight edges and correct proportions.

```
# Perspective Transformation
def perspective_transform(image, contour):
    rect = order_points(contour.reshape(4, 2))
    (tl, tr, br, bl) = rect

    widthA = np.sqrt(((br[0] - bl[0]) ** 2) + ((br[1] - bl[1]) ** 2))
    widthB = np.sqrt(((tr[0] - tl[0]) ** 2) + ((tr[1] - tl[1]) ** 2))
    maxWidth = max(int(widthA), int(widthB))

    heightA = np.sqrt(((tr[0] - br[0]) ** 2) + ((tr[1] - br[1]) ** 2))
    heightB = np.sqrt(((tl[0] - bl[0]) ** 2) + ((tl[1] - bl[1]) ** 2))
    maxHeight = max(int(heightA), int(heightB))

    dst = np.array(
        [[0, 0], [maxWidth - 1, 0], [maxWidth - 1, maxHeight - 1], [0, maxHeight - 1]],
        dtype="float32",
    )

    M = cv2.getPerspectiveTransform(rect, dst)
    warped = cv2.warpPerspective(image, M, (maxWidth, maxHeight))

    show("Warped Document", warped)
    return warped
```

Step 5: Optical Character Recognition (OCR)

The enhanced and corrected document is passed through **Tesseract OCR**, which extracts readable and searchable text from the image.
The extracted text is displayed or saved for further use.

```
def select_language():
    print("\nSelect language for OCR:")
    print("1. English")
    print("2. Hindi")
    print("3. Gujarati")

    choice = input("Enter choice (1-3): ").strip()
    language_map = {
        "1": ("eng", "English"),
        "2": ("hin", "Hindi"),
        "3": ("guj", "Gujarati"),
    }

    if choice in language_map:
        return language_map[choice]
    else:
        print("Invalid choice. Using English by default.")
        return ("eng", "English")


def text_extraction(image, lang_code="eng", lang_name="English",
output_file="extracted_text.txt"):
    print(f"\nPerforming OCR ({lang_name})...")

    # Ensure grayscale
    if len(image.shape) == 3:
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    else:
        gray = image.copy()

    # Auto-invert if background is dark (for better OCR)
    if np.mean(gray) < 100:
        gray = cv2.bitwise_not(gray)

    # Multiple OCR configurations for better results
    ocr_configs = [
```

```
        ("--oem 1 --psm 6", "Uniform text block"),
        ("--oem 1 --psm 4", "Single column text"),
        ("--oem 1 --psm 3", "Fully automatic layout"),
        ("--oem 1 --psm 11", "Sparse text"),
    ]

    results = []

    for config, description in ocr_configs:
        try:
            print(f"  Trying: {description}...")

            data = pytesseract.image_to_data(
                gray, config=f"{config} -l {lang_code}",
                output_type=pytesseract.Output.DICT
            )

            confidences = [int(c) for c in data["conf"] if str(c).isdigit() and int(c) > 0]
            avg_confidence = sum(confidences) / len(confidences) if confidences else 0

            text = pytesseract.image_to_string(gray, config=f"{config} -l {lang_code}")
            text = text.replace("\x0c", "").strip()

            word_count = len([w for w in text.split() if len(w) > 2])
            quality_score = avg_confidence * (1 + word_count / 100)

            results.append({
                "text": text,
                "confidence": avg_confidence,
                "quality": quality_score,
                "method": description,
            })

            print(f"    Confidence: {avg_confidence:.1f}%, Words: {word_count}, Score:
{quality_score:.1f}")

        except Exception as e:
            print(f"    Failed: {e}")
            continue

    if not results:
        print("All OCR attempts failed!")
        return "", 0, "None"

    best_result = max(results, key=lambda x: x["quality"])
```

```
    # Print a preview of extracted text
    print("\n" + "=" * 60)
    print("EXTRACTED TEXT PREVIEW:")
    print("=" * 60)
    print(best_result["text"][:500] if best_result["text"] else "[No text detected]")
    print("=" * 60)

    # Save extracted text to file
    with open(output_file, "w", encoding="utf-8") as f:
        f.write("AI Powered Document Scanner - OCR Result\n")
        f.write("=" * 60 + "\n")
        f.write(f"Language: {lang_name}\n")
        f.write(f"OCR Method: {best_result['method']}\n")
        f.write(f"Confidence: {best_result['confidence']:.2f}%\n\n")
        f.write("Extracted Text:\n")
        f.write("-" * 60 + "\n")
        f.write(best_result["text"])

    print(f"\nExtracted text saved to: {output_file}")

    return best_result["text"], best_result["confidence"], best_result["method"]
```

Step 6: Post-Processing

The recognized text may contain small spelling or recognition errors. To improve accuracy:

- A **spell checker** is applied to correct common mistakes.
- The text is formatted into a clean, readable structure for saving or exporting.

```
def ai_spell_correction(text, lang_code):
    if "eng" not in lang_code:
        print("Spell checking only available for English text.")
        return text

    spell = SpellChecker()
    lines = text.split("\n")
    corrected_lines = []
    corrections_made = 0

    print("\nPerforming AI Spell Correction...")

    for line in lines:
```

```
        if not line.strip():
            corrected_lines.append(line)
            continue

        words = line.split()
        corrected_words = []

        for word in words:
            prefix, suffix = "", ""
            clean_word = word

            while clean_word and not clean_word[0].isalnum():
                prefix += clean_word[0]
                clean_word = clean_word[1:]

            while clean_word and not clean_word[-1].isalnum():
                suffix = clean_word[-1] + suffix
                clean_word = clean_word[:-1]

            if clean_word and clean_word.isalpha() and len(clean_word) > 2:
                correction = spell.correction(clean_word.lower())
                if correction and correction != clean_word.lower():
                    if clean_word[0].isupper():
                        correction = correction.capitalize()
                    corrected_words.append(prefix + correction + suffix)
                    corrections_made += 1
                else:
                    corrected_words.append(word)
            else:
                corrected_words.append(word)

        corrected_lines.append(" ".join(corrected_words))

    print(f"Corrections made: {corrections_made}")
    return "\n".join(corrected_lines)
```

Step 7: Output Generation

Finally, the processed document and extracted text are saved to an output folder.
The result includes:

- The enhanced scanned image.
- A text file containing the recognized text.
- Optional visual outputs displayed through Matplotlib for step-by-step understanding.

```
def save_processed_image(image, filename):
    output_path = os.path.join(OUTPUT_DIR, filename)
    cv2.imwrite(output_path, image)
    print(f"Saved: {output_path}")
    return output_path


def enhance_image_quality(image):
    if len(image.shape) == 2:
        image_rgb = cv2.cvtColor(image, cv2.COLOR_GRAY2BGR)
    else:
        image_rgb = image

    lab = cv2.cvtColor(image_rgb, cv2.COLOR_BGR2LAB)
    l, a, b = cv2.split(lab)
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    l = clahe.apply(l)
    enhanced = cv2.merge([l, a, b])
    enhanced = cv2.cvtColor(enhanced, cv2.COLOR_LAB2BGR)

    kernel = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]])
    enhanced = cv2.filter2D(enhanced, -1, kernel)

    return enhanced

def convert_to_high_quality_jpg(image, output_filename):
    output_path = os.path.join(OUTPUT_DIR, output_filename)
    enhanced = enhance_image_quality(image)
    pil_image = Image.fromarray(cv2.cvtColor(enhanced, cv2.COLOR_BGR2RGB))
    pil_image.save(output_path, "JPEG", quality=95, optimize=True, dpi=(300, 300))
    print(f"High-quality JPG saved: {output_path}")
    return output_path


def convert_image_to_pdf(image, output_filename):
    output_path = os.path.join(OUTPUT_DIR, output_filename)
    enhanced = enhance_image_quality(image)
    pil_image = Image.fromarray(cv2.cvtColor(enhanced, cv2.COLOR_BGR2RGB))
    pil_image.save(output_path, "PDF", resolution=300.0, quality=95)
    print(f"PDF saved: {output_path}")
    return output_path


def save_text_results(text, original_text, lang_name, method, confidence,
output_file="extracted_text.txt"):
    output_path = os.path.join(OUTPUT_DIR, output_file)
```

```
    with open(output_path, "w", encoding="utf-8") as f:
        f.write("=" * 60 + "\n")
        f.write("HANDWRITTEN TEXT EXTRACTION RESULTS\n")
        f.write("=" * 60 + "\n\n")
        f.write(f"Language: {lang_name}\n")
        f.write(f"OCR Method: {method}\n")
        f.write(f"Confidence: {confidence:.2f}%\n")
        f.write("\n" + "-" * 60 + "\n")
        f.write("EXTRACTED TEXT:\n")
        f.write("-" * 60 + "\n\n")
        f.write(text)

        if original_text != text:
            f.write("\n\n" + "-" * 60 + "\n")
            f.write("ORIGINAL (BEFORE SPELL CHECK):\n")
            f.write("-" * 60 + "\n\n")
            f.write(original_text)

    print(f"\nResults saved to: {output_path}")
    return output_path
```

Step 8: Main Function

```
def main():
    os.makedirs(OUTPUT_DIR, exist_ok=True)

    print("=" * 60)
    print("AI POWERED DOCUMENT SCANNER WITH OCR")
    print("=" * 60)

    print("\nSelect preprocessing mode:")
    print("1. Normal")
    print("2. Thin handwriting")
    print("3. Thick handwriting")

    mode_choice = input("Enter choice (1-3): ").strip()
    mode_map = {"1": "normal", "2": "thin", "3": "thick"}
    mode = mode_map.get(mode_choice, "normal")

    print("\nSelect thresholding method:")
    print("1. Adaptive Thresholding (recommended for uneven lighting)")
    print("2. Normal (Otsu) Thresholding")

    thresh_choice = input("Enter choice (1-2): ").strip()
```

```
    thresh_map = {"1": "adaptive", "2": "normal"}
    threshold_type = thresh_map.get(thresh_choice, "adaptive")

    # Step 1: Preprocessing
    image, processed, final_preprocessed = preprocessing(IMG, mode=mode,
threshold_type=threshold_type)
    save_processed_image(final_preprocessed, "01_preprocessed.png")

    # Step 2: Contour detection
    contour = contour_detection(image, processed)

    # Step 3: Perspective transform
    if contour is not None:
        h, w = image.shape[:2]
        full_image_contour = np.array([[0, 0], [w - 1, 0], [w - 1, h - 1], [0, h - 1]],
dtype=np.float32)
        contour_reshaped = contour.reshape(4, 2)
        if not np.allclose(contour_reshaped, full_image_contour, atol=20):
            warped = perspective_transform(image, contour)
            warped_preprocessed = perspective_transform(final_preprocessed, contour)
            final_image = warped
            final_for_ocr = warped_preprocessed
        else:
            final_image = image
            final_for_ocr = final_preprocessed
    else:
        final_image = image
        final_for_ocr = final_preprocessed

    save_processed_image(final_for_ocr, "02_final_ocr_ready.png")
    show("Final Image for OCR", final_for_ocr)

    # Step 4: Language selection and OCR
    lang_code, lang_name = select_language()
    text, confidence, method = text_extraction(
        final_for_ocr,
        lang_code,
        lang_name,
        os.path.join(OUTPUT_DIR, "extracted_text.txt"),
    )

    # Step 5: Save output formats
    print("\n" + "=" * 60)
    print("GENERATING OUTPUT FILES")
    print("=" * 60)
```

```
    jpg_path = convert_to_high_quality_jpg(final_image, "scanned_document.jpg")
    pdf_path = convert_image_to_pdf(final_image, "scanned_document.pdf")

    # Summary
    print("\n" + "=" * 60)
    print("PROCESSING COMPLETE!")
    print("=" * 60)
    print(f"\nOutput directory: {OUTPUT_DIR}/")
    print("\nGenerated files:")
    print("  ✓ Preprocessed image: 01_preprocessed.png")
    print("  ✓ OCR-ready image: 02_final_ocr_ready.png")
    print("  ✓ High-quality JPG: scanned_document.jpg")
    print("  ✓ PDF: scanned_document.pdf")
    print("  ✓ Extracted text: extracted_text.txt")
    print(f"\nOCR Method Used: {method}")
    print(f"OCR Quality: {confidence:.1f}% confidence")
    print(f"Extracted Text Length: {len(text)} characters")

    print("\nExtracted Text (Full):")
    print("=" * 60)
    print(text)
    print("=" * 60)
if __name__ == "__main__":
    main()
```

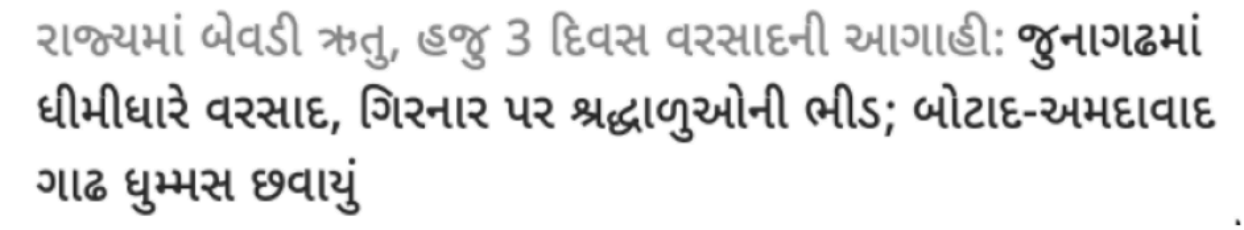
Results

Sample Example of Output we get.

```
============================================================
AI POWERED DOCUMENT SCANNER WITH OCR
============================================================

Select preprocessing mode:
1. Normal
2. Thin handwriting
3. Thick handwriting
Enter choice (1-3):  2

Select thresholding method:
1. Adaptive Thresholding (recommended for uneven lighting)
2. Normal (Otsu) Thresholding
Enter choice (1-2):  1
Loading IMAGE: /home/krushang/radhey/personal/college/Doc-Scanner-Project/test/printed_guj.png
Pre Processing Pipline
```

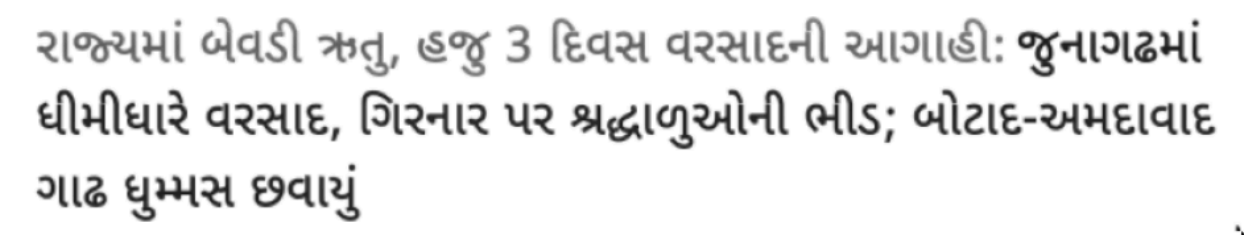
```
Grayscale
```

1. Grayscale



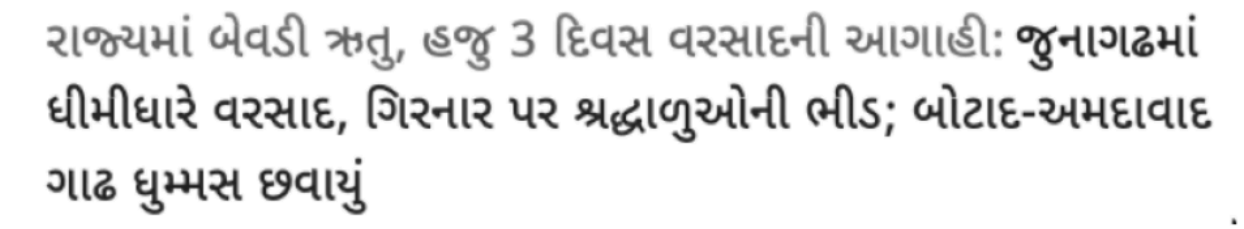

```
CLAHE Contrast Enhancement
```

2. Contrast Enhanced



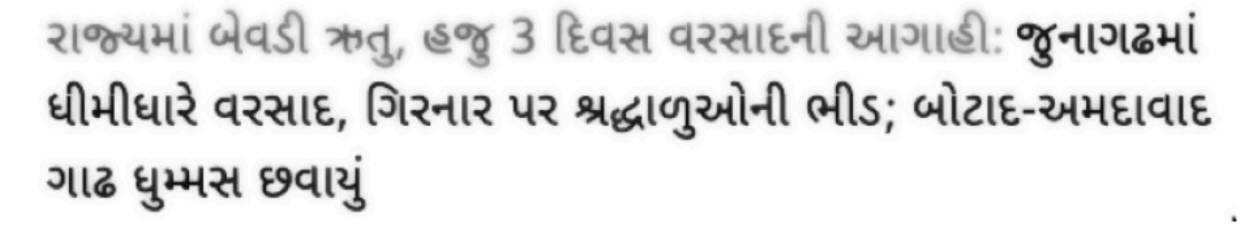

```
Denoising
```

3. Denoised



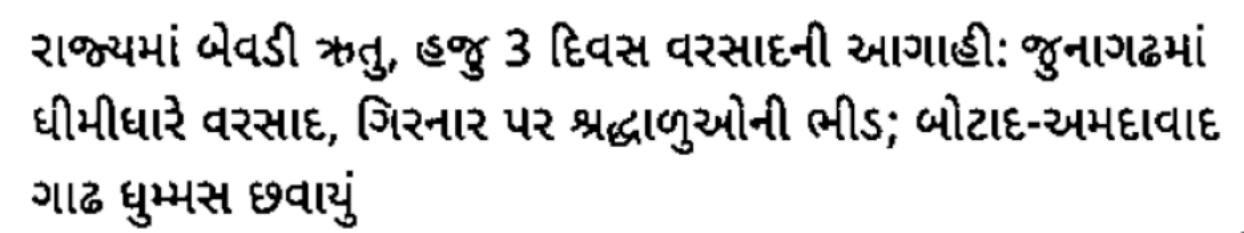

```
Bilateral Filter
```

4. Bilateral Filter



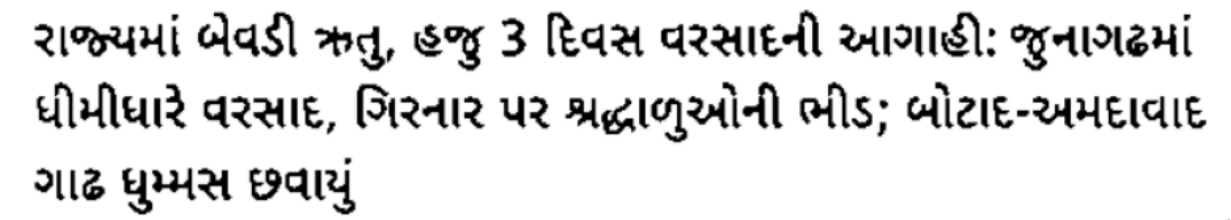

```
Applying ADAPTIVE thresholding...
Adaptive Thresholding Applied
```

5. Adaptive Threshold



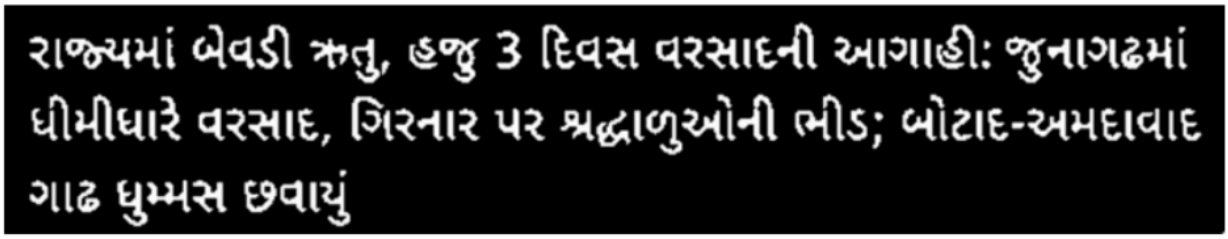
```
Morphological Operations for Thickness Adjustment
```

6. Morphological Cleanup



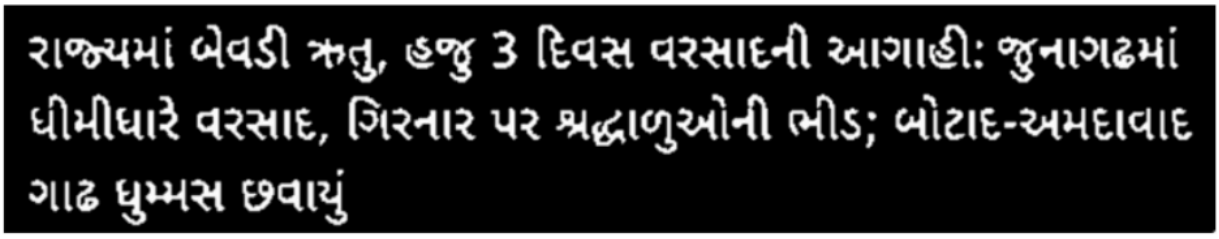

```
Upscaling
```

7. Upscaled (Final)




```
Saved: ./out/01_preprocessed.png

Contour Detection Options:
1. Auto-detect document edges
2. Use whole image
3. Manual corner selection
Enter choice (1-3):  2
Using whole image...
Saved: ./out/02_final_ocr_ready.png
```

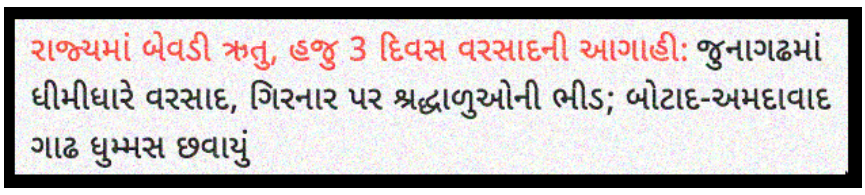
Final Image for OCR



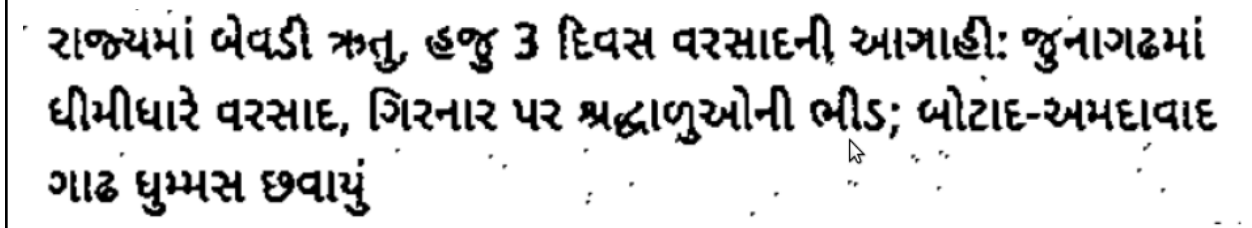

```
Select language for OCR:
1. English
2. Hindi
3. Gujarati
Enter choice (1-3):  3

Performing OCR (Gujarati)...
  Trying: Uniform text block...
    Confidence: 94.6%, Words: 17, Score: 110.6
  Trying: Single column text...
    Confidence: 94.6%, Words: 17, Score: 110.6
  Trying: Fully automatic layout...
    Confidence: 94.6%, Words: 17, Score: 110.6
  Trying: Sparse text...
    Confidence: 94.6%, Words: 17, Score: 110.6
```

```
============================================================
EXTRACTED TEXT PREVIEW:
============================================================
રાજ્યમાં બેવડી ઋતુ, હજુ 3 દિવસ વરસાદની આગાહી: જુનાગઢમાં
ધીમીધારે વરસાદ, ગિરનાર પર શ્રદ્ધાળુઓની ભીડ; બોટાદ-અમદાવાદ
ગાઢ ધુમ્મસ છવાયું
============================================================

Extracted text saved to: ./out/extracted_text.txt

============================================================
GENERATING OUTPUT FILES
============================================================
High-quality JPG saved: ./out/scanned_document.jpg
PDF saved: ./out/scanned_document.pdf

============================================================
PROCESSING COMPLETE!
============================================================

Output directory: ./out/

Generated files:
  ✓ Preprocessed image: 01_preprocessed.png
  ✓ OCR-ready image: 02_final_ocr_ready.png
  ✓ High-quality JPG: scanned_document.jpg
  ✓ PDF: scanned_document.pdf
  ✓ Extracted text: extracted_text.txt

OCR Method Used: Uniform text block
OCR Quality: 94.6% confidence
Extracted Text Length: 134 characters


Extracted Text (Full):
============================================================
રાજ્યમાં બેવડી ઋતુ, હજુ 3 દિવસ વરસાદની આગાહી: જુનાગઢમાં
ધીમીધારે વરસાદ, ગિરનાર પર શ્રદ્ધાળુઓની ભીડ; બોટાદ-અમદાવાદ
ગાઢ ધુમ્મસ છવાયું
============================================================
```

Case 1: Noise Image Testing

Original Image with Noise


After Denoising and Morphological Cleanups(as thinner characters) With Adaptive Thresolding

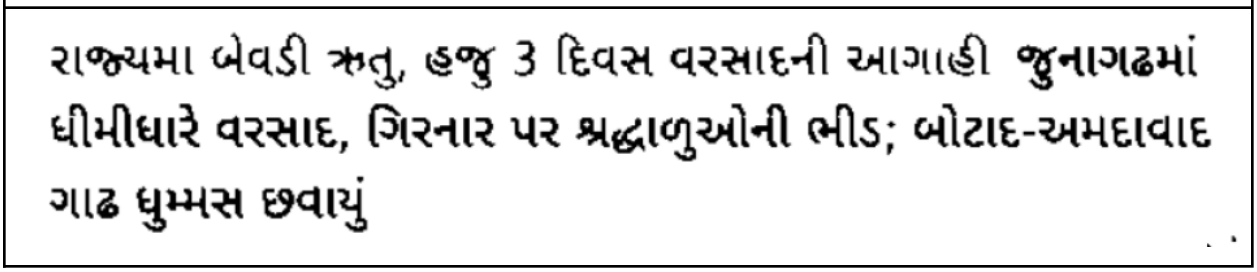


Extracted Text Output and Confidence Score

```
Performing OCR (Gujarati)...
  Trying: Uniform text block...
    Confidence: 81.0%, Words: 18, Score: 95.5
  Trying: Single column text...
    Confidence: 87.9%, Words: 17, Score: 102.8
  Trying: Fully automatic layout...
    Confidence: 87.9%, Words: 17, Score: 102.8
  Trying: Sparse text...
    Confidence: 92.6%, Words: 17, Score: 108.3

============================================================
EXTRACTED TEXT PREVIEW:
============================================================
રાજ્યમાં બેવડી ઋતુ, હજુ 3 દિવસ વરસાદની આગાહી: જુનાગઢમાં

ધીમીધારે વરસાદ, ગિરનાર પર શ્રદ્ધાળુઓની ભી ભીડ; બોટાદ-અમદાવાદ

ગાઢ ધુમ્મસ છવાયું
============================================================

Extracted text saved to: ./out/extracted_text.txt
```

Case 2: Applying Normal Thresholding on Original Image

After Normal Thresholding

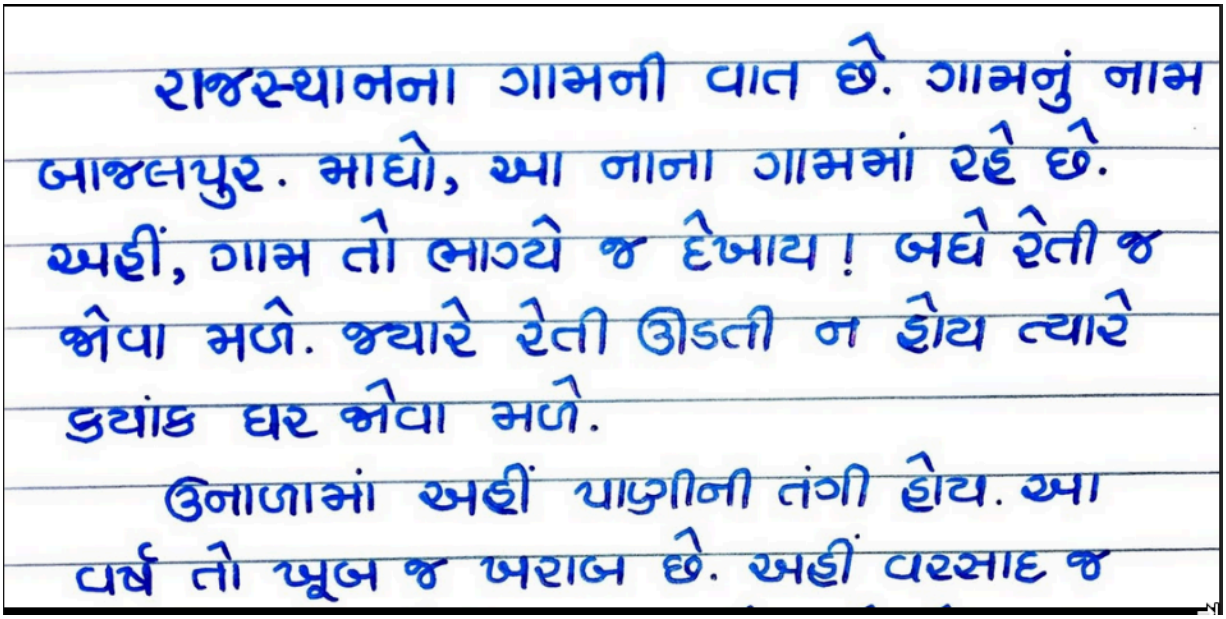


Extracted Text Output and Confidence Score

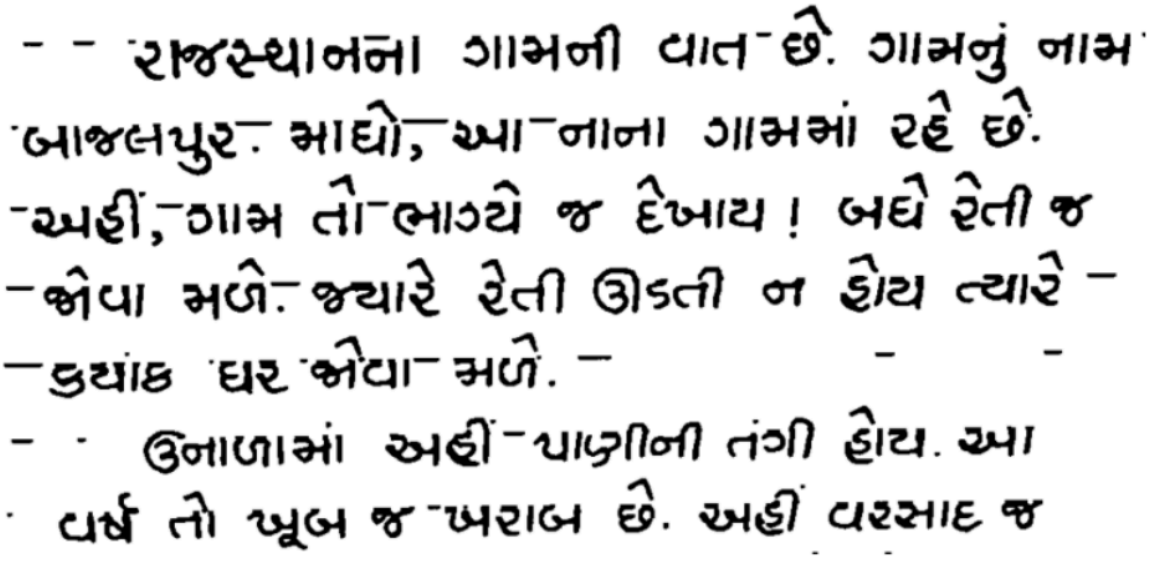
```
Performing OCR (Gujarati)...
  Trying: Uniform text block...
    Confidence: 49.3%, Words: 15, Score: 56.7
  Trying: Single column text...
    Confidence: 49.3%, Words: 15, Score: 56.7
  Trying: Fully automatic layout...
    Confidence: 49.3%, Words: 15, Score: 56.7
  Trying: Sparse text...
    Confidence: 54.5%, Words: 17, Score: 63.7

============================================================
EXTRACTED TEXT PREVIEW:
============================================================
૨... ૩હ૩ ક૨ હજુ 3 દ૨ે વ૨સ 1. ૨હ. સ૩ જુનાગઢમ.

ધ.પોધા૨ે વ૨સાધ. ગિ૨તા૨ ચ૨ શ્રદ્ધાળ્અ.ની ₹.૩: સ:૮ાદ અક૬ાવ.$

ૐ..ઢ ધુમ્ચસ છવાયુ
============================================================

Extracted text saved to: ./out/extracted_text.txt
```

Similarly Applying for gujarati.png (Handwritten Gujarati)

Original Image



After Thicker Character Morphological Cleanups

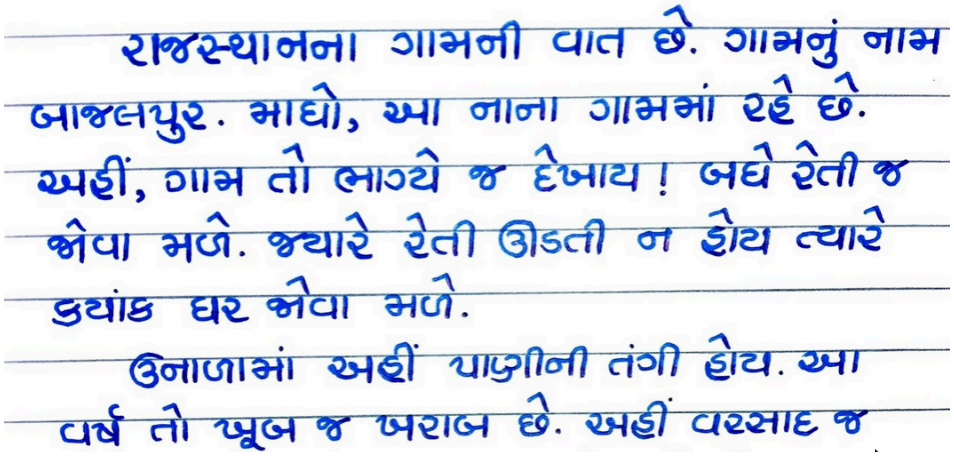


Extracted Text Output and Confidence Score

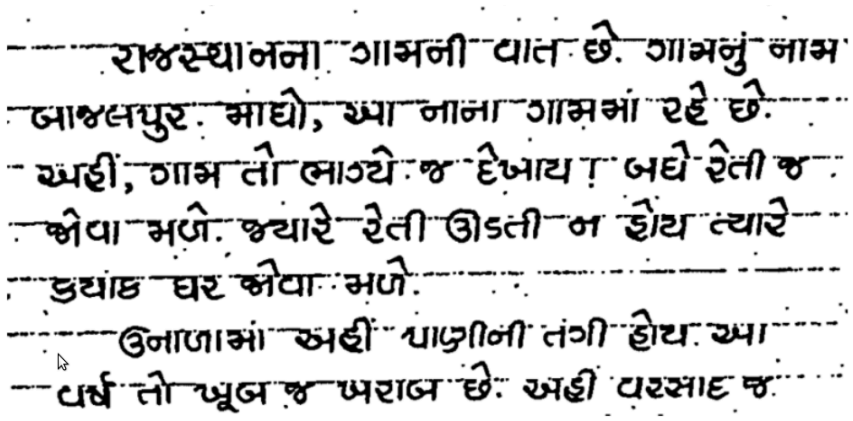
```
Performing OCR (Gujarati)...
  Trying: Uniform text block...
    Confidence: 62.4%, Words: 33, Score: 83.0
  Trying: Single column text...
    Confidence: 64.8%, Words: 31, Score: 84.9
  Trying: Fully automatic layout...
    Confidence: 62.6%, Words: 32, Score: 82.7
  Trying: Sparse text...
    Confidence: 69.2%, Words: 33, Score: 92.0

============================================================
EXTRACTED TEXT PREVIEW:
============================================================
રી રાજસ્થાનના ડાસની વાત છ: ગામનું નાગ

બાજલપુર- માદા; આ નાના ડા/મના સ્ટે છ.

-અહીં . ગાન તા-_ભાગ્થે જ દેખાય | બદ રતઉ જ

-જેવા મળે-જ્યારે રેતી ઊડતી ન હ્ય ત્યાર -

શ

શ

કયાક ઘર જવા

ઝળળ

શ

ઉનાળાઓ અહી -શાછીઉન તoઉ હાય. અ!

વર્ષ તો સ્રબ જ ખરાબ છ. અહં વરસાદ જ
============================================================

Extracted text saved to: ./out/extracted_text.txt
```

Case 1: Noise Image Testing

Original Image with Noise



After Denoising and Morphological Cleanups(as thicker characters) With Adaptive Thresolding

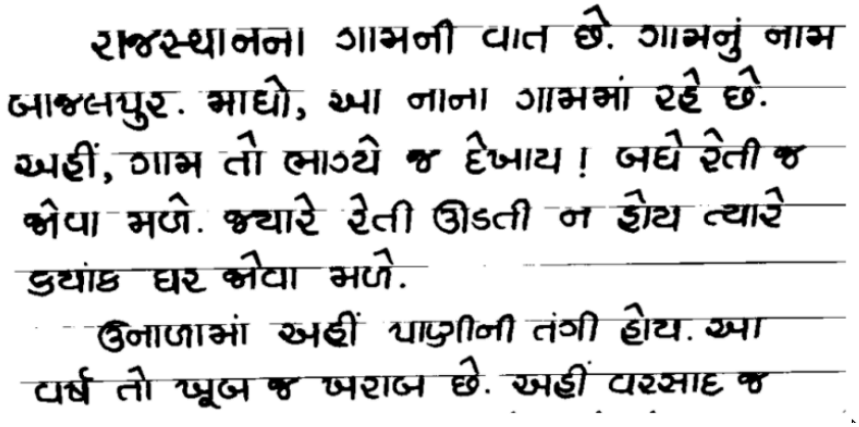


Extracted Text Output and Confidence Score

```
Performing OCR (Gujarati)...
  Trying: Uniform text block...
    Confidence: 40.0%, Words: 42, Score: 56.8
  Trying: Single column text...
    Confidence: 47.8%, Words: 30, Score: 62.2
  Trying: Fully automatic layout...
    Confidence: 41.4%, Words: 32, Score: 54.6
  Trying: Sparse text...
    Confidence: 45.1%, Words: 41, Score: 63.5

============================================================
EXTRACTED TEXT PREVIEW:
============================================================
----સક#કસ્થાનના ડા।અની વાત-છ. ગામનુ બાન

શકશ શમઃ શન શકઈ

_બાજલપુર- સધી; આ-બાના- -ઝાનના' ૨૮ છઃ

- સણી કાબ તા-ભાડટી- છ “દેખાય” બ ૨? ૪.

જીવા-અળ: જયારે રેતી ઊવ્લી- બ “કટ લ્યાર--

ઝુ:-----2:

શસઃ ૨જ શન ણ

દિ ત

લ્યાઈ ઘરર “છલતા-- અઈ.

------ઉનાળામો સહી 1110 ત૦૭-હોટા” અઈ” “--

--લર્ણસ-ખૂલ જઈ-ભરાલ છે: અહી 'વરસાદ' હક”
============================================================

Extracted text saved to: ./out/extracted_text.txt
```

Case 2: Applying Normal Thresholding on Original Image

After Normal Thresholding

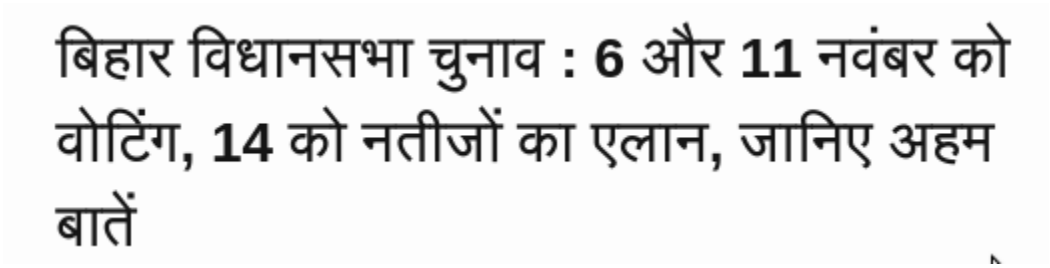


Extracted Text Output and Confidence Score

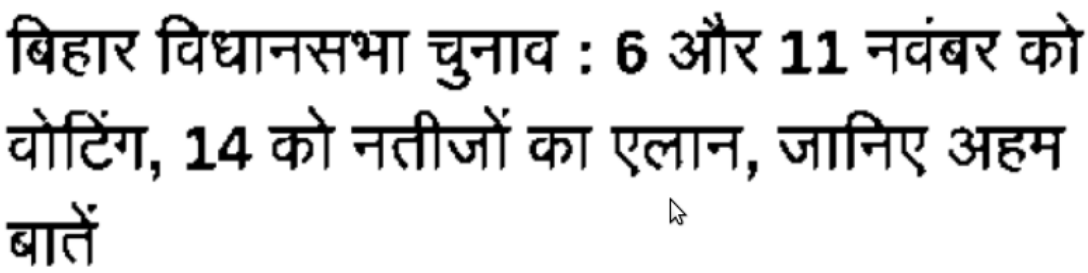
```
Performing OCR (Gujarati)...
  Trying: Uniform text block...
    Confidence: 68.1%, Words: 37, Score: 93.3
  Trying: Single column text...
    Confidence: 72.5%, Words: 34, Score: 97.2
  Trying: Fully automatic layout...
    Confidence: 76.0%, Words: 35, Score: 102.5
  Trying: Sparse text...
    Confidence: 71.9%, Words: 34, Score: 96.3

============================================================
EXTRACTED TEXT PREVIEW:
============================================================
ટજસ્શચાબંના ડામની વ:.ત છ ગામનુ આગમ
બાજલપુર . સાધો, અ નાના ગામભા રટે છ છ.

અહી, ગામ તા ભાગ્યે જ દેખાય | બધ રત0 જ

જોવા મળી. જ્યારે રેતી ઊડતી ન ફય ત્યાર
છ્ઠથાકઈ ઘર જવા મળ.

ઉનાળામા અહીં શ્ાછ0ન0 ત") હાય. અ!
વર્ષ તો ખ્રબ જ ખરાબ છે. અહીં વરસાદ જ
============================================================

Extracted text saved to: ./out/extracted_text.txt
```

Similarly Applying for printed_hin.png (Printed Hindi)

Original Image



After Thinner Character Morphological Cleanups

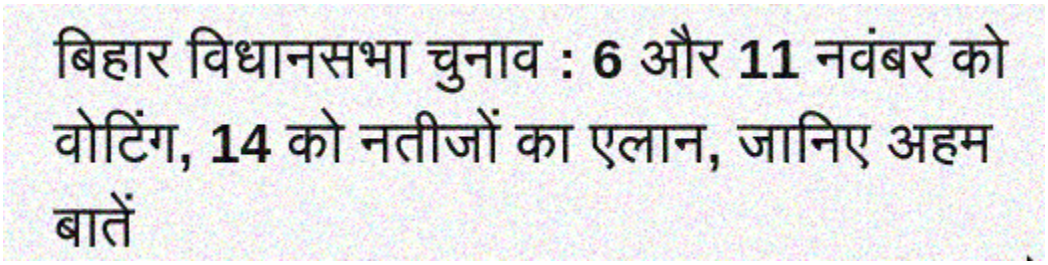


Extracted Text Output and Confidence Score

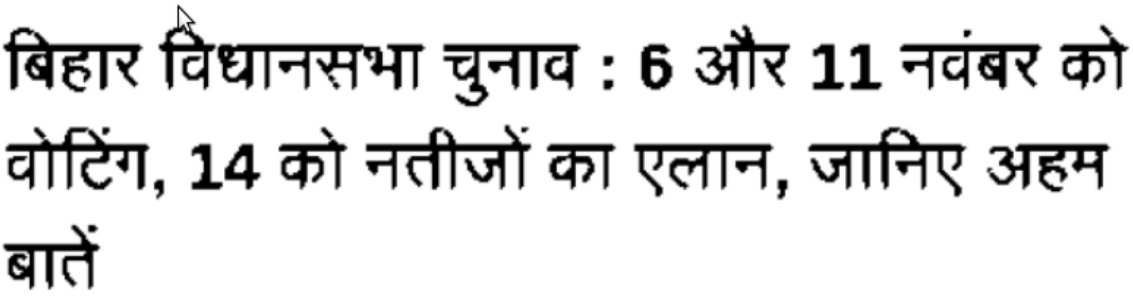
```
Performing OCR (Hindi)...
  Trying: Uniform text block...
    Confidence: 90.5%, Words: 10, Score: 99.6
  Trying: Single column text...
    Confidence: 90.5%, Words: 10, Score: 99.6
  Trying: Fully automatic layout...
    Confidence: 90.5%, Words: 10, Score: 99.6
  Trying: Sparse text...
    Confidence: 90.5%, Words: 10, Score: 99.6

============================================================
EXTRACTED TEXT PREVIEW:
============================================================
बिहार विधानसभा चुनाव : 6 और 44 नवंबर को
वोटिंग, 34 को नतीजों का एलान, जानिए अहम
बातें
============================================================

Extracted text saved to: ./out/extracted_text.txt
```

Case 1: Noise Image Testing

Original Image with Noise



After Denoising and Morphological Cleanups(as thicker characters) With Adaptive Thresolding

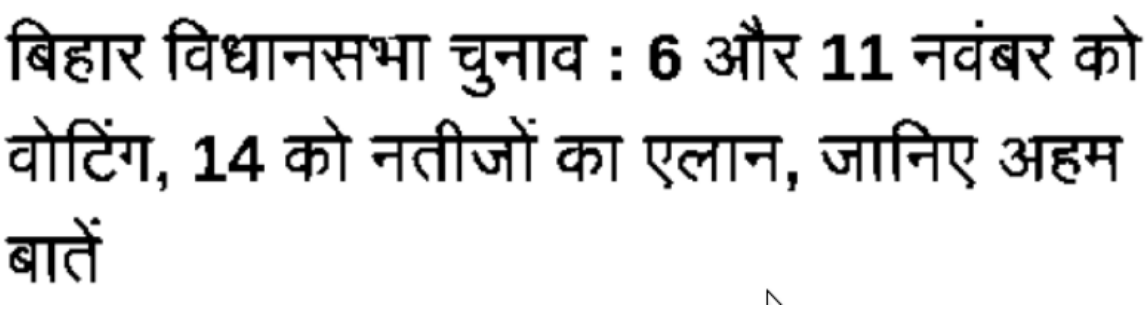


Extracted Text Output and Confidence Score

```
Performing OCR (Hindi)...
  Trying: Uniform text block...
    Confidence: 95.0%, Words: 10, Score: 104.5
  Trying: Single column text...
    Confidence: 95.0%, Words: 10, Score: 104.5
  Trying: Fully automatic layout...
    Confidence: 95.0%, Words: 10, Score: 104.5
  Trying: Sparse text...
    Confidence: 95.0%, Words: 10, Score: 104.5

============================================================
EXTRACTED TEXT PREVIEW:
============================================================
बिहार विधानसभा चुनाव : 6 और 44 नवंबर को
वोटिंग, 44 को नतीजों का एलान, जानिए अहम
बातें
============================================================

Extracted text saved to: ./out/extracted_text.txt
```

Case 2: Applying Normal Thresholding on Original Image

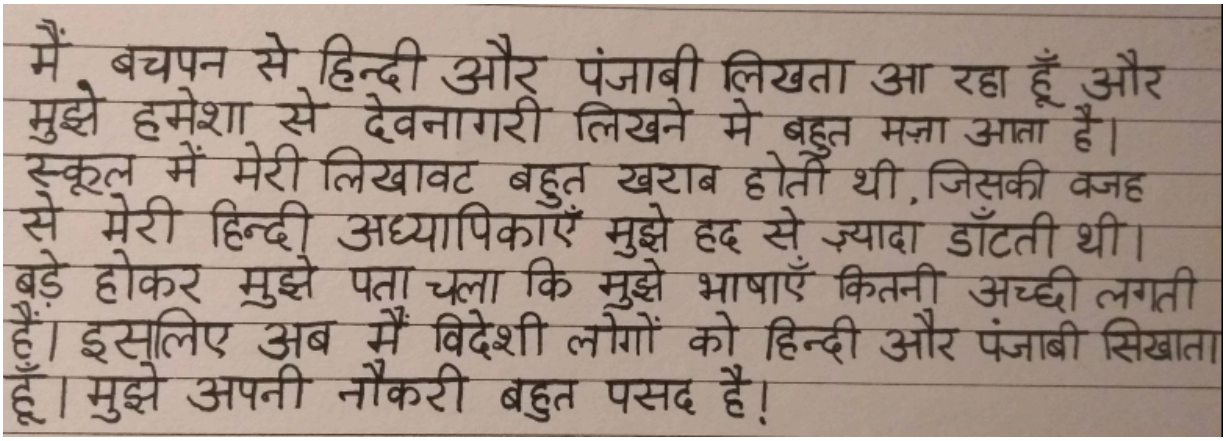
After Normal Thresholding

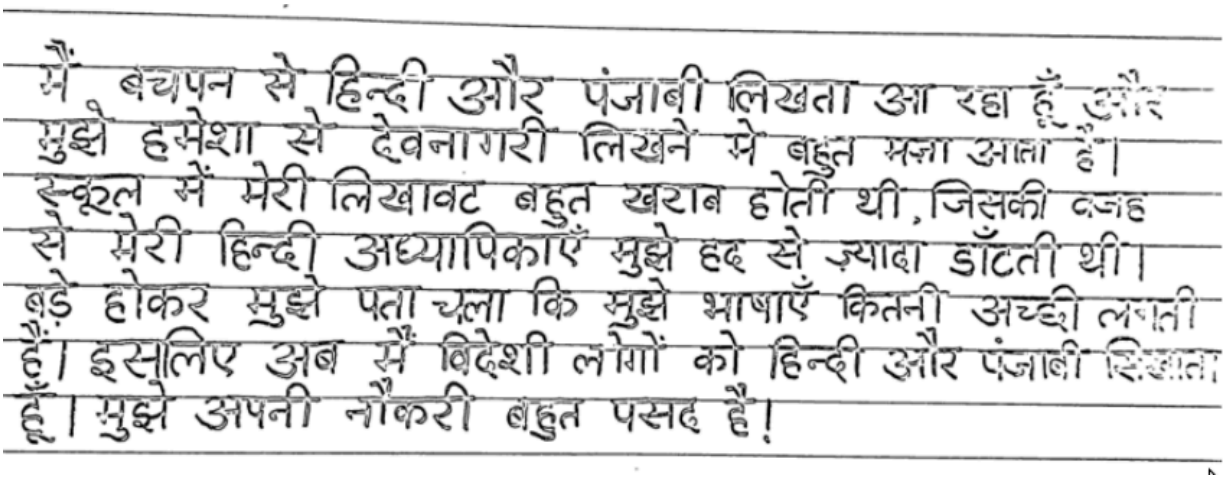


Extracted Text Output and Confidence Score

```
Performing OCR (Hindi)...
  Trying: Uniform text block...
    Confidence: 95.0%, Words: 10, Score: 104.5
  Trying: Single column text...
    Confidence: 95.0%, Words: 10, Score: 104.5
  Trying: Fully automatic layout...
    Confidence: 95.0%, Words: 10, Score: 104.5
  Trying: Sparse text...
    Confidence: 95.0%, Words: 10, Score: 104.5

============================================================
EXTRACTED TEXT PREVIEW:
============================================================
बिहार विधानसभा चुनाव : 6 और 44 नवंबर को
वोटिंग, 44 को नतीजों का एलान, जानिए अहम
बातें
============================================================

Extracted text saved to: ./out/extracted_text.txt
```

Similarly Applying for hindi.png (Handwritten Hindi)

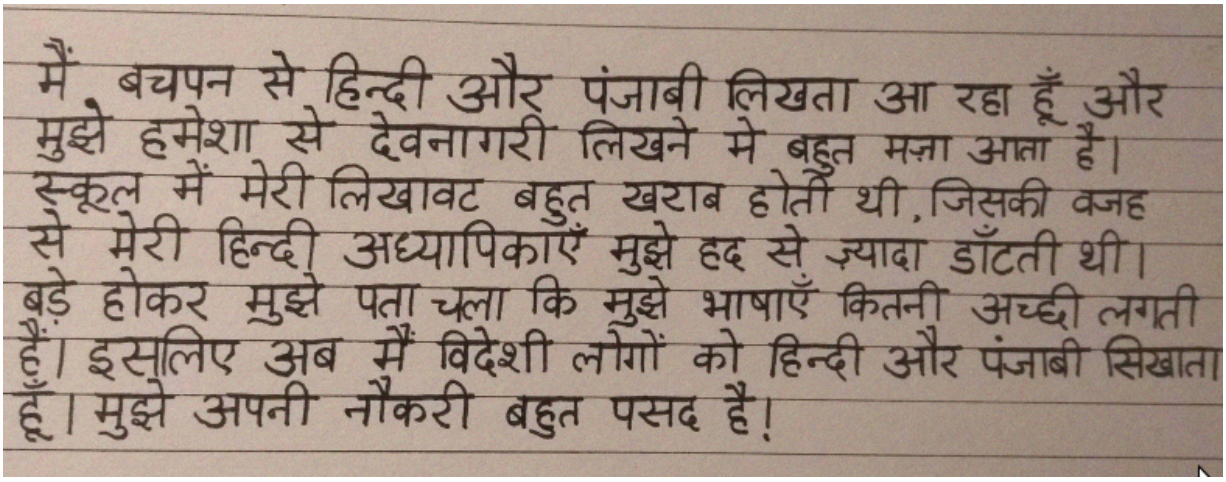
Original image


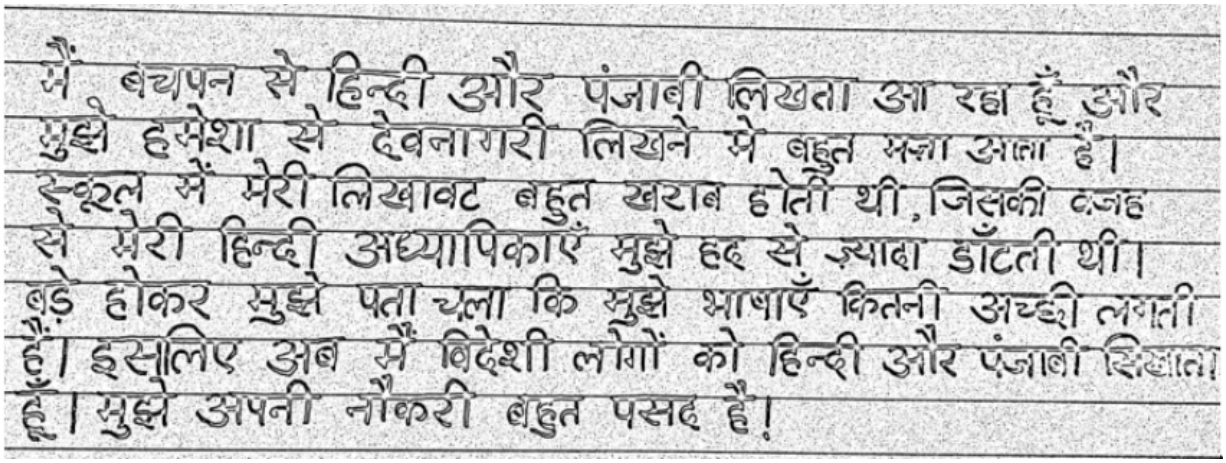
After Thicker Character Morphological Cleanups


Extracted Text Output and Confidence Score

```
Performing OCR (Hindi)...
  Trying: Uniform text block...
    Confidence: 50.9%, Words: 80, Score: 91.6
  Trying: Single column text...
    Confidence: 60.2%, Words: 56, Score: 93.9
  Trying: Fully automatic layout...
    Confidence: 61.4%, Words: 54, Score: 94.5
  Trying: Sparse text...
    Confidence: 44.5%, Words: 72, Score: 76.5

============================================================
EXTRACTED TEXT PREVIEW:
============================================================
"२६
वा ऊ छा णंने से /शन्न जज ट्ि कि _-+-ज्रे>-
। झे हिन्दी बीलेखता झा सह हैं

'जुझ्ले हग्ेशा से देवनागरी लिखने से बहुत मजा सता है] 7

ए्ो

ज्छूत सं मर्र कट कूटल भं मरी लिखावट बहत खटाब होती थी जिसकी दऊहा 7:
से सरी हिल्दी अध्यापिकाएँ सुझे हद से ज़्यादा डाँटती थी 7



(5 कर सह व पता जला कि सुझे झाषाएँ कितनी उसर्च | €थता
हा इसलिए जब झा हिदेशी लोगों को हिन्दी आर एंजाबी डिझिाता

९

८

हूँ सुझे ऊउपनी नोकरी बहुत पसद है। प्र
============================================================

Extracted text saved to: ./out/extracted_text.txt
```

Case 1: Noise Image Testing

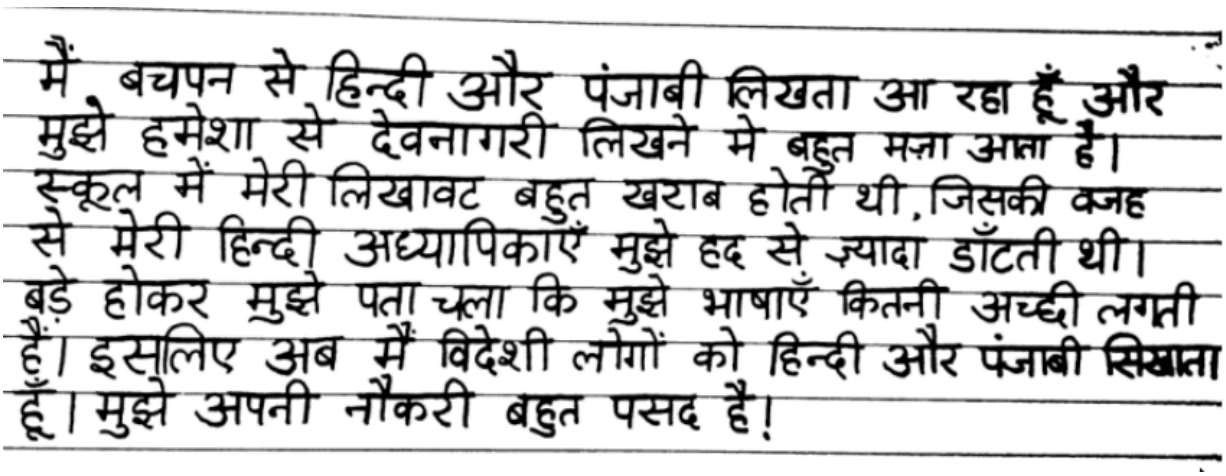
Original Image with Noise


After Denoising and Morphological Cleanups(as thinner characters) With Adaptive Thresolding


Extracted Text Output and Confidence Score
Could not able to process

Case 2: Applying Normal Thresholding on Original Imag

After Normal Thresholding

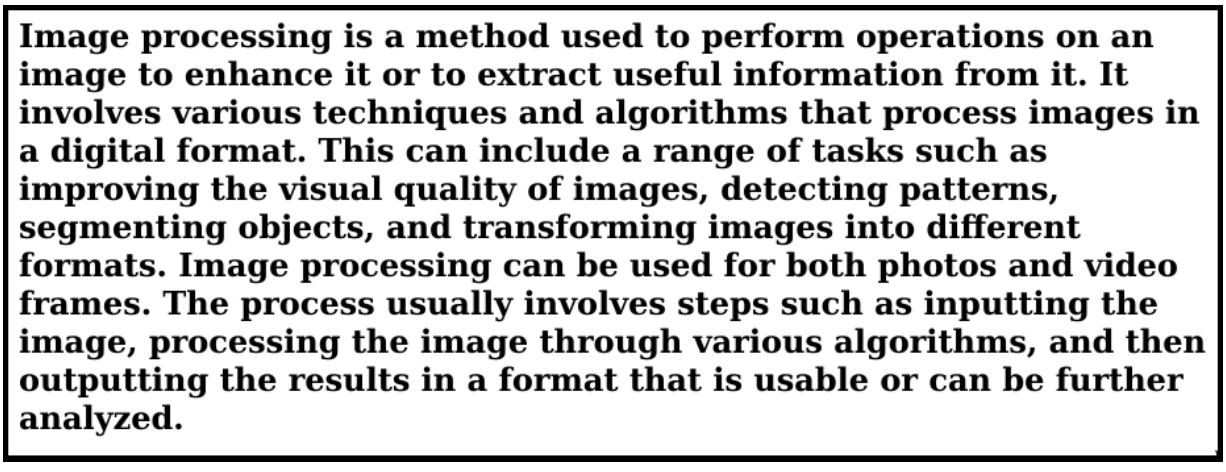


Extracted Text Output and Confidence Score

```
[No text detected]
```

Similarly Applying for printed_eng.png (Printed English)

Original Image



After Morphological Cleanups

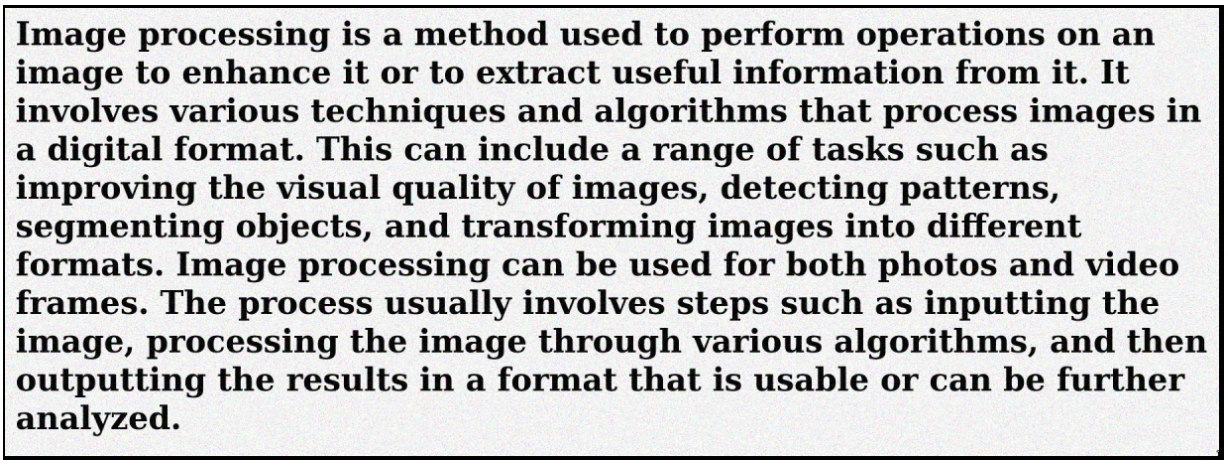
Image processing is a method used to perform operations on an image to enhance it or to extract useful information from it. It involves various techniques and algorithms that process images in a digital format. This can include a range of tasks such as improving the visual quality of images, detecting patterns, segmenting objects, and transforming images into different formats. Image processing can be used for both photos and video frames. The process usually involves steps such as inputting the image, processing the image through various algorithms, and then outputting the results in a format that is usable or can be further analyzed.

Extracted Text Output and Confidence Score

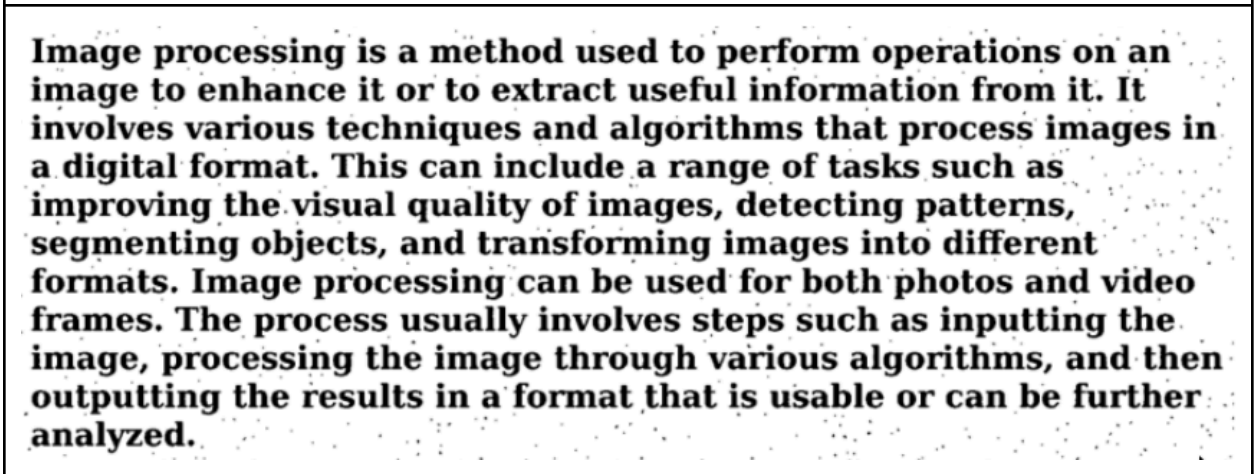
```
Performing OCR (English)...
  Trying: Uniform text block...
    Confidence: 95.5%, Words: 80, Score: 172.0
  Trying: Single column text...
    Confidence: 95.5%, Words: 80, Score: 172.0
  Trying: Fully automatic layout...
    Confidence: 95.5%, Words: 80, Score: 172.0
  Trying: Sparse text...
    Confidence: 95.5%, Words: 80, Score: 172.0

============================================================
EXTRACTED TEXT PREVIEW:
============================================================
Image processing is a method used to perform operations on an
image to enhance it or to extract useful information from it. It
involves various techniques and algorithms that process images in
a digital format. This can include a range of tasks such as
improving the visual quality of images, detecting patterns,
segmenting objects, and transforming images into different
formats. Image processing can be used for both photos and video
frames. The process usually involves steps such as inputting the
============================================================

Extracted text saved to: ./out/extracted_text.txt
```

Case 1: Noise Image Testing

Original Image with Noise



After Denoising and Morphological Cleanups



Extracted Text Output and Confidence Score

```
Performing OCR (English)...
  Trying: Uniform text block...
    Confidence: 88.5%, Words: 81, Score: 160.1
  Trying: Single column text...
    Confidence: 86.6%, Words: 80, Score: 155.8
  Trying: Fully automatic layout...
    Confidence: 86.9%, Words: 80, Score: 156.5
  Trying: Sparse text...
    Confidence: 92.0%, Words: 80, Score: 165.6

============================================================
EXTRACTED TEXT PREVIEW:
============================================================
Image processing is a method used to perform operations on an

image to enhance it or to extract useful information from it. It

involves various techniques and algorithms that process images in

a. digital format. This can include a range of tasks such as

improving the.visual quality of i images, detecting patterns,

segmenting objects, and transforming i images into different

formats. Image processing ‘can be used for both photos and video

frames. The process usually involves steps such as
============================================================

Extracted text saved to: ./out/extracted_text.txt
```

Case 2: Applying Normal Thresholding on Original Image

After Normal Thresholding

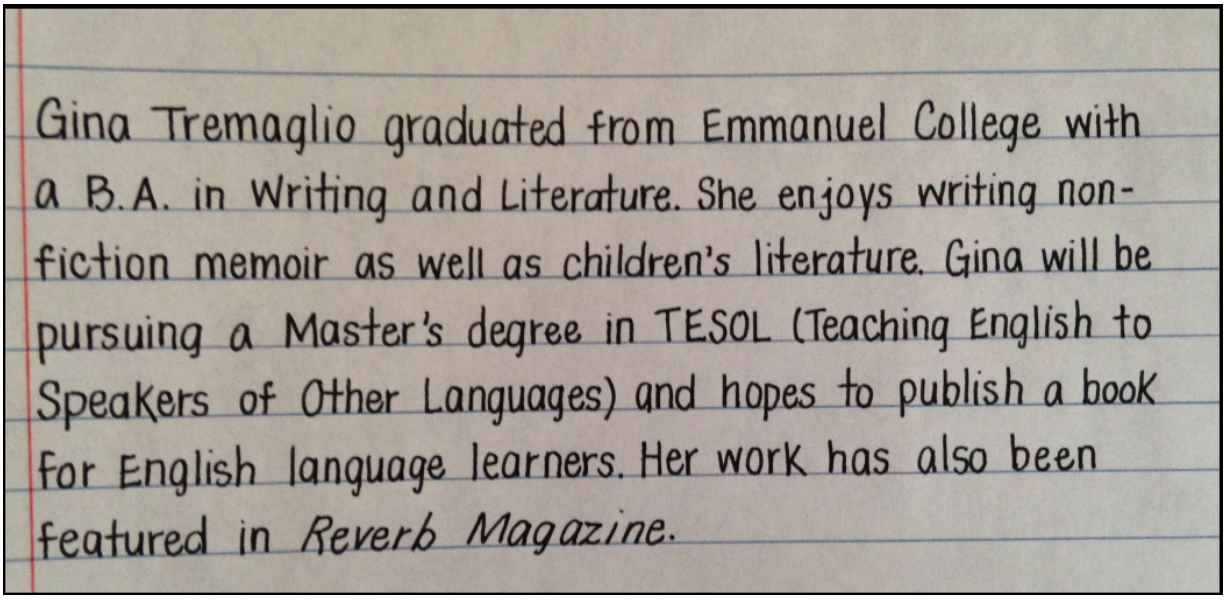
Image processing is a method used to perform operations on an image to enhance it or to extract useful information from it. It involves various techniques and algorithms that process images in a digital format. This can include a range of tasks such as improving the visual quality of images, detecting patterns, segmenting objects, and transforming images into different formats. Image processing can be used for both photos and video frames. The process usually involves steps such as inputting the image, processing the image through various algorithms, and then outputting the results in a format that is usable or can be further analyzed.

Extracted Text Output and Confidence Score

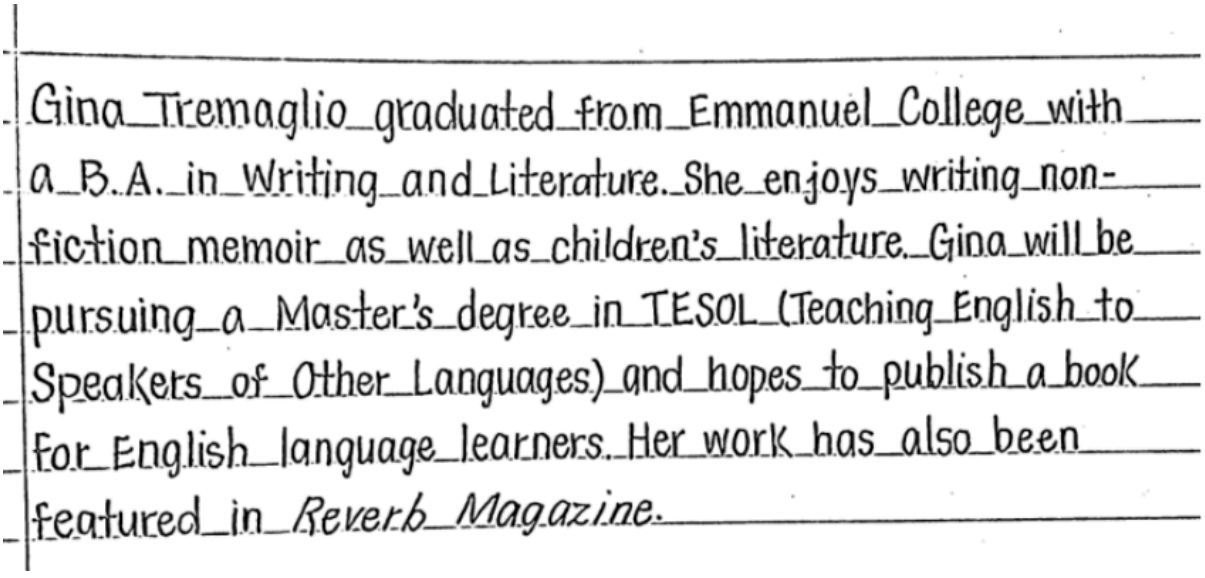
```
Performing OCR (English)...
  Trying: Uniform text block...
    Confidence: 95.3%, Words: 80, Score: 171.5
  Trying: Single column text...
    Confidence: 95.3%, Words: 80, Score: 171.5
  Trying: Fully automatic layout...
    Confidence: 95.3%, Words: 80, Score: 171.5
  Trying: Sparse text...
    Confidence: 95.3%, Words: 80, Score: 171.5

============================================================
EXTRACTED TEXT PREVIEW:
============================================================
Image processing Is a method used to perform operations on an
image to enhance it or to extract useful information from it. It
involves various techniques and algorithms that process images in
a digital format. This can include a range of tasks such as
improving the visual quality of images, detecting patterns,
segmenting objects, and transforming images into different
formats. Image processing can be used for both photos and video
frames. The process usually involves steps such as inputting the
============================================================

Extracted text saved to: ./out/extracted_text.txt
```

Similarly Applying for printed_eng.png (Printed English)

Original Image



After Morphological Cleanups

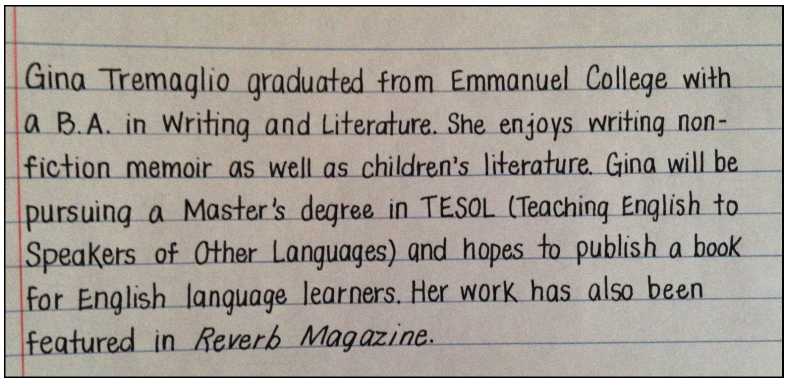


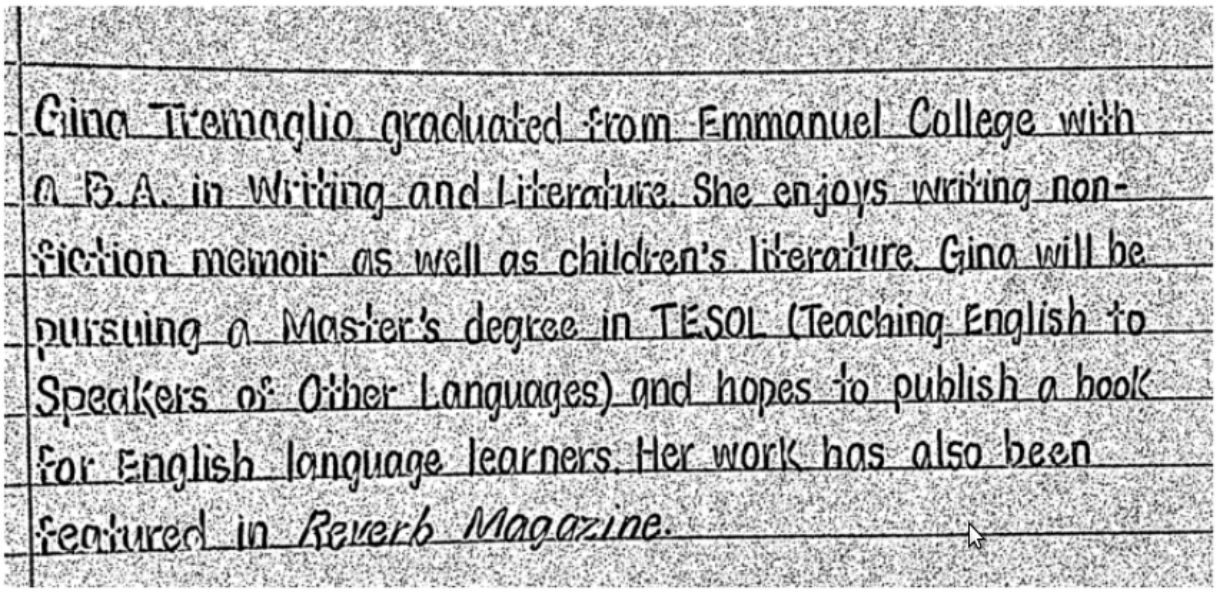
Extracted Text Output and Confidence Score

```
Performing OCR (English)...
  Trying: Uniform text block...
    Confidence: 32.9%, Words: 16, Score: 38.2
  Trying: Single column text...
    Confidence: 47.9%, Words: 12, Score: 53.6
  Trying: Fully automatic layout...
    Confidence: 47.9%, Words: 12, Score: 53.6
  Trying: Sparse text...
    Confidence: 37.6%, Words: 13, Score: 42.5

============================================================
EXTRACTED TEXT PREVIEW:
============================================================
|

|Gina –Tremaglio_qracuated_trom_Emmanuel _College_with
Q_B.A_in_Writing_and_Literature.She_enjoys_writing_non-
|fietion_memoir_as_well_as_childken's_literature_Cina_willbe
_|pursuing_a_Mastter's_degree_ in LESOL_(Teaching_English_to__
Speakets_of_Other_Languages)_and_hopes_to_publish_a_book
For_English_lanquage_leatners._Herworl_h as_also_been
feqtured_in_Aeverh Magazine:
============================================================

Extracted text saved to: ./out/extracted_text.txt
```

Case 1: Noise Image Testing

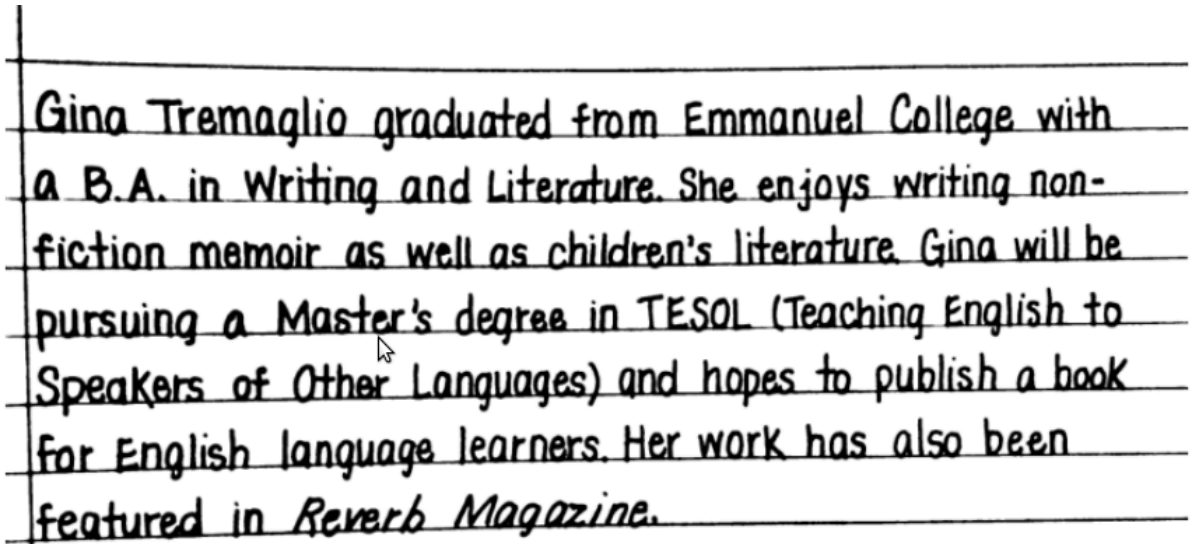
Original Image with Noise



After Denoising and Morphological Cleanups


Extracted Text Output and Confidence Score
Could not able to process

Case 2: Applying Normal Thresholding on Original Image

After Normal Thresholding


Extracted Text Output and Confidence Score

```
Performing OCR (English)...
  Trying: Uniform text block...
    Confidence: 68.3%, Words: 43, Score: 97.6
  Trying: Single column text...
    Confidence: 69.9%, Words: 37, Score: 95.8
  Trying: Fully automatic layout...
    Confidence: 73.9%, Words: 24, Score: 91.7
  Trying: Sparse text...
    Confidence: 44.1%, Words: 26, Score: 55.6

============================================================
EXTRACTED TEXT PREVIEW:
============================================================
Inc amagho Qraduated from Emmanue rilege with
4{Q_B.A. in Writing and Literature. She enjoys writing non-
iction memoir as well as children’s |rrerature. Gina will De
DUrsuing a Master's degree ir ( aAchinNa English to
seakers of Other Languages) and hopes to Publish a book
“. NC ch language jearne Mer WOR DU ALSO Vee

rE QTUred in Aeverh Maga NE
============================================================

Extracted text saved to: ./out/extracted_text.txt
```

Overall Evaluation for All Language Specific Document

Confidence Scoring

	Original	Noise Image	Normal Thresholding
Gujarati Handwritten	69.2 %	45.1 %	76.0 %
Hindi Handwritten	61.4 %	-	No Text Detected
English Handwritten	47.9 %	-	68.3 %

Gujarati Printed	94.6 %	92.6 %S	86.2 %
Hindi Printed	90.5 %	95.0 %	95.0 %
English Printed	95.5 %	92.0 %	95.3 %

Conclusion

The developed **AI-Powered Document Scanner with OCR** successfully demonstrates the ability to detect, enhance, and extract text from various types of documents using image processing and AI techniques.

From the confidence scores, it is observed that:

- **Printed documents** achieved consistently **high OCR accuracy (above 90%)** across all languages (Gujarati, Hindi, and English). This shows that the system performs excellently for clean, well-printed text.

- **Handwritten documents**, in contrast, showed **lower accuracy (45–76%)**, indicating that OCR struggles with variations in handwriting styles and uneven pen pressure.

- **Normal thresholding and preprocessing** improved recognition results for most cases, especially in printed text. However, handwritten text in noisy images remains more challenging.

- Among handwritten samples, **Gujarati handwriting produced the highest accuracy (76%)**, suggesting better compatibility with the selected OCR configuration, while **Hindi handwriting showed the lowest performance** due to complex character shapes and diacritics.

Overall, the system performs **reliably for printed text scanning and digitization**, producing clean, searchable outputs.
For handwritten text, the results are **acceptable for basic recognition** but can be improved further using **deep learning–based OCR models** or **language-specific handwriting datasets**.