CENTRE FOR DEVELOPMENT AND ADVANCED COMPUTING

(Ministry of Electronics and Information Technology (MeitY), India)

PROJECT REPORT ON

Graphology Based Emotional Recognition and Handwriting Analysis Using Computer Vision & Image Processing and Python Libraries.

Post Graduate Diploma

In

Big Data Analytics

SUBMITTED BY:-

Saumya Mukherjee – 250870625008

Sushovan Mondal – 250870625009

Sapna Gupta - 250870625007

CERTIFICATE OF APPROVAL

This is to Certify that the project report entitled- **"Graphology Based Emotional Recognition and Handwriting Analysis"** submitted by **Sapna Gupta, Sushovan Mondal, and Saumya Mukherjee** is a bonafide work carried out by them under the supervision of **Ms. Barnali Pal, Scientist-F, C-DAC, Kolkata.** It is approved for the partial fulfilment of the requirement *Centre for Development of Advanced Computing (C-DAC) Kolkata*, for the completion of report. This project report has not been earlier submitted to any other Institute or University for the award of any degree or diploma.

Barnali Pal.

(Barnali Pal)
Scientist-F
CDAC, Kolkata

Declaration

We hereby declare that the project work entitled English Handwriting Analysis System using Graphology and Image Processing **submitted** to the Centre for Advance Computing(C-DAC), Kolkata, is a record of an original work done by us under the guidance of **Ms. Barnali Pal, Scientist-F, CDAC Kolkata**. This project work has been performed for the award of ***Post Graduate Diploma in Big Data Analytics (PG-DBDA)*** course of C- DAC, Kolkata only and this or any similar project will not be used for any other Degree or Diploma's associateship / fellowship.

Signature:

Sushovan Mondal

Name: Sushovan Mondal
PRN No.: 250870625009

Signature:

Sapnagupta

Name: Sapna Gupta
PRN No.: 250870625007

Signature:

Saumya Mukherjee

Name: Saumya Mukherjee
PRN No.: 250870625008

Acknowledgement

We express our utmost sincere regards and deep gratitude to our project guide **Ms. Barnali Pal, Scientist-F,** C-DAC, Kolkata for her valuable guidance towards us to navigate through the difficulty of project execution at all times**.**

We are very much thankful to **Mr. Sayan Mondal**, Intern C-DAC Kolkata for their unconditional help, guidance and inspiration for giving us a wonderful opportunity to enhance our skills. Finally, we would like to extend our thanks to all those who have contributed, directly or indirectly to make this project successful.

Sushovan Mondal

Sushovan Mondal

Sapnagupta

Sapna Gupta

Saumya Mukherjee

Saumya Mukherjee

Index

Abstract

Handwriting analysis is a valuable technique for understanding writing patterns and behavioral traits, but traditional graphological methods are often manual, subjective, and time-consuming. This project presents a web-based English Handwriting Analysis System that automates the analysis of handwritten text using digital image processing and intelligent rule-based techniques. The system accepts scanned or photographed handwriting images as input and performs multiple analyses to extract meaningful handwriting features.

The proposed system analyzes key handwriting characteristics such as sentence alignment, letter slant, handwriting size, writing pressure, word spacing, line spacing, t-bar placement, and i-dot positioning. These features are processed using image preprocessing, spatial analysis, and geometric measurements to provide consistent and interpretable results. A browser-based user interface allows users to upload images, select analysis options, and visualize results in an intuitive manner.

The application is implemented in Python and follows a modular architecture that separates the user interface from the processing and analysis logic. The system operates in a local execution environment, ensuring ease of deployment and quick response time during development and testing. By automating handwriting analysis, the project aims to improve efficiency, reduce human bias, and provide a systematic approach to studying handwriting characteristics.

This project demonstrates the effective use of digital image processing techniques for handwriting analysis and highlights the potential of intelligent automation in educational, research, and preliminary personality assessment applications.

Introduction

Handwriting is a complex human activity that reflects not only linguistic content but also behavioral, emotional, and psychological traits of an individual. The analysis of handwriting, often referred to as graphology, has traditionally been performed manually by experts, making the process time-consuming and subjective. With advancements in digital image processing and intelligent systems, automated handwriting analysis has become a feasible and efficient alternative.

This project presents a web-based English Handwriting Analysis System that automatically examines handwritten text images and extracts meaningful handwriting features. The system allows users to upload scanned or photographed handwriting samples and performs multiple analyses such as sentence alignment, letter slant, handwriting size, writing pressure, word spacing, line spacing, t-bar placement, and i-dot characteristics. These features are analyzed to infer behavioral and personality-related insights in a structured and interpretable manner.

The application is implemented using the Python programming language and follows a modular architecture to separate user interaction, processing logic, and analysis components. A browser-based interface enables easy interaction without requiring specialized software installation. The system operates in a local execution environment, ensuring quick response time and ease of deployment during development and testing.

By combining digital image processing techniques with rule-based analysis, the project aims to provide consistent, objective, and reproducible handwriting assessments. This system can be useful in educational evaluation, preliminary personality assessment, and research applications where handwriting characteristics need to be studied efficiently.

Overall, the project demonstrates how intelligent automation can enhance traditional handwriting analysis by making it more accessible, scalable, and systematic while maintaining interpretability of results.

What is Graphology

Graphology is the systematic study of handwriting with the aim of understanding an individual's personality traits, behavior, and emotional characteristics. It is based on the concept that handwriting is a form of expressive movement influenced by the writer's subconscious mind, habits, and psychological state. Since writing is a learned motor skill that becomes automatic over time, graphologists believe that personal traits are reflected naturally in handwriting patterns.

In graphology, several handwriting features are analyzed, including letter size, slant, spacing between words and lines, pressure applied while writing, alignment of text, and the shape of specific letters. Each of these features is associated with certain behavioral tendencies. For example, large handwriting is often linked with confidence and extroversion, while small handwriting may indicate concentration and introversion. Similarly, right-slanted writing can suggest emotional expressiveness, whereas left-slanted writing may indicate reservation or introspection.

Graphology has been applied in various fields such as personality assessment, career guidance, education, recruitment, and forensic document examination. It is sometimes used to understand learning styles, emotional states, and compatibility in professional or personal contexts. However, graphology is not considered an exact or universally accepted science, as interpretations can vary and depend on the experience of the analyst.

With advancements in technology, graphology is increasingly supported by automated handwriting analysis using digital image processing and artificial intelligence. These systems aim to reduce subjectivity and improve consistency by applying measurable rules and computational techniques to handwriting samples.

How we used Graphology

In this project, **graphology is used as the conceptual foundation** for analyzing handwritten text and interpreting behavioral characteristics. Traditional graphology involves examining handwriting features manually to infer personality traits. In contrast, this project automates that process by converting graphological principles into **measurable, rule-based features** using digital image processing.

The system first accepts a handwritten English text image as input. Using image preprocessing techniques, the handwritten content is isolated from the background. Based on established graphology concepts, specific handwriting features are then extracted and analyzed. For example, **sentence and text alignment** are used to study emotional balance and attitude, **letter slant** is analyzed to understand emotional expressiveness, and **handwriting size** is evaluated to infer confidence and focus. Similarly, **writing pressure** is assessed through stroke thickness and intensity, while **word and line spacing** are analyzed to understand thinking patterns and organization.

Graphology also emphasizes the importance of individual letter features. In this project, **t-bar placement** is analyzed to study ambition and self-esteem, and **i-dot position and shape** are examined to assess attention to detail, imagination, and energy levels. Each feature is interpreted using predefined graphological rules rather than subjective judgment.

By implementing these graphology rules computationally, the project ensures consistency, objectivity, and repeatability in handwriting analysis. Thus, graphology is effectively transformed from a manual interpretive practice into an automated, technology-driven handwriting analysis system.

Overview of the project

The English Handwriting Analysis System is a web-based application designed to automatically analyze handwritten text using graphology principles combined with digital image processing. The project aims to convert traditional, manual handwriting analysis into an objective and automated process that can provide consistent and interpretable results.

The system allows users to upload scanned or photographed images of handwritten English text through a browser-based interface. Once an image is uploaded, it undergoes preprocessing to enhance text clarity and separate handwriting from the background. The application then extracts multiple handwriting features such as sentence alignment, letter slant, handwriting size, writing pressure, word spacing, line spacing, t-bar placement, and i-dot characteristics.

Each extracted feature is evaluated using predefined graphological rules to infer behavioral and personality-related traits. The analysis results are presented visually and textually, making them easy to understand for users without prior knowledge of graphology. The project follows a modular architecture, separating the user interface, processing logic, and analysis components to ensure maintainability and scalability.

The application runs in a local execution environment and does not require internet connectivity, making it suitable for academic, educational, and experimental use. By integrating graphology concepts with automated image processing techniques, the project demonstrates how intelligent systems can enhance traditional handwriting analysis, reduce human bias, and provide a structured approach to studying handwriting characteristics.

Programming Language – Python

Python is used as the core programming language for the project. It supports rapid development, readability, and easy integration of different components. Python enables efficient implementation of image processing logic, data handling, and application control within a single, cohesive environment.

Application Type – Web-Based Application

The project is developed as a web-based application that allows users to interact through a browser. This approach provides ease of access, platform independence, and real-time interaction, enabling users to upload handwriting images and view analysis results dynamically.

Development Environment – Visual Studio Code

Visual Studio Code is used as the primary development environment. It provides features such as syntax highlighting, debugging support, extension management, and integrated terminal access, which help in efficient coding, testing, and maintenance of the project.

Deployment Platform – Localhost

The application is deployed on a local machine using a local server environment. Localhost deployment allows easy testing, debugging, and performance evaluation without requiring internet connectivity or cloud infrastructure during the development phase.

Operating System

The system is platform-independent and can run on Windows, Linux, or macOS. As long as the operating system supports Python execution, the application can function correctly, making it portable and adaptable across different computing environments.

User Interface Technology – Browser-Based Interface

The user interface is rendered inside a web browser, providing a clean and interactive experience. Users can upload images, select analysis options, and view results visually without installing any additional desktop software.

Architecture Style – Modular Application Architecture

The project follows a modular architecture where different functionalities are divided into separate components. This separation improves code readability, reusability, debugging efficiency, and future scalability by allowing independent modification of modules.

Image Data Handling – Digital Image Processing System

The system processes digital images of handwritten text captured through scanning or photography. It handles image loading, preprocessing, and transformation, enabling accurate extraction of handwriting features from raw image data.

File Handling – Local File System

The application uses the local file system for uploading and temporarily storing handwriting images. This approach simplifies file management, ensures faster access to data, and avoids dependency on external storage or network-based file systems.

Execution Environment – Python Runtime Environment

The Python runtime environment is required to execute the application. It manages memory, program execution, and interaction between different modules, ensuring that the application runs smoothly and efficiently on the target system.

Libraries Used

The project integrates OpenCV for image processing, NumPy for numerical computation, Streamlit for user interaction, and Scikit-learn for machine learning–based handwriting analysis within a Python environment.

Library	Role in Project
OpenCV	*Image preprocessing, contour detection, handwriting feature extraction*
NumPy	*Numerical computation and statistical analysis*
Streamlit	*Web-based user interface and visualization*
Scikit-learn	*Clustering and regression algorithms*
Python	*Core programming and system integration*

OpenCV (cv2)

OpenCV (Open Source Computer Vision Library) is used as the core image processing library in this project.

Purpose in the Project:

- Converts uploaded handwriting images into grayscale and binary formats
- Performs image thresholding using Otsu's method
- Applies morphological operations such as dilation and opening to enhance text regions
- Detects contours to identify letters, words, and lines

- Draws bounding boxes, regression lines, and visual markers on detected handwriting features

Why OpenCV?
OpenCV provides fast, optimized, and reliable functions for low-level image processing, which is essential for handwriting analysis.

NumPy

NumPy (Numerical Python) is used for numerical and mathematical computations.

Purpose in the Project:

- Stores image data as multidimensional arrays
- Performs statistical calculations such as mean, median, and standard deviation
- Computes distances, angles, slopes, and ratios between handwriting components
- Reshapes and manipulates data for clustering and regression algorithms

Why NumPy?
NumPy offers efficient array-based computation, making it ideal for processing image pixels and numerical features.

Streamlit

Streamlit is a Python-based web application framework used to create the user interface.

Purpose in the Project:

- Provides a web interface for uploading handwriting images

- Allows users to select different handwriting analysis features
- Displays processed images, tables, and analysis results dynamically
- Enables quick local deployment without complex web development

Why Streamlit?
Streamlit simplifies the creation of interactive data applications and allows rapid prototyping with minimal code.

<u>Scikit-learn (sklearn)</u>

Scikit-learn is a machine learning library used for implementing unsupervised learning and regression techniques.

Purpose in the Project:

- **DBSCAN:** Groups handwritten words and text lines based on spatial proximity
- **K-Means Clustering:** Separates handwriting components such as stems and dots
- **Linear Regression:** Estimates the slope of handwritten sentences for alignment analysis

Why Scikit-learn?
Scikit-learn provides reliable, well-tested machine learning algorithms with simple APIs suitable for academic projects.

<u>Python Standard Libraries</u>

The project also relies on built-in Python capabilities.

Purpose in the Project:

- Modular function definition and code organization
- Control flow, data structures, and exception handling

- Efficient integration of multiple libraries into a single system

Why Python?

Python’s simplicity, readability, and strong ecosystem make it ideal for computer vision and machine learning applications.

Algorithms used

The project combines image processing, unsupervised machine learning, regression analysis, and rule-based algorithms. Techniques such as thresholding, morphological operations, contour analysis, clustering, and spatial measurements extract handwriting features. These features are interpreted using logical rules to analyze alignment, spacing, slant, pressure, and character structures.

Otsu's Thresholding Algorithm

- Used to convert grayscale handwriting images into binary images.
- Automatically determines the optimal threshold value.
- Helps separate foreground text from background.

Application in Project:
Text extraction for all handwriting analysis features.

Morphological Operations

- Erosion
- Dilation
- Opening (Erosion + Dilation)

Purpose:

- Noise removal
- Stroke enhancement
- Word and line segmentation

Application in Project:
Sentence alignment, word spacing, line spacing, t-bar and i-dot detection.

Contour Detection Algorithm

- Uses OpenCV's contour detection technique.
- Identifies connected components representing letters, words, and strokes.

Application in Project:
Tall & narrow letters, t-bars, i-dots, word spacing, handwriting size analysis.

Non-Maximum Suppression (NMS)

- Eliminates overlapping bounding boxes.
- Retains only the most relevant detections.

Application in Project:
Sentence alignment and line detection.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

- An **unsupervised clustering algorithm**.
- Groups data points based on density rather than fixed cluster count.

Application in Project:

- Line segmentation
- Word grouping
- Line spacing analysis

Why DBSCAN:
Handles variable handwriting spacing and unknown number of text lines.

K-Means Clustering

- An **unsupervised clustering algorithm**.
- Partitions data into clusters based on feature similarity.

Application in Project:
Separating i-dots from letter stems based on size and height.

Linear Regression Algorithm

- Supervised regression technique.
- Computes the slope of handwritten lines.

Application in Project:
Sentence alignment analysis (upward, downward, straight text).

Connected Component Analysis

- Identifies and labels connected regions in a binary image.

Application in Project:
Handwriting size detection and letter segmentation.

Aspect Ratio–Based Heuristic Algorithm

- Uses height-to-width ratios of bounding boxes.

Application in Project:
Detection of tall and narrow letters.

Stroke Thickness & Intensity Estimation Algorithm

- Calculates average stroke thickness and pixel intensity.

Application in Project:
Writing pressure analysis (heavy vs light pressure).

Minimum Area Rectangle Algorithm

- Computes the smallest enclosing rotated rectangle.

Application in Project:
Letter slant detection.

Distance-Based Spatial Analysis

- Uses Euclidean distance and geometric thresholds.

Application in Project:

- Word spacing
- Line spacing
- i-dot placement analysis

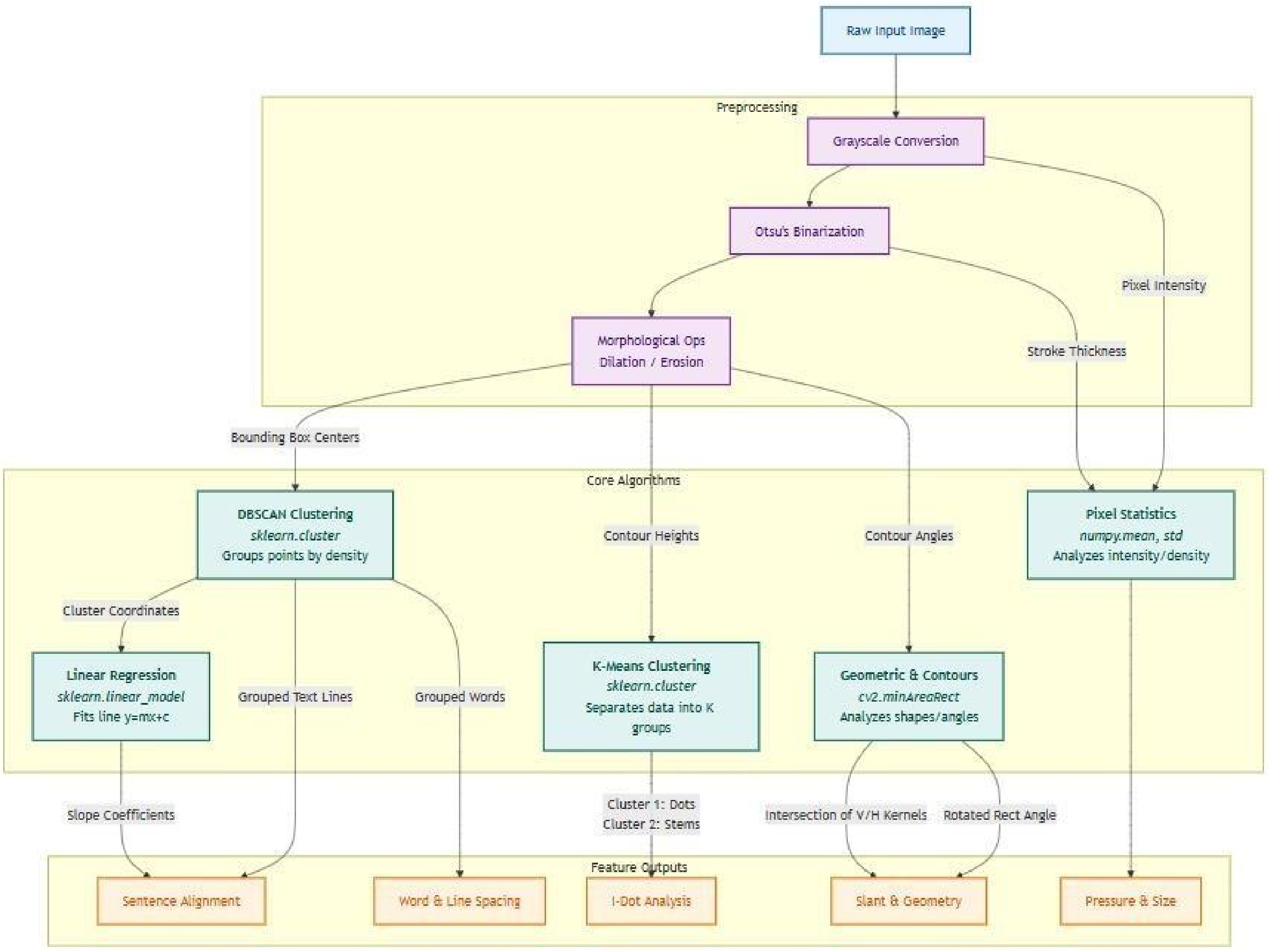
Rule-Based Decision Logic

- Uses predefined thresholds and heuristics.
- Converts numerical features into interpretable results.

Application in Project:
Final classification of handwriting traits (slant, spacing, alignment, pressure, etc.).

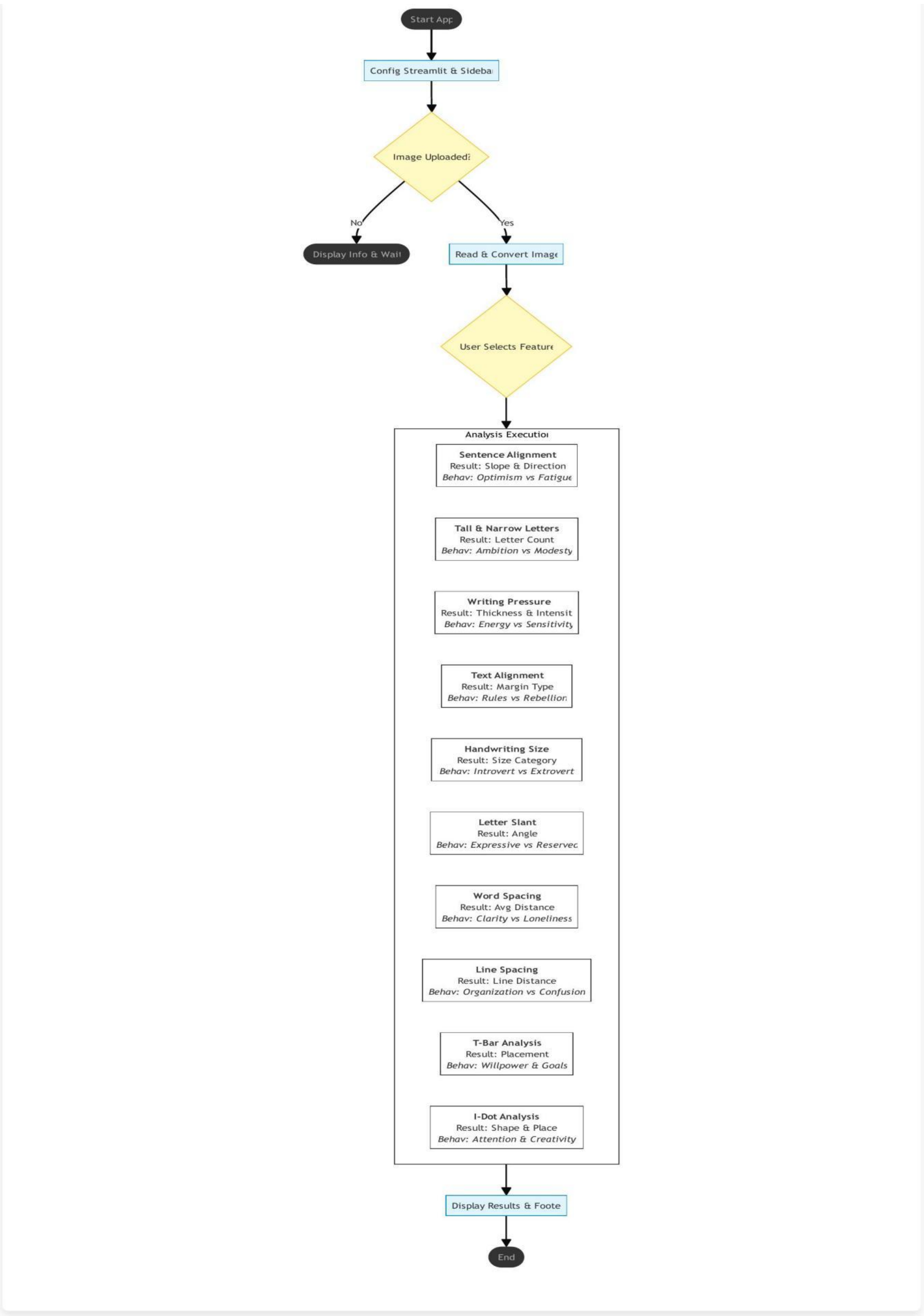
Methodology

1) Analysis



Raw Input Image
Preprocessing
Grayscale Conversion
Otsu's Binarization
Morphological Ops Dilation / Erosion
Pixel Intensity
Stroke Thickness
Bounding Box Centers
Core Algorithms
DBSCAN Clustering sklearn.cluster Groups points by density
Contour Heights
Contour Angles
Pixel Statistics numpy.mean, std Analyzes intensity/density
Cluster Coordinates
Linear Regression sklearn.linear_model Fits line y=mx+c
Grouped Text Lines
Grouped Words
K-Means Clustering sklearn.cluster Separates data into K groups
Geometric & Contours cv2.minAreaRect Analyzes shapes/angles
Slope Coefficients
Cluster 1: Dots Cluster 2: Stems
Intersection of V/H Kernels
Rotated Rect Angle
Feature Outputs
Sentence Alignment
Word & Line Spacing
I-Dot Analysis
Slant & Geometry
Pressure & Size

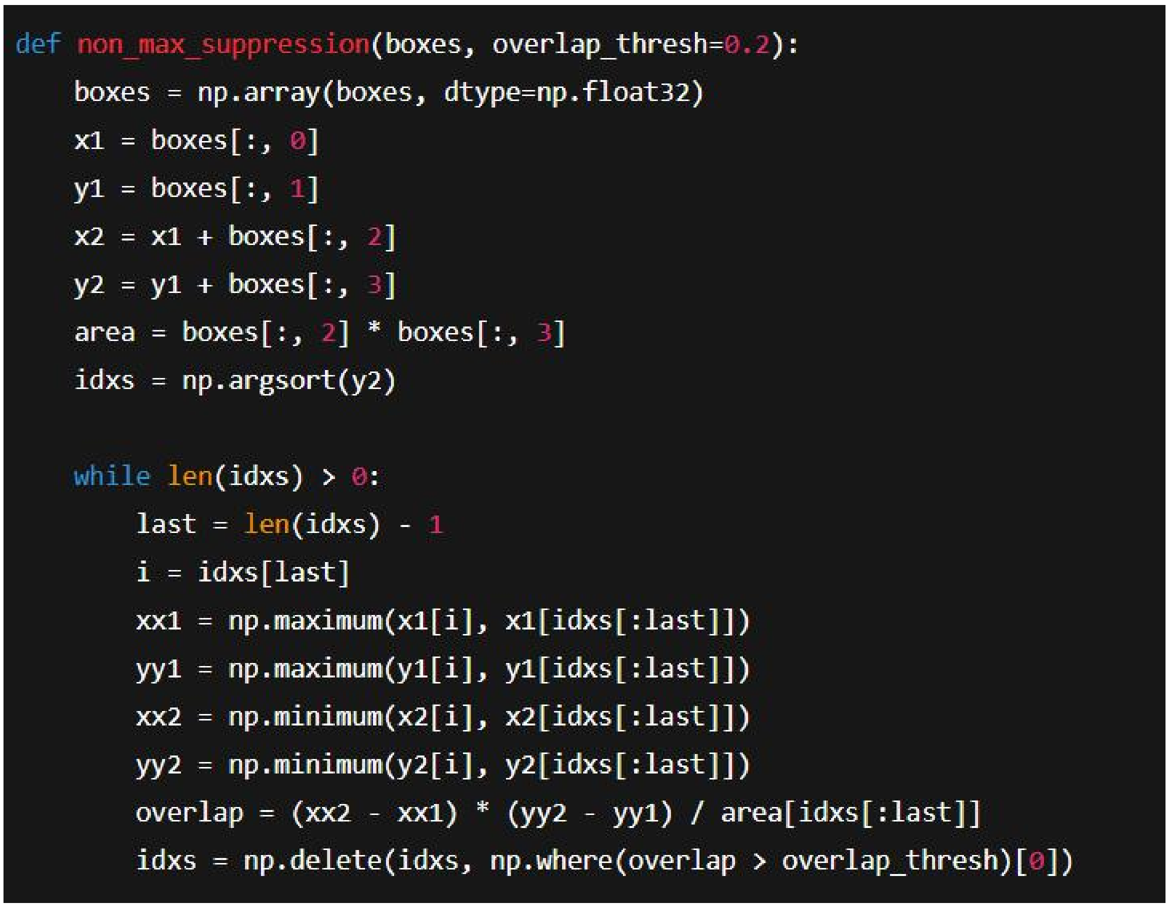
2) User Interface



Handwriting Analysis System – Code Implementation

This code implements multiple image processing and machine learning techniques to analyze different handwriting characteristics such as alignment, slant, spacing, pressure, letter structure, and stylistic features. OpenCV is used for image preprocessing and feature extraction, while Scikit-learn is used for clustering and regression analysis.

Non-Maximum Suppression (NMS)



```
def non_max_suppression(boxes, overlap_thresh=0.2):
    boxes = np.array(boxes, dtype=np.float32)
    x1 = boxes[:, 0]
    y1 = boxes[:, 1]
    x2 = x1 + boxes[:, 2]
    y2 = y1 + boxes[:, 3]
    area = boxes[:, 2] * boxes[:, 3]
    idxs = np.argsort(y2)

    while len(idxs) > 0:
        last = len(idxs) - 1
        i = idxs[last]
        xx1 = np.maximum(x1[i], x1[idxs[:last]])
        yy1 = np.maximum(y1[i], y1[idxs[:last]])
        xx2 = np.minimum(x2[i], x2[idxs[:last]])
        yy2 = np.minimum(y2[i], y2[idxs[:last]])
        overlap = (xx2 - xx1) * (yy2 - yy1) / area[idxs[:last]]
        idxs = np.delete(idxs, np.where(overlap > overlap_thresh)[0])
```

Purpose:
Removes overlapping bounding boxes and keeps only the most relevant ones.

Implementation Logic:

- Converts bounding boxes into arrays.
- Sorts boxes by bottom-right Y coordinate.
- Iteratively keeps the largest box and removes other boxes that overlap beyond a threshold.
- Prevents duplicate detection of the same text region.

Why used:
Ensures clean detection of words/letters without redundant overlaps.

<u>Sentence Alignment Detection</u>

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
_, binary = cv2.threshold(gray, 0, 255,
                          cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)

kernel_dilate = cv2.getStructuringElement(cv2.MORPH_RECT, (30, 8))
dilated = cv2.dilate(binary, kernel_dilate, iterations=1)

contours, _ = cv2.findContours(dilated, cv2.RETR_EXTERNAL,
                               cv2.CHAIN_APPROX_SIMPLE)
```

Clustering Lines

```
y_centers = np.array([[y + h//2] for (x, y, w, h) in boxes])
db = DBSCAN(eps=30, min_samples=1).fit(y_centers)
```

Slope Calculation

```
reg = LinearRegression().fit(x_coords, y_coords)
slope = reg.coef_[0]
```

Purpose:
Determines whether handwriting lines are **upward, downward, or straight**.

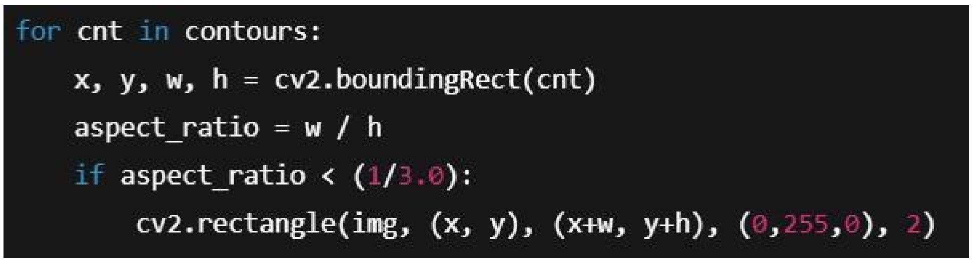
Implementation Logic:

- Converts image to grayscale and binarizes using Otsu’s thresholding.
- Uses morphological operations to merge words into line-like structures.
- Extracts bounding boxes of text contours.
- Groups text into lines using **DBSCAN clustering** on Y-coordinates.
- Applies **Linear Regression** to each line to compute slope.
- Classifies slope direction based on a dynamic threshold.

Output:

- Line direction (upward / downward / straight)
- Slope value
- Word count per line

<u>Tall & Narrow Letter Detection</u>



```
for cnt in contours:
    x, y, w, h = cv2.boundingRect(cnt)
    aspect_ratio = w / h
    if aspect_ratio < (1/3.0):
        cv2.rectangle(img, (x, y), (x+w, y+h), (0,255,0), 2)
```

Purpose:
Identifies letters like ***l, t, f*** that are tall and narrow.

Implementation Logic:

- Extracts contours from **binarized image**.
- Filters out small noise using **height thresholds**.
- Computes aspect ratio **(width ÷ height)**.
- Marks letters with very small **width-to-height** ratio.

Interpretation:
Often linked to writing style, discipline, or emphasis.

Writing Pressure Detection

```
inverted = cv2.bitwise_not(gray)
_, binary = cv2.threshold(inverted, 0, 255,
                          cv2.THRESH_BINARY + cv2.THRESH_OTSU)

stroke_thickness = np.mean(cv2.reduce(binary, 0, cv2.REDUCE_AVG)) / 255
intensity = np.mean(gray[gray < 200])
```

Purpose:
Estimates whether the writer uses **heavy or light pressure**.

Implementation Logic:

- Inverts grayscale image to highlight ink strokes.
- Measures average stroke thickness using pixel intensity reduction.
- Computes mean intensity of text pixels.
- Classifies pressure based on thickness and darkness.

Output:

- Pressure category (Heavy / Light)
- Stroke thickness
- Intensity value

Margin / Text Alignment Detection

```
ys, xs = np.where(binary > 0)
avg_x = np.mean(xs)
img_center = binary.shape[1] / 2
```

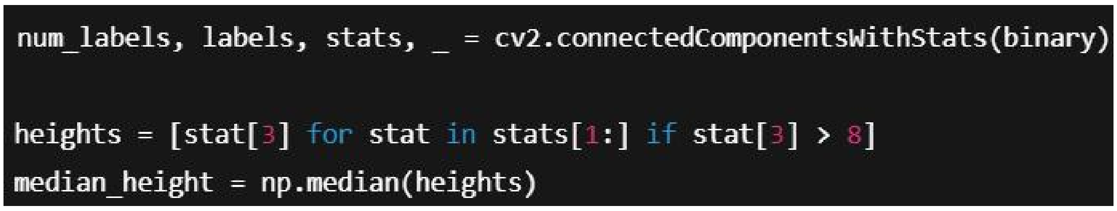
Purpose:
Detects **left, right, or center alignment** of handwriting.

Implementation Logic:

- Finds all ink pixels in the image.
- Computes average X-coordinate of text.
- Compares it with image center.
- Classifies alignment based on proximity.

Use Case:
Useful in handwriting formatting and personality analysis.

Handwriting Size Detection



```
num_labels, labels, stats, _ = cv2.connectedComponentsWithStats(binary)

heights = [stat[3] for stat in stats[1:] if stat[3] > 8]
median_height = np.median(heights)
```

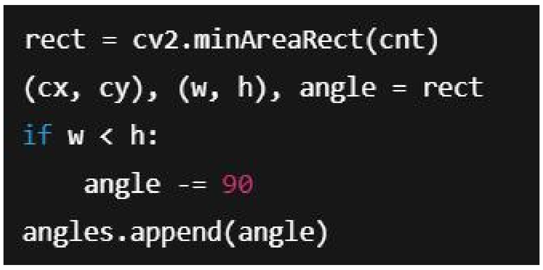
Purpose:
Determines whether handwriting is **small or large**.

Implementation Logic:

- Uses connected component analysis.
- Measures height of individual characters.
- Computes median character height.
- Compares it with overall image height.

Output:
Small Handwriting / Large Handwriting

Letter Slant Detection



```python
rect = cv2.minAreaRect(cnt)
(cx, cy), (w, h), angle = rect
if w < h:
    angle -= 90
angles.append(angle)
```

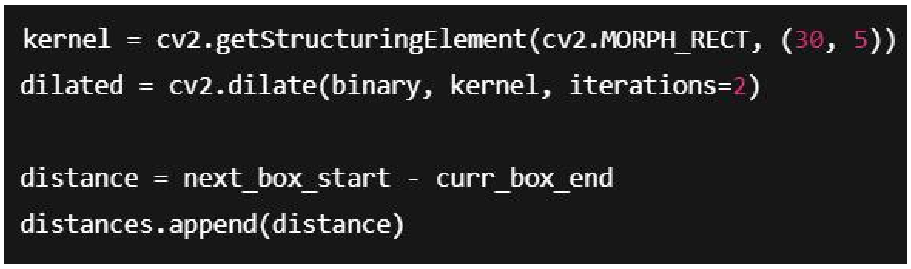
Purpose:
Detects **left slant, right slant, or vertical writing**.

Implementation Logic:

- Extracts tall letter contours.
- Uses **minimum area rectangle** to estimate angle.
- Normalizes angle values.
- Averages all detected angles.
- Classifies slant direction based on angle range.

Visual Output:
Draws rotated bounding boxes around slanted letters.

<u>Word Spacing Analysis</u>



```python
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (30, 5))
dilated = cv2.dilate(binary, kernel, iterations=2)

distance = next_box_start - curr_box_end
distances.append(distance)
```

Purpose:
Measures spacing between consecutive words.

Implementation Logic:

- Dilates text to merge letters into words.
- Extracts word bounding boxes.

- Groups words into lines using **DBSCAN**.
- Computes horizontal distances between adjacent words.
- Compares spacing with average word width.

Output:
Wide / Narrow / Consistent spacing

Line Spacing Analysis

```
distance = next_line_top - curr_line_bottom
line_height = sorted_lines[i][3] - sorted_lines[i][1]
```

Purpose:
Analyzes vertical spacing between text lines.

Implementation Logic:

- Merges words into full text lines using dilation.
- Clusters bounding boxes by Y-center.
- Computes distance between bottom of one line and top of next.
- Compares spacing with average line height.

Output:
Wide / Narrow / Consistent line spacing

T-Bar Placement Analysis

```
vertical_kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (1, 15))
stems = cv2.morphologyEx(binary, cv2.MORPH_OPEN, vertical_kernel)

horizontal_kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (10, 1))
bars = cv2.morphologyEx(binary, cv2.MORPH_OPEN, horizontal_kernel)
```

Placement Logic

```
placement_ratio = (bar_center_y - sy) / sh
```

Purpose:
Analyzes where the cross-bar of the letter **‘t’** is placed.

Implementation Logic:

- Extracts vertical stems using vertical morphology.
- Extracts horizontal bars using horizontal morphology.
- Matches bars with stems based on overlap.
- Calculates bar position relative to stem height.

Categories:
High / Middle / Low on stem

I-Dot Analysis

```
kmeans = KMeans(n_clusters=2).fit(heights_arr)
middle_zone_height = min(kmeans.cluster_centers_)
```

Dot Classification

```
aspect_ratio = w / h
if aspect_ratio > 0.7 and aspect_ratio < 1.3:
    shapes.append("Circle")
```

Purpose:
Analyzes placement and shape of **dots over ‘i’**.

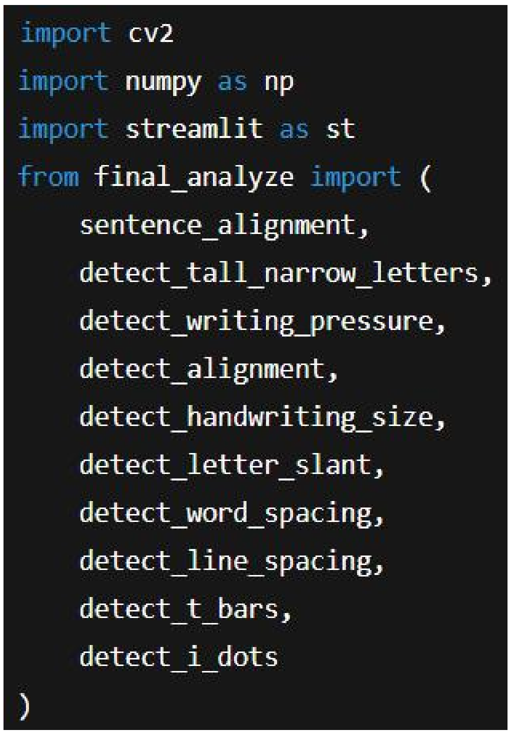
Implementation Logic:

- Separates dots and stems based on size using **KMeans clustering**.
- Matches dots above corresponding stems.
- Measures dot position (above, left, right).
- Determines dot shape using area and aspect ratio.

Output:

- Number of i-dots
- Dominant placement
- Dominant shape (Dot / Circle / Slash)

Library Imports & Module Integration



```
import cv2
import numpy as np
import streamlit as st
from final_analyze import (
    sentence_alignment,
    detect_tall_narrow_letters,
    detect_writing_pressure,
    detect_alignment,
    detect_handwriting_size,
    detect_letter_slant,
    detect_word_spacing,
    detect_line_spacing,
    detect_t_bars,
    detect_i_dots
)
```

Implementation

- **OpenCV (cv2)** → Image decoding & color conversion
- **NumPy** → Byte-to-image conversion
- **Streamlit** → Web interface
- **final_analyze.py** → Backend analysis logic

This separates **UI logic** from **processing logic**, following modular design.

Streamlit Page Configuration

```
st.set_page_config(
    page_title="Enhanced English Handwriting Analysis",
    layout="wide"
)
```

Implementation

- Sets page title and wide layout for side-by-side image display.
- Improves readability for analysis outputs.

Application Header & Description

```
st.title("🖋️ Enhanced English Handwriting Analysis Tool (V-Final)")
st.write("This version uses your 5 original features...")
```

Implementation

- Displays project title.
- Describes the tool's scope (original + enhanced features).

Image Upload Handling

```
uploaded_file = st.file_uploader(
    "Upload an image (jpg, jpeg, png)",
    type=["jpg", "jpeg", "png"]
)
```

Implementation

- Accepts handwriting image from the user.
- Supports common image formats.

Image Decoding & Preprocessing

```python
file_bytes = np.asarray(bytearray(uploaded_file.read()), dtype=np.uint8)
img_color = cv2.imdecode(file_bytes, 1)
img_gray = cv2.cvtColor(img_color, cv2.COLOR_BGR2GRAY)
```

Implementation

- Converts uploaded file bytes into OpenCV image.
- Creates **both color and grayscale versions**.
- Grayscale is reused for multiple analysis functions → efficiency.

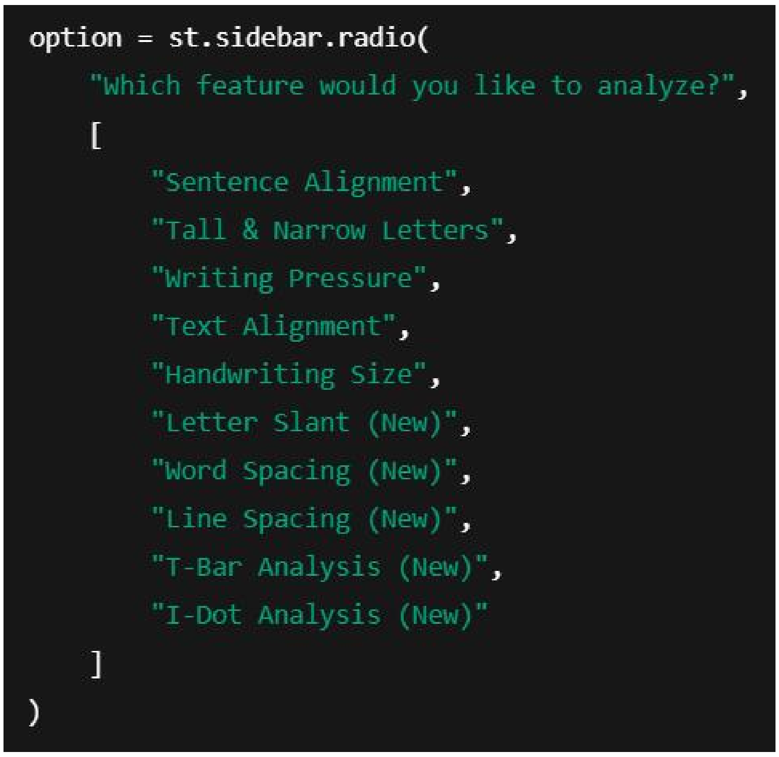
Display Uploaded Image

```python
st.image(img_color, channels="BGR",
         caption="📥 Your Uploaded Image",
         use_column_width=True)
```

Implementation

- Shows the original image to the user before analysis.
- Uses BGR channel order (OpenCV standard).

Sidebar Feature Selection Menu

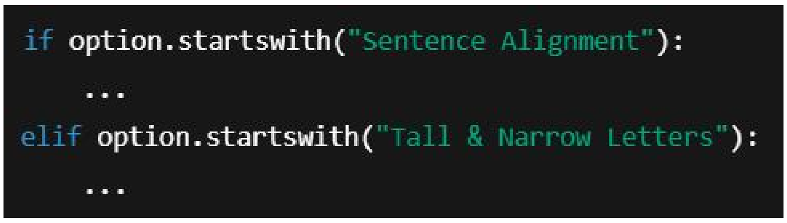


```python
option = st.sidebar.radio(
    "Which feature would you like to analyze?",
    [
        "Sentence Alignment",
        "Tall & Narrow Letters",
        "Writing Pressure",
        "Text Alignment",
        "Handwriting Size",
        "Letter Slant (New)",
        "Word Spacing (New)",
        "Line Spacing (New)",
        "T-Bar Analysis (New)",
        "I-Dot Analysis (New)"
    ]
)
```

Implementation

- Uses **radio buttons** for single-feature selection.
- Keeps UI simple and avoids multi-processing overload.

Conditional Feature Execution Logic



```python
if option.startswith("Sentence Alignment"):
    ...
elif option.startswith("Tall & Narrow Letters"):
    ...
```

Implementation

- Each feature is **executed independently**.
- Prevents unnecessary computation.
- Enables modular expansion of features.

Sentence Alignment Integration

```
processed_img, results, threshold =
    sentence_alignment(img_color.copy())
```

Implementation

- Sends a copy of image to avoid overwriting original.
- Displays:
  - Detected regression lines
  - Line slopes and directions
  - Word counts
- Converts results into a **table** for clarity.

Tall & Narrow Letters Integration

```
processed_img, count =
    detect_tall_narrow_letters(img_color.copy())
```

Implementation

- Returns annotated image + count.
- Adds behavioral interpretation based on detection count.

Writing Pressure Integration

```
pressure, thickness, intensity =
    detect_writing_pressure(img_color.copy())
```

Implementation

- Displays numerical measurements.
- Maps them to **human-readable personality traits**.

Text Alignment (Bug-Fixed Integration)

```
alignment_result =
    detect_alignment(img_gray.copy())
```

Implementation

- Original function returns **only one string**.
- UI logic updated to prevent unpacking crash.
- Demonstrates **defensive programming**.

Handwriting Size (Bug-Fixed Integration)

```
size_result =
    detect_handwriting_size(img_gray.copy())
```

Implementation

- Matches return type of backend function.
- Gracefully handles “No handwriting detected” case.

Letter Slant Integration

```
processed_img, slant_category, avg_angle =
    detect_letter_slant(img_color.copy())
```

Implementation

- Displays slanted letter bounding boxes.
- Shows average slant angle.
- Adds behavioral explanation.

Word Spacing Integration

```
processed_img, spacing_category, avg_distance =
    detect_word_spacing(img_color.copy())
```

Implementation

- Visualizes spacing lines.
- Compares spacing with average word width.
- Interprets spacing psychologically.

Line Spacing Integration (Fixed)

```
processed_img, spacing_category, avg_distance =
    detect_line_spacing(img_color.copy())
```

Implementation

- Uses corrected morphology kernel.
- Displays vertical spacing markers.
- Classifies spacing type.

T-Bar Analysis Integration

```
processed_img, report =
    detect_t_bars(img_color.copy())
```

Implementation

- Receives structured dictionary output.
- Displays dominant placement and count.
- Explains personality correlation.

I-Dot Analysis Integration

```
processed_img, report =
    detect_i_dots(img_color.copy())
```

Implementation

- Handles complex multi-feature output.
- Displays dot placement and shape.
- Combines spatial + geometric analysis.

No-Image Handling

```
else:
    st.info("Please upload an image to begin.")
```

Implementation

- Prevents app crash.
- Improves user experience.

Footer & Metadata

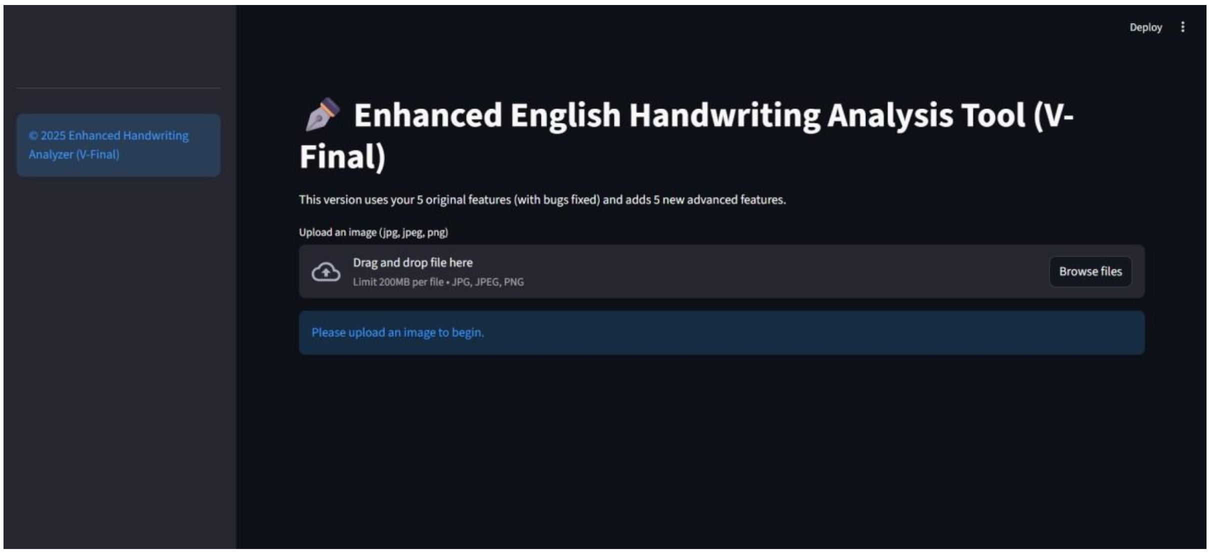
```
st.sidebar.info("© 2025 Enhanced Handwriting Analyzer (V-Final)")
```

Implementation

- Adds professional project branding.

Interface

Here below we show the interface design how it look like



Deploy
© 2025 Enhanced Handwriting Analyzer (V-Final)

Enhanced English Handwriting Analysis Tool (V-Final)

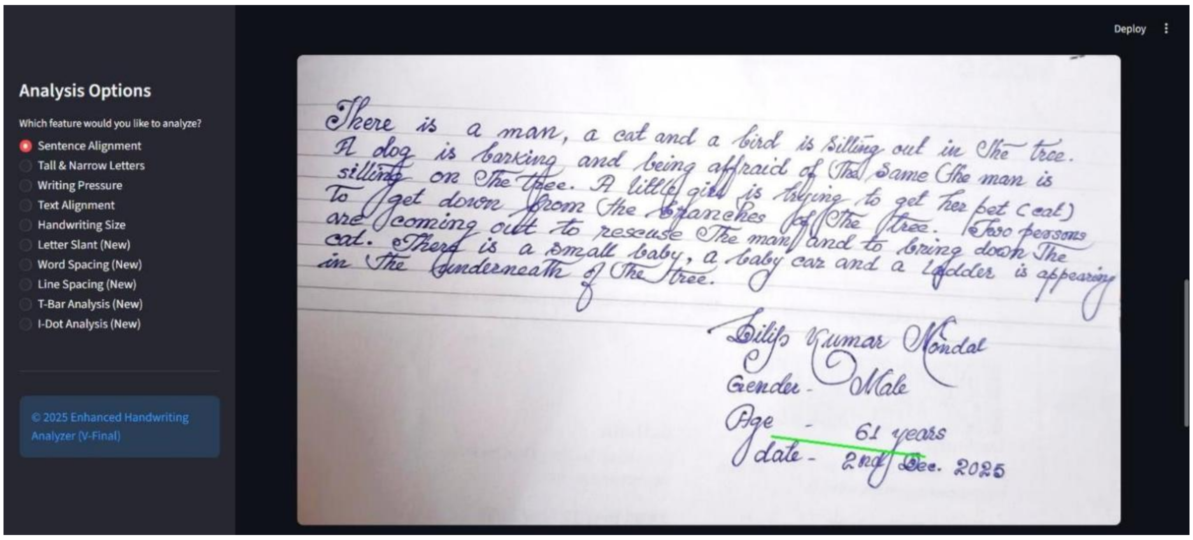
This version uses your 5 original features (with bugs fixed) and adds 5 new advanced features.

Upload an image (jpg, jpeg, png)

Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG
Browse files

Please upload an image to begin.

Here it is when used for handwriting analysis



Which feature would you like to analyze?

- [✓ Sentence Alignment]
- Tall & Narrow Letters
- Writing Pressure
- Text Alignment
- Handwriting Size
- Letter Slant (New)
- Word Spacing (New)
- Line Spacing (New)
- T-Bar Analysis (New)
- I-Dot Analysis (New)

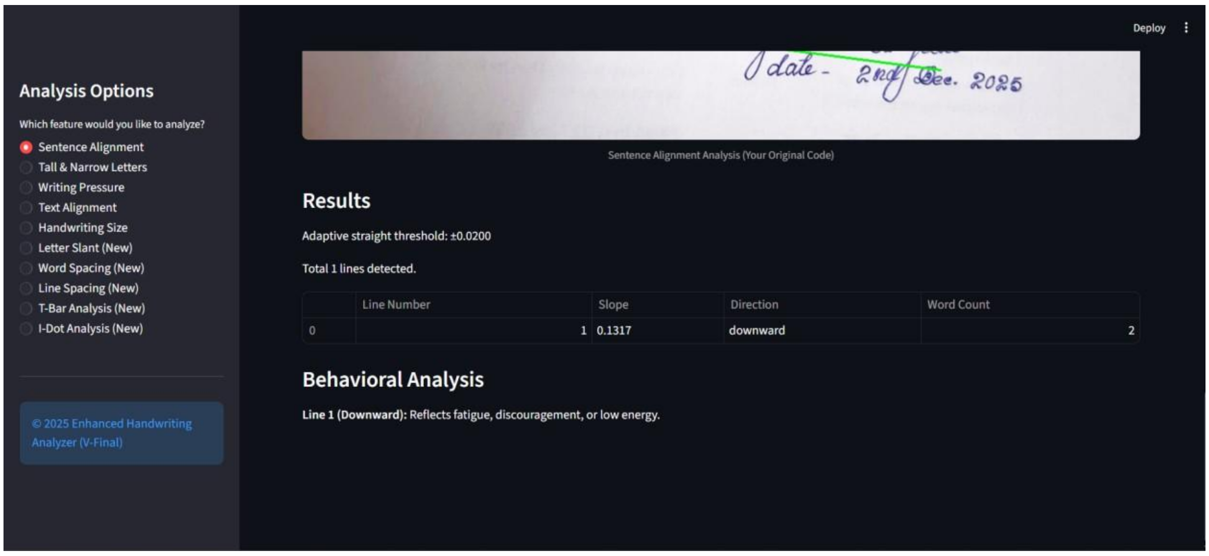
There is a man, a cat and a bird is sitting out in the tree.
A dog is barking and being affraid of the same the man is
sitting on the tree. A little girl is trying to get her pet (cat)
to get down from the branches of the tree. Two persons
are coming out to rescuse the man and to bring down the
cat. There is a small baby, a baby car and a ladder is appearing
in the underneath of the tree.

Dilip Kumar Mondal

Gender - Male

Age - 61 years

date - 2nd Dec. 2025



Deploy

Analysis Options

Which feature would you like to analyze?

- [✓ Sentence Alignment]
- Tall & Narrow Letters
- Writing Pressure
- Text Alignment
- Handwriting Size
- Letter Slant (New)
- Word Spacing (New)
- Line Spacing (New)
- T-Bar Analysis (New)
- I-Dot Analysis (New)

© 2025 Enhanced Handwriting Analyzer (V-Final)

date - 2nd Dec. 2025

Sentence Alignment Analysis (Your Original Code)

Results

Adaptive straight threshold: ±0.0200

Total 1 lines detected.

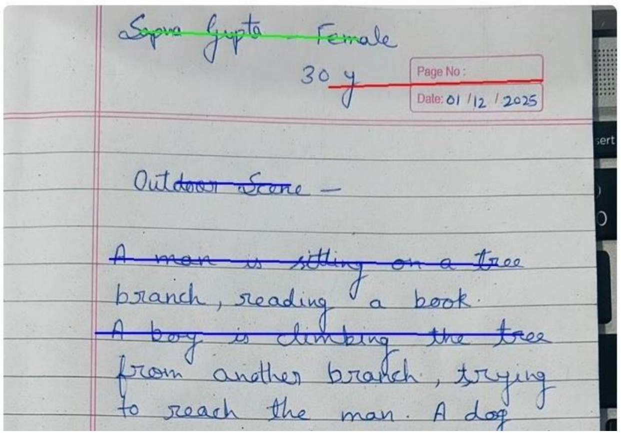
	Line Number	Slope	Direction	Word Count
0	1	0.1317	downward	2

Behavioral Analysis

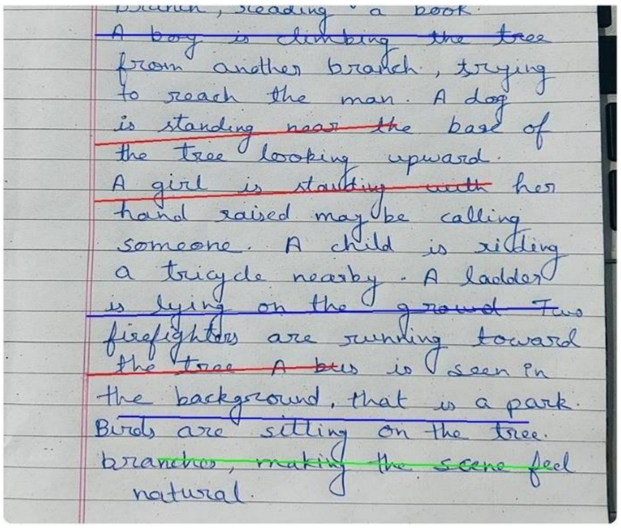
Line 1 (Downward): Reflects fatigue, discouragement, or low energy.

Here is another example

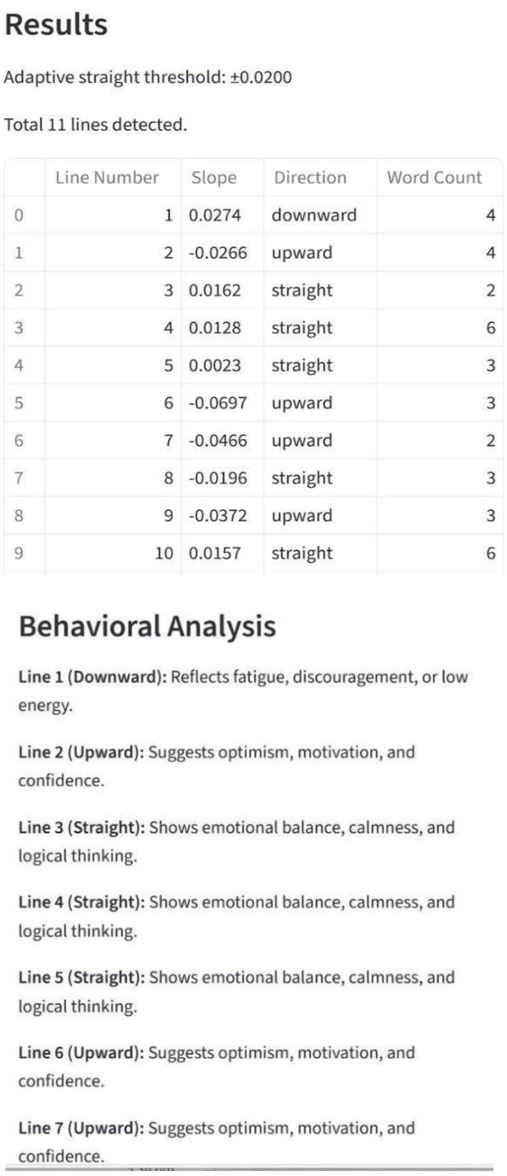
Analysis Results: Sentence Alignment



Sapna Gupta Female
30 y
Page No :
Date: 01 /12 /2025
Outdoor Scene –
A man is sitting on a tree
branch, reading a book.
A boy is climbing the tree
from another branch, trying
to reach the man. A dog



branch, reading a book.
A boy is climbing the tree
from another branch, trying
to reach the man. A dog
is standing near the base of
the tree looking upward.
A girl is standing with her
hand raised maybe calling
someone. A child is riding
a tricycle nearby. A ladder
is lying on the ground. Two
firefighters are running toward
the tree. A bus is seen in
the background, that is a park.
Birds are sitting on the tree.
branches, making the scene feel
natural.



Results

Adaptive straight threshold: ±0.0200

Total 11 lines detected.

	Line Number	Slope	Direction	Word Count
0	1	0.0274	downward	4
1	2	-0.0266	upward	4
2	3	0.0162	straight	2
3	4	0.0128	straight	6
4	5	0.0023	straight	3
5	6	-0.0697	upward	3
6	7	-0.0466	upward	2
7	8	-0.0196	straight	3
8	9	-0.0372	upward	3
9	10	0.0157	straight	6

Behavioral Analysis

Line 1 (Downward): Reflects fatigue, discouragement, or low energy.

Line 2 (Upward): Suggests optimism, motivation, and confidence.

Line 3 (Straight): Shows emotional balance, calmness, and logical thinking.

Line 4 (Straight): Shows emotional balance, calmness, and logical thinking.

Line 5 (Straight): Shows emotional balance, calmness, and logical thinking.

Line 6 (Upward): Suggests optimism, motivation, and confidence.

Line 7 (Upward): Suggests optimism, motivation, and confidence.

Future Scope

The English Handwriting Analysis System has significant potential for future enhancements and expansion. One major area of improvement is the integration of **advanced machine learning and deep learning models** to improve the accuracy and adaptability of handwriting analysis. Instead of relying only on rule-based interpretations, the system can be trained on large handwriting datasets to learn complex patterns and provide more refined insights.

Another important future scope is **cloud-based deployment**, which would allow the application to be accessed remotely through the internet. This would enable multiple users to use the system simultaneously and make it suitable for large-scale applications such as educational institutions and research organizations.

The project can also be extended to support **multiple languages and scripts**, enabling handwriting analysis beyond English. This would broaden its usability in multilingual environments. Additionally, integrating **user profile management and report generation** features would allow users to store handwriting samples and track changes over time.

Further improvements may include **mobile application support**, allowing users to capture handwriting images directly using smartphones. The system can also be enhanced by incorporating **forensic handwriting verification**, helping in document authentication and signature analysis.

Overall, the project provides a strong foundation that can be expanded into a more intelligent, scalable, and versatile handwriting analysis platform with applications in education, psychology, recruitment, and forensic studies.

Conclusion

This project successfully demonstrates the development of an automated **English Handwriting Analysis System** based on graphology principles and digital image processing techniques. By transforming traditional, manual handwriting analysis into a technology-driven process, the system provides a consistent, objective, and efficient method for examining handwriting characteristics.

The application analyzes multiple handwriting features such as sentence alignment, letter slant, handwriting size, writing pressure, word and line spacing, t-bar placement, and i-dot characteristics. These features are extracted using image preprocessing and spatial analysis and are interpreted through predefined graphological rules. A user-friendly, web-based interface allows easy interaction, making the system accessible even to users without prior knowledge of handwriting analysis.

The project follows a modular architecture, ensuring clarity, maintainability, and scalability of the codebase. Running in a local execution environment, the system offers fast performance and ease of deployment for academic and experimental use. The results demonstrate that automated handwriting analysis can effectively replicate key aspects of traditional graphology while reducing subjectivity and human bias.

Overall, this project highlights the practical application of intelligent systems in handwriting analysis and serves as a strong foundation for future enhancements involving advanced machine learning techniques, multilingual support, and large-scale deployment.

References

1. Book used "**Graphology As A PsychoDiagnostic Tool By Mohan Bose**".
2. https://www.geeksforgeeks.org/python/libraries-in-python/
3. https://www.youtube.com/playlist?list=PLjOi3GxJDwhETvcIS1B2gYDv8aMjaRS4x
4. https://pyimagesearch.com/2020/08/24/ocr-handwriting-recognition-with-opencv-keras-and-tensorflow/?utm_source=chatgpt.com
5. https://www.geeksforgeeks.org/machine-learning/ai-ml-and-data-science-tutorial-learn-ai-ml-and-data-science/