import cv2

import numpy as np

import matplotlib.pyplot as plt

def preprocess\_image(image\_path):

"""Loads and preprocesses the handwriting image."""

img = cv2.imread(image\_path, cv2.IMREAD\_GRAYSCALE)

if img is None:

raise ValueError(f"Image not found or invalid path: {image\_path}")

img = cv2.resize(img, (500, 500)) # Normalize size

\_, thresh = cv2.threshold(img, 128, 255, cv2.THRESH\_BINARY\_INV)

return img, thresh

def extract\_features(thresh):

"""Extracts handwriting features: slant, size, spacing, and pressure."""

# Feature 1: Slant (using contours to calculate the angle of the text)

contours, \_ = cv2.findContours(thresh, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

# Filter out very small contours that are unlikely to represent meaningful parts of the handwriting

contours = [cnt for cnt in contours if cv2.contourArea(cnt) > 100]

if contours:

# Find the rotated bounding box for the largest contour

rect = cv2.minAreaRect(contours[0])

angle = rect[2] # The angle of the bounding box

if angle < -45:

angle = 90 + angle

avg\_slant = angle

else:

avg\_slant = 0 # No contours detected

# Feature 2: Average Letter Size

letter\_sizes = [cv2.boundingRect(cnt)[3] for cnt in contours if cv2.boundingRect(cnt)[3] > 5]

avg\_size = np.mean(letter\_sizes) if letter\_sizes else 0

# Feature 3: Spacing Between Words

bounding\_boxes = [cv2.boundingRect(cnt) for cnt in contours]

bounding\_boxes.sort(key=lambda x: x[0])

spacings = [bounding\_boxes[i+1][0] - (bounding\_boxes[i][0] + bounding\_boxes[i][2]) for i in range(len(bounding\_boxes)-1)]

avg\_spacing = np.mean(spacings) if spacings else 0

# Feature 4: Pressure (Pixel Density)

black\_pixels = np.sum(thresh == 255)

white\_pixels = np.sum(thresh == 0)

pressure = black\_pixels / (white\_pixels + black\_pixels)

return avg\_slant, avg\_size, avg\_spacing, pressure

def predict\_personality(avg\_slant, avg\_size, avg\_spacing, pressure):

"""Predicts personality traits based on handwriting features."""

personality\_traits = []

# Slant Analysis (Refined with Ambivert)

if avg\_slant > 5:

personality\_traits.append("Extroverted & Expressive 🎭")

elif avg\_slant < -5:

personality\_traits.append("Introverted & Reserved 🤫")

else:

personality\_traits.append("Ambivert & Balanced 🤔")

# Letter Size Analysis

if avg\_size > 40:

personality\_traits.append("Confident & Outgoing 😎")

elif avg\_size < 20:

personality\_traits.append("Detail-Oriented & Focused 🔬")

else:

personality\_traits.append("Moderate & Adaptable 😊")

# Spacing Analysis

if avg\_spacing > 15:

personality\_traits.append("Independent & Loves Freedom 🌍")

elif avg\_spacing < 5:

personality\_traits.append("Social & Likes Crowds 👫")

else:

personality\_traits.append("Organized & Well-Structured 📏")

# Pressure Analysis

if pressure > 0.6:

personality\_traits.append("Emotional & Passionate ❤")

elif pressure < 0.3:

personality\_traits.append("Easy-going & Relaxed 🌿")

else:

personality\_traits.append("Emotionally Stable 😊")

return personality\_traits

def visualize\_results(original\_img, thresh\_img, slant, size, spacing, pressure, personality\_traits):

"""Displays the original and processed images along with the extracted features and personality traits."""

# Show the processed images and handwriting analysis

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)

plt.title("Original Handwriting Image")

plt.imshow(original\_img, cmap='gray')

plt.axis("off")

plt.subplot(1, 2, 2)

plt.title("Processed Handwriting Image")

plt.imshow(thresh\_img, cmap='gray')

plt.axis("off")

plt.tight\_layout()

plt.show()

# Print the extracted features

print(f"Extracted Handwriting Features:")

print(f"- Average Slant: {slant:.2f}")

print(f"- Average Letter Size: {size:.2f}")

print(f"- Average Spacing: {spacing:.2f}")

print(f"- Pressure Ratio: {pressure:.2f}")

# Print predicted personality traits

print("\nPredicted Personality Traits:")

for trait in personality\_traits:

print(f"- {trait}")

# Main Execution Flow

image\_path = r"C:\Users\potnu\Downloads\hand.jpg" # Replace with actual image path

try:

# Load and preprocess the image

original\_img, thresh\_img = preprocess\_image(image\_path)

# Extract handwriting features

slant, size, spacing, pressure = extract\_features(thresh\_img)

# Predict Personality Traits

personality\_traits = predict\_personality(slant, size, spacing, pressure)

# Visualize the results

visualize\_results(original\_img, thresh\_img, slant, size, spacing, pressure, personality\_traits)

except ValueError as e:

print(e)