Automated Document Aligner (Midterm Project)

Goal: To automatically process a distorted image of a document and output a perfectly rectified, frontal view of the page, demonstrating mastery of classical computer vision techniques covered in class.

Material Covered: This project draws primarily from **Image Processing (Class 4)**, **Features (Classes 5-6)**, and **Geometric / Planar Transforms (Classes 7-8)**.

Project Overview

Build a computer vision pipeline to take an image of a document from the provided synthetic dataset and transform it into a flat, frontal perspective. The input images show a letter-sized document (11”x8.5”) at various angles and with simulated real-world deformations. Some of the samples may pose a difficult challenge for a simple method!

Your final submission will be a single Python script that processes all images in a given input folder and saves the rectified outputs to a new folder.

The dataset is available to download from this link:
https://drive.google.com/file/d/1pAuTSutQdl25-Ifzs5Zw5Q-MCvcPGTGu/view?usp=sharing

Key Technical Steps

Your automatic system must implement the following steps:

1. **Preprocessing and Binarization (Class 4):**
    - Load the input image.
    - Apply appropriate **filtering (e.g., Gaussian, Bilateral)** to reduce noise while preserving edges.
    - Use **binarization techniques** (e.g., **Otsu's method**, **Adaptive Thresholding**, or a custom histogram-based approach) to segment the relatively bright document from the background. This will result in a clear binary image of the document contour.
2. **Feature and Contour Extraction (Classes 5-6):**
    - Find the dominant **edges** of the document, potentially using an edge detector like **Canny**.
    - Extract the **contour** corresponding to the perimeter of the document.
    - *Alternative:* Use the **Hough Transform** to detect the four most prominent lines that form the document boundaries.
3. **Corner Detection / Localization (Classes 5-6):**
    - From the extracted contour, you must determine the precise sub-pixel coordinates of the **four corners** of the document.

- *Hint 1:* Simplify the complex contour to just **four vertices** using a curve approximation algorithm (like Ramer–Douglas–Peucker which exists in OpenCV).
- *Hint 2:* If you used the Hough Transform, find the **intersections** of the four detected lines.
- *Note:* You must ensure the four corners are ordered consistently (e.g., clockwise starting from the top-left) for the next step.
- You may want to work in a lower resolution for some steps, and then refine them in a higher resolution if needed.

4. **Geometric Rectification (Classes 7-8):**
   - Define the **source points** (the four detected corners in the input image) and the **destination points** (the corners of the desired, rectified output rectangle).
   - Compute the **Homography matrix** (*H*) that maps the source points to the destination points. This will involve solving a system of linear equations using **Least Squares optimization**. Consult the OpenCV documentation or AI for finding the right functions to use.
   - Apply the perspective transform defined by *H* to the original color image to generate the rectified output.

System Requirements

Requirement	Details
Programming Language	Python 3
Libraries	**OpenCV** (for image processing/transforms), **NumPy** (for matrix and other math operations)
Execution	The system must be executable from the command line: `python3 rectify.py <folder_with_inputs>`
Automation	The system must be **automatic**. It may contain internal parameters, but these parameters **cannot be manually**

	changed once the script is executed. The system must process all images using the same, single set of parameters.
Input/Output	For every input file (`inputN.jpg` / `.png`), an output file (`outputN.jpg`) must be created in a new `output` folder in the current working directory.
Output Size	All rectified output images must have a resolution of 550x425 **pixels** (50 DPI for a 11x8.5” paper).

Deliverables

Submit a compressed file (e.g., `.zip` or `.tar.gz`) containing:

1. **`rectify.py`:** Your single, fully documented Python script.
2. **`README.md` (or `.txt`):** A document detailing:
    - A high-level description of your final pipeline (what sequence of operations are performed).
    - The specific methods chosen for: Binarization, Edge/Contour Finding, and Corner Localization.
    - Any key parameter values you settled on (e.g., Canny thresholds, binarization constants) and how you selected them.
3. **A sub-folder named `output_samples/`** containing the rectified images for a small, representative sample of the dataset (e.g., 5-10 images).

Academic Integrity and Tool Usage

You are **encouraged to use AI tools** (like Claude, Cursor, ChatGPT or Gemini) to assist with:

- Understanding API usage and documentation (e.g., how to use specific OpenCV functions).
- Debugging and exploring different implementation strategies.
- Writing the documentation.

However, all submitted code and the theoretical explanation in your `README` must be your own work. **Do not** submit code directly generated by an AI tool without fully understanding and modifying it. Your understanding of the underlying mathematical concepts is the primary focus of this project.

This assignment is individual. You may collaborate and consult with your peers, but the final submission is yours. This means you should make sure you submit work that you personally vetted and are proud of.

Grading Philosophy

Your assignment will be graded based on the overall solution, not the accuracy of your results. We will examine the results visually but the grade will reflect your choice of implementation and completeness of the approach. Focus on the task at hand and using methods we have seen or similar ones. As always, you are welcome to use your imagination!

Good luck!