Sheets Counter Documentation

1. Overall Approach

The Sheets Counter application is designed to process images of stacked sheets and estimate the number of sheets in the stack. The approach involves the following steps:

1. **Image Upload**: Users upload an image of a stack of sheets.
2. **Image Processing**: The uploaded image is processed using computer vision techniques.
3. **Edge Detection**: We apply edge detection to identify prominent lines in the image.
4. **Line Detection**: Using the Hough Line Transform, we detect lines within the image.
5. **Sheet Counting**: The number of detected lines is used as a proxy to estimate the number of sheets.
6. **Display Results**: The application displays the number of estimated sheets to the user.

This approach leverages the ability of edge detection and line detection algorithms to differentiate between sheets based on their edges.

---

2. Frameworks/Libraries

- **Streamlit**: A framework for building interactive web applications. It provides an intuitive interface for uploading images and displaying results.
- **OpenCV**: A library for computer vision tasks. It is used for image processing, including edge detection and line detection.
- **NumPy**: A library for numerical operations in Python. It is used to handle image arrays and perform various transformations.
- **Pillow**: A library for image processing. It is used to open and handle images uploaded by the user.

---

3. Challenges and Solutions

Unfortunately, accurately counting the number of sheets in the provided image is highly challenging due to several factors:

1. **Overlapping Sheets:** The sheets are stacked closely, making it difficult to distinguish individual sheets, especially at the edges.
2. **Irregular Thickness:** The thickness of the sheets varies, making it hard to establish a consistent pattern for counting.
3. **Lighting and Shadows:** The image has uneven lighting and shadows, which further obscures the sheet boundaries.
4. **Image Resolution:** The image resolution is relatively low, limiting the detail available for analysis.

- **Challenge: Accurate Sheet Counting**
  Solution: Initially, the number of detected lines was used as a proxy for sheet counting. The parameters for edge detection and line detection (e.g., thresholds, line length) were fine-tuned to improve accuracy.
- **Challenge: Parameter Tuning**
  Solution: The parameters for edge detection (cv2.Canny()) and line detection (cv2.HoughLinesP()) required adjustment to handle varying image conditions. Extensive testing was performed to find optimal values for these parameters.

---

4. Future Scope

- **Enhanced Sheet Counting Algorithm**: Develop more sophisticated algorithms using machine learning or deep learning to improve sheet counting accuracy, especially in cases with overlapping or distorted sheets.
- **User Interface Improvements**: Implement additional features such as interactive image annotation tools or real-time feedback to enhance the user experience.
- **Batch Processing**: Allow users to upload and process multiple images at once, providing a summary of sheet counts for each image.