SHEET STACK COUNTING APPLICATION

INTRODUCTION

Manual counting of sheet stacks in manufacturing plants is an essential yet labor-intensive task. This process involves workers physically counting the number of sheets in each stack, a method that is not only time-consuming but also susceptible to human error. In a high-paced manufacturing environment, these errors can lead to significant inefficiencies, impacting both productivity and operational costs.

To address this issue, there is a pressing need for an automated solution that can accurately count the number of sheets in a stack from images. Automation in this context leverages advancements in computer vision and image processing, enabling precise and rapid analysis of images. By automating the sheet counting process, manufacturing plants can ensure higher accuracy, reduce labor costs, and increase overall productivity.

This project aims to develop a robust application that uses computer vision techniques to count sheets in an image accurately. The application will preprocess the image to enhance visibility, detect edges to outline the sheets, and use line detection algorithms to count the individual sheets. The goal is to create a solution that is not only accurate but also efficient and user-friendly, ensuring minimal latency in processing.

The approach to solving this problem involves multiple steps and trials, each exploring different methodologies to achieve the best possible results. Various image processing techniques, including edge detection, contour detection, and morphological operations, were evaluated. Additionally, machine learning methods were considered, though limited by the availability of labeled data.

In the end, the most effective approach combines several preprocessing techniques with a refined edge and line detection process to accurately identify and count the sheets. This document outlines the overall approach, the frameworks and tools used, the challenges faced and solutions implemented, and potential future enhancements to the application.

TRIALS AND THEIR OUTCOMES

1. Hough Line Transform

I started with the Hough Line Transform, which seemed like a promising approach. The process involved converting the image to grayscale, applying Gaussian blur to smooth out noise, and then using the Canny edge detector to highlight the edges. The Hough Line Transform was supposed to identify lines representing sheet boundaries. Initially, the results looked promising, but I quickly noticed an issue: the number of detected sheets was excessively high. It seemed that the algorithm was picking up more lines than actual sheets, likely due to sensitivity to noise and the presence of extraneous lines in the image. This was the first sign that the approach needed refinement.

2. Morphological Closing

After the Hough Line Transform didn’t quite meet expectations, I decided to explore morphological closing. This approach involved additional image preprocessing, including Gaussian blur and edge detection, followed by morphological operations to enhance structure detection. I was inspired by research on machine vision systems for counting corrugated cardboard, which suggested that morphological operations could improve the detection of lines and structures. While this method did enhance certain features, I encountered a new challenge: the number of detected sheets was surprisingly low. It became clear that excessive noise reduction was causing the system to miss some sheets. The challenge now was to balance between too much and too little noise reduction.

3. Contour Detection

Moving on from morphological closing, I tried contour detection. This method involved converting the image to grayscale, applying Gaussian blur, and using Canny edge detection, followed by contour extraction to locate the sheets. I was hopeful that this approach would provide a more accurate count by focusing on the contours of the sheets. However, as I analyzed the results, I found that the number of detected sheets was still not quite right. The contours often overlapped or were affected by noise, leading to an overestimation of sheets. It became apparent that contour detection alone wasn't sufficient for accurate counting, prompting me to look for alternative methods.

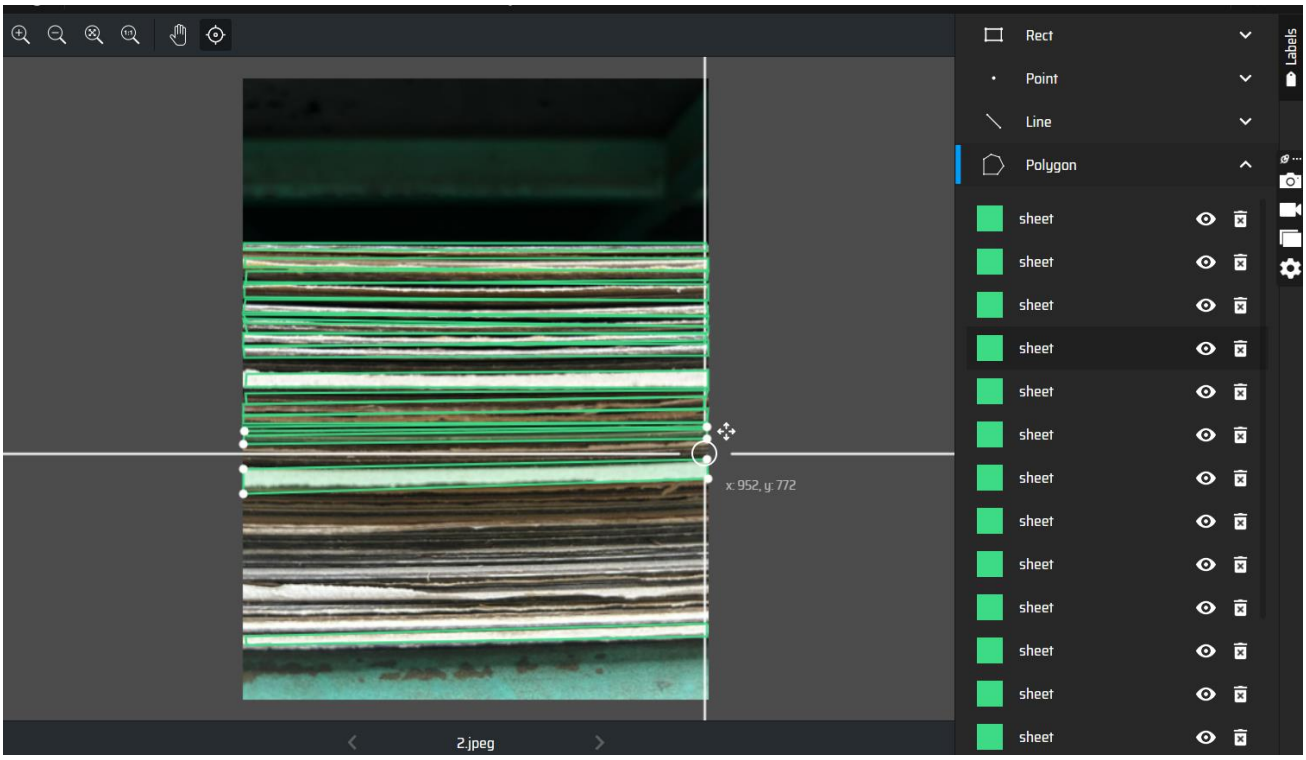
4. Line Midpoint

After facing challenges with the previous methods, I explored the line midpoint approach. This involved preprocessing the image with rotation, grayscale conversion, and contrast stretching, then using edge detection and Hough Line Transform to identify lines. By focusing on the midpoints of these lines, I aimed to refine the count and handle horizontal sheets specifically. This approach showed promise, particularly for horizontal sheets, but it was clear that it had limitations. It only worked well for horizontal sheets and required further fine-tuning to address its limitations. Despite these improvements, it was clear that a more generalized solution was needed.

5. Machine Learning Model

Finally, I turned to machine learning. With limited labeled data, I attempted data augmentation to generate more images for training a model. The idea was to use a deep learning model, potentially MobileNet for efficiency or EfficientNet for better performance, to handle sheet counting. However, the challenge of manually labeling images and limited dataset size became apparent. Although this approach showed potential, the constraints on data and time meant that it wasn't feasible to fully implement during the project. Despite this, the experience highlighted the benefits of machine learning for future improvements, particularly if more time could be available.

In the end, each trial offered valuable insights and helped refine the approach. While some methods were discarded due to various challenges, they contributed to the development of a more accurate and efficient final solution.



Manually annotating each sheet in atleast 50 images to begin training the model

FINAL APPROACH

The final approach incorporates elements from the trials but with refined preprocessing, edge detection, and line classification techniques. The steps include:

1. Convert the image to grayscale.
2. Apply median filtering and histogram equalization for better contrast.
3. Use the Canny edge detector to highlight sheet boundaries.
4. Apply the Hough Line Transform to detect lines.
5. Classify lines as vertical or horizontal, sort, and group them based on proximity.
6. Draw representative lines on the original image and count the sheets.

FRAMEWORKS/LIBRARIES/TOOLS

- OpenCV: Used for image processing tasks such as grayscale conversion, median filtering, histogram equalization, edge detection, and line detection.
- NumPy: Utilized for numerical operations and array manipulations.
- Flask: A lightweight web framework used to create the backend of the web application.
- HTML/CSS/JavaScript: For creating the frontend of the web application, providing a user-friendly interface for uploading images and displaying results.

CHALLENGES AND SOLUTIONS

1. High Number of Detected Sheets:

- Problem: Initial methods like the Hough Line Transform detected too many lines, resulting in an overestimation of sheet count.

- Solution: Improved line detection and classification techniques, and grouping lines based on proximity helped in accurately identifying the sheet boundaries.

2. Low Number of Detected Sheets:

- Problem: The morphological closing approach led to underestimation due to excessive noise reduction.

- Solution: Fine-tuning the morphological operations and connected components analysis improved detection accuracy.

3. Contour Detection Inaccuracy:

- Problem: Contour detection resulted in inaccurate sheet counts due to overlapping contours and noise.

- Solution: Additional preprocessing steps like Gaussian blur and better edge detection parameters enhanced contour accuracy.

4. Horizontal Sheets Limitation:

- Problem: The line midpoint approach only worked for horizontal sheets.

- Solution: Developed a more generalized approach that handles both vertical and horizontal sheets.

5. Limited Labeled Data for ML Model:

- Problem: Training a machine learning model required a large labeled dataset, which was not available.

- Solution: Used data augmentation to generate more images and manually labeled a few samples, though this approach was ultimately set aside due to time constraints.

Overall Approach

The overall approach to developing the automated sheet counting application centers on a systematic and methodical process of image analysis and processing. The goal was to replace a manual, error-prone sheet counting method with an efficient and accurate automated solution. The final approach integrates several stages of image processing to accurately detect and count the sheets in a stack from provided images. Here's a detailed breakdown of the approach:

1. Image Preprocessing

Grayscale Conversion: The first step in preprocessing is converting the image to grayscale. This simplifies the image by reducing it to a single channel, making subsequent processing steps more straightforward and less computationally intensive.

Median Filtering: To reduce noise and smooth the image, median filtering is applied. This technique replaces each pixel's value with the median value of its neighboring pixels, effectively removing random noise while preserving edges.

Histogram Equalization: To enhance contrast and improve the visibility of features in the image, histogram equalization is performed. This technique adjusts the intensity distribution of the image, making the boundaries of sheets more distinguishable.

2. Edge Detection

Canny Edge Detection: Edge detection is crucial for identifying the boundaries of sheets. The Canny edge detector is used to highlight the edges by detecting areas of rapid intensity change. This results in a binary image where edges are marked, which helps in subsequent line detection steps.

3. Line Detection

Hough Line Transform: The Hough Line Transform is employed to detect lines in the edge-detected image. This method is effective for identifying straight lines and is used to detect the boundaries of sheets. The parameters of the Hough Transform are carefully tuned to balance sensitivity and specificity, ensuring that only relevant lines are detected.

4. Line Classification and Grouping

Classification: Detected lines are classified based on their orientation—vertical or horizontal. This classification helps in understanding the arrangement of sheets and ensures that the counting process is accurate.

Sorting and Grouping: After classification, the lines are sorted and grouped based on their proximity. This step addresses the challenge of overlapping or closely spaced lines, which could otherwise lead to inaccurate counting. Lines that are too close to each other are grouped together, and representative lines are selected for counting.

5. Visualization and Counting

Drawing Lines: To visualize the detected lines and verify the results, representative lines are drawn on a copy of the original image. This step helps in visually assessing the accuracy of the detection and grouping process.

Counting Sheets: The final step involves counting the number of sheets based on the grouped lines. The number of sheets is calculated by counting the representative lines and adjusting for any potential errors. The count is derived from the grouped lines, with adjustments made for any overlaps or inaccuracies.

6. Output and Processing

Saving Processed Image: The processed image, with detected lines drawn on it, is saved to a designated directory. This allows for review and verification of the results.

Returning Results: The final count of sheets and the path to the processed image are returned as outputs. This provides a clear and actionable result for users, allowing them to assess the number of sheets in the stack.

Overall, this approach combines traditional image processing techniques with methodical steps to achieve accurate and efficient sheet counting. By carefully preprocessing the image, detecting and classifying lines, and grouping them effectively, the solution addresses the core challenges and provides a reliable automated counting system.

FUTURE SCOPE

Potential Improvements:

1. Enhanced Preprocessing: Implement more advanced preprocessing techniques to reduce noise and improve line detection accuracy.

2. Deep Learning Models: Train a deep learning model with a larger labeled dataset for more robust and generalized sheet counting.

3. User Feedback Loop: Incorporate a feedback mechanism where users can correct the count, and the system learns from these corrections.

4. Real-time Processing: Optimize the application for real-time processing, allowing continuous monitoring of sheet stacks.

5. Multi-angle Support: Extend the application to support images taken from different angles and perspectives.

Additional Features:

1. Batch Processing: Allow users to upload multiple images at once and receive a summary report of sheet counts.

2. Detailed Reports: Generate detailed reports with annotated images, count statistics, and processing times.

3. Mobile App: Develop a mobile application for easier access and on-the-go sheet counting.

CONCLUSION

The automated sheet stack counting application significantly improves the efficiency and The development of the automated sheet counting application successfully transitioned from initial concept to a functional solution that addresses the core challenges of accuracy and efficiency in a manufacturing setting. The final approach integrates a comprehensive image processing pipeline, combining grayscale conversion, median filtering, histogram equalization, Canny edge detection, and Hough Line Transform to accurately detect and count sheets from images. This method effectively reduces the overestimation and noise issues encountered in earlier trials, providing a reliable count of sheets by grouping detected lines and refining the results.

Throughout the project, various methods were tested, each offering valuable insights and contributing to the final solution. The Hough Line Transform and morphological closing methods revealed the importance of balancing noise reduction with accurate detection, while contour detection and line midpoint approaches highlighted the need for a more generalized solution. The final method overcomes these challenges by focusing on line detection and grouping, ensuring both accuracy and efficiency.

Looking ahead, there are several opportunities for enhancing the application further. Incorporating machine learning models like MobileNet or EfficientNet could improve performance and adaptability to different sheet orientations and conditions. Additionally, refining image preprocessing techniques, optimizing for real-time processing, and enhancing the user interface based on feedback could further advance the solution. This project underscores the value of iterative development and adaptation, demonstrating how a methodical approach to problem-solving leads to practical and effective solutions in complex scenarios.