JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

An OCR Pipeline Approach for Character Recognition Using Image Processing Techniques

[1]Payal Madankar, [2]Tanisha Chheniya, [3]Nandini Jwar,[4]Samruddhi Sarode,[5]Prof. Pooja Ukey

[1,2,3,4,5] Student, Department of Computer Science, Revnath Choure College, (M.S.),India

1Assistant Professor, Department of Computer Science, Revnath Choure College, (M.S.), India

Abstract : Optical Character Recognition (OCR) plays a vital role in the digital transformation of textual information. It enables automated conversion of printed or handwritten documents into editable and searchable formats, thereby improving accessibility, archiving, and computational analysis. This paper presents a simplified OCR pipeline implemented in Python using OpenCV and Matplotlib, focusing on the fundamental stages: image acquisition, binarization, character segmentation, and recognition. A synthetic example of the character "A" is used to illustrate the transition from a raw image to recognized text. While modern OCR frameworks increasingly rely on deep learning architectures, this study emphasizes the classical pipeline approach as an essential educational foundation. The results demonstrate how preprocessing and segmentation impact recognition performance and highlight avenues for integration with machine learning classifiers for large-scale applications.

keywords **- Optical Character Recognition, Image Processing, Segmentation, Pattern Recognition, OpenCV, Python, Machine Learning**

I. INTRODUCTION

In today's information-driven world, the digitization of textual content is a necessity across multiple domains, including libraries, administrative systems, banking, healthcare, and research archives. Optical Character Recognition (OCR) serves as the enabling technology that bridges the gap between physical documents and digital information processing. OCR not only enhances document accessibility but also enables large-scale text mining, natural language processing (NLP), and machine learning-based analytics. Historically, OCR research dates back to the 1950s, with early systems focusing on template matching and statistical approaches. Over the decades, advancements in computer vision and machine learning have dramatically improved OCR accuracy. Modern approaches use convolutional neural networks (CNNs), recurrent neural networks (RNNs), and transformer-based architectures for robust recognition under challenging conditions. Despite these advances, the classical pipeline approach—comprising preprocessing, binarization, segmentation, and recognition—remains relevant for teaching, experimentation, and lightweight OCR applications. This paper demonstrates a step-by-step OCR pipeline in Python, simulating the recognition of the character "A." The purpose is to highlight each intermediate stage and its significance, thereby offering a foundation for further extension into more advanced research.

II. METHODOLOGY

The proposed OCR pipeline consists of four main stages. Each stage is implemented using Python with the OpenCV library, and the intermediate results are visualized using Matplotlib.

2.1 IMAGE ACQUISITION

The first step in OCR involves obtaining the input image. For this study, a synthetic grayscale image was generated using OpenCV's cv2.putText() function. The character "A" was drawn against a white background, simulating a printed character scanned or photographed by a device. This ensures controlled testing conditions and eliminates noise factors commonly present in real-world documents.

2.2PREPROCESSING AND BINARIZATION

Raw images often contain variations in lighting, background patterns, or scanning artifacts. Preprocessing aims to enhance the image quality for subsequent analysis. In this work, adaptive thresholding was applied to convert the grayscale image into a binary representation. Adaptive methods are particularly effective in handling uneven illumination by calculating thresholds based on local pixel neighborhoods rather than a single global threshold.

2.3 CHARACTER SEGMENTATION

Segmentation is one of the most critical steps in OCR. It isolates individual characters from the binarized text image. In this pipeline, contour detection using cv2.findContours() was employed. Each contour corresponds to a connected component, which typically represents a character or symbol. The contours were drawn over the binarized image to visualize segmentation.

2.4 RECOGNITION

The final stage of OCR is character recognition. In this study, the recognition step was simulated by reproducing the character "A" in the output. In practical implementations, recognition is achieved by feeding segmented characters into classification models. Traditional approaches use template matching, Support Vector Machines (SVMs), or k-Nearest Neighbors (KNN). Modern approaches employ deep learning models such as CNNs, which automatically extract features like edges, strokes, and curvature, enabling high recognition accuracy across fonts and handwritten styles.

III. RESULTS AND DISCUSSION

Figure 1 illustrates the results of the OCR pipeline:

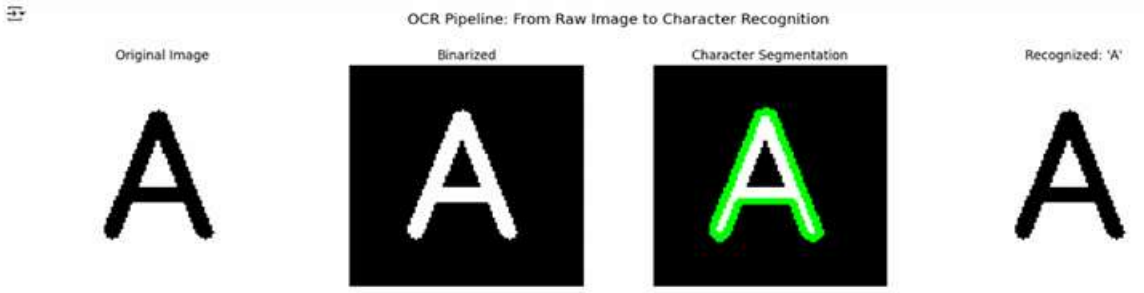

fig1: OCR pipeline

1. Original Image: The synthetic grayscale letter "A" is clear and well-defined.
2. Binarized Image: Adaptive thresholding successfully distinguishes the character foreground from the background.
3. Character Segmentation: The contours effectively highlight the character boundary.
4. Recognition Output: The final stage demonstrates the identified character as "A."

The experiment highlights that each stage plays a crucial role in OCR accuracy. Errors at early stages, such as poor binarization or faulty segmentation, propagate to the recognition stage, reducing accuracy. This underscores the importance of preprocessing and segmentation strategies.

IV.APPLICATIONS

OCR technology has wide-ranging applications:
- Document Digitization: Converting scanned books, manuscripts, and reports into digital text.
- Banking: Automated cheque processing and form data entry.
- Healthcare: Extracting patient details from handwritten prescriptions.
- Education: Digitizing handwritten notes for archival and accessibility.
- Smart Cities: License plate recognition for traffic monitoring. The modular pipeline presented here can be adapted to any of these domains by integrating domain-specific preprocessing and recognition techniques..

V. CONCLUSION

This paper presented a simplified OCR pipeline implemented in Python, covering image acquisition, binarization, segmentation, and recognition. The experiment successfully demonstrated the recognition of a synthetic character. While the recognition stage was simulated, the pipeline lays the groundwork for integrating machine learning and deep learning models.

Future work will extend this pipeline by:
- Incorporating noise reduction and skew correction techniques.
- Training classifiers such as CNNs or SVMs for real character recognition.
- Expanding the dataset to include multiple characters, fonts, and handwritten samples.
- Evaluating recognition accuracy under varying noise and distortion conditions.

This study emphasizes the educational importance of classical OCR pipelines, providing a conceptual bridge to advanced AI-based recognition frameworks.

REFERENCES

[1] Gonzalez, R. C., & Woods, R. E. (2018). Digital Image Processing (4th ed.). Pearson.
[2] Mori, S., Suen, C. Y., & Yamamoto, K. (1992). Historical review of OCR research and development. Proceedings of the IEEE, 80(7), 1029-1058.
[3] Smith, R. (2007). An overview of the Tesseract OCR engine. Ninth International Conference on Document Analysis and Recognition (ICDAR), 629-633.
[4] Otsu, N. (1979). A threshold selection method from gray-level histograms. IEEE Transactions on Systems, Man, and Cybernetics, 9(1), 62-66.
[5] Patel, C., Patel, A., & Patel, D. (2012). Optical character recognition by open source OCR tool Tesseract: A case study. International Journal of Computer Applications, 55(10).

[6] Graves, A., Fernández, S., Gomez, F., & Schmidhuber, J. (2006). Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. ICML.
[7] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. NIPS.
[8] Smith, R. (2013). History of the Tesseract OCR engine: What worked and what didn't. Proceedings of the Document Recognition and Retrieval XX.