Enhanced Image-to-Text Conversion for License Plate Recognition Using OpenCV and Tesseract

Faezah Ayyub [1] Dr.B.Sasi Kumar [2]

[1] M.Tech Student - CSE, Department of Computer Science Engineering, Dr.V.R.K Women's College of Engineering & Technology, Hyderabad, Telangana, India.

[2] Principal & Professor, Department of Computer Science Engineering, Dr.V.R.K College of Engineering & Technology, Hyderabad, Telangana, India.

1 ABSTRACT

This document explains in detail the process of converting images to text. It describes the different procedures required to extract text from image files (such as jpeg or png) and create a separate text file containing the extracted information. The paper addresses the limitations of current image processing applications and aims to improve them through different levels of image processing and filtering. The implementation uses the CV2 OpenCV library with Python for image processing and Tesseract to extract text from processed images. The different levels of processing applied to each image help to achieve improved text results. After processing, the resulting text files are cleaned by removing commas, semicolons, quotes, periods, and other non-standard characters using ASCII filtering, as these characters are not typically found in standard license plates (Palekar et al., 2017).

2 INTRODUCTION

Traffic management in cities remains a constant challenge, especially in India, where unlicensed vehicles are on the rise due to population growth. Frequent traffic congestion often leads citizens to violate traffic rules in order to reach their destinations faster. To address this issue, it is important to develop a system that streamlines the administrative work associated with issuing fines. An efficient and reliable system for automating and monitoring parking barriers is also required. For large commercial enterprises, managing and monitoring only licensed vehicles is a huge resource in terms of time and cost. Therefore, a fast, accurate, and effective system is urgently needed. Capturing license plate images and retrieving vehicle details from a database is a promising solution to streamline the process (Palekar et al., 2017).

Figure 1 Traffic Police taking pictures of vehicle license plates

The captured images need to be processed to extract important information. Techniques such as image segmentation, dilation, erosion, contour finding, and thresholding are used to enhance the images. Image processing is important to extract the required information from the images and connect them to the database (Singh & Bhushan, 2019). The most effective way to convert the image information into a usable format is through

text extraction, which can be done using Tesseract software. Tesseract is an optical character recognition (OCR) tool that converts images into text (Sambana et al., 2023).

2.1 SOFTWARE'S USED

The field of image processing is primarily software-driven, and OCR software such as Tesseract is used to convert images to text. However, Tesseract is prone to errors, especially for images that are not of high quality. To improve accuracy, you should first process the images using OpenCV before inputting them into Tesseract.

2.1.1 OpenCV

OpenCV (Open Source Computer Vision) is a free library suitable for both academic and commercial use, designed for real-time computer vision applications. The functionality of this library includes a wide range of functions, including 2D and 3D toolkits, self-motion estimation, facial recognition, gesture recognition, action understanding, object segmentation, recognition, and tracking (Howse et al., 2016).

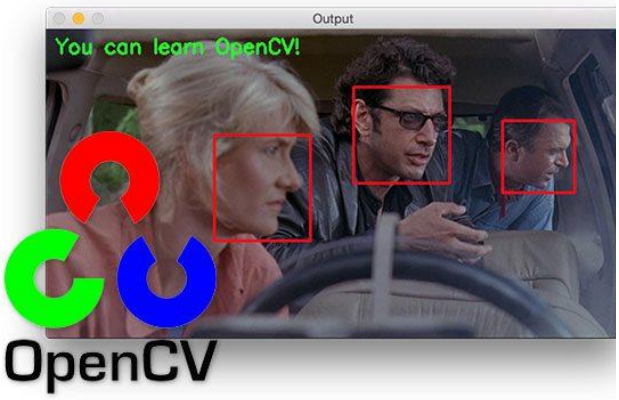

Figure 2 Open CV Real Time working

OpenCV is written primarily in C++, with a powerful C++ interface and a less comprehensive but extensive C interface. This library contains many predefined functions useful for image processing. Given its open-source nature, OpenCV was chosen for this project, allowing the implementation of several image processing techniques such as RGB to grayscale conversion, erosion, and dilation.

2.1.2 Python

Python is a popular high-level interpreted programming language known for its focus on code readability. The syntax allows programmers to express concepts concisely compared to languages like C++ or Java. Python supports multiple programming paradigms, including object-oriented, imperative, and functional programming. It has a dynamic type system and automatic memory management along with a comprehensive standard library. Python's simplicity and efficiency allowed us to create short, clean code snippets for each processing technique and facilitated the development of multi-level processing systems. As a result, Python has proven to be very effective for digital image processing, with simple, easy-to-understand code (S. Walker & Martinez, 2018).

```
#Take the users input
words = raw_input("Enter some text to translate to pig latin: ")
print "You entered: ", words

#Now I need to break apart the words into a list
words = words.split(' ')

#Now words is a list, so I can manipulate each one usinga loop

for i in words:
    if len(i) >= 3: #I only want to translate words greater than 3 characters
        i = i + "%say" % (i[0])
        i = i[1:]
        print i
    else:
        pass
```

Figure 3 Working of Python Language

2.1.3 Tesseract

The Tesseract package includes libtesseract, an OCR engine developed by Ray Smith, and tesseract, a command-line tool. Tesseract supports Unicode (UTF-8), recognizes over 100 languages natively, and can learn new languages. It offers a variety of output formats, including plain text, hOCR (HTML), and PDF. Originally developed as proprietary software by Hewlett-Packard in the late 1980s and early 1990s, Tesseract has since evolved through updates and migrations from C to C++. It is compatible with Linux, Windows, and macOS, but has been tested primarily on Windows and Ubuntu. Previous versions of Tesseract only supported TIFF images with single-column text and lacked layout analysis, resulting in errors with multiple columns or complex text. Starting with version 3.00, Tesseract introduced text formatting, OCR geolocation, and page layout analysis. Version 3.04, released in July 2015, expanded language support to over 100 languages. Tesseract can be

used as a backend for complex OCR tasks, including layout analysis, when combined with a frontend such as OCRopus (Seiter, 1993).

Figure 4 Tesseract using OCR to detect texts

3 METHODOLOGY

3.1 IMAGE ACQUISITION:

Collect image files in formats like JPEG, PNG, etc. that contain text information for extraction.

3.2 PREPROCESSING:

Image Loading: Load image files using the CV2 OpenCV library in Python.

Preprocessing: Apply basic preprocessing steps like resizing or contrast adjustment to improve image quality.

3.3 IMAGE PROCESSING:

Grayscale Conversion: Convert images from RGB to grayscale to simplify processing.

Noise Reduction: Reduce noise using image processing techniques like Gaussian Blur.

Binarization: Apply thresholding to convert grayscale images to binary images where text can be clearly distinguished from background.

Formal Operations: Enhance text by performing stretching and erosion to remove small artifacts.

3.4 TEXT EXTRACTION:

OCR Application: Input processed binary images into Tesseract OCR software to extract text.

Text Recognition: Convert processed images to text format using Tesseract functions.

3.5 POST-PROCESSING:

Text Cleanup: Clean the extracted text by removing non-standard characters such as commas, semicolons, quotation marks, and periods using ASCII filtering.

Text Formatting: Format the cleaned text as needed to ensure that the final output is consistent and readable.

3.6 OUTPUT GENERATION:

Create Text File: Save the cleaned and formatted text to a separate text file.

Validation: Validate the extracted text to ensure accuracy and completeness.

3.7 EVALUATION:

Accuracy Evaluation: Evaluate the accuracy of the text extraction by comparing it to manually verified text.

Performance Metrics: Measure the performance of the image processing and text extraction steps to identify areas for improvement.

This methodology provides a comprehensive approach to converting images into text by addressing image quality issues and improving the text extraction process.

4 CONCLUSION

In conclusion, the proposed methodology effectively addresses the problem of converting images to text. By leveraging advanced image processing techniques using the CV2 OpenCV library and the Tesseract OCR application for text extraction, the process significantly improves the accuracy and reliability of text conversion. Preprocessing steps including grayscale conversion, noise reduction, and morphological operations ensure that the images are optimized for OCR, resulting in more accurate text extraction. Postprocessing measures such as ASCII filtering further enhance the extracted text by removing non-standard characters, ensuring that the final output meets the standards required for readability and conformance (2022) (Larsen & Becker, 2021).

The iterative approach of the methodology, combining image enhancement and sophisticated text recognition, provides a powerful solution to the limitations of current image processing applications. By generating clean and accurate text files from image inputs, the approach not only improves the efficiency of text extraction but also addresses real-world problems in applications such as license plate recognition and automated document processing. Future work may include further improving the process, exploring additional image processing techniques, and extending the capabilities of the system to handle a wider range of text and image types (Larsen & Becker, 2021).

5 REFERENCE

(2022) Optimization and machine learning [Preprint]. doi:10.1002/9781119902881.

Howse, J., Joshi, P. and Beyeler, M. (2016) OpenCV: Computer vision projects with python: Get savvy with opencv and actualize Cool Computer Vision Applications: A course in three modules. Birmingham, UK: Packt Publishing.

Larsen, K.R. and Becker, D.S. (2021) 'Why use automated machine learning?', Automated Machine Learning for Business, pp. 1–22. doi:10.1093/oso/9780190941659.003.0001.

Palekar, R.R. et al. (2017) 'Real time license plate detection using opencv and Tesseract', 2017 International Conference on Communication and Signal Processing (ICCSP) [Preprint]. doi:10.1109/iccsp.2017.8286778.

S. Walker, J. and Martinez, A. (2018) Python. Chicago: Jonathan Wee.

Sambana, H. et al. (2023) 'License plate recognition using a sequential model and opencv', 2023 7th International Conference on Computing Methodologies and Communication (ICCMC) [Preprint]. doi:10.1109/iccmc56507.2023.10083767.

Seiter, C. (1993) OCR: The recognition you deserve. Estados Unidos: PC World Communications.

Singh, J. and Bhushan, B. (2019) 'Real time indian license plate detection using deep neural networks and Optical Character Recognition Using LSTM TESSERACT', 2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS) [Preprint]. doi:10.1109/icccis48478.2019.8974469