Comparative Analysis of Text Extraction from Color Images using Tesseract and OpenCV

Revathi AS
Electronics and Telecommunication
D J Sanghvi College of Engineering
Mumbai, India
revathi.as@djsce.ac.in

Nishi A Modi
Electronics and Telecommunication
D J Sanghvi College of Engineering
Mumbai, India
nishimodi99@gmail.com

***Abstract*— Image-based Text Extraction has a growing requirement in today's generation. Students, doctors, and engineers generate a lot of images every day. It is very important to extract text from these images in a simple yet effective manner. We can obtain useful information by testing these images. We aim is to summarize the visual information and retrieve its content. The Optical Recognition System involves several algorithms that fulfill this purpose. Text Extraction involves a lot of processes from text detection, localization, segmentation and, text recognition. Tesseract is the most optimized OCR Engine build by HP Labs and owned by Google. Text Detection involves the recognition of text from desired input images. Text Localization involves identifying the position of text on the images. Tesseract works pretty well on the light-colored background but unable to recognize text on darker shades. We have tried to apply various image processing techniques. This method will allow us to recognize text from most types of background. We propose to provide methods for easy text extraction. Track bar allows the user to adjust various parameters to extract a required text from an Image. This method is gaining huge importance in years to come. For Automation, we can use a set of image processing techniques such as edge detection, filtering and, blurring for better results. A series of these steps will enable us to extract text from images efficiently. This experiment compares the optimized result by two methods for efficient Text Extraction.**

Keyword – Text Extraction, Tesseract, Image Processing Threshold

I. INTRODUCTION

Text Extraction is gaining importance in today's fast-growing world. It is used by students, doctors, engineers and, many others use in day to day life. It reduces the chances of mistakes made by humans and optimizes the process. OpenCV is a library consisting of programming functions for real-time Computer Vision. It helps to process an image and apply various functions like Image Resizing, object detection, etc. OCR is used for converting images into machine-encoded language. It was used for the conversion of typed, handwritten or printed text. It includes a lot of sub-processes: Preprocessing, Text detection, Text recognition and Post processing.

Tesseract is an open-source recognition engine either used directly or using an API to extract text from images. Although supported by many languages, it does not have a provision of a built-in GUI. It is used in conjunction along with the external detector or existing layout analysis within a document.

Extracting and recognizing text in images has become a potential application in many fields like robotics, intelligent transport systems, etc [1]. The main modules of these types of applications are object localization, object extraction and, text recognition [1]. The textual part in an image is determined by text detection. Text recognition is carried out by OCR. It is used in the following manner:

- Handwriting Recognition Technique: A lot of advancement is going on in this field. It takes the input as a mathematical equation and generates step wise solution of each process.
- Digitalization: It is preferred to reduce manual work. It is used to extract data from business documents, receipts, passports, etc. It can be stored safely and used for future references.
- Book Scanning: It can be used to convert raw images into digital text format. This method is very effective as such copies can be stored and shared with people worldwide.
- Self-Driving Cars: OCR has a wide range of applications in building these models. Self-driving cars are highly dependent on OCR for signs as well as other vehicles on road.

Several techniques have been developed for extracting the text from an image. The proposed methods were based on morphological operators, wavelet transform, artificial neural network, skeletonization operation, edge detection algorithm, histogram technique, etc. [2]

II. LITERATURE SURVEY

Image processing is a software-focused domain, it has a wide range of applications. Tesseract is an OCR Software used for the conversion of an image into text. If the quality of an image is not pristine, this software is prone to errors. So the image needs processing using OpenCV and then processed using Tesseract for better results. In this project, we use an open source, programming language and OCR Engine.

978-93-80544-43-4/21/$31.00 ©2021 IEEE

OpenCV (Open Source Computer Vision) is available for both academic and commercial use. It aims at real-time computer vision and has a library of programming. It has wide area applications including 2D and 3D features and toolkits. Emotion estimation, Facial recognition system, Gesture recognition, Motion understanding, Object identification Segmentation and, recognition and Motion tracking. [3] OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. OpenCV contains libraries of pre-defined functions helpful in image processing.

It is widely used as a platform for implementation and testing. OpenCV libraries can be used to implement processes such as dilation, erosion, RGB to Gray conversion and many more. Python is a high-level, general process and a dynamic programming language that is widely used and interpreted.. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library. This language enabled us to short code snippets that applied to all processing techniques. This language allowed us to develop a mechanism for multi-level processing. This method is very useful for various kinds of images along with the simplicity in writing and executing our code.

Tesseract is a high-end package that contains an OCR engine called libtesseract and a command line program called a tesseract. The lead developer of this package is Ray Smith. It is trained in such a manner that it can recognize other languages as well. It supports several output formats such as plain-text, HTML, etc.[4] It also has Unicode support which recognizes more than a hundred languages. Tesseract engine was developed at Hewlett Packard lab in Bristol, England as proprietary software. It was developed between 1985 and 1994. It was later ported to Windows in 1996. A major part of the code was written in C followed by the other changes in a C++. The whole code was created in such a manner that it could compile in C++ compiler.

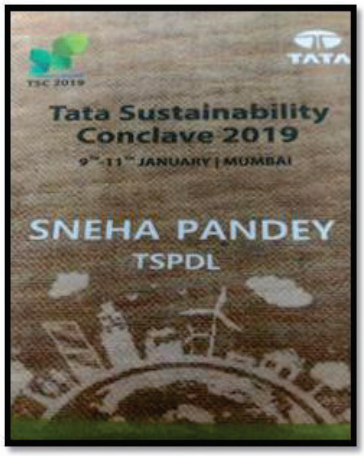

Fig. 1. Original Image

Today, Tesseract is available in all operating systems such as Linux, Windows, and Mac OS X. Few limited resources restrict its usage to Windows and Ubuntu. [5] It does not provide us flexibility in accepting input. It only accepts TIFF images in a one-column format. The early versions of this package do not include layout analysis. If we provide multi-columned text with various backgrounds or images produces a garbage output. This is a major problem when we have to extract text from color backgrounds. This method is usually used for number plate recognition. It has improved a lot in this area. The first version of Tesseract supported only the English language but the later versions were compatible with 39 languages. It can be trained to work for different languages as well. It is highly proficient for working in the back end but it can be used at the front end such as OCRopus[6] Figure 1 shows the original image used and Figure 2 depicts the output for image without any processing.

In this project, OCR is used at the final stage after the image is processed to provide the desired output for various kinds of background and gaining the optimum results.

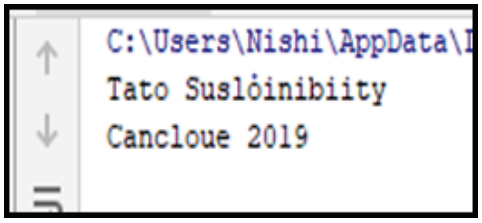

Fig. 2. Output of the image without processing

III. Proposed Solution

We have analyzed a lot of parameters and proposed these three methods for effective Text Extraction from almost all types of backgrounds.

A. Using Manual Trackbar.

The Track bar is used to apply the value for changing the image threshold in runtime. Creating a track bar is creating a progress bar for an image and adjusting various parameters. This track bar has six parameters: LH (Lower Hue), LS (Lower Saturation), LV (Lower value), UH (Upper Hue), US (Upper Saturation) and UV (Upper Value).Different parameters are adjusted as per the background image. If the background color is having a lighter shade then the values such as lower value and lower hue are changed. If the background color has a darker shade then the higher values are changed. This procedure helps us in creating a mask of a particular image and then extracting text from that image. This works in a precise manner for almost all types of background images. It improves the efficiency of Tesseract and thus allows us to extract text with ease and clarity.

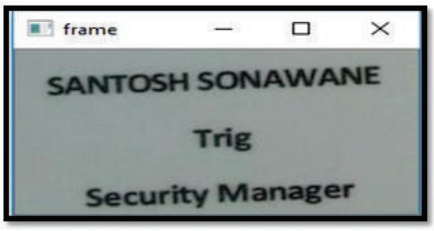

Fig. 3. Original Image in light background

This method is applicable for darker background which provides great accuracy s compared to those images which are not processed.

Fig. 4. Image after changing values of track bar

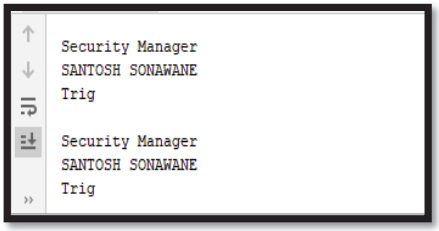

Fig. 5. Output of the masked image

Fig. 6. Original Image in darker background

We have taken an original image in lighter (Figure 3) and darker (Figure 6) background. We apply mask on both the images (Figure 4 for light background and Figure 7 for dark background). The output is displayed on the terminal for both types of background (Figure 5 for light background and Figure 8 for dark background).

Fig. 7. Image after changing the value of track bar

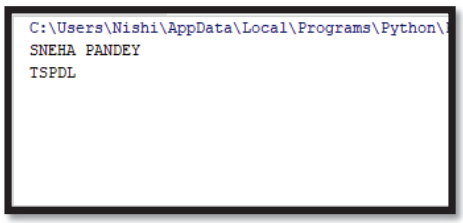

Fig. 8. Result for the darker background

This method is very effective for both dark as well as light shades. However, this method can be tedious as we need to change parameters for individual images so we tried to implement an autonomous mode for this method.

B. Track bar in Autonomous mode

In this method, we take input from the user confirming the shade of the background color. Once provided with the color, it changes the parameters autonomously until it produces the desired output. This method has more probability of producing errors as compared to the manual method. Using Figure 6 as the input image, we implement a code to calculate the value of the parameters and find the result in the terminal for dark background. The background matters a lot in this method. However, this code is available for light and dark shades. It can be customized for more diverse range of background shades,

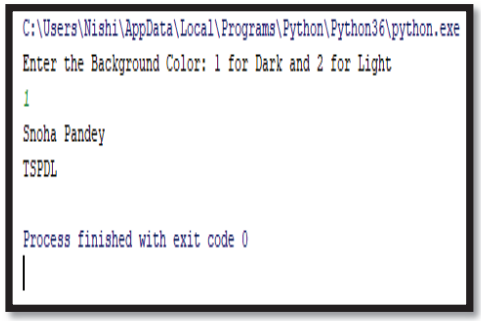

Fig. 9. Autonomous track bar for dark background

Using Figure 3 as the input image, we implement a code to calculate the value of the parameters and find the result in the terminal for light background.

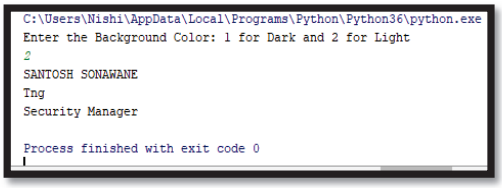

Fig. 10. Autonomous trackbar for light background

This method can be used to prevent adjusting parameters by the manual method. Results produced by this method is quite accurate but not as accurate as compared to the manual method.

Tesseract might produce different results when applied for the same image containing a large number of words. Thus, we need to run the code several number of times which might be a tedious task in this method.

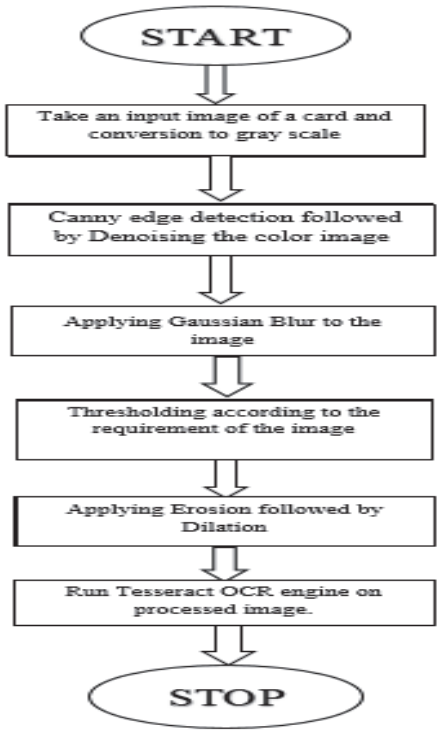

Fig. 11. Flowchart for Text Extraction

C. Using Image Processing

One of the best methods that provide the required results using Image processing is explained using a flowchart.[7] It includes a series of steps which provides the best result for images of almost all background and text of all colors. It includes the following steps:

1) Canny Edge Detection: It is used to detect edges in an image. It requires a multistage algorithm and accepts gray scale image s input. It accepts the grayscale image along with the edge and two threshold values.

2) Denoising: It refers to the reconstruction of the signal using the noisy image. It removes the unwanted noise and allows us to analyze the image in a better format. (Figure 12, 13 and 18)

3) Gaussian Blur: This method uses a Gaussian kernel. It is a low pass filter which in removing high-frequency components. This method accepts the following parameters: source and destination point, size of kernel and, kernel standard deviation. (Figure 14 and 19)

4) Erosion: It is used for removing small white noises in an image. It also allows us to separate two connected images. It is a morphological operation on an image. It shrinks the image while removing noise. (Figure 16)

5) Dilation: In case of the removal of noise, erosion is followed by dilation. It helps in increasing the object area. It is widely used to connect two detached objects in an image.(Figure 17)

We apply these steps on both darker background as well as light background and compare the accuracy in both cases. Figure7 to demonstrate image processing for accurate text extraction. Gray scale conversion is done on an image followed by edge detection for extracting particular text from images.

Fig. 12. Gray scale conversion of input image

De noising of the colored image is displayed and used for Gaussian blurring of the image and other morphological operations.

Fig. 13. Denoising of the previous image

Fig. 14. Gaussian blur on the previous image

Fig. 15. hresholding according to the requirement of the image

Fig. 16. Erosion on the threshold image

Fig. 17. Dilation of the eroded image

A similar process is applied for light background and the results obtained are compared with the other two methods. Image processing is usually required for darker shades as compared to lighter shades.[8]

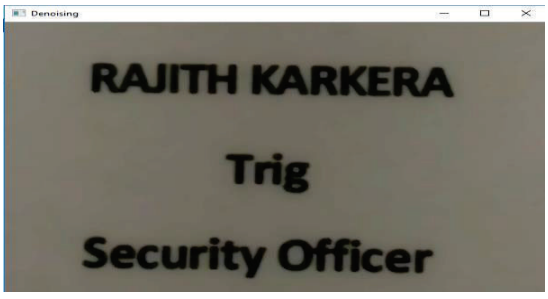

Fig. 18. Denoising on light background images

For light shades, rigorous image processing is not required. If the output of Gaussian blur is applied to tesseract, it provides the optimum result with great accuracy. It can also be used for the processing of encrypted images. [9]

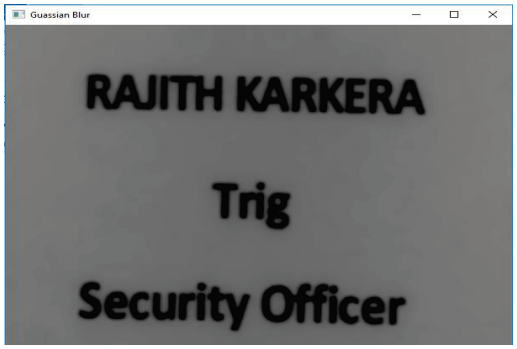

Fig. 19. Gaussian Blur on the previous image

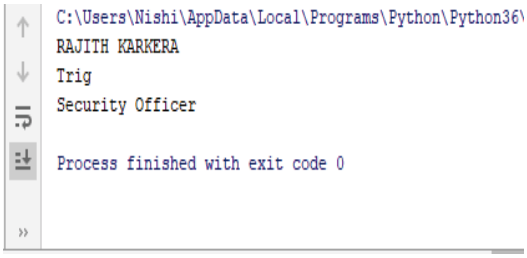

Fig. 20. Output of processed image

IV. Result

We have applied three different methods for improving the text extraction process using Tesseract. After applying these methods on 25 different images, we have tried to extract almost 180 characters from these images. The accuracy percentage is depicted in the table below:

TABLE I. Result of three proposed solution

	Without processing	**Trackbar in Manual mode**	**Trackbar in autonomous mode**	**Image processing**
Total characters	180	180	180	180
Extracted characters	104	174	158	165
Accuracy	57.78%	96.67%	87.78%	91.67%

Hence, it can be concluded that if we apply Tesseract on a processed image, it provides far better results as compared to the unprocessed images. Hence, this method should be implemented for better results. Track bar in manual mode provides the highest accuracy followed by image processing and track bar in autonomous mode.

V. Conclusion

Image processing plays a crucial role in extracting information from any image. The comparative analysis of the same algorithm on similar as well as different images helped us to understand the variation in output of images with the darker background as compared to a lighter background. The output of various images has been subjected to OCR and the best output has been considered for obtaining results.

Some errors are possible due to the orientation of the image, variation in the background color and, shakiness of the image. We can also improve the results by using the Neuro-fuzzy network. Since we have processed the images, it is not essential for our project. This method will help us in significantly reducing errors and help us in predicting the accuracy for further cases.

VI. Future Scope

This application is not extremely useful for extracting employee information or for making a database but also for wide range of applications. It can be used for number plate extraction [10] [11] , newspaper industries, book store publications, web applications and many more. This project provides the best method for successful text extraction and compares the result for future analysis.

Convolutional Neural networks can be used in addition to the above possible methods to imply in effective text extraction. This method can be further improved using Deep learning models. We can also create an Android App to support effective text extraction in real time. This allows individuals to use this method in everyday life and work on various methods to improve its accuracy.

References

[1] R. Chandrasekaran and RM. Chandrasekaran , "Morphology based Text Extraction in Images", in proceeding of IJCST Vol 2, Issue 4, ISSN:0976-8491(Online)| ISSN:2229- 4333(Print) Oct-Dec. 2011.

[2] Agaian SS et al (2001) Transform-based image enhancement algorithms with performance measure. IEEE Trans Image Process 10(3):367–382

[3] Automated Text Extraction from Images using OCR System, Published 2019, Computer Science, 2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)

[4] Bradski, Gary; Kaehler, Adrian (2008). Learning OpenCV: Computer vision with the OpenCV library. O'Reilly Media.

[5] "The RedMonk Programming Language Rankingsz: June 2015 – tecosystems". Redmonk.com. 1 July 2015. Retrieved 10 September 2015. [13] Kay, Anth

[6] Vincent, Luc (August 2006). "Announcing Tesseract OCR". Archived from the original on October 26, 2006.

[7] Text detection and removal from image using inpainting with smoothing; Priyanka Deelip Wagh, D.R. Patil

[8] Text extraction in document images: highlight on using corner points; Nicolas Ragot, Vikas Yadav

[9] An image encryption approach using particle swarm optimization and chaotic map Musheer Ahmad, Mohammad Zaiyan Alam, Zeya Umayya, Sarah Khan, Faiyaz Ahmad Bharati Vidyapeeth's Institute of Computer Applications and Management 2018

[10] Multiple license plate detection for complex background; Ching-Tang Hsieh, Yu-Shan Juan, Kuo-Ming Hung.

[11] Number plate recognition through image using morphological algorithm. In 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), pages 3157-3160, March 2016.