# Real Time License Plate Detection Using OpenCV and Tesseract

Rahul R. Palekar, Sushant U. Parab and Dhrumil P. Parikh, *Member, IEEE,* Prof. Vijaya N. Kamble

***Abstract*— This paper presents the implementation of image to text conversion. The paper describes various steps required to extract text from any image file (jpeg/png) and create a separate text file consisting of information extracted from image file. It considers the shortcomings of various image processing applications available and works on overcoming them by employing variable level of image processing and filtration. The CV2 OpenCV library using Python language is used for image processing and Tessaract is used for text extraction from the processed image. The variable level of image processing ensures that different images get different levels of treatment in order to produce optimized text results. After the image processing step is employed the output text file are formatted by filtering out commas, semicolons, apostrophes, colons and other such characters using ASCII filtering as these characters are not part of any standard license plate.**

***Index Terms*—Gaussian blur, License-Plate, OpenCV, Tesseract.**

## I. Introduction

Traffic regulation in urban regions is an ever present and persistent problem. With the increasing population in India, the number of unlicensed cars has increased. The frequent traffic jams have led to many citizens breaking the traffic rules to get to their destinations faster. In such a difficult scenario, it is imperative to introduce a system to simplify the paperwork involved in handing out fines. It is also necessary to construct a system for parking barrier automation and society surveillance which is fast, robust and reliable. For huge commercial enterprises the allowance of only authorized vehicles and their surveillance in the premises sometimes requires large amount of resources in terms of money as well as time. Hence a quick, error-free and efficient system is in urgent need. Thus the most efficient and fastest way to achieve these goal would be capturing the image of the license plate and using that image to extract details of the vehicle from the available database which will assist in speeding up of all the process.

Rahul R. Palekar, Sushant U. Parab and Dhrumil P. Parikh are with the Sardar Patel Institute of Technology, Mumbai, India, 40055 (phone: 99200 20691; e-mail: rahulpalekar401@gmail.com,sushant.sp11@gmail.com, tauras108dhrumil@gmail.com, vijaya_kamble@spit.ac.in ).

The captured image needs to be processed in order to extract important information from the image. Different image processing techniques of image segmentation and image enhancement such as dilation, erosion, contour finding and thresholding are used. Hence image processing becomes an essential part of this system where the required information from the image can be obtained and thus linked to the database. The best and most suitable form of information from the image would be in text form which can be done using Tesseract software. The Tesseract software which makes use of Optical Character Recognition (OCR) is an OCR Machine.

Section II briefly explains about the literature survey we have done prior to implement our work. The software tools that were used while implementing our project are discussed in section III. The algorithm of our code is explained in section IV while a comparative analysis of different images is given in section V. The user interface which we developed is showcased in section VI and the goals we have achieved and which we desire to fulfill in the near future are pointed out in section VII

### A. *Theory Behind Image Processing*

Digital image processing is basically the use of computer algorithms to perform image processing on digital images. It allows a much wider range of algorithms to be applied to the input data and can avoid problems such as the build-up of noise and signal distortion during processing. [1] Since images are defined over two dimensions, digital image processing may be used in the form of multidimensional systems. In 2002 Raanan Fattel introduced Gradient domain image processing, a new way to process images in which the differences between pixels are manipulated rather than the pixel values themselves. [2] Some techniques which are used in digital image processing includes the following: Pixilation, Linear filtering, Image editing, Image restoration, Principal components analysis, Independent component analysis, Hidden Markov models, Anisotropic diffusion, Partial differential equations, Self-organizing maps, Neural networks and Wavelets.

## II. Related Work

Image processing is a hot research topic for a long time and has scope for many innovations in technological applications. Image processing is required in all the major developing and advanced sectors of society like medical, security, engineering,

978-1-5090-3800-8/17/$31.00 ©2017 IEEE

entertainment, media and much more. Different image preprocessing techniques necessary to achieve higher accuracy using methods like RGB to grey, blurring, thresholding and contouring [3] is important. In certain areas large number of images are required to access certain information. The Histogram technique is implemented to obtain statistical data from an image and categorize the image as poor or good [4]. It is also used for the normalization and equalizing the image. For small, redundant or unwanted fragments in an image a two stage frame work of Histogram processing for filtering and selection to remove unwanted small texts from an image is used [5]. In License Plate Detection the text extraction is the important goal thus image processing becomes the most crucial part prior to image to text conversion. The approach in [6] uses the method of corner points for text extraction in document images. It has fixed parameters assigned for different types of images like handwritten, typewritten, skewed, etc and it is quite fast. Paper [7] proposes the method of detecting license plate in an image taken from varied distance and different illuminations using wavelet transform and masking out potential license part. After the processing is completed then the processed image can be given to the Tesseract OCR Machine to convert the image into text using command line interface [8]. Text detection solely based on Tesseract is restricted to various parameters of image and thus preprocessing is necessary.

## III. SOFTWARE USED

Image processing is a software focused domain. OCR Software like Tesseract can be used to convert the image to text form. But this software is prone to errors, especially if the quality of the image is not pristine. The image needs to be processed using OpenCV and the processed image can be fed to Tesseract to get much better results.

### A. *OpenCV*

OpenCV (Open Source Computer vision) is free for both academic and commercial use. It is a library of programming functions mainly aimed at real-time computer vision. [9] OpenCV's application has wide areas which includes 2D and 3D feature toolkits, Egomotion estimation, Facial recognition system, Gesture recognition, Motion understanding, Object identification Segmentation and recognition and Motion tracking. [10] OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. [11] OpenCV contains libraries of pre-defined functions helpful in image processing. Since it is open source, it was chosen as the platform to test the project. Using OpenCV libraries we have implemented image processing mechanisms like RGB to grayscale conversion, erosion, dilation.

### B. *Python*

Python is a widely used high-level, general-purpose, interpreted, dynamic programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale. Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library. [12]

Python programming language made us enable to write short code snippets for each processing techniques. It also enabled us to develop a multi-level processing mechanism. Thus python programming language was indeed very helpful in the digital processing of the stock images by writing simple and easily understandable python codes.

### C. *Tesseract*

Tesseract package contains an OCR engine - libtesseract and a command line program - tesseract. The lead developer is Ray Smith. Tesseract has unicode (UTF-8) support, and can recognize more than 100 languages "out of the box". It can be trained to recognize other languages. Tesseract supports various output formats: plain-text, hocr(html), pdf. [13]

The Tesseract engine was originally developed as proprietary software at Hewlett Packard labs in Bristol, England and Greeley, Colorado between 1985 and 1994, with some more changes made in 1996 to port to Windows, and some migration from C to C++ in 1998. A lot of the code was written in C, and then some more was written in C++. Since then all the code has been converted to at least compile with a C++ compiler. [14]

Tesseract is available for Linux, Windows and Mac OS X, however, due to limited resources only Windows and Ubuntu are rigorously tested by developers. Tesseract up to and including version 2 could only accept TIFF images of simple one column text as inputs. These early versions did not include layout analysis and so inputting multi-columned text, images, or equations produced a garbled output. Since version 3.00 Tesseract has supported output text formatting, hOCR positional information and page layout analysis. Tesseract can detect whether text is monospaced or proportional. The initial versions of Tesseract could only recognize English language text. V3.04, released in July 2015, added an additional 39 language/script combinations, bringing the total count of support languages to over 100. Tesseract can be trained to work in other languages too. Tesseract is suitable for use as a backend, and can be used for more complicated OCR tasks including layout analysis by using a frontend such as OCRopus.

In this project, Tesseract is used as the final step for OCR after the image has been sufficiently processed so as to get optimum output.

## IV. ALGORITHM FOR LICENSE PLATE DETECTION

A few images of license plates were taken for processing.

The images were in different states in terms of quality. Processing of these images by a single common code would not yield equal results. Therefore, a common code with various stages of processing was decided upon, with output of each stage fed to the Tesseract OCR. The best output from the OCR would be taken as the converted text value.

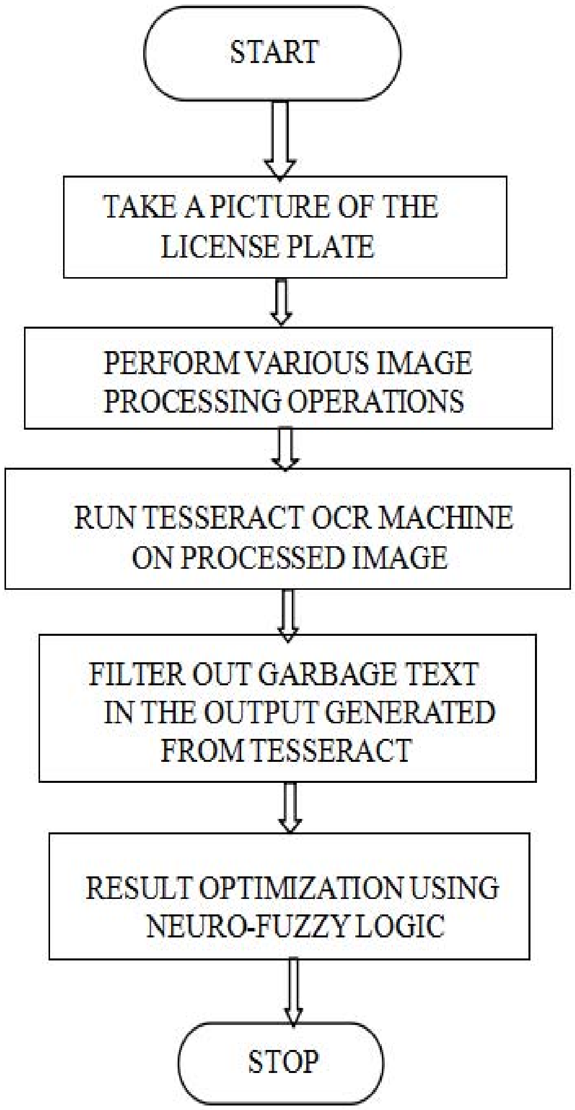

START
TAKE A PICTURE OF THE LICENSE PLATE
PERFORM VARIOUS IMAGE PROCESSING OPERATIONS
RUN TESSERACT OCR MACHINE ON PROCESSED IMAGE
FILTER OUT GARBAGE TEXT IN THE OUTPUT GENERATED FROM TESSERACT
RESULT OPTIMIZATION USING NEURO-FUZZY LOGIC
STOP

Fig.1 Flowchart for image processing and obtaining text

Fig.1 shows the algorithm for digital image processing and text extraction from the image.

Fig.2 shows the input image taken from the stock camera of cell phone. Image was taken in daylight environment with decent ambient conditions. Running of Tesseract over this image yielded blank output.

Fig.3 is the image obtained after performing thresholding on the processed original image. In this image, portion of license plate is clearly visible. Text result obtained after running Tesseract over this image was 'MHo'2cz 5712'. The single inverted comma after 'o' in text output was due to a stray mark present very close to the license plate number in the image.

Fig.4 shows the output as seen in the IDLE terminal. The image processing is immediately followed by a Tesseract operation on each of the different processing stages. The

## V. COMPARATIVE ANALYSIS

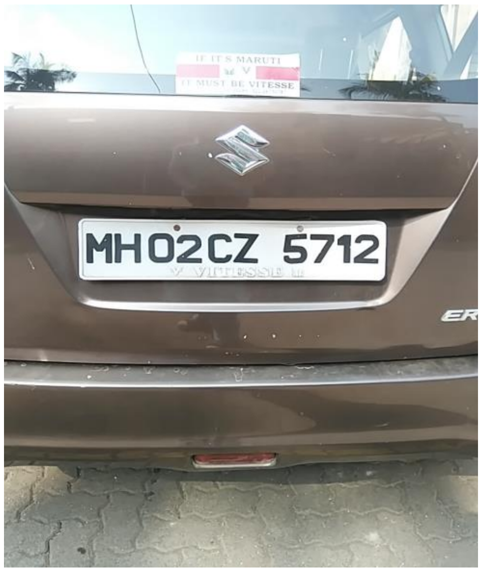


Fig.2 Stock image taken from camera

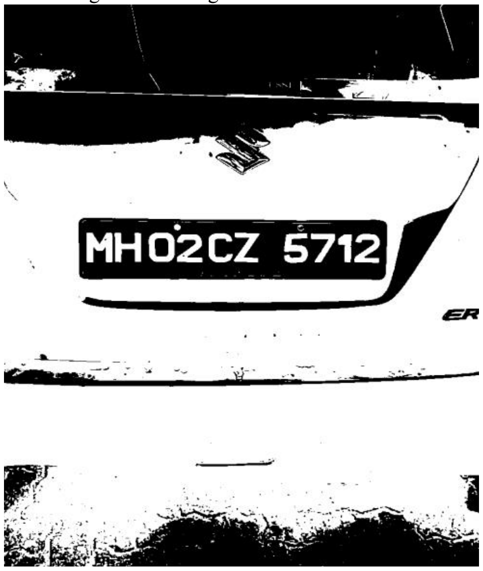
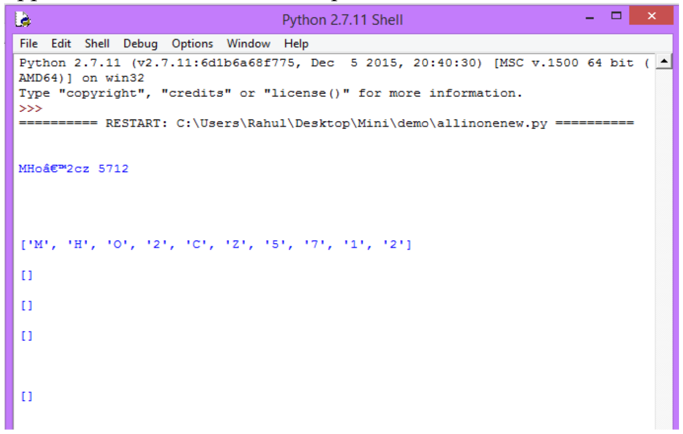


Fig. 3. Image after thresholding

outputs are then compiled and displayed on the terminal. The erroneous characters which are not part of a license plate are filtered out, the lowercase characters are converted to uppercase. The result is the sequence MH02CZ 5712.



```
Python 2.7.11 (v2.7.11:6d1b6a68f775, Dec  5 2015, 20:40:30) [MSC v.1500 64 bit (
AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:\Users\Rahul\Desktop\Mini\demo\allinonenew.py ==========


MHoâ€™2cz 5712




['M', 'H', 'O', '2', 'C', 'Z', '5', '7', '1', '2']

[]

[]

[]



[]
```

Fig. 4. Output of image 1 as seen in the IDLE terminal

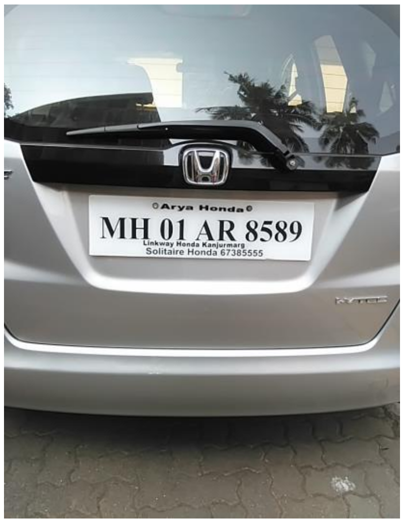


Fig. 5. Stock image taken from camera

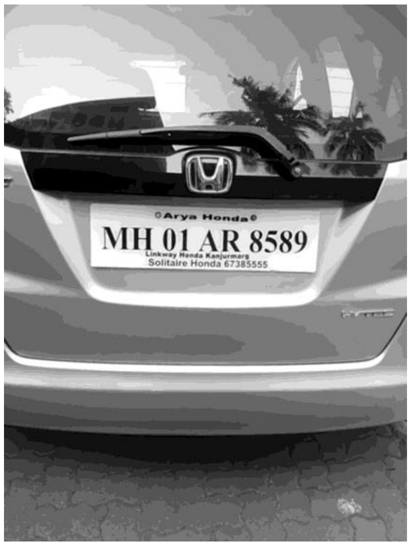
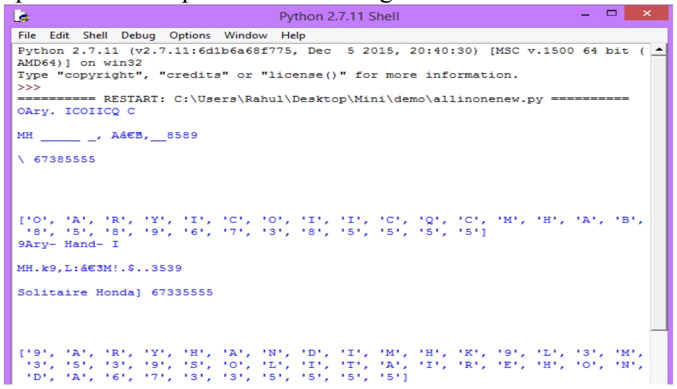


Fig. 6. Image after median blur

Fig.5 represents 2nd image captured using stock camera of different cell phone in different ambient conditions. Notice the font style of license number is different. '8589' is misinterpreted as '3539' after performing Tesseract over this image.

Fig.6 shows the image after performing the median blur operation on the processed 2nd image.



```
Python 2.7.11 (v2.7.11:6d1b6a68f775, Dec  5 2015, 20:40:30) [MSC v.1500 64 bit (
AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:\Users\Rahul\Desktop\Mini\demo\allinonenew.py ==========
OAry. ICOIICQ C

MH ______ _, Aâ€B,__8589

\ 67385555




['O', 'A', 'R', 'Y', 'I', 'C', 'O', 'I', 'I', 'C', 'Q', 'C', 'M', 'H', 'A', 'B',
 '8', '5', '8', '9', '6', '7', '3', '8', '5', '5', '5', '5']
9Ary- Hand- I

MH.k9,L:â€3M!.$..3539

Solitaire Honda] 67335555




['9', 'A', 'R', 'Y', 'H', 'A', 'N', 'D', 'I', 'M', 'H', 'K', '9', 'L', '3', 'M',
 '3', '5', '3', '9', 'S', 'O', 'L', 'I', 'T', 'A', 'I', 'R', 'E', 'H', 'O', 'N',
 'D', 'A', '6', '7', '3', '3', '5', '5', '5', '5']
```

Fig.7. Output of image 2 as seen in the IDLE terminal

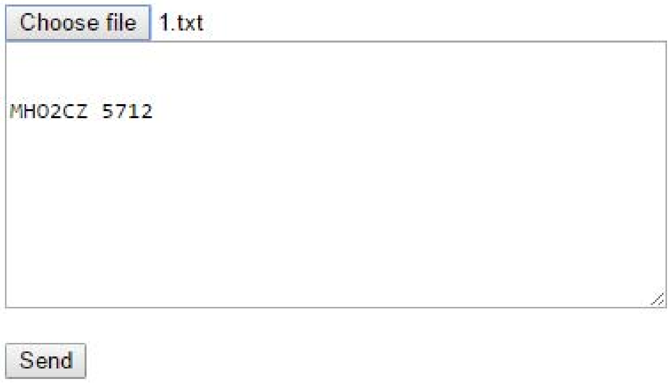
After running Tesseract over this image, the sequence 'CMH ______ _, A'B,__8589' is obtained among a larger output text file which indicates some level of progress, but not a completely error free output. Stray characters were obtained in text output while certain data was missing. This happened due to different thickness and style of the font of the license plate number in the image.

Thus depending on the characteristics of the image, different font styles and ambient conditions, various levels of processing are used and best possible text output is obtained.

## VI. RESULT

A user interface was designed which options to select image file and its equivalent text had output was generated. The image files as well as the OCR output were simultaneously displayed.

Fig.8 shows the extracted text output from Sample 1 license plate image file. Text output obtained was nearly equal to license number on image of license plate.



```
MH02CZ 5712
```

Fig. 8. Output with image file converted to text file

## VII. CONCLUSION

Image Processing is very crucial for extracting any information from an image. We first applied image processing algorithms to images and then those images were used in Tesseract software to obtain the text from the images. Different images have different text styles, length, width and font, so different images require different levels of digital image processing techniques. For this reason we have obtained image results after every level of processing i.e. thresholding, Gaussian blur, dilation and erosion. Out of these image results for a single image the best suited image is then applied to Tesseract for obtaining the text from an image. Therefore after digitally processing an image we have achieved better and near to perfection outputs.

The comparative analysis of the same algorithm on different sample images of license plates shows that optimum output will vary from stage to stage for each photo. Therefore the output of each stage must be observed and subjected to OCR and the best text output should be considered as the result. This would require the development of another algorithm to sort the different outputs and predict the optimum output.

The final output of the license plates was found to be very close to the actual value on the plates. However, some errors creep in depending upon the quality of the original image, the lighting, the shakiness and general condition of the license plate in question.

The different stages were passed through Tesseract OCR and their respective outputs passed through ASCII filter. The filter is successful in removing special characters from the text file, as well as converting lowercase characters to uppercase.

This software needs further text processing in order to be truly useful in a wide range of scenarios. Use of Neuro-Fuzzy networks to train the software to recognize the valid license plate number combinations is planned as the next stage of the software. This processing will be done after the OCR filtering stage and will be a part of the next phase of the software development. This will significantly reduce errors and be useful in predicting the most likely output based on previous results.

This software has prospective applications not only in public traffic enforcement, but also in housing societies and malls. Small establishments can maintain records of the vehicles registered to them and employ a system using the License Plate detection algorithm to effectively recognize and keep out intruder vehicles.

## ACKNOWLEDGMENT

We take it a deemed entitlement to express our sincere thanks to all concerned. We submit our gratitude and sincere thanks to our Institute and Department of Electronics and Telecommunication Engineering, for their constant motivation and support and having helped us shape the problem and providing intuition towards the solution. We are thankful to the Sardar Patel Institute of Technology, India for providing the necessary facilities for carrying out this work. Last but not the least, we would like to thank Prof. Dayanand D. Ambawade and Prof. Nataasha Raul for being there to support and help us..

## REFERENCES

[1] Azriel Rosenfeld, Picture Processing by Computer, New York: Academic Press, 1969.
[2] Space Technology Hall of Fame:Inducted Technologies/1994. Space Foundation. 1994. Retrieved 7 January 2010.
[3] Image edge detection based on OpenCV; Guobo Xie, Wen lu , \emph{International Journal of Electronics and Electrical Engineering Vol. 1, No. 2, June 2013}
[4] Image processing with OpenCV; Fabrizio Dini & Giuseppe Lisanti
[5] Text detection and removal from image using inpainting with smoothing; Priyanka Deelip Wagh, D.R. Patil
[6] Text extraction in document images: highlight on using corner points; Nicolas Ragot, Vikas Yadav
[7] Multiple license plate detection for complex background; Ching-Tang Hsieh, Yu-Shan Juan, Kuo-Ming Hung , \emph{Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA'05) 2005 IEEE}
[8] Optical Character Recognition by Open Source OCR Tool Tesseract: A Case Study; Chirag Patel, Atul Patel, Dharmendra Patel , \emph{International Journal of Computer Applications (0975 – 8887) Volume 55– No.10, October 2012}
[9] Itseez leads the development of the renowned computer vision library OpenCV. http://itseez.com
[10] Bradski, Gary; Kaehler, Adrian (2008). Learning OpenCV: Computer vision with the OpenCV library. O'Reilly Media, Inc. p. 6.
[11] CPAN: http://search.cpan.org/~yuta/Cv-0.29/
[12] "The RedMonk Programming Language Rankings: June 2015 – tecosystems". Redmonk.com. 1 July 2015. Retrieved 10 September 2015.
[13] Kay, Anthony (July 2007). "Tesseract: an Open-Source Optical Character Recognition Engine". Linux Journal. Retrieved 28 September 2011.
[14] Vincent, Luc (August 2006). "Announcing Tesseract OCR". Archived from the original on October 26, 2006. Retrieved 2008-06-26