

Automatic License Plate Recognition using Python and OpenCV

K.M. Sajjad

Department of Computer Science and Engineering
M.E.S. College of Engineering, Kuttippuram, Kerala
me@sajjad.in

Abstract—Automatic License Plate Recognition system is a real time embedded system which automatically recognizes the license plate of vehicles. There are many applications ranging from complex security systems to common areas and from parking admission to urban traffic control. Automatic license plate recognition (ALPR) has complex characteristics due to diverse effects such as of light and speed. Most of the ALPR systems are built using proprietary tools like Matlab. This paper presents an alternative method of implementing ALPR systems using Free Software including Python and the Open Computer Vision Library.

Keywords—License plate, Computer Vision, Pattern Recognition, Python, OCR.

I. INTRODUCTION

The scientific world is deploying research in intelligent transportation systems which have a significant impact on peoples' lives. Automatic License Plate Recognition (ALPR) is a computer vision technology to extract the license number of vehicles from images. It is an embedded system which has numerous applications and challenges. Typical ALPR systems are implemented using proprietary technologies and hence are costly. This closed approach also prevents further research and development of the system. With the rise of free and open source technologies the computing world is lifted to new heights. People from different communities interact in a multi-cultural environment to develop solutions for mans never ending problems. One of the notable contribution of the open source community to the scientific world is Python. Intel's researches in Computer Vision bore the fruit called Open Computer Vision (OpenCV) library, which can support computer vision development.

II. PROPOSED SYSTEM

In India, basically, there are two kinds of license-plates, black characters in white plate and black characters in yellow plate. The former for private vehicles and latter for commercial, public service vehicles. The system tries to address these two categories of plates. The high-level block diagram of the proposed system is shown in Fig. 1.

A. Capture

The image of the vehicle is captured using a high resolution photographic camera. A better choice is an Infrared (IR) camera. The camera may be rolled and pitched with respect to the license plates.

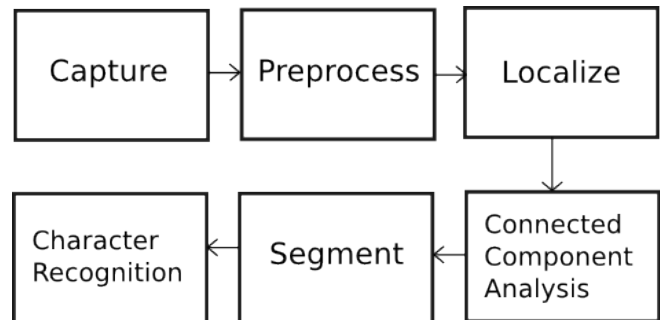


Fig. 1. Proposed System



Fig. 2. Captured Image (Original image)

Character recognition is generally very sensitive to the skew. The readable characters can become distorted due to the obliqueness of the camera. Using a better camera with more definition and resolution will increase the success ratio of the system.

B. Preprocess

Preprocessing is the set algorithms applied on the image to enhance the quality. It is an important and common phase in any computer vision system. For the present system preprocessing involves two processes: **Resize** – The image size from the camera might be large and can drive the system slow. It is to be resized to a feasible aspect ratio. **Convert Colour Space** – Images captured using IR or photographic cameras will be either in raw format or encoded into some multimedia standards. Normally, these images will be in RGB mode, with three channels (viz. red, green and blue).



Fig. 3. Localization (Thresholded image)

Number of channels defines the amount colour information available on the image. The image has to be converted to grayscale.

C. Localize

Rear or front part of the vehicle is captured into an image. The image certainly contains other parts of the vehicle and the environment, which are of no requirement to the system. The area in the image that interests us is the license plate and needs to be localized from the noise. Localization is basically a process of binarizing the image. As shown in Fig. 3, the image is converted to black and white. There are two motivations for this operation – **1. Highlighting characters** and **2. Suppressing background**.

Localization is done by an image processing technique called **Thresholding**. The pixels of the image are truncated to two values depending upon the value of threshold. Threshold requires pre-image analysis for identifying the suitable threshold value. Adaptive thresholding technique determines a local optimal threshold value for each image pixel so as to avoid the problem originating from non-uniform illumination.

D. Connected Component Analysis



Fig. 4. Connected Components (Blobs)

In order to eliminate undesired image areas, a connected component algorithm is first applied to the binarized plate candidate. Connected component analysis is performed to identify the characters in the image. Basic idea is to traverse

through the image and find the connected pixels. Each of the connected components (blobs) are labelled and extracted. Fig. 4 shows the filtered blobs.

E. Segmentation

Segmentation is the process of cropping out the labelled blobs. These blobs are expected to be the required portion of the license number. A special algorithm called **Image Scissoring**[1] is introduced here. In this algorithm, the license plate is vertically scanned and scissored at the row on which there is no white pixel and the scissored area is copied into a new matrix, as in Fig. 5.



Fig. 5. Segmented blobs

There are unwanted blobs even after segmentation. These are classified using special algorithms.



Fig. 6. Classified Blobs

F. Character Recognition

Finally, the selected blobs are send to a Optical Character Recognition (OCR) Engine, which returns the ASCII of the license number.

III. TOOLS

A. Python

Python is a remarkably powerful dynamic, object-oriented programming language that is used in a wide variety of application domains. It offers strong support for integration with other languages and tools, and comes with extensive standard libraries. To be precise, the following are some distinguishing features of Python:

- Very clear, readable syntax.
- Strong introspection capabilities.
- Full modularity.
- Exception-based error handling.

- High level dynamic data types.
- Supports object oriented, imperative and functional programming styles.
- Embeddable.
- Scalable
- Mature

With so much of freedom, Python helps the user to think problem centric rather than language centric as in other cases. These features makes Python a best option for scientific computing.

B. OpenCV

OpenCV is a library of programming functions for real time computer vision originally developed by Intel and now supported by Willowgarage. It is free for use under the open source BSD license. The library has more than five hundred optimized algorithms. It is used around the world, with forty thousand people in the user group. Uses range from interactive art, to mine inspection, and advanced robotics. The library is mainly written in C, which makes it portable to some specific platforms such as Digital Signal Processor. Wrappers for languages such as C, Python, Ruby and Java (using JavaCV) have been developed to encourage adoption by a wider audience. The recent releases have interfaces for C++. It focuses mainly on real-time image processing. OpenCV is a cross-platform library, which can run on Linux, Mac OS and Windows. To date, OpenCV is the best open source computer vision library that developers and researchers can think of.

C. Tesseract

Tesseract is a free software OCR engine that was developed at HP between 1984 and 1994. HP released it to the community in 2005. Tesseract was introduced at the 1995 UNLV Annual Test OCR Accuracy [2] and is currently developed by Google released under the Apache License. It can now recognize 6 languages, and is fully UTF8 capable. Developers can train Tesseract with their own fonts and character mapping to obtain perfect efficiency.

IV. IMPLEMENTATION

OpenCV with Python is a deadly combination for computer vision. We will now revisit each phase in the pipeline and see how they are implemented using the above tools.

A. Capture

Starting from point one of capturing the image, OpenCV library has highly optimized algorithms for all image processing operations. OpenCV provides interface for different camera models. The following code snippet explains how to interface an in-built web camera and capture a frame.

```
from opencv import highgui as hg
capture = hg.cvCreateCameraCapture(0)
hg.cvNamedWindow("Snapshot")
frame = hg.cvQueryFrame(capture)
hg.cvShowImage("Snapshot", frame)
```

B. Preprocess

As seen before, preprocessing involves resizing and changing colour spaces of the source image. Like any other image processing toolkits, OpenCV also provides fast and quick procedures.

Resize

```
original = cv.LoadImageM("image.jpg")
thumbnail = cv.CreateMat(original.rows
/ 10, original.cols / 10, original.type)
cv.Resize(original, thumbnail)
```

The original image is resized to the dimensions specified in the thumbnail object. **Colour space conversion**
`CvtColor(original, gray, CV_RGB2GRAY)`
The above line of code converts the original image to gray. More image conversion codes are available at the OpenCV Wiki[3].

C. Localize

Threshold operation is performed in this phase. To retain the image quality, adaptive threshold algorithms are to be used. Previous researches[4] have concluded that Otsu's[5] thresholding algorithm is the efficient way of binarizing the image. OpenCV provides complex and efficient adaptive thresholding algorithms including Otsu method.
`cvThreshold(image, binary_image, 128, 255, CV_THRESH_OTSU)`
The above line of code returns a binary image which is adaptively thresholded. The arguments follow the order: 1. Source image, 2. Destination image, 3. Threshold value, 4. Resultant value, and 5. Type of threshold. The type `CV_THRESH_OTSU` performs Otsu algorithm on the source image.

D. Connected Component Analysis

cvBlobsLib is a library to perform binary images connected component labelling. It also provides functions to manipulate, filter and extract results from the extracted blobs. The library provides two basic functionalities:

- Extract 8-connected components in binary or grayscale images.
- Filter the obtained blobs to get the interest objects in the image. This is performed using the Filter method from `CBlobResult`.

```
The library is thread-safe if different objects per thread are used.
myblobs = CBlobResult(binary_image, mask, 0, True)
myblobs.filter_blobs(325, 2000)
blob_count = myblobs.GetNumBlobs()
```

The connected components are labelled using the above code snippet. `filter_blobs` method is used to filter out the blobs of required dimensions.

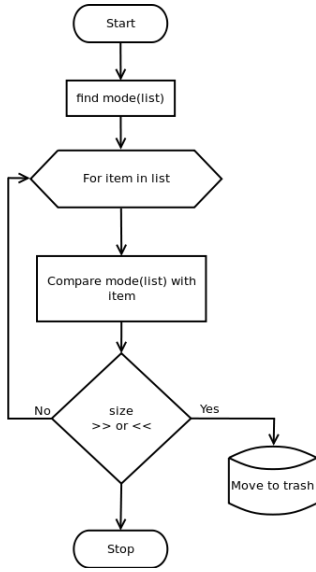


Fig. 7. Aspect ratio based elimination

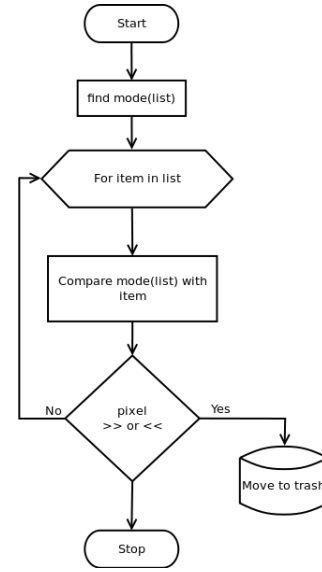


Fig. 8. Pixel coordinate based selection

E. Segmentation

Image Scissoring is hard-coded in Python by scanning the image vertically and cropping out white portions. The algorithm, is fast and efficient than compared to other pre-defined image cropping techniques. Segmentation phase also involves classification of the collected blobs and recording only the essential ones. Undesirable blobs occur even after segmentation. These are removed by two methods:

- 1) Aspect ratio based elimination.
- 2) Pixel coordinate based selection.

Aspect ratio based elimination: The aspect ratio (row/column) of each blob is calculated and recorded. A binarized candidate is sure of containing more characters than unwanted blobs. The mean of the aspect ratios are calculated and compared to all the blobs in turn. The process is explained briefly in the flowchart Fig. 7.

If anyone of them has a larger deviation, that blob is removed from the candidate. This algorithm was devised based on research and experiment through out the process. The dynamic nature of Python is exploited in every step of this algorithm.

Pixel coordinate based selection: This algorithm thrives on the fact that license numbers are occurring in the plate in a single set of rows. Effectively, we can detect the edge of the license plate, and select the blobs coming between the minimum and maximum row coordinates. This can reduce the amount of unwanted blobs and make the system more accurate. The algorithm is explained in Fig. 8.

F. Character Recognition

Tesseract OCR engine has a Python wrapper, which make character recognition quick and easy. from tesseract import image_to_string

```

image = open("blob.jpg")
text = image_to_string(image)
  
```

V. STATISTICS AND RESULTS

The system has been put to test for various measurements of performance and accuracy.

A. Accuracy Analysis

TABLE I
ACCURACY ANALYSIS

Operation	Sample	Success	Fail	Success Ratio
License plate localization	100	92	8	92%
Character Separation	92	88	4	95.7%
Character Recognition	88	83	5	94.3%

B. Performance Analysis

During the initial days, the system suffered severe performance faults. It took more than eighteen seconds for recognizing the license plate and extracting the number. Subsequent research on algorithm and code optimization drastically brought down the operation time to two seconds. One of the key factor that determined performance was the size of input image. The Fig. 9 indicates this relationship.

VI. CONCLUSION

The message of this research is to show that free and open source technologies are matured enough for scientific computing domains. Python and OpenCV are good points of start for researchers and students of computer vision.

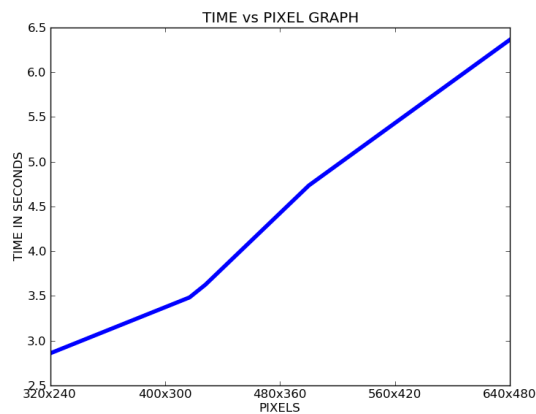


Fig. 9. Performance Analysis

The system works satisfactorily for wide variations in illumination conditions and different types of number plates commonly found in India. It is definitely a better alternative to the existing proprietary systems, even though there are known restrictions.

ACKNOWLEDGMENT

Bunch of thanks to my associates A. Nijad, J. Saalim and A. Shehzad; N. Sajith, Assistant Professor, Department of Computer Science and Engineering, M.E.S. College of Engineering, for mentoring this research.

REFERENCES

- [1] Prathamesh Kulkarni, Ashish Khatri, Prateek Banga, Kushal Shah, *Automatic Number Plate Recognition (ANPR) System for Indian conditions*
- [2] S.V. Rice, F.R. Jenkins, T.A. Nartker, *The Fourth Annual Test of OCR Accuracy, Technical Report 95-03*. Information Science Research Institute, University of Nevada, Las Vegas,(1995).
- [3] <http://opencv.willowgarage.com/documentation/python/>
- [4] A.Conci, J. E. R. de Carvalho, T. W. Rauber, *A Complete System for Vehicle Plate Localization, Segmentation and Recognition in Real Life Scene*, IEEE LATIN AMERICA TRANSACTIONS, VOL. 7, NO. 5, SEPTEMBER 2009
- [5] Nobuyuki Otsu (1979). *A threshold selection method from gray-level histograms*. IEEE Trans. Sys., Man., Cyber. 9: 62-66.
- [6] Chih-Hai Fana, Yu-Hang Peng, *Vehicle License Plate Recognition System Design*, Chung Hua Journal of Science and Engineering, Vol. 7, No. 2, pp. 47-52 (2009)