International Journal of Computer Applications Technology and Research
Volume 3– Issue 12, 756 - 761, 2014, ISSN:- 2319–8656

Automatic License Plate Recognition using OpenCV

Pratiksha Jain
Department of CSE
IGIT, GGSIPU
New Delhi, India

Neha Chopra
Department of ECE
IGIT, GGSIPU
New Delhi, India

Vaishali Gupta
Department of CSE
IGIT, GGSIPU
New Delhi, India

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

1. 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.

2. 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.[Reference 1]

2.1 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.

Figure. 1 Example of a Number Plate with acceptable resolution

2.2. 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 Color 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).

Figure. 2 Image converted in RGB mode

After performing the steps 1 and 2, the image is passed to next component.

2.3. License Plate Extractor

This is most critical process in License Plate Recognition System. In this process we apply different techniques on image to detect and extract license plate. This process is divided in two parts.

2.3.1 License Plate Detection through Haar-like features

In image processing techniques, Haar-like features are used to recognize objects from image . If our proposed system is selected to detect only license plates then the Haar-like features are used for this purpose and no further processing is done. This technique is old and laborious and more over needs a large database to store the collected samples nearly about 10000 images of the plates and characters

2.3.2.2 License Plate Detection through Edge Detection

In the other case, if our proposed system has to recognize license plates, then the binary image is created from the image. After that following steps are performed to extract license plate from binary image:
1. Four Connected Points are searched from binary image.

2. Width/Height ratio is matched against those connected points.

3. License Plate region is extracted from image.

4. Transformation of extracted license plate is performed.

Then the extracted license plate is passed to next component for further processing.
This approach is quick and takes less execution time and memory with high a efficiency ratio. That's why we have adopted this technique in our project

Figure 3: License Plate Extraction

2.4 Character Segmentation

In this part further image processing is done on extracted license plate to remove unnecessary data. After character segmentation, the extracted license plate has only those characters that belong to license number.
This also achieved with the width height ratios matching with the contours detected on extracted number plate.

Figure 4: Character Segmentation

2.5. Optical Character Recognition

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

3.WHY OPENCV??

Advantages of OpenCV over MATLAB

- **Speed:** Matlab is built on Java, and Java is built upon C. So when you run a Matlab program, your computer is busy trying to interpret all that Matlab code. Then it turns it into Java, and then finally executes the code. OpenCV, on the other hand, is basically a library of functions written in C/C++. You are closer to directly provide machine language code to the computer to get executed. So ultimately you get more image processing done for your computers processing cycles, and not more interpreting. As a result of this, programs written in OpenCV run much faster than similar programs written in Matlab. So, conclusion? OpenCV is damn fast when it comes to speed of execution. For example, we might write a small program to detect peoples smiles in a sequence of video frames. In Matlab, we would typically get 3-4 frames analyzed per second. In OpenCV, we would get at least 30 frames per second, resulting in real-time detection.

- **Resources needed:** Due to the high level nature of Matlab, it uses a lot of your systems resources. And I mean A LOT! Matlab code requires over a gig of RAM to run through video. In comparison, typical OpenCV programs only require ~70mb of RAM to run in real-time. The difference as you can easily see is HUGE! [Reference 5].

- **Cost**: List price for the base (no toolboxes) MATLAB (commercial, single user License) is around USD 2150. OpenCV (BSD license) is *free!* Now, how do you beat that?

- **Portability:** MATLAB and OpenCV run equally well on Windows, Linux and MacOS. However, when it comes to OpenCV, any device that can run C, can, in all probability, run OpenCV.

- **Specific:** OpenCV was made for image processing. Each function and data structure was designed with the Image Processing coder in mind. Matlab, on the other hand, is quite generic. You get almost anything in the world in the form of toolboxes. All the way from financial toolboxes to highly specialized DNA toolboxes.

Despite all these amazing features, OpenCV does lose out over MATLAB on some points:

- **Ease of use:** Matlab is a relatively easy language to get to grips with. Matlab is a pretty high-level scripting language, meaning that you don't have to worry about libraries, declaring variables, memory management or other lower-level programming issues. As such, it can be very easy to throw together some code to prototype your image processing idea. Say for example I want to read in an image from file and display it**.**

- **Memory Management:** OpenCV is based on C. As such, every time you allocate a chunk of memory you will have to release it again. If you have a loop in your code where you allocate a chunk of memory in that loop and forget release it afterwards, you will get what is called a "leak". This is where the program will use a growing amount of memory until it crashes from no remaining memory. Due to the high-level nature of
Matlab, it is "smart" enough to automatically allocate and release memory in the background.
- -Matlabs memory management is pretty good. Unless your careful with your OpenCV memory allocation and releasing, you can still be frustrated beyond belief.

- **Development Environment**: Matlab comes with its own development environment. For OpenCV, there is no particular IDE that you have to use. Instead, you have a choice of any C programming IDE depending on whether you are using Windows, Linux, or OS X. For Windows, Microsoft Visual Studio or NetBeans is the typical IDE used for OpenCV. In Linux, its Eclipse or NetBeans, and in OSX, we use Apple's Xcode**.**

- **Debugging:**Many of the standard dedugging operations can be used with both Matlab and OpenCV: breakpoints can be added to code, the execution of lines can be stepped through, variable values can be viewed during code execution etc. Matlab however, offers a number of additional debugging options over OpenCV. One great feature is that if you need to quickly see the output of a line of code, the semi-colon at the end can be omitted. Also, as Matlab is a scripting language, when execution is stopped at a particular line, the user can type and execute their own lines of code on the fly and view the resulting output without having to recompile and link again. Added to this is are Matlab's powerful functions for displaying data and images, resulting in Matlab being our choice for the easiest development environment for debugging code.

3.1 Conclusion

	Matlab	OpenCV
Ease of Use	9	3
Speed	2	9
Resources Needed	4	9
Cost	4	10
Development Environment	8	6
Memory Management	9	4
Portability	3	8
Development of usefull programming skills	3	8
Help and Sample Code	8	9
Debugging	9	5
Total:	59	71

Figure 5: NECKBEARD INDEX SCORES

From the final scores we can see that OpenCV has the edge over Matlab for image and video processing development . Although Matlab has an easy learning curve, built in memory management, a great help section, it is very slow to execute code, and is expensive to get started in. While OpenCV can be difficult to debug and requires much "housework code" needed for memory management, header files, etc., it wins out due to its free cost, the magnitude of sample code available on the internet, the short development path from prototype code to embedding code, the useful programming skills learnt from its use, and its super-fast speed. Matlab is a more "generic" programming language in that it was designed for many uses, demonstrated by its numerous toolboxes ranging from financial to specialized DNA analyzing tools. On the other hand, OpenCV was made for image processing. Each function and data structure was designed with the image processing coder in mind**.**

4. PROPOSED SOLUTION

4.1 Assumptions

The objective of our thesis is to detect and recognize license plate. Our application is based on following assumptions:

1. Maximum expected distance between car and camera: 5 meters.

2.Minimum Camera angle: 90 degree (looking straight at the license plate).

3.Image should be captured in daylight.

4.Minimum Camera resolution: 3 Mega Pixel.
It is expected that it would not work efficiently during night time, rainy and cloudy days because mobiles cameras are not equipped with proper lightning. It is also expected that it will give results with decreasing accuracy with angles deviating significantly from the 90-degree (ideal) angle.

5. The new algorithm proposed for character recognition would give results with considerable percentage of errors on implementation.

6. The efficiency of the proposed system can be measured only in terms of number of license plates successfully and correctly recognized which can only be measured upon implementation.

7. Efficiency and Performance of new system may decline due to discard of OCR library but the memory requirements will decrease and also the effort for installing, configuring and running the system would decrease.

4.2 New Components of Proposed System as compared to traditional system

- **DETECTION ALGORITHM**
  We are designing this system specifically for the new proposed high security number plates which have black boundary across the number plate and also have a uniform font all across the country. So we are going to utilize this black boundary in our system by using edge based license plate detection method in our system. traditionally haar like features are used for detection. This algorithm needs a large number of license plate images which are manually obtained from a number of images including the backgrounds. It requires a larger memory to run, which is not suitable for embedded systems. Another problem with the systems using AdaBoost is that they are slower than the edge-based methods. This system is very sensitive to the distance between the camera and the license plate as well as the view angle. So we can eliminate all the above problems by using edge based detection method for our system. however detection rate of edge based method is slightly less than haar like features. This can be supported by the study conducted by some research students of Linnaeus university. Haar like feature were 96% accurate while edge based method was 87% accurate.

- **OCR LIBRARY NOT USED**
  In traditional system OCR Library is used which has to be installed, configured and run and which actually recognize the characters. We are not using this library. Instead we are developing our own algorithm for character reading. also OCR engines occupy more than 25 MB space and configuration of OCR engine has to be done with the source code. Compiler takes quite long time in compilation of typical OCR code because of the specific quality checks, spell checks, valid word checks etc. these checks are not required in ALPR case because spell checks, valid word checks are useless in case of number plates. so our algorithm is simple, fast and occupies less memory than an OCR engine. also it is expected that it will provide correct results upon implementation

4.3 Proposed Algorithm

DESCRIPTION OF THE NEW ALGORITHM FOR CHARACTER RECOGONITION

In this part, character segmented license plate is passed to optical character recognition algorithm designed by us which uses a matrix transformation of the pixel value of the binary and thus applying various filtration and extraction techniques which uniquely identifies the characters. OCR Algorithm returns license plate in text format. Which is later stored in a text file thus reducing the space in the memory storage.[Reference 3]

- Our algorithm uses a 3-4 MB database of 36 files(images).
- These 36 images are samples containing capital alphabets(A-Z) and numerals(0-9).
- These images will be colored images but only of one color say red. So pixel values where there is character is 255,0,0.
- and where the space is empty the value is 255,255,255. then the characters obtained after character segmentation are mapped with the characters in the data base one by one
- The character obtained from segmentation is mapped to a matrix one by one.
- then this matrix is compared with the sample images in database one by one.
- if the character matches then the value of the character is returned. Else next character is matched.
- if any of the 36 characters don't match with the image then either there is a distorted image or the number plate is invalid. In this condition a message will be returned.
- The matrix used will be preferably 20x20.
- for mapping between sample image and actual character we are using green intensity pixels. Because their value is 0 at every point where there is character and 255 where there is white background.
- we could have used blue intensity as well.
- this algorithm will thus possibly be able to detect similar characters like 8 and B because percentage of matching of one character will be higher than other.
- It is assumed that if any image is matched with 70-80% pixel intensities we assume that character matches
- then matrix is refreshed and new character gets copied in matrix.

the process continues until all the characters in license plate gets matched.

4.3.1 Algorithm for OCR Reader

```
OCRReader(image temp)
{
        Int mat[30][30]
        for(y=0, y<temp->height , y++)
        {
                for (x=0 , x<temp->width , x++)
                {
                /value is a structure/
                value=get RGB values of temp
                /b,g,r are integers/
                b=value.val[0]
                g=value.val[1]
                r=value.val[2]

                mat[y][x]=g;
                }
        }
        stringcopy(file,"folder of 36 files")
        for(int j=0 , j<36 , j++)
        {
                count=0
                stringcopy(file,"folder of 36 files")
                ext=get file j
                l= length of string (file)
                file[l]=ext
                file[l+1]='\0'
                file=file+"jpg"
         lchar=create image of frame of given size
         lchar= load image of name  file +"jpg"
                for(y=0; y<lchar->height; y++)
                {
                for (x=0;x<lchar->width;x++)
                {
                value= get RGB values of lchar
                b=value.val[0]
                g=value.val[1]
                r=value.val[2]

                l_mat[y][x]=g;
                }
                }

        for(y=0;y<30;y++)
        {
                for(x=0;x<30;x++)
                {
                        if(mat[y][x]==l_mat[y][x])
                        count++;
                }
        }
        if(count>400)
                {
                cout<<ext<<"in";
                }
      }
}
```

4.4 Asymptotic analysis of Algorithm

Complexity of above code is O(mn^2). Where m=36(A-Z , 0-9)and n is the pixel resolution. this is same as complexity of OCR reader.
But In traditional System OCR engine has database of 2^16 symbols(Unicode). So there value of m=2^16. Hence significant reduction in Time complexity. also since database is of 36 symbols instead of 2^16 it results in significant reduction in Space complexity.

5. CONCLUSION

The message of this research is to show that free and open source technologies are matured enough for scientific computing domains**.** 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

5.1 Future Work

Currently We have proposed the algorithms for our ALPR system. In future we would implement this system on Open CV library and would also do the performance check of the system designed. We would do the performance analysis in terms of number of plates successfully recognized. So far the algorithms looks good and suitable but if the OCR algorithm won't work than we will try to give some new algorithm or would do the comparative study of different OCR present in the market and would try to choose the best among them and implement the system.

6. ACKNOWLEDGMENTS

Our Sincere Thanks to Dr Kalpana Yadav ,our mentor for providing her guidance and cooperation in this Research.

7. REFERENCES

[1] 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

[2] Ahmed Gull Liaqat, "Real Time Mobile License Plate Recognition System" IEEE White paper California, VOL.2 2011-12-05,Linnaeus University.

[3] Ondrej Martinsky (2007). "Algorithmic and mathematical principles of automatic number plate recognition systems" (PDF). Brno University of Technology. http://javaanpr.sourceforge.net/anpr.pdf.

[4] P. Kreling, M. Hatsonn "A License Plate Recognition algorithm for Intelligent Transportation System applications". University of the Aegean and National Technical University of Athens. 2006. Archived from the original on 2008-04-20.

[5] K.M Sajjad , "ALPR Using Python and Open CV" Dept Of CSE, M.E.S College of Engineering Kuttipuram, kerala.2008-06-21

[6] Nicole Ketelaars "Final Project : ALPR", 2007-12-11

[7] Steven Zhiying Zhou , Syed Omer Gilani and Stefan Winkler "Open Sourc framework Using Mobile Devices" Interactive Multimedia Lab, Department of Electrical and Computer Engineering National University of Singapore, 10 Kent Ridge Crescent, Singapore 117576

[8] Yunggang Zhang, Changshui Zhang "A new Algorithm for Character Segmentation Of license plate" Beijing University,China, 2007-5-8