A Project Report

on

**AUTOMATIC LICENSE NUMBER PLATE
RECOGNITION**

Submitted in partial fulfillment of the

requirement for the award of thedegree

of

Bachelors Of Technology
(Computer Science and
Engineering)

GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

**Under The Supervision
of Mr. E Gautham
Assistant Professor**

Submitted By
Siddharth satyarthi
(20SCSE1010661)
Dhawal sharma
(19SCSE1010162)

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA
DECEMBER,
2021**

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

We hereby certify that the work which is being presented in the project entitled **"AUTOMATIC LICENSE NUMBER PLATE RECOGNITION"** in partial fulfillment of the requirements for the award of the Bachelor's of Technology submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of month, Year to Month and Year, under the supervision of Mr. Tarun Kumar Associate Professor, Department of Computer Science and Engineering of School of Computing Science and Engineering , Galgotias University, Greater Noida

   The matter presented in the project has not been submitted by us for the award of any other degree of this or any other places.

<div align="right">

Siddharth(20SCSE1010661)

Dhawal(19SCSE1010162)

</div>

   This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

<div align="right">

E Gautham

Assistant

Professor

Department of Computer Science and Engineering

</div>

CERTIFICATE

The Final Project Viva-Voce examination of Siddharth(20SCSE1010661) Dhawal(19SCSE1010162) has

been held on_____and his/her work is recommended for the award of Bachelor's of Technology.

Signature of Examiner(s)   Signature of Supervisor(s)

Signature of Project Coordinator   Signature of Dean

Date: December, 2021

Place: Greater Noida

Abstract

Python is an experiment in how much freedom programmers need. Too much freedom and nobody can read another's code; too little and expressiveness is endangered. Application of python extends widely when it comes to visualizing one's ideas. Our idea of designing an Automatic License Number Plate Recognition System is also a python-based project implemented using OpenCV. The OpenCV library of python used for image processing and image visualization is on the rise. This project aims to recognize license number plates. In order to detect license number plates, we will use OpenCV to identify number plates and python pytesseract to extract characters and digits from the number plates. The project program will give an interface that will accept the input as sample images of the vehicle and will give the output as the License/Registration Number of the vehicle. This program will recognize the number plate area and will make the number displayed in a text format.

TABLE OF CONTENTS

CHAPTER-1

Introduction

1.1 Formulation of the Problem: -

This project aims to recognize license number plates, the project could be useful for security, monitoring, e-challan, etc. In order to detect license number plates, we will use OpenCV to identify number plates and python pytesseract to extract characters and digits from the number plates.

OpenCV is an open-source machine learning library and provides a common infrastructure for computer vision.

Whereas Pytesseract is a Tesseract-OCR Engine to read image types and extract the information present in the image. Deep learning is a branch of machine learning that uses several layers of nonlinear processing units for feature extraction and transformation. Each layer uses output from the previous layer as input.

The projections of the proposed system are: 1) Single Entry 2) Single Exit Gate 3) Single-route traffic, further improvement to double-route routes. The video source is located at the entrance of the Institute which is able to provide live feeds with some accuracy or poorly illuminated. The video source is located in an area that can hold the entire vehicle and the required frames containing the vehicle are selected accordingly.

1.2 Tools and Technology Used: -

OpenCV: - OpenCV is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itrez. The library is cross-platform and free for use under the open- source Apache 2 License.

Sobel Operator: - The Sobel operator, sometimes called the Sobel–Feldman operator or Sobel filter, is used in image processing and computer vision, particularly within edge detection algorithms where it creates an image emphasizing edges.

Pytesseract: - Python-tesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and "read" the text embedded in images.

Python-tesseract is a wrapper for Google's Tesseract-OCR Engine. It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Pillow and Leptonica imaging libraries, including jpeg, png, gif, bmp, tiff, and others.

OpenCV: -

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching streetview images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDAand OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

The first OpenCV version was 1.0. OpenCV is released under a BSD license and hence it's free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. When OpenCV was designed the main focus was real-time applications for computational efficiency. All things are written in optimized C/C++ to take advantage of multi-core processing.

Applications of OpenCV: There are lots of applications which are solved using OpenCV, some of them are listed below.
- face recognition
- Automated inspection and surveillance
- number of people – count (foot traffic in a mall, etc)
- Vehicle counting on highways along with their speeds
- Interactive art installations
- Anamoly (defect) detection in the manufacturing process (the odd defective products)
- Street view image stitching
- Video/image search and retrieval
- Robot and driver-less car navigation and control
- object recognition
- Medical image analysis
- Movies – 3D structure from motion
- TV Channels advertisement recognition

OpenCV Functionality:
- Image/video I/O, processing, display (core, imgproc, highgui)
- Object/feature detection (objdetect, features2d, nonfree)
- Geometry-based monocular or stereo computer vision (calib3d, stitching, videostab)
- Computational photography (photo, video, superres)
- Machine learning & clustering (ml, flann)
- CUDA acceleration (gpu)

Image-Processing: -

Image processing is a method to perform some operations on an image, in order to get an enhanced image and or to extract some useful information from it.

If we talk about the basic definition of image processing then "Image processing is the analysis and manipulation of a digitized image, especially in order to improve its quality".

Digital-Image:

An image may be defined as a two-dimensional function f(x, y), where x and y are spatial(plane) coordinates, and the amplitude of fat any pair of coordinates (x, y) is called the intensity or grey level of the image at that point.

In another word An image is nothing more than a two-dimensional matrix (3-D in case of coloured images) which is defined by the mathematical function f(x, y) at any point is giving the pixel value at that point of an image, the pixel value describes how bright that pixel is, and what colour it should be.

Image processing is basically signal processing in which input is an image and output is image or characteristics according to requirement associated with that image.

Image processing basically includes the following three steps:

- Importing the image
- Analyzing and manipulating the image
- Output in which result can be altered image or report that is based on image analysis

Sobel Operator: -

Using the sobel operation, you can detect the edges of an image in both horizontal and vertical directions. You can apply sobel operation on an image using the method Sobel(). Following is the syntax of this method −

Sobel (src, dst, ddepth, dx, dy)

This method accepts the following parameters −

src − An object of the class Mat representing the source (input) image.

dst − An object of the class Mat representing the destination (output) image.

ddepth − An integer variable representing the depth of the image (-1)

dx − An integer variable representing the x-derivative. (0 or 1)

dy − An integer variable representing the y-derivative. (0 or 1)

Let us consider the following image for sobel operator program: -

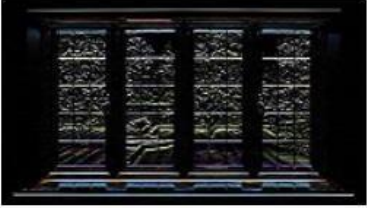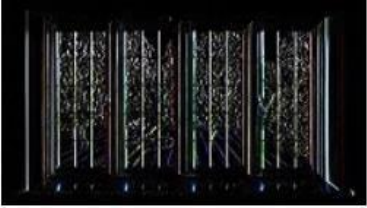Output of the above image when passed through sobel operator will be: -

Sobel Variants: -
On passing different values to the last to parameters (dx and dy) (among 0 and 1), you will get different outputs −

// Applying sobel on the Image

Imgproc.Sobel(src, dst, -1, 1, 1);

The following table lists various values for the variables dx and dy of the method Sobel() and their respective outputs.

X-derivative	Y-derivative	Output
0	1	
1	0	
1	1	

Sobel Edge Detection: -

Sobel Edge Detection is one of the most widely used algorithms for edge detection. The Sobel Operator detects edges that are marked by sudden changes in pixel intensity, as shown in the figure below.

Graph of pixel intensity as a function of t
Pixel intensity as a function of t (Source)
The rise in intensity is even more evident, when we plot the first derivative of the intensity function.

Graph of first derivative of pixel intensity as a function of t
First Derivative of Pixel intensity as a function of t (Source)
The above plot demonstrates that edges can be detected in areas where the gradient is higher than a particular threshold value. In addition, a sudden change in the derivative will reveal a change in the pixel intensity as well. With this in mind, we can approximate the derivative, using a 3×3 kernel. We use one kernel to detect sudden changes in pixel intensity in the X direction, and another in the Y direction.

These are the kernels used for Sobel Edge Detection:

(1)
$$\begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$
X-Direction Kernel

(2)
$$\begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$
Y-Direction Kernel

When these kernels are convolved with the original image, you get a 'Sobel edge image'.

If we use only the Vertical Kernel, the convolution yields a Sobel image, with edges enhanced in the X-direction
Using the Horizontal Kernel yields a Sobel image, with edges enhanced in the Y-direction.
Let G_x and G_y represent the intensity gradient in the x and y directions respectively. If A and B denote the X and Y kernels defined above:

(3)
$$G_x = A * I$$
(4)
$$G_y = B * I$$
where * denotes the convolution operator, and I represents the input image.

The final approximation of the gradient magnitude, G can be computed as:

(5)
$$G = \sqrt{G_x^2 * G_y^2}$$
And the orientation of the gradient can then be approximated as:

(6)
$$\Theta = arctan(G_y / G_x)$$
In the code example below, we use the Sobel() function to compute:

the Sobel edge image individually, in both directions (x and y),
the composite gradient in both directions (xy)
The following is the syntax for applying Sobel edge detection using OpenCV:

Sobel(src, ddepth, dx, dy)

The parameter ddepth specifies the precision of the output image, while dx and dy specify the order of the derivative in each direction. For example:

 If dx=1 and dy=0, we compute the 1st derivative Sobel image in the x-direction.
 If both dx=1 and dy=1, we compute the 1st derivative Sobel image in both directions

Edge detection involves mathematical methods to find points in an image where the brightness of pixel intensities changes distinctly.

The first thing we are going to do is find the gradient of the grayscale image, allowing us to find edge-like regions in the x and y direction. The gradient is a multi-variable generalization of the derivative. While a derivative can be defined on functions of a single variable, for functions of several variables, the gradient takes its place.
The gradient is a vector-valued function, as opposed to a derivative, which is scalar-valued. Like the derivative, the gradient represents the slope of the tangent of the graph of the function. More precisely, the gradient points in the direction of the greatest rate of increase of the function, and its magnitude is the slope of the graph in that direction.

Morphological Transformation: -

Morphological transformations are some simple operations based on the image shape. It is normally performed on binary images. It needs two inputs, one is our original image, second one is called structuring element or kernel which decides the nature of operation. Two basic morphological operators are Erosion and Dilation. Then its variant forms like Opening, Closing, Gradient etc also comes into play. We will see them one-by-one with help of following image:

Erosion: -

The basic idea of erosion is just like soil erosion only, it erodes away the boundaries of foreground object (Always try to keep foreground in white). So what it does? The kernel slides through the image (as in 2D convolution). A pixel in the original image (either 1 or 0) will be considered 1 only if all the pixels under the kernel is 1, otherwise it is eroded (made to zero).

So what happends is that, all the pixels near boundary will be discarded depending upon the size of kernel. So the thickness or size of the foreground object decreases or simply white region decreases in the image. It is useful for removing small white noises (as we have seen in colorspace chapter), detach two connected objects etc.
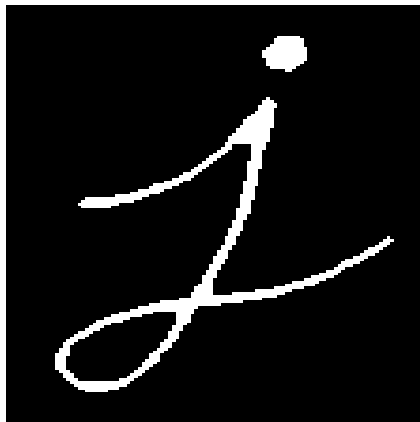
These image processing operations are applied to grayscale or binary images and are used for preprocessing for OCR algorithms, detecting barcodes, detecting license plates, and more.

And sometimes a clever use of morphological operations can allow you to avoid more complicated (and computationally expensive) machine learning and deep learning algorithms.

Morphological operations are simple transformations applied to binary or grayscale images. More specifically, we apply morphological operations to shapes and structures inside of images.

We can use morphological operations to increase the size of objects in images as well as decrease them. We can also utilize morphological operations to close gaps between objects as well as open them.

Morphological operations "probe" an image with a structuring element. This structuring element defines the neighborhood to be examined around each pixel. And based on the given operation and the size of the structuring element we are able to adjust our output image.

This explanation of a structuring element may sound vague — that's because it is. There are many different morphological transformations that perform "opposite" operations from one another — just as addition is the "opposite" of subtraction, we can think of the erosion morphological operation as the "opposite" of dilation.

Pytesseract: -

Python-tesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and "read" the text embedded in images.

Python-tesseract is a wrapper for Google's Tesseract-OCR Engine. It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Pillow and Leptonica imaging libraries, including jpeg, png, gif, bmp, tiff, and others. Additionally, if used as a script, Python-tesseract will print the recognized text instead of writing it to a file.

Functions: -

- get_languages Returns all currently supported languages by Tesseract OCR.
- get_tesseract_version Returns the Tesseract version installed in the system.
- image_to_string Returns unmodified output as string from Tesseract OCR processing
- image_to_boxes Returns result containing recognized characters and their box boundaries
- image_to_data Returns result containing box boundaries, confidences, and other information. Requires Tesseract 3.05+. For more information, please check the Tesseract TSV documentation
- image_to_osd Returns result containing information about orientation and script detection.
- image_to_alto_xml Returns result in the form of Tesseract's ALTO XML format.
- run_and_get_output Returns the raw output from Tesseract OCR. Gives a bit more control over the parameters that are sent to tesseract.

Parameters: -

- image Object or String - PIL Image/NumPy array or file path of the image to be processed by Tesseract. If you pass object instead of file path, pytesseract will implicitly convert the image to RGB mode.
- lang String - Tesseract language code string. Defaults to eng if not specified! Example for multiple languages: lang='eng+fra'
- config String - Any additional custom configuration flags that are not availablevia the pytesseract function. For example: config='--psm 6'
- nice Integer - modifies the processor priority for the Tesseract run. Not supported

on Windows. Nice adjusts the niceness of unix-like processes.

- output_type Class attribute - specifies the type of the output, defaults to string. For the full list of all supported types, please check the definition of pytesseract.Output class.
- timeout Integer or Float - duration in seconds for the OCR processing, after which, pytesseract will terminate and raise RuntimeError.
- pandas_config Dict - only for the Output.DATAFRAME type. Dictionary with custom arguments for pandas.read_csv. Allows you to customize the output of image_to_data.

Installation: -

- Python-tesseract requires Python 3.6+
- You will need the Python Imaging Library (PIL) (or the Pillow fork). Under Debian/Ubuntu, this is the package python-imaging or python3-imaging.
- Install Google Tesseract OCR (additional info how to install the engine on Linux, Mac OSX and Windows). You must be able to invoke the tesseract command as tesseract. If this isn't the case, for example because tesseract isn't in your PATH, you will have to change the "tesseract_cmd" variable
- pytesseract.pytesseract.tesseract_cmd. Under Debian/Ubuntu you can use the package tesseract-ocr. For Mac OS users. please install homebrew package tesseract.

Pytesseract or Python-tesseract is an Optical Character Recognition (OCR) tool for Python. It will read and recognize the text in images, license plates etc. Python-tesseract is actually a wrapper class or a package for Google's Tesseract-OCR Engine. It is also useful and regarded as a stand-alone invocation script to tesseract, as it can easily read all image types supported by the Pillow and Leptonica imaging libraries, which mainly includes –

- jpg
- png
- gif
- bmp
- tiff

1.3 System Design: -

Contrast Extention: -
To extend the contrast of an image means equalization of the histogram of that image will be used. In other words, the contract extension makes the image sharpen. The gray-level histogram of an image is the distribution of the gray level values in an image. The histogram equalization is a popular technique to improve the appearance of a poor contrasted image. The process of equalizing the histogram of an image consists of 4 steps: (i) Find the sum of the histogram values. (ii) Normalize these values dividing by the total number of pixels. (iii) Multiply these normalized values by the maximum gray-level value. (iv) Map the new gray level values.

Median Filtering: -
Median filter is used for eliminating the unwanted noisy regions. In this filtering method, the 3x3 matrices is passed around the image. The dimension of these matrices can be adjusted according to the noise level. The process of working is (i) one pixel is chosen as center pixel of the 3x3 matrices, (ii) the surrounding pixels are assigned as neighborhood pixels, (iii) the sorting process are employed between these nine pixels from smaller to the bigger, (iv) the fifth element is assigned as median element, (v) these procedures are implemented to the all pixels in plate image.

Character Segmentation: -
The characters of the identified number plate region are segmented using Region props function of MATLAB to obtain bounding boxes for each of the characters. Region props function returns the smallest bounding box that contains a character. This method is used to obtain the bounding boxes of all characters in the number plate.

The selected image is pre-processed by passing it over gray scale filter and edge detection is applied to isolate the region of interest, which is the number plate itself. A gray scale digital image is an image in which each pixel is quantized exclusively the shades of neutral gray, varying from black at the weakest intensity to white at the strongest intensity. The obtained gray image is then binarized, that is, it is converted to logical matrix by giving the pixel values of 1 for white shade and 0 for black shade.

1.4 OCR using Template Matching: -

Template matching is one of the Character Recognition techniques. It is the process of finding the location of a sub image called a template, inside an image. Template matching involves determining similarities between a given template and windows of the same size in an image and identifying the window that produces the highest similarity measure. It works by pixel-by-pixel comparison of the image and the template for each possible displacement of the template. This process involves the use of a database of characters or templates. There exists a template for all possible input characters. Templates are created for each of the alphanumeric characters (from A-Z and 0-9) using 'Regular' font style.

For recognition to occur, the current input character is compared to each template to find either an exact match, or the template with the closest representation of the input character. It can capture the best position where the character is by moving standard template, thereby carry out the exact match. Moving template matching method is based on the template of target character, using the template of standard character to match the target character from eight directions of up, down, left, right, upper left, lower left, upper right, lower right. The results of template matching for character recognition on some of the Indian number plates taken from static images are shown in Table.

Actual Plate	Predicted Plate	Mismatched characters	Accuracy
DL 4C AF 4943	DL 4C AF 4943	0	100%
KA 19 P 8488	KA 19 P U4UU	3	67%
MH 12 DE 1433	88 12 DE 1433	2	80%
WB 02 W 6886	X8 02 X 6886	2	80%

OCR (Optical Character Recognition): -

OCR = Optical Character Recognition. In other words, OCR systems transform a two-dimensional image of text, that could contain machine printed or handwritten text from its image representation into machine-readable text. OCR as a process generally consists of several sub-processes to perform as accurately as possible. The subprocesses are:

- Preprocessing of the Image
- Text Localization
- Character Segmentation
- Character Recognition
- Post Processing

The sub-processes in the list above of course can differ, but these are roughly steps needed to approach automatic character recognition. In OCR software, it's main aim to identify and capture all the unique words using different languages from written text characters.

For almost two decades, optical character recognition systems have been widely used to provide automated text entry into computerized systems. Yet in all this time, conventional OCR systems (like zonal OCR) have never overcome their inability to read more than a handful of type fonts and page formats. Proportionally spaced type (which includes virtually all typeset copy), laser printer fonts, and even many non-proportional typewriter fonts, have remained beyond the reach of these systems. And as a result, conventional OCR has never achieved more than a marginal impact on the total number of documents needing conversion into digital form.

Next-generation OCR engines deal with these problems mentioned above really good by utilizing the latest research in the area of deep learning. By leveraging the combination of deep models and huge datasets publicly available, models achieve state-of-the-art accuracies on given tasks. Nowadays it is also possible to generate synthetic data with different fonts using generative adversarial networks and few other generative approaches.

Optical Character Recognition remains a challenging problem when text occurs in unconstrained environments, like natural scenes, due to geometrical distortions, complex backgrounds, and diverse fonts. The technology still holds an immense potential due to the various use-cases of deep learning-based OCR like: -

- building license plate readers
- digitizing invoices
- digitizing menus
- digitizing ID cards

Open-Source OCR Tools: -

There are a lot of optical character recognition software available. I did not find any quality comparison between them, but I will write about some of them that seem to be the most developer-friendly.

Tesseract - an open-source OCR engine that has gained popularity among OCR developers. Even though it can be painful to implement and modify sometimes, there weren't too many free and powerful OCR alternatives on the market for the longest time. Tesseract began as a Ph.D. research project in HP Labs, Bristol. It gained popularity and was developed by HP between 1984 and 1994. In 2005 HP released Tesseract as an open-source software. Since 2006 it is developed by Google.

OCRopus - OCRopus is an open-source OCR system allowing easy evaluation and reuse of the OCR components by both researchers and companies. A collection of document analysis programs, not a turn-key OCR system. To apply it to your documents, you may need to do some image preprocessing, and possibly also train new models. In addition to the recognition scripts themselves, there are several scripts for ground truth editing and correction, measuring error rates, determining confusion matrices that are easy to use and edit.

Ocular - Ocular works best on documents printed using a hand press, including those written in multiple languages. It operates using the command line. It is a state-of-the-art historical OCR system. Its primary features are:

- Unsupervised learning of unknown fonts: requires only document images and a corpus of text.
- Ability to handle noisy documents: inconsistent inking, spacing, vertical alignment
- Support for multilingual documents, including those that have considerable word-level code-switching.
- Unsupervised learning of orthographic variation patterns including archaic spellings and printer shorthand.
- Simultaneous, joint transcription into both diplomatic (literal) and normalized forms.

SwiftOCR: - I will also mention the OCR engine written in Swift since there is huge development being made into advancing the use of the Swift as the development programming language used for deep learning. Check out blog to find out more why. SwiftOCR is a fast and simple OCR library that uses neural networks for image recognition. SwiftOCR claims that their engine outperforms well known Tesseract library.

Tesseract OCR: -

Tesseract is an open-source text recognition (OCR) Engine, available under the Apache 2.0 license. It can be used directly, or (for programmers) using an API to extract printed text from images. It supports a wide variety of languages. Tesseract doesn't have a built-in GUI, but there are several available from the 3rdParty page. Tesseract is compatible with many programming languages and frameworks through wrappers that can be found here. It can be used with the existing layout analysis to recognize text within a large document, or it can be used in conjunction with an external text detector to recognize text from an image of a single text line.

Tesseract 4.00 includes a new neural network subsystem configured as a text line recognizer. It has its origins in OCRopus' Python-based LSTM implementation but has been redesigned for Tesseract in C++. The neural network system in Tesseract pre-dates TensorFlow but is compatible with it, as there is a network description language called Variable Graph Specification Language (VGSL), that is also available for TensorFlow.

OCR Process Flow
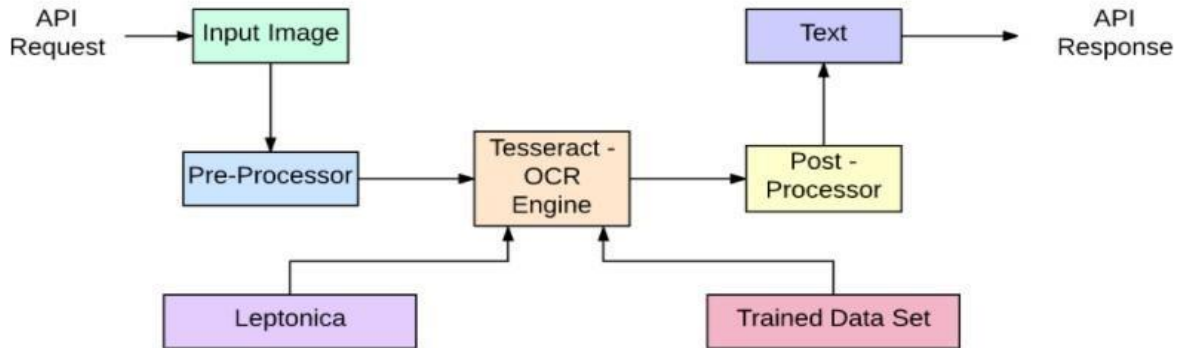
To recognize an image containing a single character, we typically use a Convolutional Neural Network (CNN). Text of arbitrary length is a sequence of characters, and such problems are solved using RNNs and LSTM is a popular form of RNN. Read this post to learn more about LSTM.

Connected components: -

Connected components labeling scans an image and groups its pixels into components based on pixel connectivity, i.e., all pixels in a connected component share similar pixel intensity values and are in some way connected with each other. Once all groups have been determined, each pixel is labeled with a gray level or a color (color labeling) according to the component it was assigned to. After the Localization of the number plate of the vehicle involved, we need to recognize the number plate into a standard form. The vehicular number plates maybe of Nonstandard forms and may vary in their fonts.

Process of acquisition: -
Image acquisition is the process of obtaining an image from the camera. This is the first step of any vision-based systems. In our current research we acquire the images using a digital camera placed by the road side facing towards the incoming vehicles .Here our aim is to get the frontal image of vehicles which contains license plate. The remaining stages of the system works in offline mode. Grayscale image: After acquiring the image, the very next step is to derive the gray scale image. Pseudo code to convert an image to a grayscale: STEP1 : Load the image STEP2 : Retrieve the properties of image like width, height and nchannels STEP3: Get the pointer to access image data STEP4: For each height and for each width of the image, convert image to grayscale by calculating average of r,g,b channels of the image convert to grayscale manually STEP5 : Display the image after converting to grayscale.

Pre- Processing: -

The pre-processing is the first step in number plate recognition. It consists the following major stages: 1.Binarization, 2.Noise Removal

• Binarization: The input image is initially processed to improve its quality and prepare it to next stages of the system. First, the system will convert RGB images to gray-level images.

• Noise Removal: In this noise removal stage we are going to remove the noise of the image i.e., while preserving the sharpness of the image. After the successful Localization of the Number Plate, we go on with Optical Character Recognition which involves the Segmentation, Feature extraction and Number plate Recognition.
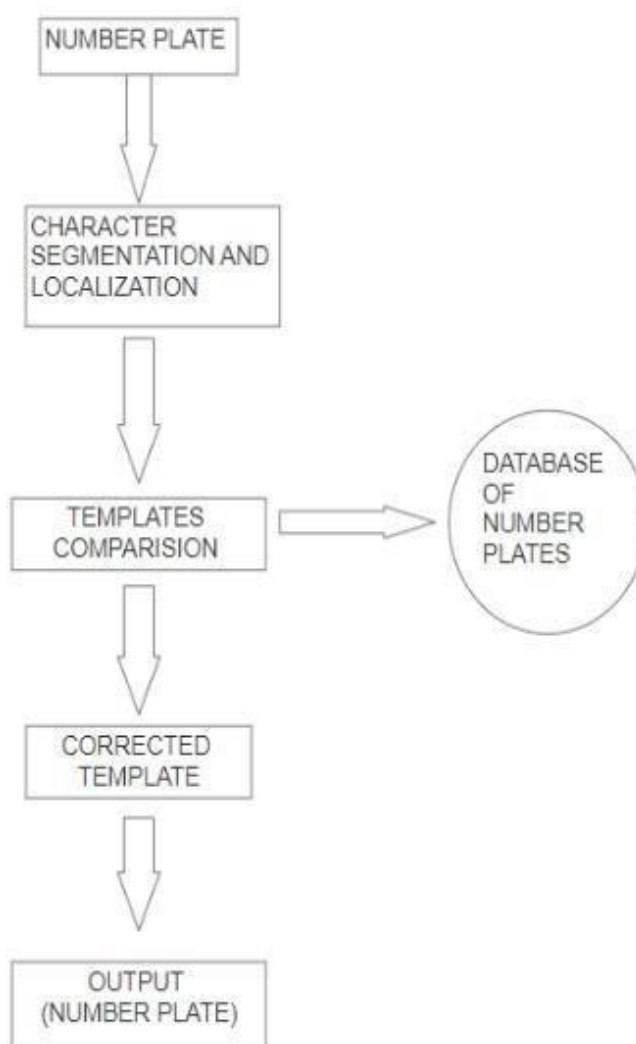
Character Segmentation : -

Segmentation is one of the most important processes in the automatic number plate recognition, because all further steps rely on it. If the segmentation fails, a character can be improperly divided into two pieces, or two characters can be improperly merged together. We can use a horizontal projection of a number plate for the segmentation, or one of the more sophisticated methods, such as segmentation using the neural networks. In this segmentation we use two types of segmentation: 1. Horizontal segmentation 2. Vertical segmentation. First we have performed vertical segmentation on the number plate then the characters are vertically segmented. After performing vertical segmentation we have to perform horizontal segmentation by doing this we get character from the plate.

Character Recognition: -

We have to recognize the characters we should perform feature extraction which is the basic concept to recognize the character. The feature extraction is a process of transformation of data from a bitmap representation into a form of descriptors, which are more suitable for computers. The recognition of character should be invariant towards the user font type, or deformations caused by a skew. In addition, all instances of the same character should have a similar description. A description of the character is a vector of numeral values, so called descriptors or patterns.

Block Diagram for the system design: -

```
        ┌──────────────────┐
        │   NUMBER PLATE   │
        └──────────────────┘
                  │
                  ▼
        ┌──────────────────┐
        │   CHARACTER      │
        │ SEGMENTATION AND │
        │  LOCALIZATION    │
        └──────────────────┘
                  │
                  ▼
        ┌──────────────────┐              ╭──────────╮
        │   TEMPLATES      │              │ DATABASE │
        │  COMPARISION     │ ───────────▶ │ OF       │
        └──────────────────┘              │ NUMBER   │
                  │                       │ PLATES   │
                  ▼                       ╰──────────╯
        ┌──────────────────┐
        │   CORRECTED      │
        │   TEMPLATE       │
        └──────────────────┘
                  │
                  ▼
        ┌──────────────────┐
        │    OUTPUT        │
        │ (NUMBER PLATE)   │
        └──────────────────┘
```

Template Creation: -

To make it happen, the current input character is compared to each template to get exactly the same, or a template with the closest image of the inserted character. It can capture the best place where the character is by moving the standard template, thus doing the exact same. Moving template matching method is based on a target character template, using a standard character template to match the target character from eight directions up, down, left, right, top left, bottom left, top right, bottom right. The effects of the alignment of the image recognition on some Indian number plates taken on still Images are considered.
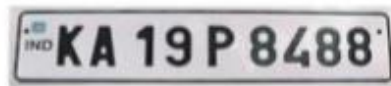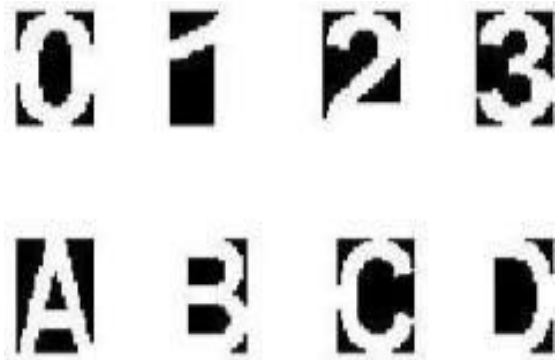
Character segmentation: -

This is the second major part of the License Plate detection algorithm. There are many factors that cause the character segmentation task difficult, such as image noise, plate frame, rivet, space mark, and plate rotation and illumination variance. We here propose the algorithm that is quite robust and gives significantly good results on images having the above-mentioned problems. The Steps involved in character Segmentation are:

• Preprocessing: Preprocessing is very important for the good performance of character segmentation. Our preprocessing consists of conversion to grayscale and binarization using a object enhancement technique. The steps involved are: Conversion to Grayscale, Binarization. Compared with the usual methods of image binarization, this algorithm uses the information of intensity and avoids the abruption and conglutination of characters that are the drawbacks of usual image binarization techniques.

• Object enhancement algorithm: The quality of plate images varies much in different capture conditions. Illumination variance and noise make it difficult for character segmentation. Then some image enhancement should be adopted to improve the quality of images. As we all know, the image enhancement methods of histogram equalization and gray level scaling have some side effects. They may have the noise enhanced as well. For character segmentation, only the character pixels need to be enhanced and the background pixels should be weakened at the same time. In fact, a license plate image contains about 20% character pixels. So these 20% character pixels need to be enhanced and the rest pixels need to be weakened. It is called object enhancement. The object enhancement algorithm consists of two steps: Firstly, gray level of all pixels is scaled into the range of 0 to 100 and compared with the original range 0 to 255, the character pixels and the background pixels are both weakened. Secondly, sorting all pixels by gray level in descending order and multiply the gray level of the top 20% pixels by 2.55. Then most characters pixels are enhanced while background pixels keep weakened. The following figure shows the result of object enhancement. It can be seen from the figure that after object enhancement the contrast of peaks and valleys of the projection is more significant than the original.

Horizontal Segmentation: -

For this we calculate the horizontal and vertical projections of intensity. Then we find the local minima for horizontal projection. Based on the threshold calculated from the above local minima"s, we find x locations of the segmented regions.

In order to locate the right and left edges of license plate from candidate region, the vertical projection after mathematical morphology deal is changed into binary image. The arithmetic for doing this is: F = Image between i & l where, fT (1, i) is the vertical projection after mathematical morphology, T is the threshold.

Then scan the function of fT (1, i) and register the potions where values change from 0 to 1 and from 1 to 0 in stack1 and stack2 respectively. So the candidate position of the left and right edge of the license plate are in stack1 (1,i) and stack2(1,i) respectively, and the candidate"s width of the license plate is calculated by:

Width(1,i)=Stack2(1,i)-Stack1(1,i)

Character Recognition: -

- Preprocessing: The image obtained after segmentation is Grayscale. Follow the preprocessing steps used for the training of the characters. Calculate the score for each of the characters: We calculate the matching score of the segmented character from the templates of the character stored by the following algorithm.

We compare the pixel values of the matrix of segmented character and the template matrix, and for every match we add 1 to the matching score and for every mis-match we decrement 1.

This is done for all 225 pixels. The match score is generated for every template and the one which gives the highest score is taken to be the recognized character. Character sets used for training the OCR: This is contained in a directory named "OCR_Training_Data".

Segmented Number Plate: -
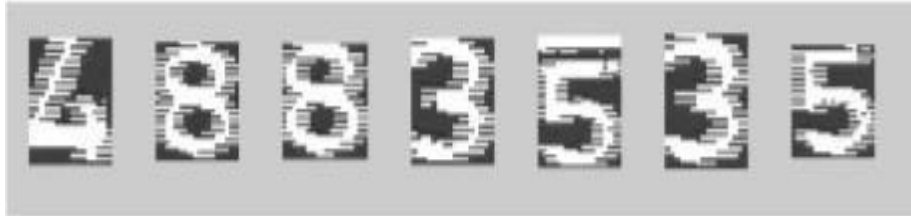
We see that the Tesseract OCR engine mostly predicts all of the license plates correctly with 100% accuracy. For the license plates, the Tesseract OCR Engine predicted incorrectly (i.e. GWT2180, OKV8004, JSQ1413), we will apply image processing techniques on those license plate files and pass them to the Tesseract OCR again. Applying the image processing techniques would increase the accuracy of the Tesseract Engine for the license plates of GWT2180, OKV8004, JSQ1413.

Actual License Plate	Predicted License Plate	Accuracy
OLA1208	OLA1208	100%
OYJ9557	OYJ9557	100%
PJG0783	PJG0783	100%
OUP9563	OUP9563	100%
OLC4728	OLC4728	100%
ODJ1599	ODJ1599	100%
GWT2180	GWT2120	86.0%
OKV8004	QKV8004	86.0%
PJB2414	PJB2414	100%
AYO9034	AYO9034	100%
JSQ1413	JSQ|413	86.0%
OKS0078	OKS0078	100%
NTK5785	NTK5785	100%
PJD2685	PJD2685	100%
NZW2197	NZW2197	100%
PJB7392	PJB7392	100%
NYY1710	NYY1710	100%
OCX4764	OCX4764	100%

Several compounding factors make ANPR incredibly challenging, including finding a dataset you can use to train a custom ANPR model! Large, robust ANPR datasets that are used to train state-of-the-art models are closely guarded and rarely (if ever) released publicly:

These datasets contain sensitive identifying information related to the vehicle, driver, and location.

ANPR datasets are tedious to curate, requiring an incredible investment of time and staff hours to annotate.

ANPR contracts with local and federal governments tend to be highly competitive. Because of that, it's often not the trained model that is valuable, but instead the dataset that a given company has curated.

Real Time ALNPR System interface image: -

Automatic License Number Plate Recognition Systems are available in all shapes and sizes:

ANPR executed in measured lighting situations with predictable number plate types can utilize basic techniques for image processing.

More advanced ANPR systems use dedicated object detectors, like HOG + Linear SVM, SSDs, YOLO, and Faster R-CNN to localize license number plates in images.

State-of-the-art ANPR software uses Recurrent Neural Networks (RNNs) and Long Short-Term Memory networks (LSTMs) in order to aid in better OCRing of the text from the number plates themselves.

Even more advanced ANPR systems utilize specialized neural network architectures in order to preprocess and clean images before they are OCRed, thereby developing the accuracy of ANPR.

For instance, let us consider an ANPR system that is mounted on a toll road. It has to be able to detect the number plate of each vehicle passing by, OCR the characters on the plate, and then store this data in a database so the vehicle's owner can be billed for the toll.

Few compounding factors make ANPR extremely challenging, involving finding a set of data we can utilize in order to train a custom model for ANPR. Large, robust datasets of ANPR that are utilized to train state-of-the-art models are tightly guarded and hardly (if ever) released publicly:

These datasets consist of sensitive identifying details associated with the vehicle, driver, and location.

The datasets of ANPR are tedious to curate, needing an unbelievable time investment and staff hours to interpret.

The contracts of ANPR with local and federal governments tend to be extremely reasonable.

It is often not the trained model that is valuable; however, instead of the dataset that a specified company has curated.

For the same cause, we will observe ANPR industries acquired not for their ANPR system but for the data itself.

Features of OpenCV:-

In the following Python project, we will utilize the following features of OpenCV in order to identify the number plate in the input image:

Gaussian Blur: Here, we will use a Gaussian Kernel for image smoothening. This technique is quite efficient for the removal of Gaussian noise. OpenCV offers a function called the GaussianBlur() function for this task.

Morphological Transformation: These are the operations grounded on image shapes and are processed on binary images. The fundamental morphological operations include Opening, Closing, Erosion, Dilation and many more. Some of the functions offered in OpenCV are as follows:

cv2.erode()
cv2.dilate()
cv2.morphologyEx()

Sobel: Here, we will calculate the derivatives from the image. This feature is significant for various tasks based on the vision of the computer. With the help of these derivatives, we can calculate the gradients and a higher alteration in gradient denotes a noteworthy change in the image.

OpenCV offers a Sobel() function in order to calculate Sobel operators.

Contours: Contours are the curves that consist of all the continuous points of the same intensity. These curves are quite useful utilities for the recognition of the object.

OpenCV offers the findContours() function for this feature.

In the above snippet of code, we have imported the image module from the matplotlib library and used the for-loop to extract the image from the designated folder. We have then used the imread() function to read the extracted image. We have then used the plot module of the matplotlib library to display the image for the users.

Image Resizing: We can resize the image file by a factor of 2x in both the horizontal and vertical directions with the help of resize.

Converting to Gray-scale: Then, we can convert the resized image file to grayscale in order to optimize the detection and reduce the number of colours available in the image drastically, which will allow us to detect the number plates easily.

Denoising the Image: We can use the Gaussian Blur technique to denoise the images. It makes the edges of the image clearer and smoother, making the characters more readable.

```
Original Number Plate      Predicted Number Plate        Accuracy

------------------         ----------------------        --------

DL3CAM0857                 DL3CAM0857                     100 %

MD06NYW                    MDOGNNS                        40 %

TN21TC706                  TN21TC706                      100 %

TN63DB5481                 TN63DB5481                     100 %

UP14DR4070                 UP14DR4070                     100 %

W5KHN                      WSKHN                          80 %
```

CHAPTER-2
Literature Survey

Computer viewing and character recognition, licensed plate recognition techniques play an important role in digital video image analysis. They therefore built the basic modules in any ALNPR system.

The car license plate recognition system includes a camera, frame, computer, and software designed to perform image processing, analysis and recognition. Car identification has been an effective study for the past few years. Much research has been done to identify the type of vehicle such as a car, truck, motorcycle or motorcycle.

Car identification has been an effective study in the past a few years ago. Much research has been done identifying the type of vehicle such as car, truck, scooter or motorcycle.

In **[3] A Roy and D.P Ghoshal**, a Soble filter was used to fix the problem to find the edges of the used car see car type.

Contourlet Transform and Vector Support Machine (SVM) was used in **[5] Anton Satria Prabuwono and Ariff Idris** detection car model. Show the numerical results to data set of about 70 images.

However, they did not use it how to stream video in real time. **[7] Jianbin Jiao, Qixiang Ye, and Qingming Huang** in monocular images are used for vehicle recognition. Use a canny edge acquisition to detect the presence of a vehicle and SVM in order know the car.

In **[9] Ying Wen**, Maximum Average Correlation Length Filter (MACH) and Log r-theta Mapping techniques was used to identify the type of car no matter what it was scale and fluctuations of vehicles.

The MACH filter was used for targeting detection in a crowded area. In **[8] Zhigang Zhang and Cong Wang**, MACH is used to filter targeted perceptions in orientations consistency and use a log r-theta map to create flight rotation and scale rotation while visualizing.

CHAPTER-3
Proposed System

The proposed plan focuses on two main solutions common problems in educational institutions, viz keeping track of the number and type of vehicles currently in existence foundation while also helping owners with their specific time the car had left the scene in the event of a robbery. The program contains of two main components, 1) Video capture source. 2) Application built using MATLAB.

The assumptions for the proposed system are: 1) Single Entry gate 2) Single Exit gate 3) Single Lane traffic, further improving to double lane traffic. The video capturing source is placed at the entry of Institute which is capable of providing live feed of certain accuracy even in poor lighting. The video source is placed at a position which can capture the entire vehicle and the required frames consisting of a vehicle are consequently selected.

The usual pattern matching pattern is an easy way to make recognition of a single font and a character of a fixed size, viz is an appropriate form of ALNPR systems. Wrong characters that differ from the classification of characters,
when the characters are not in the expected position or a few of them missed, can affect OCR recognition. Neural networks and mathematical dividers, which give a better result compare the usual pattern matching process, can overcame this problem because of their strong memory too the ability to adapt.

However, in order to achieve good performance, large number of samples and neurons are needed to obtain the neural networks. MATLAB has been used for the implementation of the algorithm on a PC equipped with a Dual Core 2.4GHz and 3G RAM. It has also been used to generate the weights of neural network. 6436 binary images with varying resolutions from the previous character segmentation stage were used. First of all, the binary images of the characters are resized to the same size. To choose the right size, several sizes of input images have been used for neural network training. High recognition rates can be achieved by using large character images but this will result in a more complex structure of the neural network as the number of weights will increase. The size corresponding to the best suitable result is used for the final neural network. Each system proposed for vehicle identification and number plate recognition in the literature survey has its own pros and cons.

CHAPTER-4
Result

In this work, existing methods and proposed algorithms in the books Vehicle recognition and Number Plate was updated.

Due to the unavailability of the ALNPR system without a shelf according to our needs, it is our effort customize the ALNPR educational institution program.

Matching template was used on number plates found in still images with anaverage accuracy of 82.7% was found.

This accuracy can be greatly improved positioning the camera properly to capture the best frame again using two layers of neural networks.

Implementation of the proposed system may be extended for approval multi-vehicle numbers in a single image frame with using multi-level genetic algorithms.

Also, a more complex version of this program can be used to take live video feed input once to choose the best car frame for car classification types and visual identificationof numbers using neural networks.

This system presents a recognition method in which the vehicle plate image is obtained by the digital cameras and the image is processed to get the number plate information. A rear image of a vehicle is captured and processed using various algorithms. Further we are planning to study about the characteristics involved with the automatic number plate system for better performance.

References

[1] **Xiaojun Zhai, Faycal Bensaali**, "StandardDefinition ANPR System on FPGA and an Approach to Extend it to HD" in 2013 IEEE GCC Conference and exhibition, November 17-20, Doha, Qatar. pp.214

[2] **H. Erdinc Kocer and K. Kursat Cevik**, "Artificial neural networks based vehiclelicense plate recognition," Procedia Computer Science, vol. 3, pp. 1033-1037, 2011

[3] **A Roy and D.P Ghoshal**, "Number Plate Recognition foruse in different countries using an improved segmentation," in 2nd National Conference on Emerging Trends and Applications in Computer Science(NCETACS), 2011, pp. 1-5

[4] **Fikriye Öztürk and Figens Özen**, "A New License PlateRecognition System Based on Probabilistic NeuralNetworks," Procedia Technology, vol. 1, pp. 124-128,2012

[5] **Anton Satria Prabuwono and Ariff Idris**, "A Study of CarPark Control System Using Optical Character Recognition," in International Conference on Computer and Electrical Engineering, 2008, pp. 866-870

[6] **Ch. Jaya Lakshmi, Dr. A. Jhansi Rani, Dr. K. Sri Ramakrishna, and M. Kanti Kiran**, "A Novel Approachfor Indian License Recognition System," InternationalJournal of Advanced EngineeringSciences and Technologies, vol. 6, no. 1, pp. 10-14,2011

[7] **Jianbin Jiao, Qixiang Ye, and Qingming Huang**, "A configurable method for multi-style license

platerecognition," Pattern Recognition, vol. 42, no. 3, pp. 358-369, 2009

[8] Zhigang Zhang and Cong Wang, "The Research of Vehicle Plate Recognition Technical Based on BP Neural Network," AASRI Procedia, vol. 1, pp. 74-81, 2012

[9] Ying Wen, "An Algorithm for License Plate recognition Applied to IntelligentTransportation System", IEEE Transactions of Intelligent Transportation Systems. pp. 1-16, 2011

[10] Chirag Patel, Dipti Shah, Atul Patel," Automatic Number Plate Recognition System (ANPR): A Survey", International Journal of Computer Applications, 2013

[11] SaimaRafique, Mahboob Iqbal and Hafiz Adnan Habib, "Space Invariant Vehicle Recognition for Toll Plaza Monitoring and Auditing System", Multitopic Conference,2009. INMIC 2009,IEEE 13th International, pp. 1-6

[12] https://vectorified.com/images/ocr-icon-14.png

[13] http://www.team-bhp.com/forum/attachments/indian-car-scene/69590d1226627898-high-security-registration-plates-hsrp-india-d1.jpg

[14] Fajas F., Farhan Yousuf, Remya P. R., Adarsh P. Pavanan, Sajan Ambadiyil and Varsha Swaminathan, "Automatic Number Plate Recognition for Indian Standard Number Plates", Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2012 4th International Congress, pp. 1026-1028

[15] X. Pan, X. Ye and S. Zhang, "A hybrid method for robust car plate character recognition," presented at the IEEE International Conference on Systems, Man and Cybernetics, 2004

[16] C. Anantha Reddy, C. Shoba Bindu, "Multi-Level Genetic Algorithm for Recognizing Multiple License Plates in a Single Image", Journal of Innovation in Computer Science and Engineering, 2015

[17] Anand Sumatilal Jain, Jayshree M. Kundargi, "Automatic Number Plate Recognition Using Artificial Neural Network", International Research Journal of Engineering and Technology (IRJET), 2015

[18] Weihua Wang, "Reach on Sobel Operator for Vehicle Recognition, "International Joint Conference on Artificial Intelligence, pp.448-451, 2009

[19] Saeid Rahati, Reihaneh Morvejian, Ehsan M. Kazemi and Farhad M. Kazem "Vehicle Recognition Using Contourlet Transform and SVM," Proceedings of the Fifth International Conference on Information Technology, 2008

[20] M.A. Sotelo , J. Nuevo , L.M. Bergasa and M. Ocana, "Road Vehicle Recognition in Monocular Images," IEEE Symposium on Industrial Electronics, 2005

[21] Bone P, Young R, Chatwin C. "Position, rotation, scale, and orientation-invariant multiple object recognition from cluttered scenes," Opt Eng2006; 45:077203

[22] Xiaojun Zhai, Faycal Bensaali and Reza Sotudeh, "OCR- Based Neural Network for ANPR" in IEEE, 2012. Pp1

[23] Y. Amit, D. Geman, and X. Fan, "A coarse-to-fine strategy for multiclass shape detection," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, pp. 1606-1621, 2004

[24] C. Oz, and F. Ercal, "A Practical License Plate Recognition System for Real-Time Environments. Computational Intelligence and Bio-inspired System," Lecture Notes in Computer Science, vol. 3512/2005, pp.497-538, 2005

[25] S.L. Chang, L.S. Chen, Y.C. Chung and S.W. Chen, "Automatic license plate recognition," IEEE Transactions on Intelligent Transportation Systems, vol. 5, pp. 42-53, 2004

[26] Y. Wen, Y. Lu, J. Yan, Z. Zhou, von Deneen K.M. and P. Shi, "An Algorithm for License Plate Recognition Applied to Intelligent Transportation System," IEEE