CHAPTER-1

INTRODUCTION

1.1 IMAGE PROCESSING

Image processing is a method to convert an image into digital form and perform some operations on it, in order to get an enhanced image or to extract some useful information from it. It is a type of signal dispensation in which input is image, like video frame or photograph and output may be image or characteristics associated with that image. Usually Image Processing system includes treating images as two dimensional signals while applying already set signal processing methods to them.

1.1.1 PURPOSE OF IMAGE PROCESSING

The purpose of image processing is divided into 5 groups. They are:

1. Visualization - Observe the objects that are not visible.
2. Image sharpening and restoration - To create a better image.
3. Image retrieval - Seek for the image of interest.
4. Measurement of pattern – Measures various objects in an image.
5. Image Recognition – Distinguish the objects in an image.

1.2 CLUSTERING

Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters).

TYPES OF CLUSTERING

Broadly speaking, clustering can be divided into two sub groups:

Hard Clustering: In hard clustering, each data point either belongs to a cluster completely or not. For example, in the above example each customer is put into one group out of the 10 groups.

Soft Clustering: In soft clustering, instead of putting each data point into a separate cluster, a probability or likelihood of that data point to be in those clusters is assigned. For example, from the above scenario each costumer is assigned a probability to be in either of 10 clusters of the retail store.

1.2.1 Support Vector Clustering

In SVC data points are mapped from data space to a high dimensional feature space using a kernel function. In the kernel's feature space the algorithm searches for the smallest sphere that encloses the image of the data using the Support Vector Domain Description algorithm. This sphere, when mapped back to data space, forms a set of contours which enclose the data points. Those contours are then interpreted as cluster boundaries, and points enclosed by each contour are associated by SVC to the same cluster.

SVC uses the Support Vector Domain Description (SVDD) to delineate the region in data space where the input examples are concentrated. SVDD belongs to the general category of kernel based learning. In its "linear" version SVDD looks for the smallest sphere that encloses the data. When used in conjunction with a kernel function, it looks for the smallest enclosing sphere in the feature space defined by the kernel function.

SVC is a nonparametric clustering algorithm that does not make any assumption on the number or shape of the clusters in the data. In our experience it works best for low-dimensional data, so if your data is high-dimensional, a preprocessing step, e.g. using principal component analysis, is usually required. Several enhancements of the original algorithm were proposed that provide specialized algorithms for computing the clusters by only computing a subset of the edges in the adjacency matrix.

1.2.2 K-MEANS CLUSTERING

K-means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining.

K-means clustering aims to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells.

The problem is computationally difficult (NP-hard); however, efficient heuristic algorithm converge quickly to a local optimum. These are usually similar to the expectation maximization algorithm for mixtures of Gaussian distribution via an iterative refinement approach employed by both k-means and Gaussian mixture modelling. They both use cluster centre to model the data; however, k-means clustering tends to find clusters of comparable spatial extent, while the expectation-maximization mechanism allows clusters to have different shapes.

The algorithm has a loose relationship to the k-nearest neighbour classifier, a popular machine learning technique for classification that is often confused with *k*-means due to the name. Applying the 1-nearest neighbour classifier to the cluster centers obtained by *k*-means classifies new data into the existing clusters. This is known as nearest centroid classifier or Rocchio algorithm.

1.2.3 HIERARCHICAL CLUSTERING

Hierarchical clustering, as the name suggests is an algorithm that builds hierarchy of clusters. This algorithm starts with all the data points assigned to a cluster of their own. Then two nearest clusters are merged into the same cluster. In the end, this algorithm terminates when there is only a single cluster left.

The results of hierarchical clustering can be shown using dendrogram. The dendrogram can be interpreted as:

At the bottom, we start with 25 data points, each assigned to separate clusters. Two closest clusters are then merged till we have just one cluster at the top. The height in the dendrogram at which two clusters are merged represents the distance between two clusters in the data space.

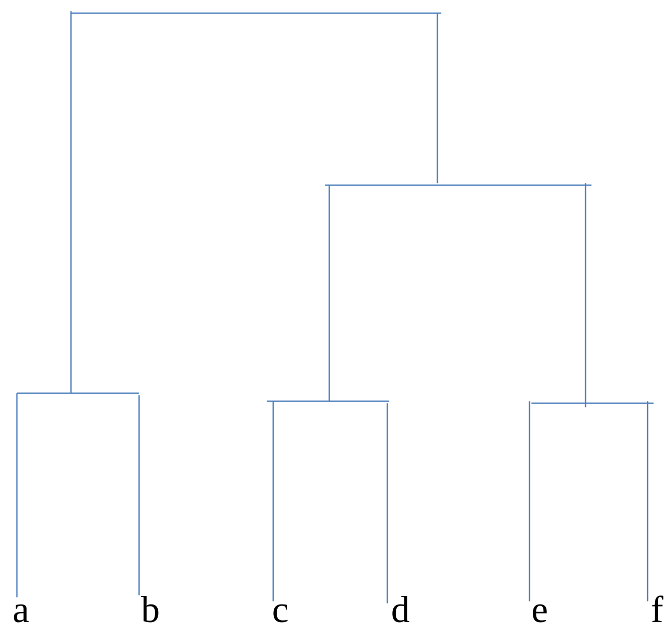


Fig 1.2.1 dendrogram

This fig 1.2.1 describes the decision of the no. of clusters that can best depict different groups can be chosen by observing the dendrogram. The best choice of the no. of clusters is the no. of vertical lines in the dendrogram cut by a horizontal line that can transverse the maximum distance vertically without intersecting a cluster.

In the above example, the best choice of no. of clusters will be 4 as the red horizontal line in the dendrogram below covers maximum vertical distance AB.

1.3 EDGE DETECTION

1.3.1 SOBEL OPERATOR

A very common operator for doing this is a Sobel Operator, which is an approximation to a derivative of an image. It is separate in the y and x directions. If we look at the x-direction, the gradient of an image in the x-direction is equal to this operator here. We use a kernel 3 by 3 matrix, one for each x and y direction. The gradient for x-direction has minus numbers on the left hand side and positive numbers on the right hand side and we are preserving a little bit of the center pixels. Similarly, the gradient for y-direction has minus numbers on the bottom and positive numbers on top and here we are preserving a little bit on the middle row pixels.

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Fig 1.3.1 kernel of sobel operator

Essentially what we are trying to do here with the Sobel Operator is trying to find out the amount of the difference by placing the gradient matrix over each pixel of our image. We get two images as output, one for X- Direction and other for Y-Direction. By using Kernel Convolution, we can see in the example image below there is an edge between the column of 100 and 200 values.

100	100	200	200
100	100	200	200
100	100	200	200
100	100	200	200

-1	0	1
-2	0	2
-1	0	1

fig 1.3.2 size of the original image

fig 1.3.3 size of the kernel

-100
-200
-100
+200
+400
+200
=400

fig 1.3.4 result of multiplication of original image and kernel

The above example shows the result of doing convolution by placing the Gradient matrix X over a red marked 100 of images. The calculation is shown on the right which sums up to 400, which is non-zero, hence there is an edge. If all the pixels of images were of the same value, then the convolution would result in a resultant sum of zero. So the gradient matrix will provide a big response when one side is brighter. Another point to note here is that the sign of the output resultant does not matter.

1.3.2 CANNY EDGE DETECTION

Canny edge detection is a technique to extract useful structural information from different vision objects and dramatically reduce the amount of data to be processed. It has been widely applied in various computer vision systems. Canny has found that the requirements for the application of edge detection on diverse vision systems are relatively similar. Thus, an edge detection solution to address these requirements can be implemented in a wide range of situations. The general criteria for edge detection include:

1. Detection of edge with low error rate, which means that the detection should accurately catch as many edges shown in the image as possible
2. The edge point detected from the operator should accurately localize on the center of the edge.
3. A given edge in the image should only be marked once, and where possible, image noise should not create false edges.

To satisfy these requirements Canny used the calculus of variations – a technique which finds the function which optimizes a given functional. The optimal function in Canny's detector is described by the sum of four exponential terms, but it can be approximated by the first derivative of a Gaussian.

Among the edge detection methods developed so far, Canny edge detection algorithm is one of the most strictly defined methods that provides good and reliable detection. Owing to its optimality to meet with the three criteria for edge detection and the simplicity of process for implementation, it became one of the most popular algorithms for edge detection.

1.3.3 PREWITT OPERATOR

Prewitt operator is used for edge detection in an image. It detects two types of edges

- Horizontal edges
- Vertical Edges

Edges are calculated by using difference between corresponding pixel intensities of an image. All the masks that are used for edge detection are also known as derivative masks. Because as we have stated many times before in this series of tutorials that image is also a signal so changes in a signal can only be calculated using differentiation. So that’s why these operators are also called as derivative operators or derivative masks.

All the derivative masks should have the following properties:

- Opposite sign should be present in the mask.
- Sum of mask should be equal to zero.
- More weight means more edge detection.

VERTICAL DIRECTION

-1	0	1
-1	0	1
-1	0	1

Table 1.3.1

Above mask will find the edges in vertical direction and it is because the zeros column in the vertical direction. When you will convolve this mask on an image, it will give you the vertical edges in an image.

HORIZONTAL DIRECTION

-1	-1	-1
0	0	0
1	1	1

Table 1.3.2

Above mask will find edges in horizontal direction and it is because that zeros column is in horizontal direction. When you will convolve this mask onto an image it would prominent horizontal edges in the image.

1.4 GAUSSIAN FILTER

Gaussian filtering is used to blur images and remove noise and detail. In one dimension, the Gaussian function is:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma}} e - \frac{x^2}{2\sigma}$$

Where σ is the standard deviation of the distribution. The distribution is assumed to have a mean of 0.

CHAPTER 2
LITERATURE SURVEY

2.1 License number plate recognition system using entropy-based features selection approach with SVM

Author: Muhammd Attique Khan, Muhammed Sharif, Muhammed Younus Javed, Tallha Akram.

The authors says that a License plate recognition (LPR) system plays a vital role in security applications which include road traffic monitoring, street activity monitoring, identification of potential threats, and so on. Numerous methods were adopted for LPR but still, there is enough space for a single standard approach which can be able to deal with all sorts of problems such as light variations, occlusion, and multi-views. The proposed approach is an effort to deal under such conditions by incorporating multiple features extraction and fusion. The proposed architecture is comprised of four primary steps: (i) selection of luminance channel from CIE Lab colour space, (ii) binary segmentation of selected channel followed by image refinement, (iii) a fusion of Histogram of oriented gradients (HOG) and geometric features followed by a selection of appropriate features using a novel entropy-based method, and (iv) features classification with support vector machine (SVM). To authenticate the results of proposed approach, different performance measures are considered. The selected measures are False positive rate (FPR), False negative rate (FNR), and accuracy which is achieved maximum up to 99.5%. Simulation results reveal that the proposed method performs exceptionally better compared with existing works.

2.2 Eyes on the Target: Super-Resolution and License-Plate Recognition in Low-Quality surveillance Videos

Author: Hilario Seibel, Siome Goldenstein and Anderson Rocha.

Hilario Seibel, Siome Goldenstein and Anderson Rocha says that a Low-quality surveillance cameras throughout the cities could provide important cues to identify a suspect, for example, in a crime scene. However, the license-plate recognition is especially difficult under poor image resolutions. In this vein, super-resolution (SR) can be an inexpensive solution, via software, to overcome this limitation. Consecutive frames in a video may contain different information that could be integrated into a single image, richer in details. Authors design and develop a novel, free and open-source framework underpinned by SR and automatic license-plate recognition (ALPR) techniques to identify license-plate characters in low-quality real-world traffic videos, captured by cameras not designed specifically for the ALPR task, aiding forensic analysts in understanding an event of interest. The framework handles the necessary conditions to identify a target license plate, using a novel methodology to locate, track, align, super-resolve, and recognize its alphanumerics. The user receives as outputs the rectified and super resolved license-plate, richer in detail, and also the sequence of license-plates characters that have been automatically recognized in the super-resolved image. Additionally, we also design and develop a novel SR method that projects the license-plates separately onto the rectified grid, and then fill in the missing pixels using inpainting techniques. We Our experiments show that SR can indeed increase the number of correctly recognized characters posing the framework as an important step toward providing forensic experts and practitioners with a solution for the license-plate recognition problem under difficult acquisition conditions.

2.3 Deep Classifiers-Based License Plate Detection, Localization and Recognition on GPU-Powered Mobile Platform

Author: Syed Tahir Hussain Rizvi , Denis Patti.

Syed Tahir Hussain Rizvi, Denis Patti. Says that the realization of a deep neural architecture on a mobile platform is challenging, but can open up a number of possibilities for visual analysis applications. A neural network can be realized on a mobile platform by exploiting the computational power of the embedded GPU and simplifying the flow of a neural architecture trained on the desktop workstation or a GPU server. This paper presents an embedded platform-based Italian license plate detection and recognition system using deep neural classifiers. In this work, trained parameters of a highly precise automatic license plate recognition (ALPR) system are imported and used to replicate the same neural classifiers. A CUDA-based framework is used to realize these neural networks. The flow of the trained architecture is simplified to perform the license plate recognition in real-time. Results show that the tasks of plate and character detection and localization can be performed in real-time on a mobile platform by simplifying the flow of the trained architecture. However, the accuracy of the simplified architecture would be decreased.

2.4 Number Plate Recognition Using an Improved Segmentation

Author: Mr. G. T. Sutar, Mr. A.V. Shah.

The authors say that a NPR (Number Plate Recognition) using is a system designed to help in recognition of number plates of vehicles. This system is designed for the purpose of the security system. This system is based on the image processing system. This system helps in the functions like detection of the number plates of the vehicles, processing them and using processed data for further processes like storing, allowing vehicle to pass or to reject vehicle. NPR is an image processing technology which uses number (license) plate to identify the vehicle. The objective is to design an efficient automatic authorized vehicle identification system by using the vehicle number plate. The system is implemented on the entrance for security control of a highly restricted area like military zones or area around top government offices e.g. Parliament, Supreme Court etc. The developed system first captures the vehicle image. Vehicle number plate region is extracted using the image segmentation in an image. Optical character recognition technique is used for the character recognition. The resulting data is then used to compare with the records on a database. The system is implemented and simulated in Matlab, and it performance is tested on real image. It is observed from the experiment that the developed system successfully detects and recognize the vehicle number plate on real images.

2.5 Algorithm for License Plate Localization and Recognition for Tanzania Car Plate Numbers

Author: Isack Bulugu

The author says that a License plate lo calization and recognition (LPLR) is presented. It uses image processing and character recognition technology in order to identify the license number plates of the vehicles automatically. This system is considerable interest because of its good application in traffic monitoring systems, surveillance devices and all kind of intelligent transport system. The objective of this work is to design algorithm for License Plate Localization and Recognition (LPLR) of Tanzanian License Plates. The plate numbers used are standard ones with black and yellow or black and white colors. Also, the letters and numbers are placed in the same row (identical vertical levels), resulting in frequent changes in the horizontal intensity. Due to that, the horizontal changes of the intensity have been easily detected, since the rows that contain the number plates are expected to exhibit many sharp variations. Hence, the edge finding method is exploited to find the location of the plate. To increase readability of the plate number, part of the image was enhanced, noise removal and smoothing median filter is used due to easy development. The algorithm described in this paper is implemented using MATLAB.

2.6 Multi-task Convolutional Neural Network System for License Plate Recognition

Author: Hong-HyunKim, Je-KangPark, Joo-HeeOh and Dong-JoongKang.

The author says that a License plate recognition is an active research field as demands sharply increase with the development of Intelligent Transportation System (ITS). However, since the license plate recognition (LPR) is sensitive to the conditions of the surrounding environment such as a complicated background in the image, viewing angle and illumination change, it is still difficult to correctly recognize letters and digits on LPR. This study applies Deep Convolutional Neural Network (DCNN) to the license plate recognition. The DCNN is a method of which the performance has recently been proven to have an excellent generalization error rate in the field of image recognition. The proposed layer structure of the DCNN used in this study consists of a combination of a layer for judging the existenceofalicenseplateandalayerforrecognizingdigitsandcharacters. This learning method is based on Multi Task Learning (MTL). Through experiments using real images, this study shows that this layer structure classifies digits and characters more accurately than the DCN nusing a conventional layer does. Weal souse artificial images generated directly for training model. Keywords: Deep convolutional neural network, license plate recognition, machine learning, multi task learning.

CHAPTER 3
SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

- The Existing System of Number Plate detection and recognition techniques are implemented in MATLAB Programming Language.
- It cannot displays the Number Plate of Poor Quality Image and Night Captured pictures.
- Its uses the traditional method of character recognition techniques named super resolution method.
- Its convert the image Colour from RGB to HSV image, this image are low in quality.
- It removes less noise in the image uses mean blur filtering.
- It finds all Contours values using obtained threshold values by applying adaptive threshold to remove unimportant region in the image.
- It identifies only Boundary Box of the license plates.

3.2 PROPESED SYSTEM

Number Plate Recognition is an image processing technology which uses number plate to identify the vehicle. This objective is to design an efficient automatic authorized vehicle identification system by using the vehicle number plate. The system first captures the vehicles image. The vehicle image is preprocessed to enhanced the quality of image. Localization is the process of identifying number plate from the image captured, rear and front part of the vehicle is captured into an image other part of the vehicle are cropped. The number plate region is extracted using the image segmentation in an image. Optical character recognition OCR involves photo scanning of the text character by character, analysis of the scanned in image into character codes.

CHAPTER 4
SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

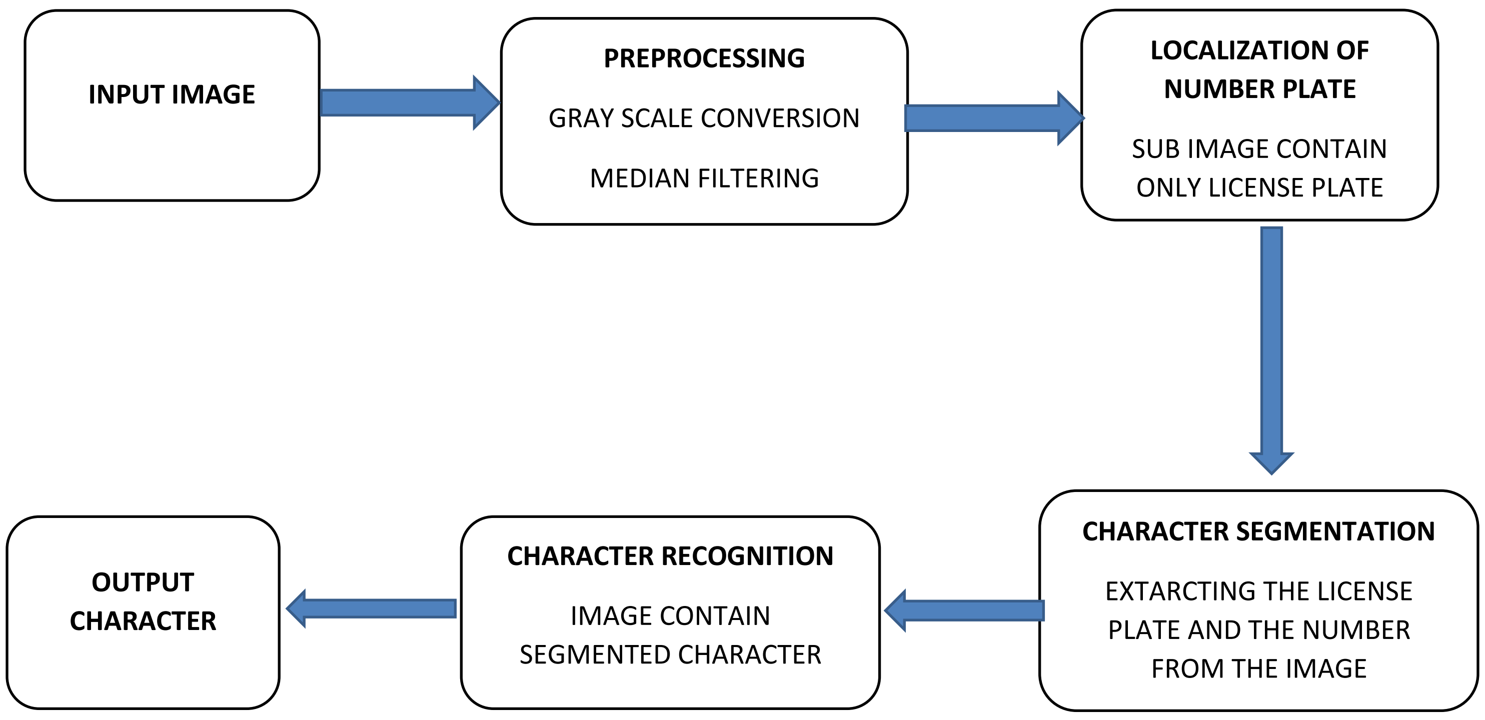
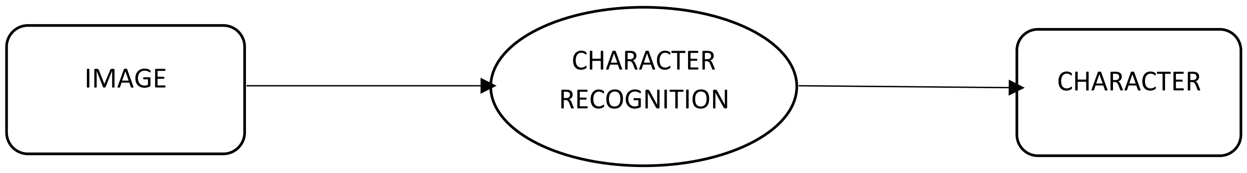

INPUT IMAGE
PREPROCESSING
GRAY SCALE CONVERSION
MEDIAN FILTERING
LOCALIZATION OF NUMBER PLATE
SUB IMAGE CONTAIN ONLY LICENSE PLATE
CHARACTER SEGMENTATION
EXTARCTING THE LICENSE PLATE AND THE NUMBER FROM THE IMAGE
CHARACTER RECOGNITION
IMAGE CONTAIN SEGMENTED CHARACTER
OUTPUT CHARACTER

Fig.2

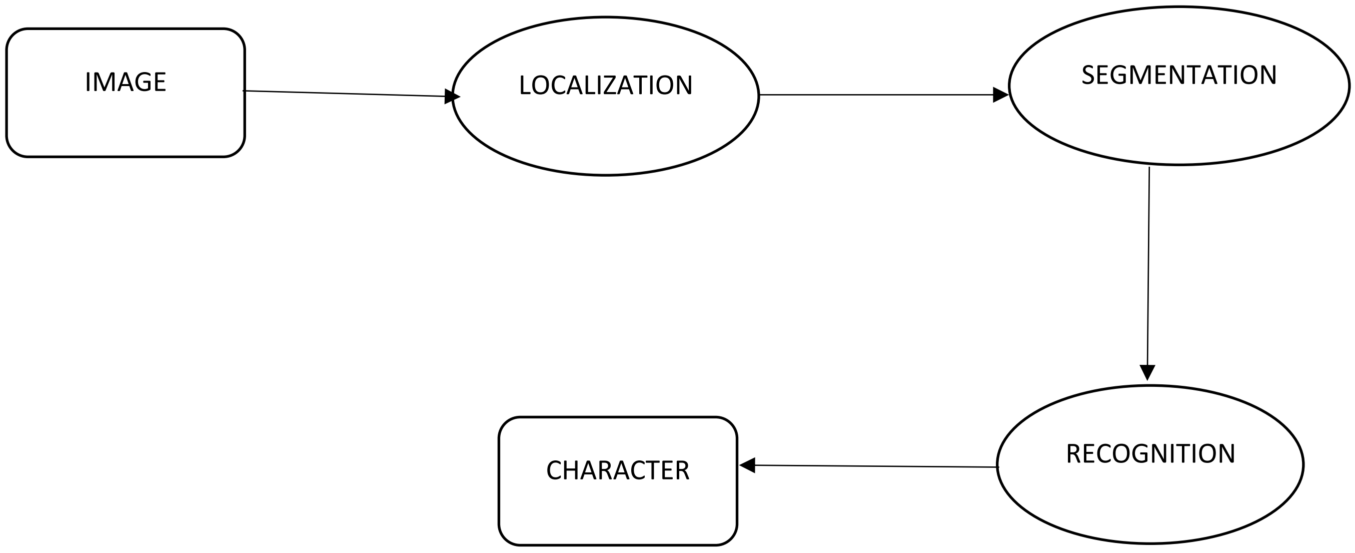
4.2 DATA FLOW DIAGRAM

A data-flow diagram (DFD) is a way of representing a flow of a data of a process or a system (usually an information system) The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow, there are no decision rules and no loops. Specific operations based on the data can be represented by a flowchart.

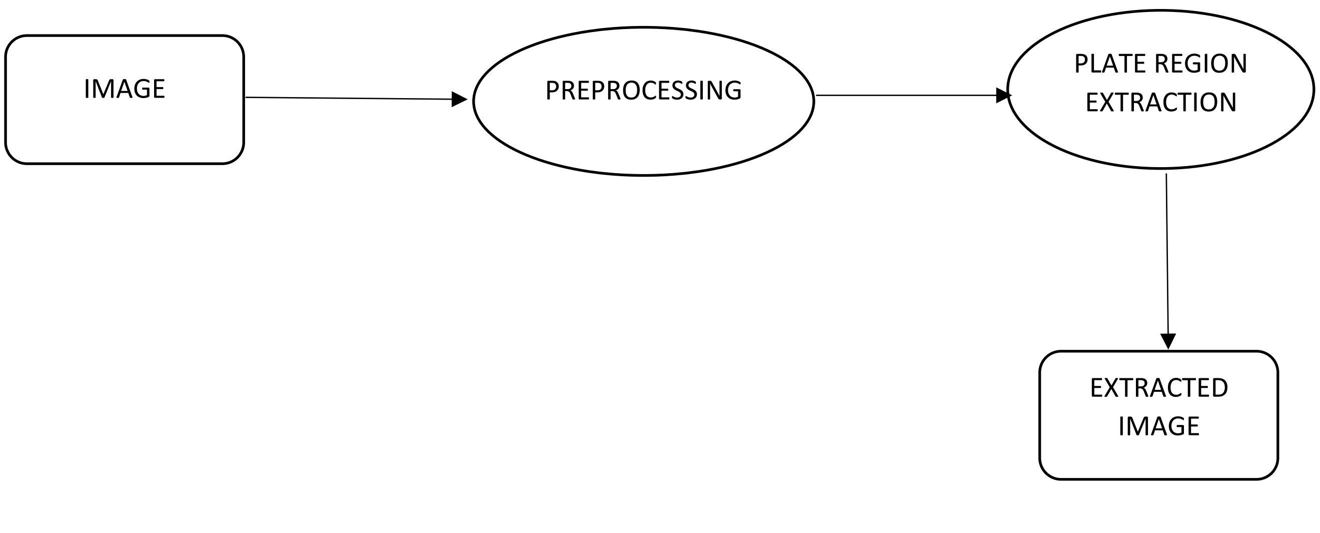
LEVEL 0:



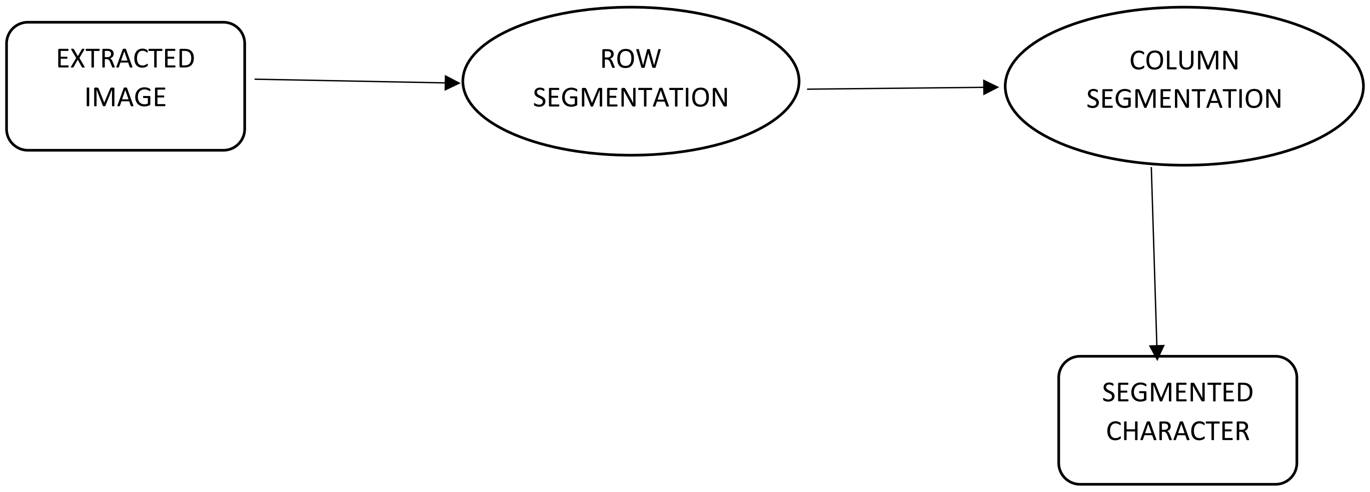
LEVEL 1:



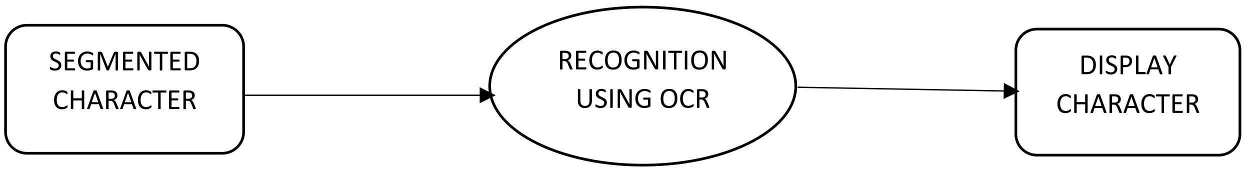
LEVEL 2:



LEVEL 3:



LEVEL 4:



4.3 MODULES DESCRIPTION

4.3.1 PREPROCESSING

Pre-processing improves the quality of image, have to be processed before the Number Plate localization algorithm are applied to increase the accuracy of Number Plate.

PROCESS:

- Read image
- Resize image

- Remove noise(Denoise)
- Segmentation
- Morphology(smoothing edges)

READ IMAGE

In this step, we store the path to our image dataset into a variable then we created a function to load folders containing images into arrays.

RESIZE IMAGE

In this step in order to visualize the change, we are going to create two functions to display the images the first being a one to display one image and the second for two images. After that, we then create a function called processing that just receives the images as a parameter.

REMOVE NOISE

Still, inside the function Processing () we add this code to smooth our image to remove unwanted noise. We do this using gaussian blur.

Gaussian blur (also known as Gaussian smoothing) is the result of blurring an image by a Gaussian function. It is a widely used effect in graphics software, typically to reduce image noise. The visual effect of this blurring technique is a smooth blur resembling that of viewing the image through a translucent screen, distinctly different from the bokeh effect produced by an out-of-focus lens or the shadow of an object under usual illumination. Gaussian smoothing is also used as a pre-processing stage in computer vision algorithms in order to enhance image structures at different scales.

SEGMENTATION AND MORPHOLOGY

In this step, we step we are going to segment the image, separating background from foreground objects and we are going to further improve our segmentation with more noise removal.

4.3.2 LOCALIZATION

Localization is the process of identifying Number Plate from the image captured rear and front part of the vehicle is captured into an image other part of the vehicles are cropped

TWO OPERATIONS

- Highlighting Characters
- Suppressing Background

Pre-processing techniques are used to convert the three scale image to binary image and it is used to highlight the characters of number plate and to suppress the background details

Localization using characteristics of Alphanumeric characters

Alphanumeric characters have specific properties in binary image such as size, white pixel density, atomicity etc. The steps of proposed approach for localization of license plate from an image are explained below. (A)Inverted 'L': Masking mask having size equal to maximum character size is used, having shape of inverted L. At each Position, these conditions are tested:

a) All pixels in first row and first column of a box should be black.

b) There should be at least a single white pixel on second row and second column of a box.

Input image of the number plate is pre-processed for the removal of background noise . The pre-processed image is binarized to make it suitable for localizing the number plate. Otsu's algorithm is employed to for fast binarization.

4.3.3 SEGMENTATION

Image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as super-pixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze.

Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain characteristics.

The result of image segmentation is a set of segments that collectively cover the entire image, or a set of contours extracted from the image (see edge detection). Each of the pixels in a region are similar with respect to some characteristic or computed property, such as color, intensity, or texture. Adjacent regions are significantly different with respect to the same characteristic(s).

THRESHOLDING

The simplest method of image segmentation is called the thresholding method. This method is based on a clip-level (or a threshold value) to turn a gray-scale image into a binary image. The key of this method is to select the threshold value (or values when multiple-levels are selected). Several popular methods are used in industry including the maximum entropy method, balanced histogram thresholding, Otsu's method (maximum variance), and k-means clustering.

4.3.4 RECOGNITION

The ability of an software to identify object, place, people, writing and actions in image. It use computer vision technology in combination with camera and Artificial Intelligence to achieve image recognition Automatic Number Plate

Recognition (ANPR) is the ability to automatically recognize the symbols contained in the number plates of a motor vehicle when read from an image provided by video surveillance cameras for the purposes of further processing by a security system.

Image recognition is the process of identifying and detecting an object or a feature in a digital image or video. This concept is used in many applications like systems for factory automation, toll booth monitoring, and security surveillance. Typical image recognition algorithms include: Optical character recognition.

Automatic number plate recognition (ANPR) is an image processing technology which uses number (license) plate to identify the vehicle. The objective is to design an efficient automatic authorized vehicle identification system by using the vehicle number plate. The system is implemented on the entrance for security control of a highly restricted area like military zones or area around top government offices e.g. Parliament, Supreme Court etc. The developed system first detects the vehicle and then captures the vehicle image. Vehicle number plate region is extracted using the image segmentation in an image. Optical character recognition technique is used for the character recognition.

OPTICAL CHARACTER RECOGNITION

It is the recognition of printed or written text character by a computer. This involves photo scanning of the text character by character analysis of the scanned in image and translation of the character image into character code such as ASCII commonly used in data processing.

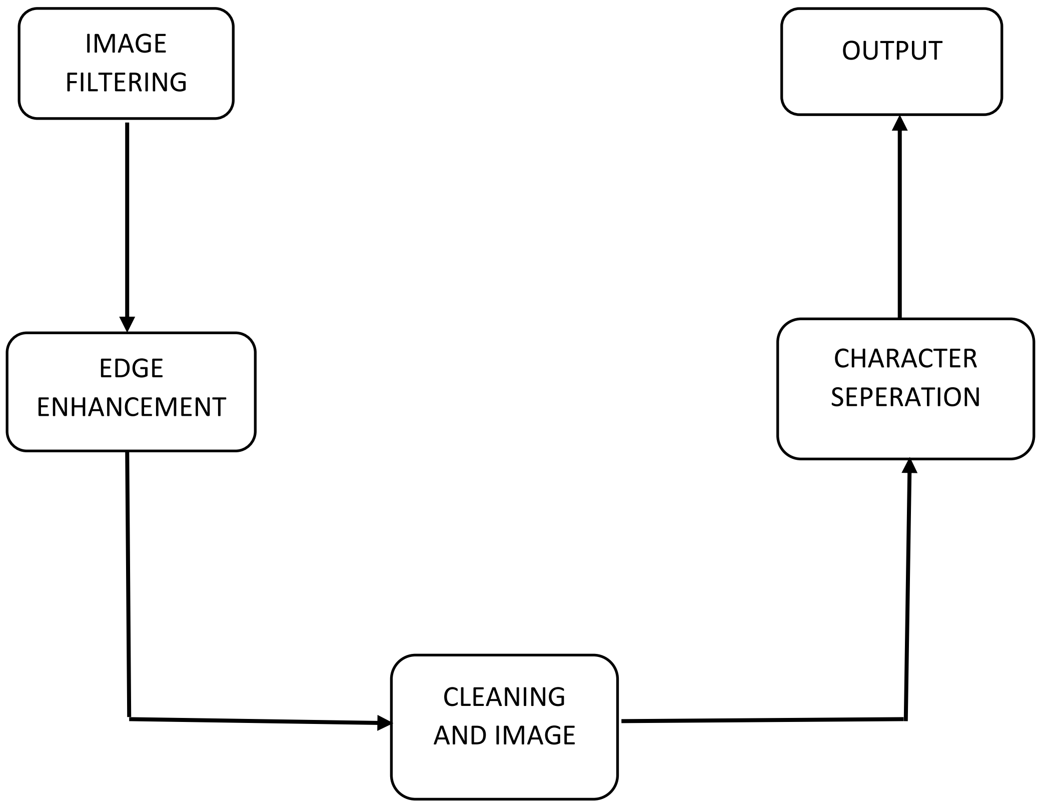


Fig 4.3.4 recognition

4.3.5 EXPERIMENTAL RESULT

The Number plate detection and recognition system first captures the vehicle image. Then the image is prepeocessed to enhance the quality of image. The RGB to Gray Scale Conversion has been done in the preprocessing stage. Localization is the process of identifying number plate from the image captured. Rear ad front part of the vehicle is captured into an image and other part of the vehicle are cropped. The Region is extracted using an image segmentation in an image. Then the gray scale image is binarized to remove the noise and to get the enhanced quality of image. Optical Character Recognition (OCR) technique is used for the character recognition. OCR involves photoscanning of the text character by character analysis of the scanned image into character codes. Then the Alphanumeris of the number plate has been detected and displayed.

CHAPTER 5

SYSTEM SPECIFICATION

5.1 HARDWARE REQUIREMENT

The Hardware requirement may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. There are used by software engineer as the starting point for the system design.

PROCESSOR	: Core I5
MONITOR	: 15” COLOR
HARD DISK	: 1TB
KEYBOARD	: STANDARD 102 KEYS
MOUSE	: 3 BUTTONS
OUTPUT DEVICE	: LED MONITOR
RAM	: 4GB

6.2 SOFTWARE REQUIREMENT

The software requirement document is the specification of the system. It should include both a definition and a specification of requirements.

OPERATING SYSTEM	: Windows 10
FRONT END	: Python or Anaconda

CHAPTER 6

SOFTWARE DESCRIPTION

7.1 GENERAL

This chapter is about the software language and the tools used in the development of the project. The primary language used here are PYTHON.

7.2 PYTHON

Python is an object-oriented, high-level programming language with integrated dynamic semantics primarily for web and app development. It is extremely attractive in the field of Rapid Application Development because it offers dynamic typing and dynamic binding options.

Python is relatively simple, so it's easy to learn since it requires a unique syntax that focuses on readability. Developers can read and translate Python code much easier than other languages. In turn, this reduces the cost of program maintenance and development because it allows teams to work collaboratively without significant language and experience barriers.

Additionally, Python supports the use of modules and packages, which means that programs can be designed in a modular style and code can be reused across a variety of projects. Once you've developed a module or package you need, it can be scaled for use in other projects, and it's easy to import or export these modules.

One of the most promising benefits of Python is that both the standard library and the interpreter are available free of charge, in both binary and source form. There is no exclusivity either, as Python and all the necessary tools are available on all major platforms. Therefore, it is an enticing option for developers who don't want to worry about paying high development costs.

It’s easy to learn – the time needed to learn Python is shorter than for many other languages; this means that it’s possible to start the actual programming faster;

It’s **easy to teach** – the teaching workload is smaller than that needed by other languages; this means that the teacher can put more emphasis on general (language-independent) programming techniques, not wasting energy on exotic tricks, strange exceptions and incomprehensible rules;

It’s **easy to use** for writing new software – it’s often possible to write code faster than using Python;

It’s **easy to understand** – it’s also often easier to understand someone else’s code faster if it is written in Python;

It’s **easy to obtain**, install and deploy – Python is free, open and multiplatform; not all languages can boast that.

7.3 ANACONDA NAVIGATOR

Anaconda is a free and open-source distribution of the python and RProgramming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analysis, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system conda.

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, macOS and Linux

Conda is an open source, cross platform language-agnostic package manager and environment management system that installs, runs, and updates packages and their dependencies. It was created for Python programs, but it can package and distribute software for any language (e.g., R), including multi-language projects. The Conda package and environment manager is included in all versions of Anaconda, Miniconda, and Anaconda Repository

SPYDER

Spyder is an open source cross-platform integrated development environment (IDE) for scientific programming in the python language. Spyder integrates with a number of prominent packages in the scientific Python stack, including NumPy, SciPy, Matplotlib, pandas, IPython, symPy and Cython, as well as other open source software. It is released under the MIT license.

Spyder is extensible with first- and third-party plugins, includes support for interactive tools for data inspection and embeds Python-specific code quality assurance and introspection instruments, such as Pyflakes, Pylint and Rope. It is available cross-platform through Anaconda on Windows, on macOS through MacPorts, and on major Linux distributions such as Arch Linux, Debian, Fedora, Gentoo Linux, openSUSE and Ubuntu.

Spyder uses Qt for its GUI, and is designed to use either of the PyQt or PySide Python bindings. QtPy, a thin abstraction layer developed by the Spyder project and later adopted by multiple other packages, provides the flexibility to use either backend.

Spyder, the Scientific Python Development Environment, is a free integrated development environment (IDE) that is included with Anaconda. It includes editing, interactive testing, debugging and introspection features

JUPYTER

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modelling, data visualization, machine learning, and much more.

Jupyter is a browser-based interpreter that allows you to interactively work with Python and R. Anaconda provides Jupyter as well. You can think of Jupyter as a digital notebook that gives you an ability to execute commands, take notes and draw charts. It's primarily used by Data Scientists.

The Jupyter Notebook is an incredibly powerful tool for interactively developing and presenting data science projects. A notebook integrates code and its output into a single document that combines visualisations, narrative text, mathematical equations, and other rich media. The intuitive workflow promotes iterative and rapid development, making notebooks an increasingly popular choice at the heart of contemporary data science, analysis, and increasingly science at large. Best of all, as part of the open source Project Jupyter, they are completely free.

The Jupyter project is the successor to the earlier IPython Notebook, which was first published as a prototype in 2010. Although it is possible to use many different programming languages within Jupyter Notebooks, this article will focus on Python as it is the most common use case.

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 Python Imaging Library, including jpeg, png, gif, bmp, tiff, and others, whereas tesseract-ocr by default only supports tiff and bmp. Additionally, if used as a script, Python-tesseract will print the recognized text instead of writing it to a file.

OPENCV

OpenCV was started at Intel in 1999 by Gary Bradsky, and the first release came out in 2000. Vadim Pisarevsky joined Gary Bradsky to manage Intel's Russian software OpenCV team. In 2005, OpenCV was used on Stanley, the vehicle that won the 2005 DARPA Grand Challenge. Later, its active development continued under the support of Willow Garage with Gary Bradsky and Vadim Pisarevsky leading the project. OpenCV now supports a multitude of algorithms related to Computer Vision and Machine Learning and is expanding day by day.

OpenCV supports a wide variety of programming languages such as C++, Python, Java, etc., and is available on different platforms including Windows, Linux, OS X, Android, and iOS. Interfaces for high-speed GPU operations based on CUDA and OpenCL are also under active development.

OpenCV-Python is the Python API for OpenCV, combining the best qualities of the OpenCV C++ API and the Python language OpenCV-Python.

OpenCV-Python is a library of Python bindings designed to solve computer vision problems.

Python is a general purpose programming language started by Guido van Rossum that became very popular very quickly, mainly because of its simplicity and code readability. It enables the programmer to express ideas in fewer lines of code without reducing readability. Compared to languages like C/C++, Python is slower. That said, Python can be easily extended with C/C++, which allows us to write computationally intensive code in C/C++ and create Python wrappers that can be used as Python modules. This gives us two advantages: first, the code is as fast as the original C/C++ code (since it is the actual C++ code working in background) and second, it easier to code in Python than C/C++. OpenCV-Python is a Python wrapper for the original OpenCV C++ implementation.

OpenCV-Python makes use of NumPy, which is a highly optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from NumPy arrays. This also makes it easier to integrate with other libraries that use NumPy such as SciPy and Matplotlib.

TENSORFLOW

Tensor Flow is an open source library for fast numerical computing. It was created and is maintained by Google and released under the Apache 2.0 open source license. The API is nominally for the Python programming language, although there is access to the underlying C++ API.

CHAPTER 7

CONCLUSION

7.1 CONCLUSION

Number plate recognition system proposed for LPR based on the features selection. The proposed system comprises of four major step: (i) preprocessing (ii) localization (Plate Region Extraction) (iii) Segmentation (iv) Recognition. With proposed method, we have tried our best to deal with different problems of light variation, occlusion, and so on. The simulation results confirm that, with our novel idea, we have managed to tackle above mentioned problems. Moreover, it is also concluded that a cascaded design can comfortably manage mentioned problem at its early stages and also the selection of good features results in improved accuracy. As a future work, we will add few more set of features and will implement improved features selection technique to reduce the error rate and increase the recognition rate.

APPENDIX 1

SOURCE CODE

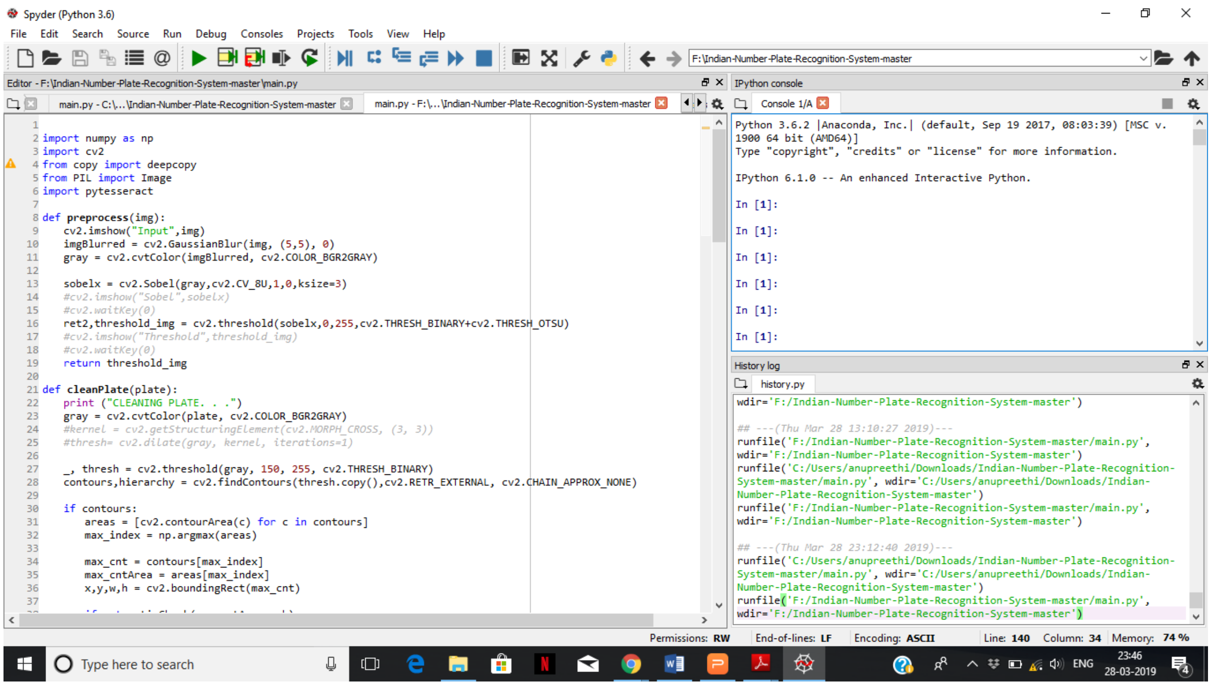
```
import numpy as np
import cv2
from copy import deepcopy
from PIL import Image
import pytesseract

def preprocess(img):
	cv2.imshow("Input",img)
	imgBlurred = cv2.GaussianBlur(img, (5,5), 0)
	gray = cv2.cvtColor(imgBlurred, cv2.COLOR_BGR2GRAY)
	sobelx = cv2.Sobel(gray,cv2.CV_8U,1,0,ksize=3)
	#cv2.imshow("Sobel",sobelx)
	#cv2.waitKey(0)
	ret2,threshold_img                                                =
cv2.threshold(sobelx,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
	#cv2.imshow("Threshold",threshold_img)
	#cv2.waitKey(0)
	return threshold_img

def cleanPlate(plate):
	print ("CLEANING PLATE. . .")
	gray = cv2.cvtColor(plate, cv2.COLOR_BGR2GRAY)
	#kernel = cv2.getStructuringElement(cv2.MORPH_CROSS, (3, 3))
	#thresh= cv2.dilate(gray, kernel, iterations=1)

	_, thresh = cv2.threshold(gray, 150, 255, cv2.THRESH_BINARY)
```

```
        contours,hierarchy                                                        =
cv2.findContours(thresh.copy(),cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_NONE)
        if contours:
                areas = [cv2.contourArea(c) for c in contours]
                max_index = np.argmax(areas)
                max_cnt = contours[max_index]
                max_cntArea = areas[max_index]
                x,y,w,h = cv2.boundingRect(max_cnt)
        if not ratioCheck(max_cntArea,w,h):
                        return plate,None
        cleaned_final = thresh[y:y+h, x:x+w]
                #cv2.imshow("Function Test",cleaned_final)
                return cleaned_final,[x,y,w,h]
        else:
                return plate,None

def extract_contours(threshold_img):
        element=cv2.getStructuringElement(shape=cv2.MORPH_RECT,
ksize=(17, 3))
        morph_img_threshold = threshold_img.copy()
        cv2.morphologyEx(src=threshold_img,          op=cv2.MORPH_CLOSE,
kernel=element, dst=morph_img_threshold)
        cv2.imshow("Morphed",morph_img_threshold)
        cv2.waitKey(0)
contours,hierarchy=cv2.findContours(morph_img_threshold,mode=cv2.RETR_
EXTERNAL,method=cv2.CHAIN_APPROX_NONE)
        return contours
def ratioCheck(area, width, height):
        ratio = float(width) / float(height)
        if ratio < 1:
                ratio = 1 / ratio
        aspect = 4.7272
        min = 15*aspect*15  # minimum area
        max = 125*aspect*125  # maximum area
        rmin = 3
        rmax = 6

        if (area < min or area > max) or (ratio < rmin or ratio > rmax):
                return False
        return True

def isMaxWhite(plate):
        avg = np.mean(plate)
        if(avg>=115):
                return True
        else:
                return False

def validateRotationAndRatio(rect):
        (x, y), (width, height), rect_angle = rect

        if(width>height):
                angle = -rect_angle
        else:
                angle = 90 + rect_angle

        if angle>15:
                return False
        if height == 0 or width == 0:
                return False

        area = height*width
        if not ratioCheck(area,width,height):
                return False
        else:
                return True

def cleanAndRead(img,contours):
        pytesseract.pytesseract.tesseract_cmd='C:\\ProgramFiles(x86)\\Tesseract-
OCR\\tesseract.exe'
        for i,cnt in enumerate(contours):
                min_rect = cv2.minAreaRect(cnt)

                if validateRotationAndRatio(min_rect):
                    x,y,w,h = cv2.boundingRect(cnt)
                    plate_img = img[y:y+h,x:x+w]

                    if(isMaxWhite(plate_img)):
                        #count+=1
                        clean_plate, rect = cleanPlate(plate_img)

                        if rect:
                            x1,y1,w1,h1 = rect
                            x,y,w,h = x+x1,y+y1,w1,h1
                            cv2.imshow("Cleaned Plate", clean_plate)
                            plate_im = Image.fromarray(clean_plate)
```

```
                                        plate_im.save('out.jpg')

text=pytesseract.image_to_string(Image.open('out.jpg'), lang='eng')
                                print ("Detected Text : ",text)
                                img=cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
                                        cv2.imshow("Detected Plate", img)
                                        cv2.waitKey(0)

        #print "No. of final cont : " , count
if __name__ == '__main__':
        Print ("DETECTING PLATE . . .")
        #img = cv2.imread ("testData/Final.JPG")
        Img = cv2.imread ("testdata/test3.jpg")
        threshold_img = preprocess (img)
        contours= extract_contours(threshold_img)
        if len (contours)!=0:
                print len (contours) #Test
                cv2.drawContours(img, contours, -1, (0,255,0), 1)
                cv2.imshow("Contours", img)
                cv2.waitKey(0)
cleanAndRead(img,contours)
```

APPENDIX 2

SCREENSHOTS



Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
F:\Indian-Number-Plate-Recognition-System-master
Editor - F:\Indian-Number-Plate-Recognition-System-master\main.py
main.py - C:\...\Indian-Number-Plate-Recognition-System-master
main.py - F:\...\Indian-Number-Plate-Recognition-System-master
IPython console
Console 1/A
History log
history.py
Type here to search

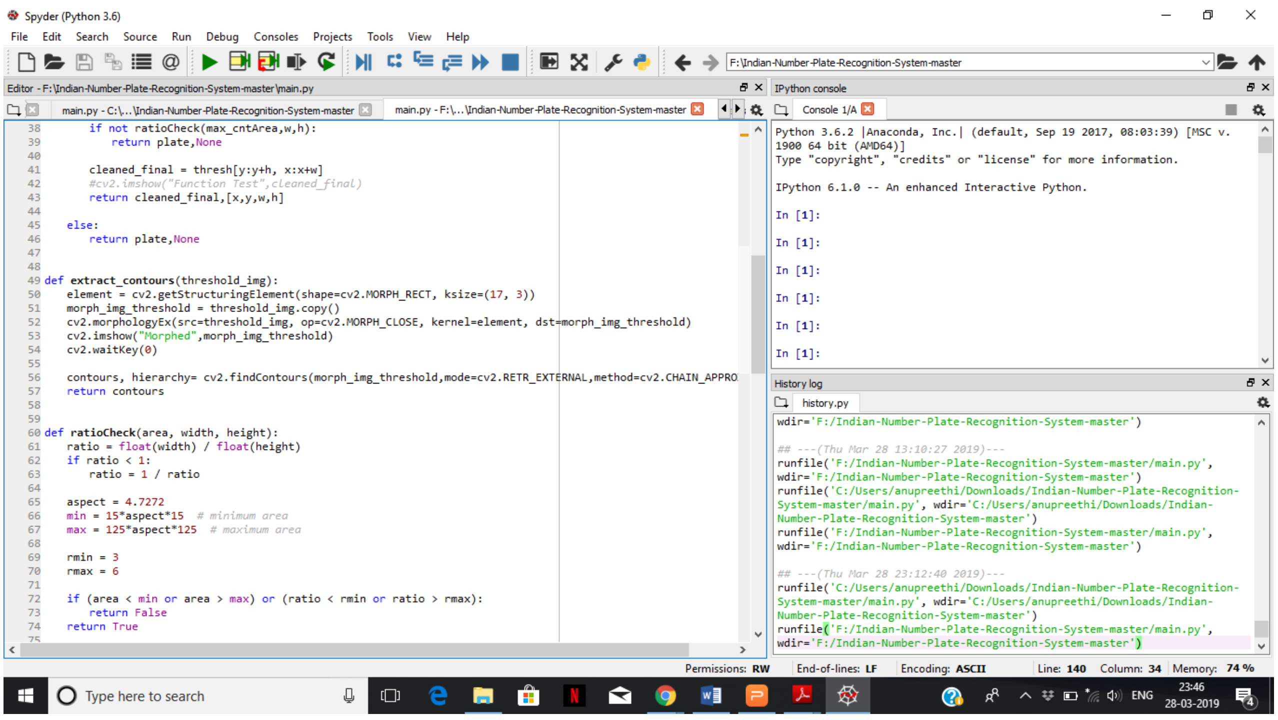
```python

import numpy as np
import cv2
from copy import deepcopy
from PIL import Image
import pytesseract

def preprocess(img):
    cv2.imshow("Input",img)
    imgBlurred = cv2.GaussianBlur(img, (5,5), 0)
    gray = cv2.cvtColor(imgBlurred, cv2.COLOR_BGR2GRAY)

    sobelx = cv2.Sobel(gray,cv2.CV_8U,1,0,ksize=3)
    #cv2.imshow("Sobel",sobelx)
    #cv2.waitKey(0)
    ret2,threshold_img = cv2.threshold(sobelx,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
    #cv2.imshow("Threshold",threshold_img)
    #cv2.waitKey(0)
    return threshold_img

def cleanPlate(plate):
    print ("CLEANING PLATE. . .")
    gray = cv2.cvtColor(plate, cv2.COLOR_BGR2GRAY)
    #kernel = cv2.getStructuringElement(cv2.MORPH_CROSS, (3, 3))
    #thresh= cv2.dilate(gray, kernel, iterations=1)

    _, thresh = cv2.threshold(gray, 150, 255, cv2.THRESH_BINARY)
    contours,hierarchy = cv2.findContours(thresh.copy(),cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

    if contours:
        areas = [cv2.contourArea(c) for c in contours]
        max_index = np.argmax(areas)

        max_cnt = contours[max_index]
        max_cntArea = areas[max_index]
        x,y,w,h = cv2.boundingRect(max_cnt)

```

```
Python 3.6.2 |Anaconda, Inc.| (default, Sep 19 2017, 08:03:39) [MSC v.
1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 6.1.0 -- An enhanced Interactive Python.

In [1]:

In [1]:

In [1]:

In [1]:

In [1]:

In [1]:
```

```
wdir='F:/Indian-Number-Plate-Recognition-System-master')

## ---(Thu Mar 28 13:10:27 2019)---
runfile('F:/Indian-Number-Plate-Recognition-System-master/main.py',
wdir='F:/Indian-Number-Plate-Recognition-System-master')
runfile('C:/Users/anupreethi/Downloads/Indian-Number-Plate-Recognition-
System-master/main.py', wdir='C:/Users/anupreethi/Downloads/Indian-
Number-Plate-Recognition-System-master')
runfile('F:/Indian-Number-Plate-Recognition-System-master/main.py',
wdir='F:/Indian-Number-Plate-Recognition-System-master')

## ---(Thu Mar 28 23:12:40 2019)---
runfile('C:/Users/anupreethi/Downloads/Indian-Number-Plate-Recognition-
System-master/main.py', wdir='C:/Users/anupreethi/Downloads/Indian-
Number-Plate-Recognition-System-master')
runfile('F:/Indian-Number-Plate-Recognition-System-master/main.py',
wdir='F:/Indian-Number-Plate-Recognition-System-master')
```

Permissions: RW End-of-lines: LF Encoding: ASCII Line: 140 Column: 34 Memory: 74 %

23:46 28-03-2019


Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
F:\Indian-Number-Plate-Recognition-System-master
Editor - F:\Indian-Number-Plate-Recognition-System-master\main.py
main.py - C:\...\Indian-Number-Plate-Recognition-System-master
main.py - F:\...\Indian-Number-Plate-Recognition-System-master
IPython console
Console 1/A
History log
history.py
Type here to search
ENG

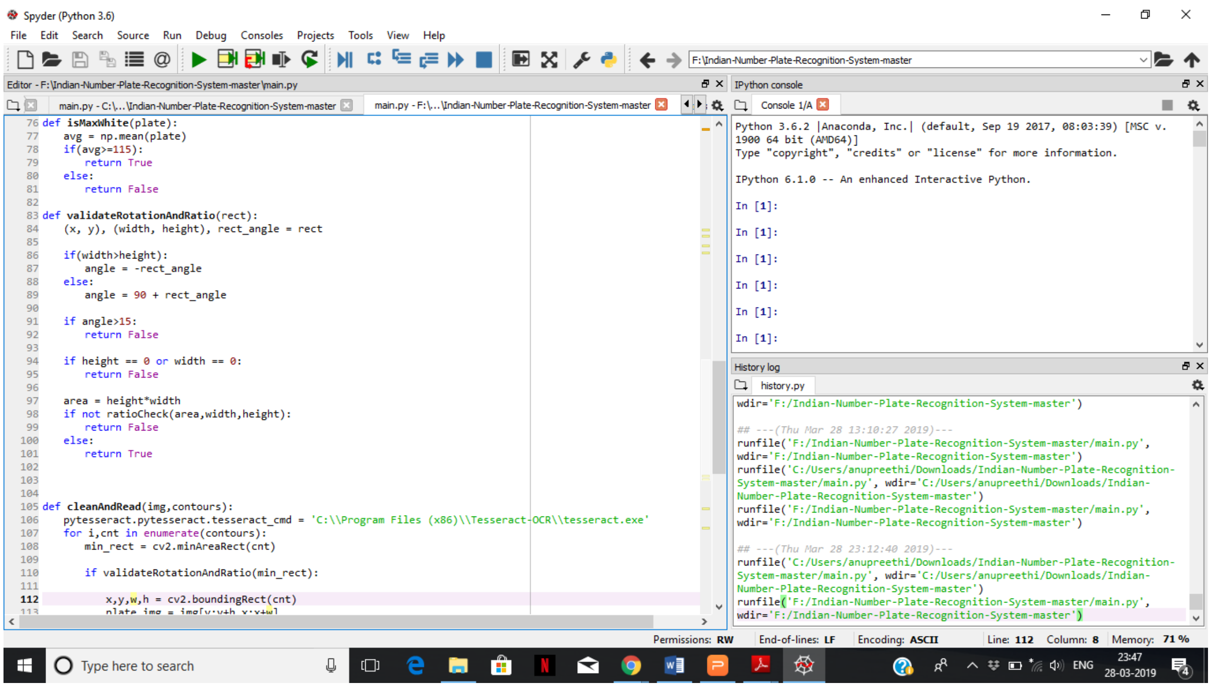
```
        if not ratioCheck(max_cntArea,w,h):
            return plate,None

        cleaned_final = thresh[y:y+h, x:x+w]
        #cv2.imshow("Function Test",cleaned_final)
        return cleaned_final,[x,y,w,h]

    else:
        return plate,None


def extract_contours(threshold_img):
    element = cv2.getStructuringElement(shape=cv2.MORPH_RECT, ksize=(17, 3))
    morph_img_threshold = threshold_img.copy()
    cv2.morphologyEx(src=threshold_img, op=cv2.MORPH_CLOSE, kernel=element, dst=morph_img_threshold)
    cv2.imshow("Morphed",morph_img_threshold)
    cv2.waitKey(0)

    contours, hierarchy= cv2.findContours(morph_img_threshold,mode=cv2.RETR_EXTERNAL,method=cv2.CHAIN_APPRO
    return contours


def ratioCheck(area, width, height):
    ratio = float(width) / float(height)
    if ratio < 1:
        ratio = 1 / ratio

    aspect = 4.7272
    min = 15*aspect*15  # minimum area
    max = 125*aspect*125  # maximum area

    rmin = 3
    rmax = 6

    if (area < min or area > max) or (ratio < rmin or ratio > rmax):
        return False
    return True
```

```
Python 3.6.2 |Anaconda, Inc.| (default, Sep 19 2017, 08:03:39) [MSC v.
1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 6.1.0 -- An enhanced Interactive Python.

In [1]:

In [1]:

In [1]:

In [1]:

In [1]:

In [1]:
```

```
wdir='F:/Indian-Number-Plate-Recognition-System-master')

## ---(Thu Mar 28 13:10:27 2019)---
runfile('F:/Indian-Number-Plate-Recognition-System-master/main.py',
wdir='F:/Indian-Number-Plate-Recognition-System-master')
runfile('C:/Users/anupreethi/Downloads/Indian-Number-Plate-Recognition-
System-master/main.py', wdir='C:/Users/anupreethi/Downloads/Indian-
Number-Plate-Recognition-System-master')
runfile('F:/Indian-Number-Plate-Recognition-System-master/main.py',
wdir='F:/Indian-Number-Plate-Recognition-System-master')

## ---(Thu Mar 28 23:12:40 2019)---
runfile('C:/Users/anupreethi/Downloads/Indian-Number-Plate-Recognition-
System-master/main.py', wdir='C:/Users/anupreethi/Downloads/Indian-
Number-Plate-Recognition-System-master')
runfile('F:/Indian-Number-Plate-Recognition-System-master/main.py',
wdir='F:/Indian-Number-Plate-Recognition-System-master')
```

Permissions: RW   End-of-lines: LF   Encoding: ASCII   Line: 140   Column: 34   Memory: 74 %

23:46 28-03-2019


Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
F:\Indian-Number-Plate-Recognition-System-master
Editor - F:\Indian-Number-Plate-Recognition-System-master\main.py
main.py - C:\...\Indian-Number-Plate-Recognition-System-master
main.py - F:\...\Indian-Number-Plate-Recognition-System-master
IPython console
Console 1/A
History log
history.py
Type here to search
ENG

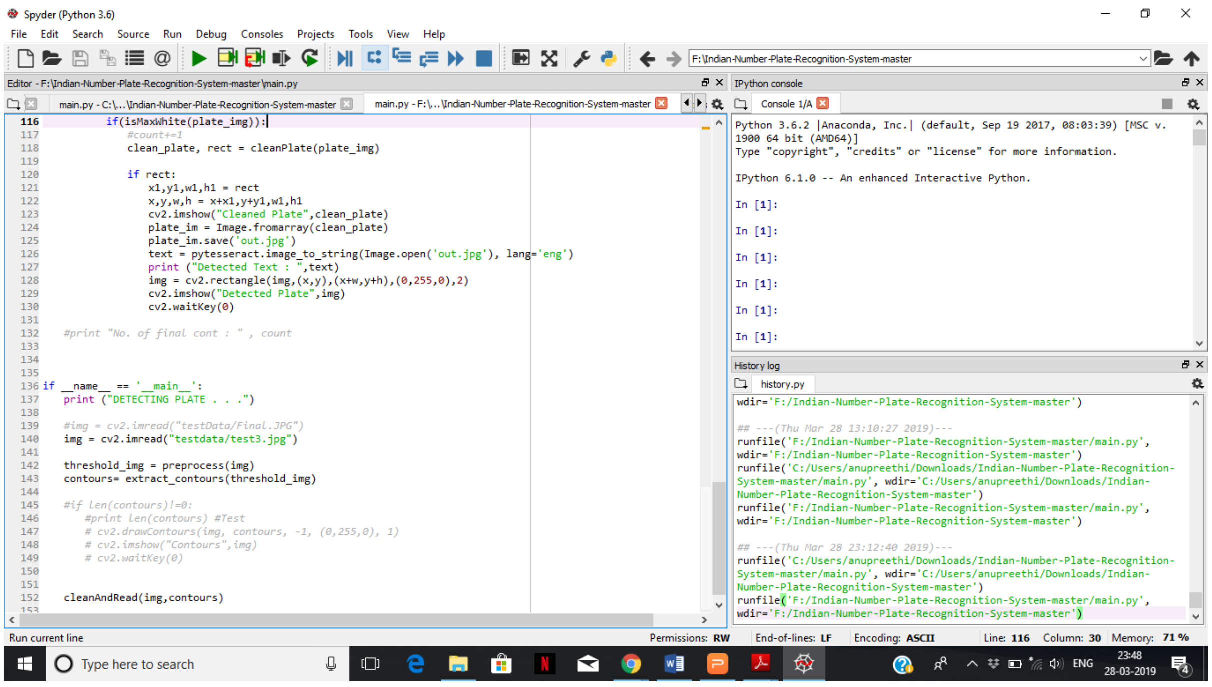
```
def isMaxWhite(plate):
    avg = np.mean(plate)
    if(avg>=115):
        return True
    else:
        return False

def validateRotationAndRatio(rect):
    (x, y), (width, height), rect_angle = rect

    if(width>height):
        angle = -rect_angle
    else:
        angle = 90 + rect_angle

    if angle>15:
        return False

    if height == 0 or width == 0:
        return False

    area = height*width
    if not ratioCheck(area,width,height):
        return False
    else:
        return True



def cleanAndRead(img,contours):
    pytesseract.pytesseract.tesseract_cmd = 'C:\\Program Files (x86)\\Tesseract-OCR\\tesseract.exe'
    for i,cnt in enumerate(contours):
        min_rect = cv2.minAreaRect(cnt)

        if validateRotationAndRatio(min_rect):

            x,y,w,h = cv2.boundingRect(cnt)
            plate_img = img[y:y+h, x:x+w]
```

```
Python 3.6.2 |Anaconda, Inc.| (default, Sep 19 2017, 08:03:39) [MSC v.
1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 6.1.0 -- An enhanced Interactive Python.

In [1]:

In [1]:

In [1]:

In [1]:

In [1]:

In [1]:
```

```
wdir='F:/Indian-Number-Plate-Recognition-System-master')

## ---(Thu Mar 28 13:10:27 2019)---
runfile('F:/Indian-Number-Plate-Recognition-System-master/main.py',
wdir='F:/Indian-Number-Plate-Recognition-System-master')
runfile('C:/Users/anupreethi/Downloads/Indian-Number-Plate-Recognition-
System-master/main.py', wdir='C:/Users/anupreethi/Downloads/Indian-
Number-Plate-Recognition-System-master')
runfile('F:/Indian-Number-Plate-Recognition-System-master/main.py',
wdir='F:/Indian-Number-Plate-Recognition-System-master')

## ---(Thu Mar 28 23:12:40 2019)---
runfile('C:/Users/anupreethi/Downloads/Indian-Number-Plate-Recognition-
System-master/main.py', wdir='C:/Users/anupreethi/Downloads/Indian-
Number-Plate-Recognition-System-master')
runfile('F:/Indian-Number-Plate-Recognition-System-master/main.py',
wdir='F:/Indian-Number-Plate-Recognition-System-master')
```

Permissions: RW   End-of-lines: LF   Encoding: ASCII   Line: 112   Column: 8   Memory: 71 %

23:47 28-03-2019



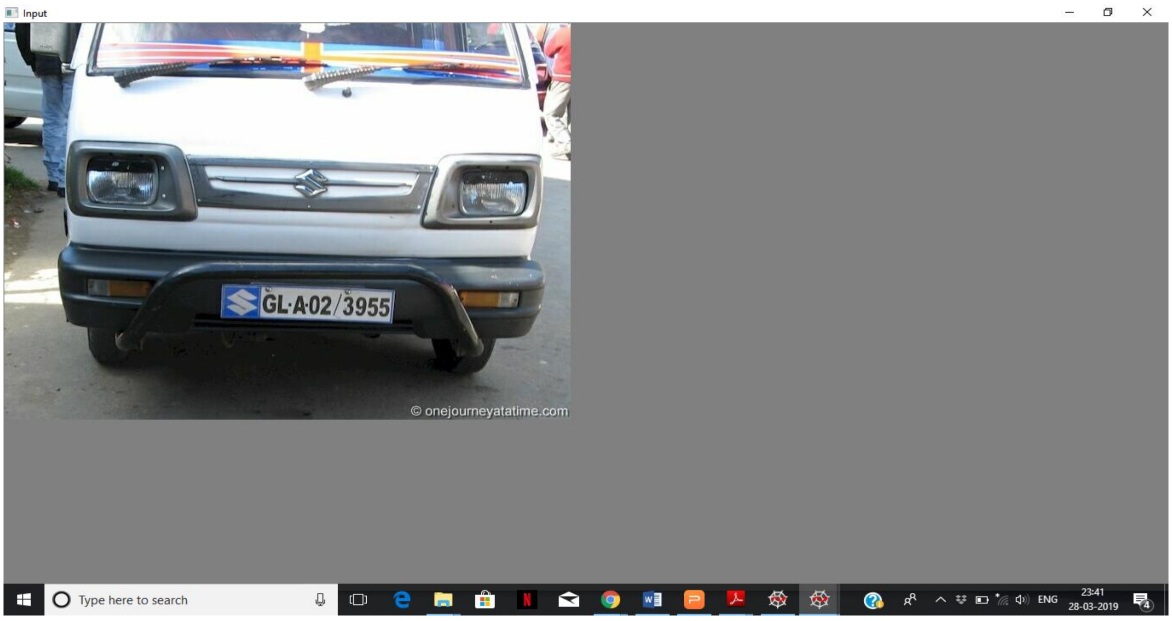
```python
            if(isMaxWhite(plate_img)):
                #count+=1
                clean_plate, rect = cleanPlate(plate_img)

                if rect:
                    x1,y1,w1,h1 = rect
                    x,y,w,h = x+x1,y+y1,w1,h1
                    cv2.imshow("Cleaned Plate",clean_plate)
                    plate_im = Image.fromarray(clean_plate)
                    plate_im.save('out.jpg')
                    text = pytesseract.image_to_string(Image.open('out.jpg'), lang='eng')
                    print ("Detected Text : ",text)
                    img = cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
                    cv2.imshow("Detected Plate",img)
                    cv2.waitKey(0)

    #print "No. of final cont : " , count



if __name__ == '__main__':
    print ("DETECTING PLATE . . .")

    #img = cv2.imread("testData/Final.JPG")
    img = cv2.imread("testdata/test3.jpg")

    threshold_img = preprocess(img)
    contours= extract_contours(threshold_img)

    #if len(contours)!=0:
        #print len(contours) #Test
        # cv2.drawContours(img, contours, -1, (0,255,0), 1)
        # cv2.imshow("Contours",img)
        # cv2.waitKey(0)


    cleanAndRead(img,contours)
```

```
Python 3.6.2 |Anaconda, Inc.| (default, Sep 19 2017, 08:03:39) [MSC v.
1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 6.1.0 -- An enhanced Interactive Python.

In [1]:

In [1]:

In [1]:

In [1]:

In [1]:

In [1]:
```

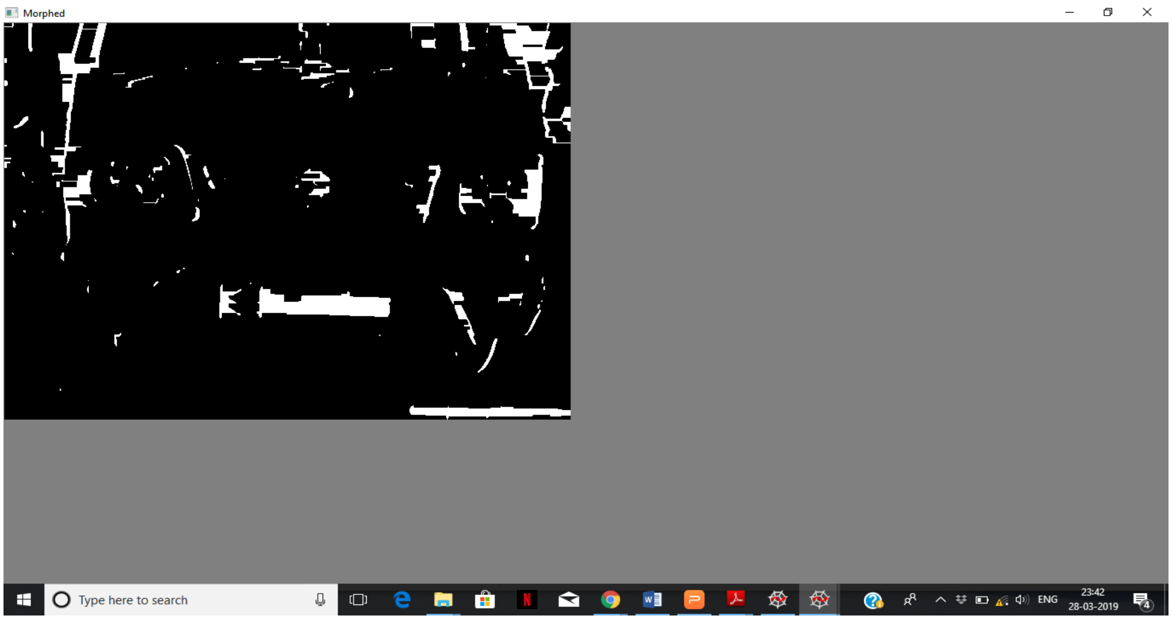
```
wdir='F:/Indian-Number-Plate-Recognition-System-master')

## ---(Thu Mar 28 13:10:27 2019)---
runfile('F:/Indian-Number-Plate-Recognition-System-master/main.py',
wdir='F:/Indian-Number-Plate-Recognition-System-master')
runfile('C:/Users/anupreethi/Downloads/Indian-Number-Plate-Recognition-
System-master/main.py', wdir='C:/Users/anupreethi/Downloads/Indian-
Number-Plate-Recognition-System-master')
runfile('F:/Indian-Number-Plate-Recognition-System-master/main.py',
wdir='F:/Indian-Number-Plate-Recognition-System-master')

## ---(Thu Mar 28 23:12:40 2019)---
runfile('C:/Users/anupreethi/Downloads/Indian-Number-Plate-Recognition-
System-master/main.py', wdir='C:/Users/anupreethi/Downloads/Indian-
Number-Plate-Recognition-System-master')
runfile('F:/Indian-Number-Plate-Recognition-System-master/main.py',
wdir='F:/Indian-Number-Plate-Recognition-System-master')
```

Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
F:\Indian-Number-Plate-Recognition-System-master
Editor - F:\Indian-Number-Plate-Recognition-System-master\main.py
main.py - C:\...\Indian-Number-Plate-Recognition-System-master
main.py - F:\...\Indian-Number-Plate-Recognition-System-master
IPython console
Console 1/A
History log
history.py
Run current line
Permissions: RW
End-of-lines: LF
Encoding: ASCII
Line: 116
Column: 30
Memory: 71 %
Type here to search
ENG
23:48
28-03-2019

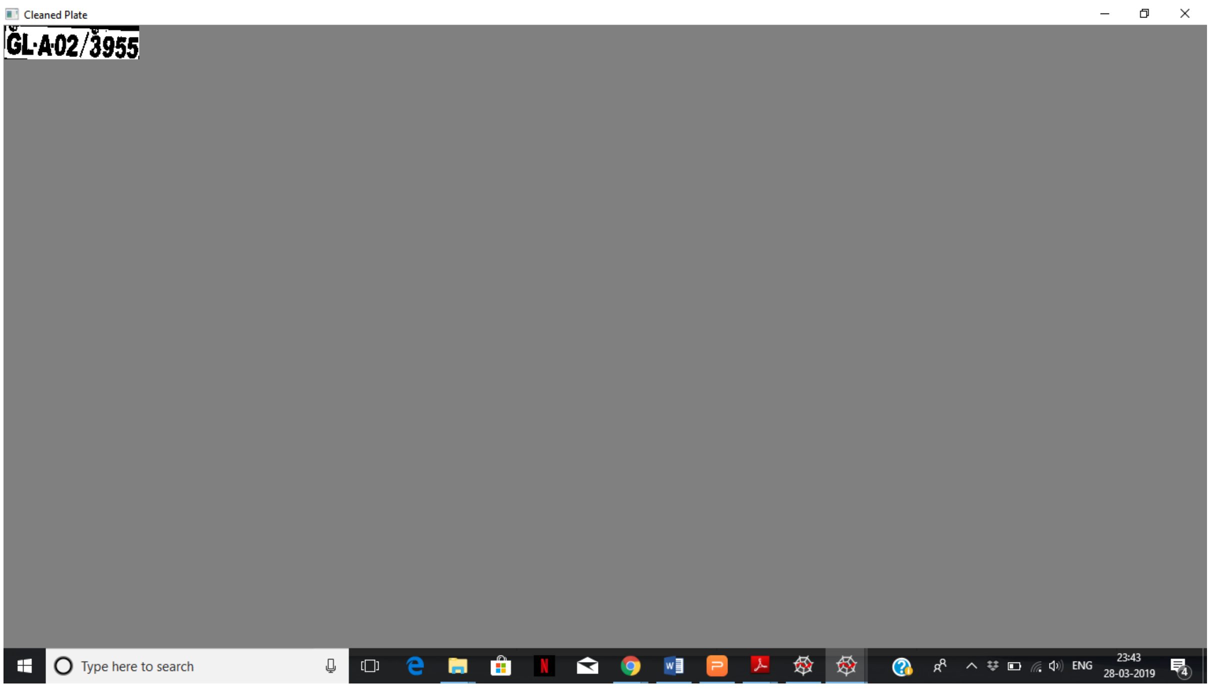
OUTPUT



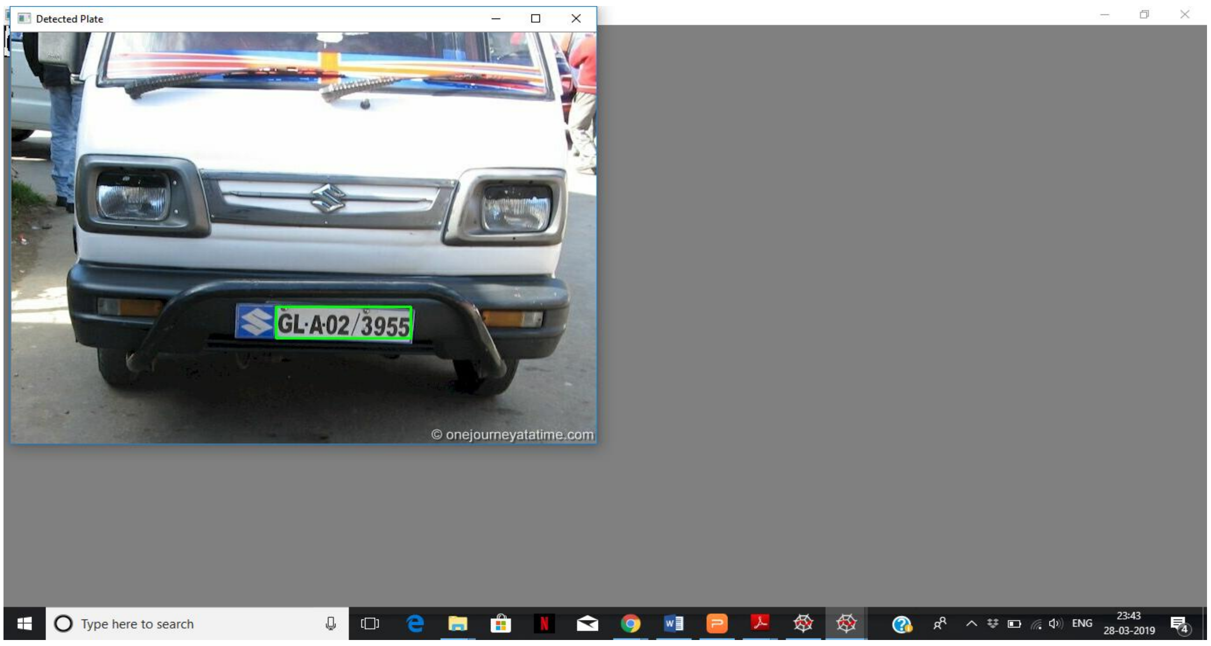
F1-System captures the image and send to the preprocessing stage to enhance the quality of image



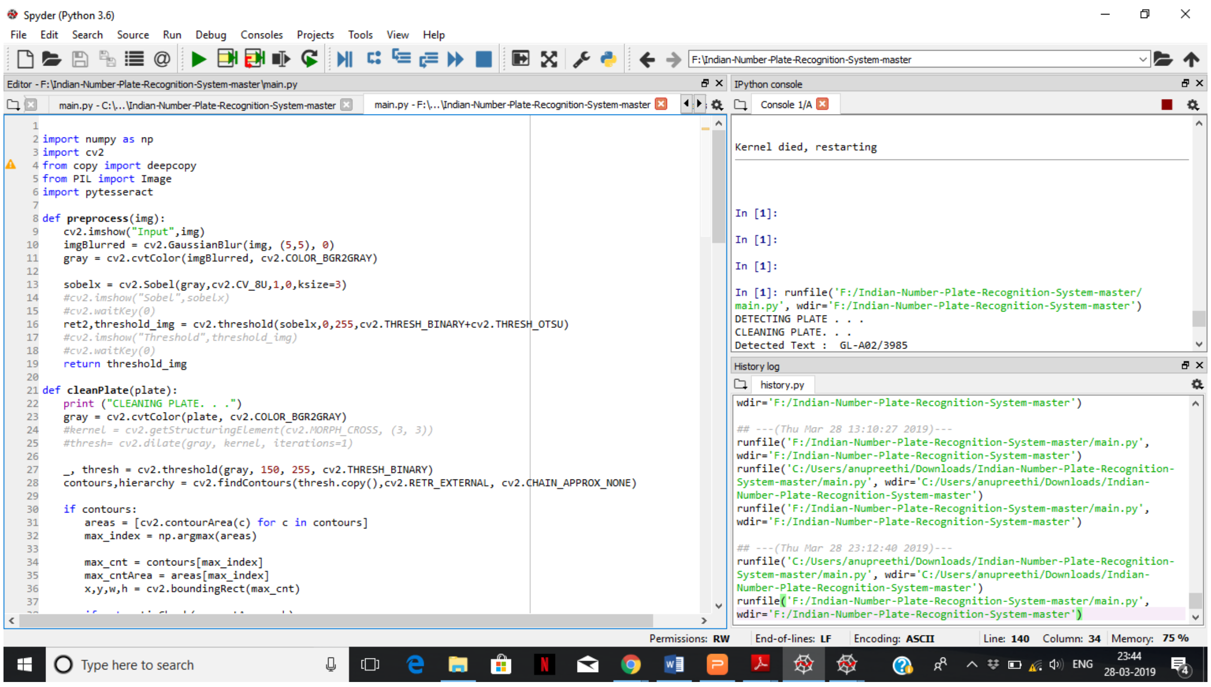
F2- The process of RGB to Gray Scale Convertion has been done using threshold value. It is also known as Thershold image



F3- Here localization process has been done through binary image convertion



F4- Detecting the number plate using image processing technique.


Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
F:\Indian-Number-Plate-Recognition-System-master
Editor - F:\Indian-Number-Plate-Recognition-System-master\main.py
main.py - C:\...\Indian-Number-Plate-Recognition-System-master
main.py - F:\...\Indian-Number-Plate-Recognition-System-master
IPython console
Console 1/A
History log
history.py
Permissions: RW End-of-lines: LF Encoding: ASCII Line: 140 Column: 34 Memory: 75 %
Type here to search
ENG
23:44
28-03-2019

```

import numpy as np
import cv2
from copy import deepcopy
from PIL import Image
import pytesseract

def preprocess(img):
    cv2.imshow("Input",img)
    imgBlurred = cv2.GaussianBlur(img, (5,5), 0)
    gray = cv2.cvtColor(imgBlurred, cv2.COLOR_BGR2GRAY)

    sobelx = cv2.Sobel(gray,cv2.CV_8U,1,0,ksize=3)
    #cv2.imshow("Sobel",sobelx)
    #cv2.waitKey(0)
    ret2,threshold_img = cv2.threshold(sobelx,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
    #cv2.imshow("Threshold",threshold_img)
    #cv2.waitKey(0)
    return threshold_img

def cleanPlate(plate):
    print ("CLEANING PLATE. . .")
    gray = cv2.cvtColor(plate, cv2.COLOR_BGR2GRAY)
    #kernel = cv2.getStructuringElement(cv2.MORPH_CROSS, (3, 3))
    #thresh= cv2.dilate(gray, kernel, iterations=1)

    _, thresh = cv2.threshold(gray, 150, 255, cv2.THRESH_BINARY)
    contours,hierarchy = cv2.findContours(thresh.copy(),cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

    if contours:
        areas = [cv2.contourArea(c) for c in contours]
        max_index = np.argmax(areas)

        max_cnt = contours[max_index]
        max_cntArea = areas[max_index]
        x,y,w,h = cv2.boundingRect(max_cnt)

```

```
Kernel died, restarting



In [1]:

In [1]:

In [1]:

In [1]: runfile('F:/Indian-Number-Plate-Recognition-System-master/
main.py', wdir='F:/Indian-Number-Plate-Recognition-System-master')
DETECTING PLATE . . .
CLEANING PLATE. . .
Detected Text :  GL-A02/3985
```

```
wdir='F:/Indian-Number-Plate-Recognition-System-master')

## ---(Thu Mar 28 13:10:27 2019)---
runfile('F:/Indian-Number-Plate-Recognition-System-master/main.py',
wdir='F:/Indian-Number-Plate-Recognition-System-master')
runfile('C:/Users/anupreethi/Downloads/Indian-Number-Plate-Recognition-
System-master/main.py', wdir='C:/Users/anupreethi/Downloads/Indian-
Number-Plate-Recognition-System-master')
runfile('F:/Indian-Number-Plate-Recognition-System-master/main.py',
wdir='F:/Indian-Number-Plate-Recognition-System-master')

## ---(Thu Mar 28 23:12:40 2019)---
runfile('C:/Users/anupreethi/Downloads/Indian-Number-Plate-Recognition-
System-master/main.py', wdir='C:/Users/anupreethi/Downloads/Indian-
Number-Plate-Recognition-System-master')
runfile('F:/Indian-Number-Plate-Recognition-System-master/main.py',
wdir='F:/Indian-Number-Plate-Recognition-System-master')
```

F5-The alphanumerics of the number plate were detected and displayed using Optical Character Recognition technique.

REFERENCE

[1] S. Du, M. Ibrahim, M. Shehata, and W. Badawy, Automatic license plate recognition (ALPR): A state-of-the-art review," IEEE Trans. Circuits Syst. Video Technol., vol. 23, no. 2, pp. 311_x0015_325, Feb. 2013

[2] W. Shi et al., ``Real-time single image and video super-resolution using an ef_x001C_cient sub-pixel convolutional neural network," in Proc. IEEE Conf. Compute. Vis. Pattern Recognit. (CVPR), Jun. 2016.

[3] C. Dong, C. C. Loy, K. He, and X. Tang, ``Image super-resolution using deep convolutional networks," IEEE Trans. Pattern Anal. Mach. Intell.,vol. 38, no. 2, pp. 295_x0015_307, Feb. 2015.

[4] C. N. E. Anagnostopoulos, I. E. Anagnostopoulos, I. D. Psoroulas,V. Loumos, and E. Kayafas, ``License plate recognition from still images and video sequences: A survey," IEEE Trans. Intell. Transp. Syst., vol. 9,no. 3, pp. 377_x0015_391, Sep. 2008.

[5] R. Smith, ``An overview of the Tesseract OCR engine," in *Proc. IEEE Int. Conf. Document Anal. Recognit.*, Sep. 2007, pp. 629_633.

[6] T. M. Breuel, ``The OCRopus open source OCR system," *Proc. SPIE,* vol. 6815, pp. 68150F-1_68150F_15, Jan. 2008.

[6] R. Szeliski, *Computer Vision: Algorithms and Applications*, 1st ed. New York, NY, USA: Springer-Verlag, 2010.

[7] P. Chatterjee, S. Mukherjee, S. Chaudhuri, and G. Seetharaman, ``Application of Papoulis_Gerchberg method in image super-resolution and inpainting," *Comput. J.*, vol. 52, no. 1, pp. 80_89, 2009.

[8]Y.-W. Tai, S. Liu, M. Brown, and S. Lin, ``Super resolution using edge prior and single image detail synthesis," in *Proc. IEEE Conf. Comput. Vis.Pattern Recognit.*, Jun. 2010, pp. 2400_2407.

[9] E. Meijering, ``A chronology of interpolation: From ancient astronomy to modern signal and image processing," *Proc. IEEE*, vol. 90, no. 3, pp. 319_342, Mar. 2002.

[10] S. C. Park, M. K. Park, and M. G. Kang, ``Super-resolution image reconstruction: A technical overview," *IEEE Signal Process. Mag.*, vol. 20, no. 3,pp. 21_36, May 2003.

[11] J. Yang, J. Wright, T. S. Huang, and Y. Ma, ``Image super-resolution via sparse representation," *IEEE Trans. Image Process.*, vol. 19, no. 11, pp. 2861_2873, Nov. 2010.

[12] R. Y. Tsai and T. S. Huang, ``Multiframe image restoration and registration," *Adv. Comput. Vis. Image Process.*, vol. 1, no. 2, pp. 317_339, 1984.

[13] K. Nasrollahi and T. B. Moeslund, ``Super-resolution: A comprehensive survey," *Mach. Vis. Appl.*, vol. 25, no. 6, pp. 1423_1468, 2014.