CHAPTER 1

1. INTRODUCTION

Vehicle identification is an essential stage in intelligent traffic systems. Nowadays vehicles play a very big role in transportation. Also the use of vehicles has been increasing because of population growth and human needs in recent years. Therefore, control of vehicles is becoming a big problem and much more difficult to solve. Vehicle Identification Systems are used for the purpose of effective control. There were several researches to identify vehicles like by license recognition, number-plate recognition, etc.

- Lotufo, Morgan and Johnson proposed automatic number-plate recognition using optical character recognition (OCR) techniques.
- The algorithm for this system is based on recognition of characters on the number-plate as it will be unique for each vehicle.

The software is designed in a planned way and using various technologies including various new facilities like using any of the DBMS software for storage of data. The software is programmed in such a way that it is having a centralized storage of the database. Using Vehicle's Identification System software, within a moment, all the details regarding the particular vehicle can be found out.

1.1 VEHICLE IDENTIFICATION SYSTEM (VIS)

VIS is an image processing system designed to help in recognition of the owner of the vehicles with the help of number-plate of vehicles. The system is based on OCR technique. This system helps in the functions like detection of vehicle owner, storing of information of vehicles in a database like name of owner, when was it purchase, address of owner, etc., for further processing or later enquiries.
The main objective is to build a system that will let us know the identity of a person the vehicle belongs to. The vehicles should have a sort of control as if at all any sort of traffic violations or we want to

know the person to whom the vehicle belongs then we can know about him/her by just passing the image of vehicle consisting the number-plate region. The system will provide to us almost all the essential details about the owner of the vehicle, thus helping people or authorities in different ways.

1.2 APPLICATIONS

This software will prove to be very useful in various ways such as various companies who provide their services in the stream of vehicle selling or vehicle maintenance by keep the details of the vehicle owners and physical details and verification about the vehicles. It can be used by police departments also for theft catching. It makes the data storage of vehicles in various streams, very easy and versatile.

- It can be used by police or investigating authorities.
- It can be used by security personnel in a society.
- It can be used by parking managing authorities.

Some of its notable features are:

- ❖ Easy to use and implement
- ❖ Convenient use
- ❖ More secure
- ❖ Less time consuming system.
- ❖ Vehicle Detection in efficient manner.

1.3 CHALLENGES

The challenge was to see if we can somehow improve the performance and accuracy. For a neatly scanned document, the character recognition process would be easy as pie.But when the case is, a receipt which is captured using a camera device, there would be problems like overexposure, underexposure, lighting condition varied throughout the image and many other worse conditions.

Its key weakness is probably its use of a polygonal approximation as input to the classifier instead of the raw outlines.

After considering the conditions and comparing different OCR software, we found that no OCR software was good enough to give good results due to underexposure or overexposure of the images and to make the condition worse the unequal distribution of lighting over the image.

1.4 RELATED WORK

A **vehicle identification number** (**VIN**), also called a **chassis number**, is a unique code, including a serial number, used by the automotive industry to identify individual motor vehicles, towed vehicles, motorcycles, scooters and mopeds, as defined in ISO 3833.VINs were first used in 1954. From 1954 to 1981, there was no accepted standard for these numbers, so different manufacturers used different formats. In 1981, the National Highway Traffic Safety Administration of the United States standardized the format. It required all over-the-road vehicles sold to contain a 17-character VIN, which does not include the letters I (i), O (o), and Q (q) (to avoid confusion with numerals 1 and 0).
The car's **vehicle identification number** (**VIN**) is the identifying code for a SPECIFIC automobile. The **VIN** serves as the car's fingerprint, as no two vehicles in operation have the same **VIN**. A **VIN** is composed of 17 characters (digits and capital letters) that act as a unique identifier for the vehicle.
VIN could be considered as the related work as it can be used in the same way by knowing the identity of the owner after checking the identity of vehicle owner against the VIN in the database.

1.5 PROPOSED SYSTEM

- Image of the vehicle number plate is taken and set as the input for the system.
- System reads the characters from the number plate using OCR engine.

- The readable characters are further processed by comparing them with the information in the database.
- Database contains all the details like vehicle number, name of vehicle owner, etc.If the same number plate characters are found in database on comparison then the owner name against that entry is displayed as desired output.

Optical Character Recognition is used to recognize characters from number plate. It is implemented in this system with the help of Open Source tool Google Tesseract.

CHAPTER 2

2. OPTICAL CHARACTER RECOGNITION WITH TESSERACT

2.1 Introduction

The mechanical or conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo (for example the text on signs and billboards in a landscape photo) or from subtitle text superimposed on an image.

Tesseract is an optical character recognition engine for various operating systems.Tesseract began as a PhD research project in HP Labs,Bristol.Tesseract is an open-source OCR engine that was developed at HP between 1984 and 1994. It is free software, released under the Apache License, Version 2.0, and development has been sponsored by Google since 2006. Tesseract is an open-source OCR engine that was developed at HP between 1984 and 1994.Tesseract is considered one of the most accurate open source OCR engines currently available. The Tesseract engine was originally developed as proprietary software at Hewlett Packard labs in Bristol, England between 1985 and 1994. Vehicle Identification System makes use of tesseract engine for Optical Character Recognition.

After a joint project between HP Labs Bristol, and HP's scanner division in Colorado, Tesseract had a significant lead in accuracy over the commercial engines, but did not become a product. The next stage of its development was back in HP Labs Bristol as an investigation of OCR for compression. Work concentrated more on improving rejection efficiency than on base-level accuracy.

Perfect rejection is shown to be achievable, but only at the cost of reduced classification accuracy in most practical situations.Most of the test results submitted for scoring were accompanied by confidence files, and most of the rest by rejection files. Rejection files contain integers from the set {0,1}, one integer per test-character image. A 1 indicates that the hypothetical classification should be scored as a reject rather than as correct or incorrect, and a 0 indicates that the classification should be scored as correct if identical to the correct classification, and incorrect otherwise. Each rejection file defines one point e(r) on the error rate e versus rejection rate r curve, so many rejection files per hypothesis file are needed to show the detailed shape of the curve.

$$e(r) = \frac{e(0) - f(r)}{1 - r}$$

At the end of this project,at the end of 1994, development ceased entirely. The engine was sent to UNLV for the 1995 Annual Test of OCR Accuracy, where it proved its worth against the commercial engines of the time. In late 2005, HP released Tesseract for open source.
It is now available at http://code.google.com/p/tesseract-ocr.

2.2 OCR Applications

OCR engines have been developed into many kinds of domain-specific OCR applications, such as receipt OCR, invoice OCR, check OCR, legal billing document OCR.

They can be used for:

- Data entry for business documents, e.g. check, passport, invoice, bank statement and receipt
- Automatic number plate recognition
- Automatic insurance documents key information extraction
- Extracting business card information into a contact list
- More quickly make textual versions of printed documents, e.g. book scanning for Project Gutenberg
- Make electronic images of printed documents searchable, e.g. Google Books

- Converting handwriting in real time to control a computer (pen computing)
- Defeating CAPTCHA anti-bot systems, though these are specifically designed to prevent OCR
- Assistive technology for blind and visually impaired users

2.3 OCR Techniques

2.3.1 Pre-processing

OCR software often "pre-processes" images to improve the chances of successful recognition. Techniques include:

- **De-skew** – If the document was not aligned properly when scanned, it may need to be tilted a few degrees clockwise or counterclockwise in order to make lines of text perfectly horizontal or vertical.
- **De-speckle** – remove positive and negative spots, smoothing edges
- **Binarisation** – Convert an image from color or greyscale to black-and-white (called a "binary image" because there are two colours). The task of binarisation is performed as a simple way of separating the text (or any other desired image component) from the background. The task of binarisation itself is necessary since most commercial recognition algorithms work only on binary images since it proves to be simpler to do so. In addition, the effectiveness of the binarisation step influences to a significant extent the quality of the character recognition stage and the careful decisions are made in the choice of the binarisation employed for a given input image type; since the quality of the binarisation method employed to obtain the binary result depends on the type of the input image (scanned document, scene text image, historical degraded document etc.).
- **Line removal** – Cleans up non-glyph boxes and lines
- **Layout analysis or "zoning"** – Identifies columns, paragraphs, captions, etc. as distinct blocks. Especially important in multi-column layouts and tables.
- **Line and word detection** – Establishes baseline for word and character shapes, separates words if necessary.
- **Script recognition** – In multilingual documents, the script may change at the level of the words and hence, identification

of the script is necessary, before the right OCR can be invoked to handle the specific script.

- **Character isolation or "segmentation"** – For per-character OCR, multiple characters that are connected due to image artifacts must be separated; single characters that are broken into multiple pieces due to artifacts must be connected.
- **Normalise aspect ratio and scale**

Segmentation of fixed-pitch fonts is accomplished relatively simply by aligning the image to a uniform grid based on where vertical grid lines will least often intersect black areas. For proportional fonts , more sophisticated techniques are needed because whitespace between letters can sometimes be greater than that between words, and vertical lines can intersect more than one character.

2.3.2 Character recognition

There are two basic types of core OCR algorithm, which may produce a ranked list of candidate characters.

Matrix matching involves comparing an image to a stored glyph on a pixel-by-pixel basis; it is also known as "pattern matching", "pattern recognition", or "image correlation". This relies on the input glyph being correctly isolated from the rest of the image, and on the stored glyph being in a similar font and at the same scale. This technique works best with typewritten text and does not work well when new fonts are encountered. This is the technique the early physical photocell-based OCR implemented, rather directly.

Feature extraction decomposes glyphs into "features" like lines, closed loops, line direction, and line intersections. These are compared with an abstract vector-like representation of a character, which might reduce to one or more glyph prototypes. General techniques of feature detection in computer vision are applicable to this type of OCR, which is commonly seen in "intelligent" handwriting recognition and indeed most modern OCR software. Nearest neighbour classifiers such as the k-nearest neighbors algorithm are used to compare image features with stored glyph features and choose the nearest match.

Software such as Cuneiform and Tesseract use a two-pass approach to character recognition. The second pass is known as "adaptive recognition" and uses the letter shapes recognised with high confidence on the first pass to recognise better the remaining letters on the second pass. This is advantageous for unusual fonts or low-quality scans where the font is distorted (e.g. blurred or faded).

The OCR result can be stored in the standardised ALTO format, a dedicated XML schema maintained by the United States Library of Congress.

2.3.3 Post-processing

OCR accuracy can be increased if the output is constrained by a lexicon – a list of words that are allowed to occur in a document. This might be, for example, all the words in the English language, or a more technical lexicon for a specific field. This technique can be problematic if the document contains words not in the lexicon, like proper nouns. Tesseract uses its dictionary to influence the character segmentation step, for improved accuracy.

The output stream may be a plain text stream or file of characters, but more sophisticated OCR systems can preserve the original layout of the page and produce, for example, an annotated PDF that includes both the original image of the page and a searchable textual representation.

"Near-neighbor analysis" can make use of co-occurrence frequencies to correct errors, by noting that certain words are often seen together. For example, "Washington, D.C." is generally far more common in English than "Washington DOC".

Knowledge of the grammar of the language being scanned can also help determine if a word is likely to be a verb or a noun, for example, allowing greater accuracy.

The Levenshtein Distance algorithm has also been used in OCR post-processing to further optimize results from an OCR API.

2.3.4 Application-specific optimizations

In recent years, the major OCR technology providers began to tweak OCR systems to better deal with specific types of input. Beyond an application-specific lexicon, better performance can be had by taking into account business rules, standard expression, or rich information contained in color images. This strategy is called "Application-Oriented OCR" or "Customized OCR", and has been applied to OCR of license plates, business cards, invoices, screenshots, ID cards, driver licenses, and automobile manufacturing.

2.3.5 OCR Tools

Google, ABBYY, Adobe Acrobat, LEAD Technologies and ScanSnap provide tools that can extract text from images or convert images into text-searchable document formats. For any project related to paperless offices, the use of OCR tools will be required to achieve the objectives of paperless offices and homes.

2.4 ARCHITECTURE

Since HP had independently-developed page layout analysis technology that was used in products, (and therefore not released for open-source) Tesseract never needed its own page layout analysis. Tesseract therefore assumes that its input is a binary image with optional polygonal text regions defined.

Processing follows a traditional step-by-step pipeline. The first step is a connected component analysis in which outlines of the components are stored. At this stage, outlines are gathered together, purely by nesting, into Blobs.Connected-component labeling (alternatively connected-component analysis, blob extraction, region labeling, blob discovery, or region extraction) is an algorithmic application of graph theory, where subsets of connected components are uniquely labeled based on a given heuristic. Connected-component labeling is not to be confused with segmentation.

Connectivity checks are carried out by checking neighbor pixels' labels (neighbor elements whose labels are not assigned yet are ignored), or say, the North-East, the North, the North-West and the West of the current pixel (assuming 8-connectivity). 4-connectivity uses only North and West neighbors of the current pixel. The following conditions are checked to determine the value of the label to be assigned to the current pixel (4-connectivity is assumed)

Conditions to check:

1. Does the pixel to the left (West) have the same value as the current pixel?
    1. Yes – We are in the same region. Assign the same label to the current pixel
    2. No – Check next condition
2. Do both pixels to the North and West of the current pixel have the same value as the current pixel but not the same label?
    1. Yes – We know that the North and West pixels belong to the same region and must be merged. Assign the current pixel the minimum of the North and West labels, and record their equivalence relationship
    2. No – Check next condition
3. Does the pixel to the left (West) have a different value and the one to the North the same value as the current pixel?
    1. Yes – Assign the label of the North pixel to the current pixel
    2. No – Check next condition
4. Do the pixel's North and West neighbors have different pixel values than current pixel?
    1. Yes – Create a new label id and assign it to the current pixel

The algorithm continues this way, and creates new region labels whenever necessary. The key to a fast algorithm, however, is how this merging is done. This algorithm uses the union-find data structure which provides excellent performance for keeping track of equivalence relationships.

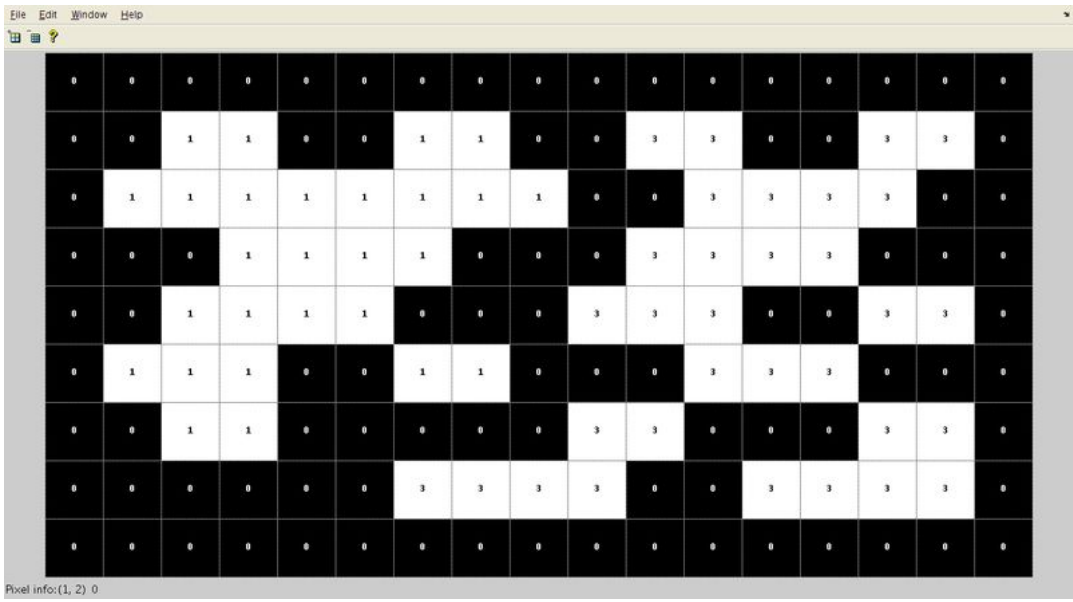

Fig. 2.4.1 Connected Component Analysis

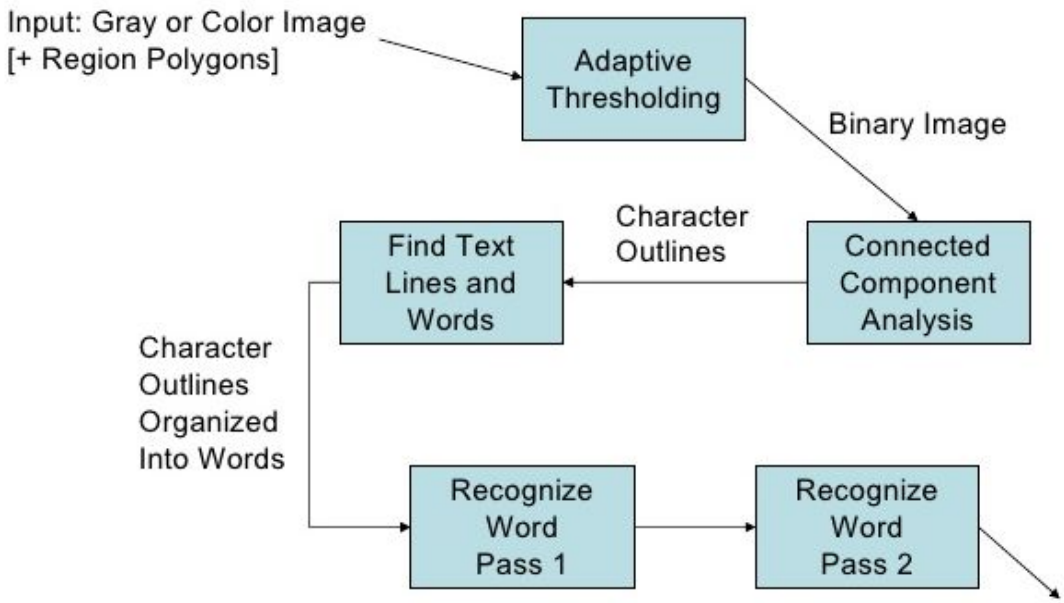

Fig. 2.4.2 Tesseract Architecture

Blobs are organized into text lines, and the lines and regions are analyzed for fixed pitch or proportional text. Text lines are broken into words differently according to the kind of character spacing. Fixed pitch text is chopped immediately by character cells. Proportional text is broken into words using definite spaces and fuzzy spaces.

Recognition then proceeds as a two-pass process. In the first pass, an attempt is made to recognize each word in turn. Each word that is satisfactory is passed to an adaptive classifier as training data. The adaptive classifier then gets a chance to more accurately recognize text lower down the page.

Since the adaptive classifier may have learned something useful too late to make a contribution near the top of the page, a second pass is run over the page, in which words that were not recognized well enough are recognized again.

A final phase resolves fuzzy spaces, and checks alternative hypotheses for the x-height to locate small cap text. Fig. 2.4.2 shows Tesseract Architecture.

2.5 FIXED PITCH DETECTION AND CHOPPING

Tesseract tests the text lines to determine whether they are fixed pitch. Where it finds fixed pitch text, Tesseract chops the words into characters using the pitch, and disables the chopper and associator on these words for the word recognition step. Fig. 2.5 shows a typical example of a fixed-pitch word.

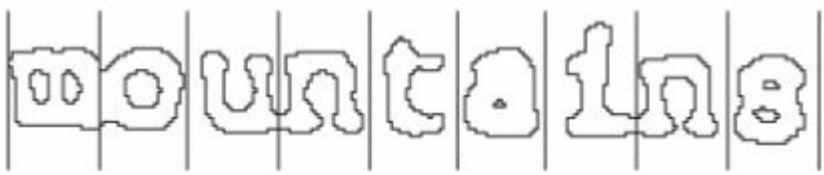

Fig. 2.5 A fixed-pitch chopped word.

2.6 Word Recognition

Part of the recognition process for any character recognition engine is to identify how a word should be segmented into characters. The initial segmentation output from line finding is classified first. The rest of the word recognition step applies only to non-fixed pitch text.

2.7 Chopping Joined Characters

Tesseract attempts to improve the result by chopping the blob with worst confidence from the character classifier. Candidate chop points are found from concave vertices of a polygonal approximation of the outline, and may have either another concave vertex opposite, or a line segment. It may take up to 3 pairs of chop points to successfully separate joined characters from the ASCII set.

Fig.2.7 shows a set of candidate chop points with arrows and the selected chop as a line across the outline where the 'r' touches the 'm'.

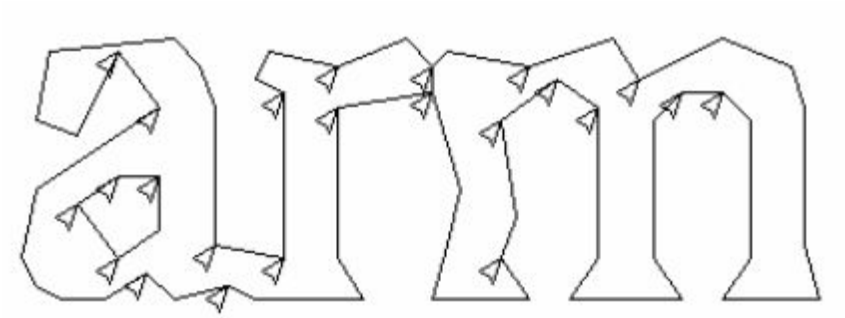

Fig. 2.7 Candidate Chop points and chop

Chops are executed in priority order. Any chop that fails to improve the confidence of the result is undone, but not completely discarded so that the chop can be re-used later by the associator if needed.

2.8 Associating Broken Characters

When the potential chops have been exhausted, if the word is still not good enough, it is given to the associator. The associator makes an A* (best first) search of the segmentation graph of possible combinations of the maximally chopped blobs into candidate characters. It does this without actually building the segmentation graph, but instead maintains a hash table of visited states. The A* search proceeds by pulling candidate new states from a priority queue and evaluating them by classifying unclassified combinations of fragments.

It may be argued that this fully-chop-then-associate approach is at best inefficient, at worst liable to miss important chops, and that may well be the case. The advantage is that the chop-then-associate scheme simplifies the data structures that would be required to maintain the full segmentation graph.

When the A* segmentation search was first implemented in about 1989, Tesseract's accuracy on broken characters was well ahead of the commercial engines of the day. Fig. 2.8 is a typical example. An essential part of that success was the character classifier that could easily recognize broken characters.

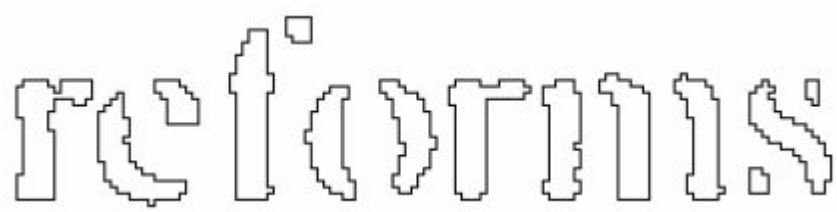

Fig. 2.8 An easily recognized word

The features extracted from the unknown are thus 3-dimensional, (x, y position, angle), with typically 50- 100 features in a character, and the prototype features are 4-dimensional (x, y, position, angle, length), with typically 10-20 features in a prototype configuration.

2.9 Static Character Classifier

It is based on a classical 1-NN algorithm and operates on retinal images of the full character whereas the dynamic one works on stroke or strokes' combination.Obviously, its robustness is lesser than the dynamic's, owing to the clustering method adaptation to on-line data.

2.9.1 Features

An intermediate idea involved the use of segments of the polygonal approximation as features, but this approach is also not robust to damaged characters. For example, in Fig. 2.9(a), the right side of the shaft is in two main pieces, but in Fig. 2.9(b) there is just a single piece.

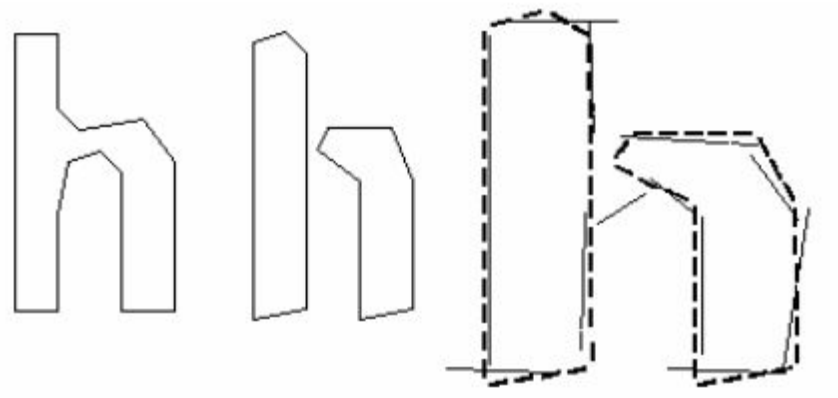

Fig. 2.9.1 (a) Pristine ‘h, (b) broken ‘h’,
(c)features matched to prototypes.

During training, the segments of a polygonal approximation are used for features, but in recognition, features of a small, fixed length (in normalized units) are extracted from the outline and matched many-to-one against the clustered prototype features of the training data. In Fig. 2.9.1(c), the short, thick lines are the features extracted from the unknown, and the thin, longer lines are the clustered
segments of the polygonal approximation that are used as prototypes.

2.9.2 Classification

Classification proceeds as a two-step process. In the first step, a class pruner creates a shortlist of character classes that the unknown might match. Each feature fetches, from a coarsely quantized 3-dimensional lookup table, a bit-vector of classes that it might match, and the bit-vectors are summed over all the features. The classes with the highest counts (after correcting for expected number of features) become the short-list for the next step.

Each feature of the unknown looks up a bit vector of prototypes of the given class that it might match, and then the actual similarity between them is computed. Each prototype character class is represented by a logical sum-of-product expression with each term called a configuration, so the distance calculation process keeps a record of the total similarity evidence of each feature in each configuration, as well as of each prototype. The best combined distance, which is calculated from the summed feature and prototype evidences, is the best over all the stored configurations of the class.

2.9.3 Training Data

Since the classifier is able to recognize damaged characters easily, the classifier was not trained on damaged characters. In fact, the classifier was trained on a mere 20 samples of 94 characters from 8 fonts in a single size, but with 4 attributes (normal, bold, italic, bold italic), making a total of 60160 training samples.

2.10 Linguistic Analysis

Tesseract contains relatively little linguistic analysis. Whenever the word recognition module is considering a new segmentation, the linguistic module (mis-named the permuter) chooses the best available word string in each of the following categories: Top

frequent word, top dictionary word, top numeric word, top uppercase word, top lowercase word (with optional initial upper), Top classifier choice word. The final decision for a given segmentation is simply the word with the lowest total distance rating, where each of the above categories is multiplied by a different constant.

2.11 Adaptive Classifier

OCR engines can benefit from the use of an Adaptive classifier. Since the static classifier has to be good at generalizing to any kind of font, its ability to discriminate between different characters or between characters and non-characters is weakened. A more font-sensitive adaptive classifier that is trained by the output of the static classifier is therefore commonly used to obtain greater discrimination within each document, where the number of fonts is limited.

Tesseract does not employ a template classifier, but uses the same features and classifier as the static classifier. The only significant difference between the static classifier and the adaptive classifier, apart from the training data, is that the adaptive classifier uses isotropic baseline/x-height normalization, whereas the static classifier normalizes characters by the centroid (first moments) for position and second moments for anisotropic size normalization.

The baseline/x-height normalization makes it easier to distinguish upper and lower case characters as well as improving immunity to noise specks. The main benefit of character moment normalization is removal of font aspect ratio and some degree of font stroke width. It also makes recognition of sub and superscripts simpler, but requires an additional classifier feature to distinguish some upper and lower case characters.

Fig. 2.11 Baseline and moment normalized letters.

CHAPTER 3

3. VEHICLE IDENTIFICATION SYSTEM USING OCR

The system has a set of input images and desirable output. For VIS, the input and output is described in the section.
The system involves use of :

- User
- Database
- Tesseract (OCR Engine)

3.1 INPUT:-

The image of vehicle number plate region is set as the input for the system. User will provide a set of images and that will be the input for the system. The characters are recognized from the vehicle number plate region.The image below shows input i.e number plate region image.

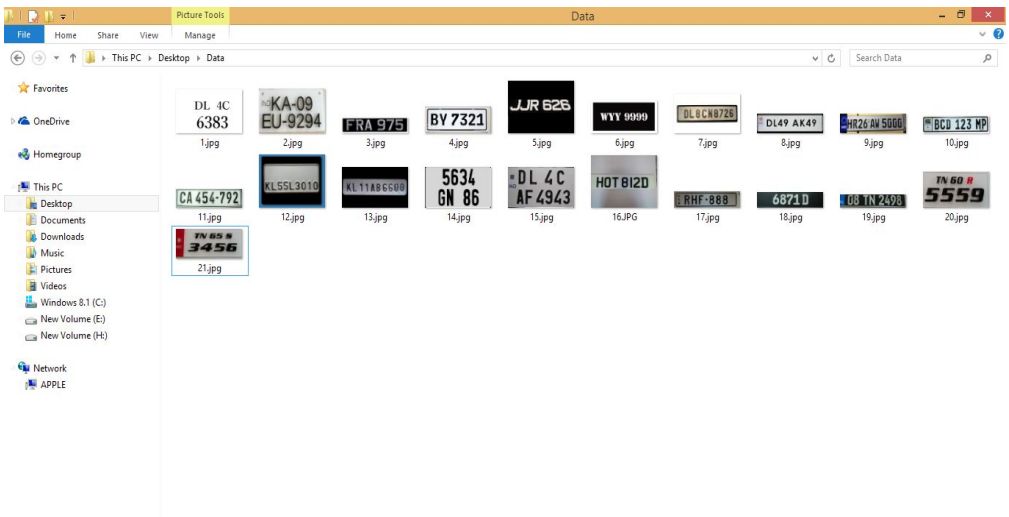

Fig. 3.1.1 Input images

Fig. 3.1.2 Input image

3.2 OUTPUT

a. The output of the system will be the name of the vehicle owner.
b. It will be obtained by matching the vehicle number against the name of the vehicle owner in the database.
c. The edit distance, similarity score of the system is also calculated.
d. Accuracy is evaluated finally for the system.
e. System thus provide an efficient way to know the vehicle owner.

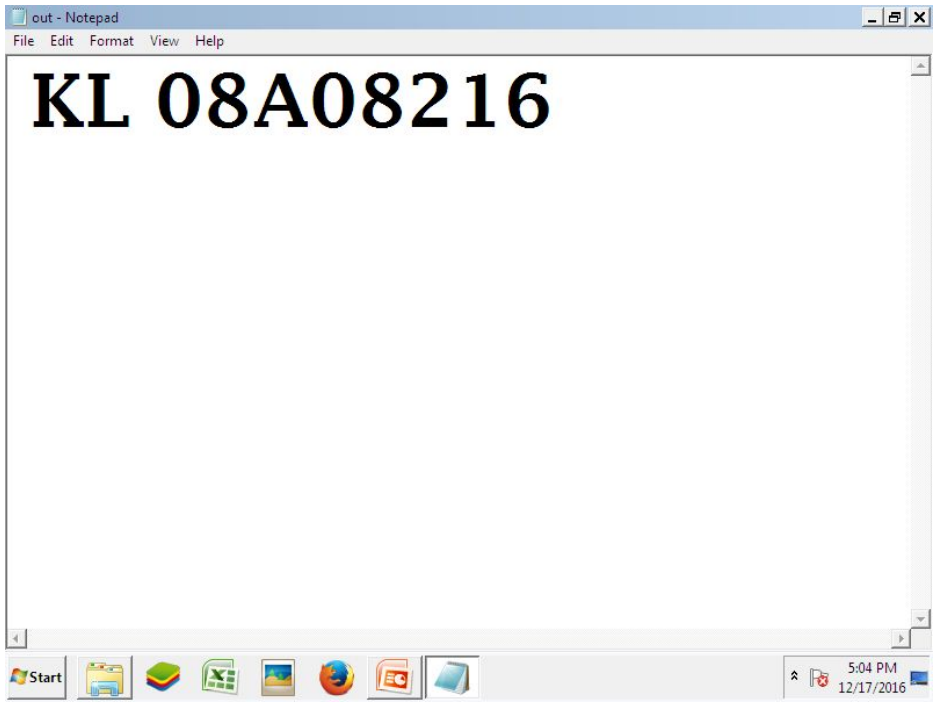

Fig. 3.2. Output text file

3.3 FLOWCHART

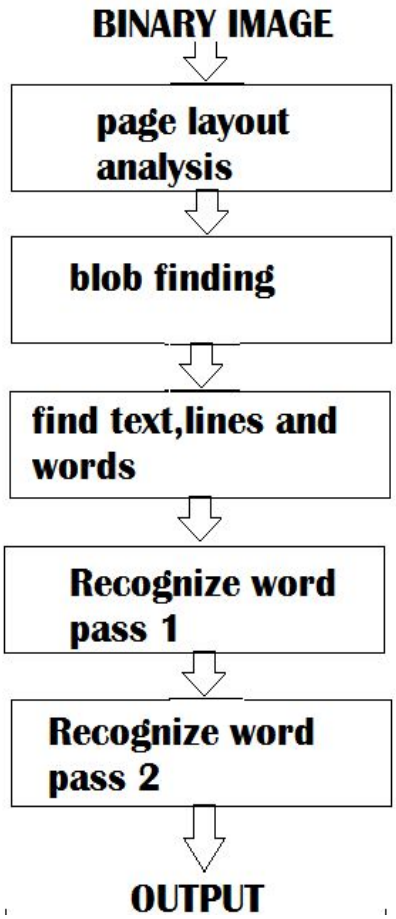

Fig. 3.3 Flowchart for character recognition

CHAPTER 4

4. EXPERIMENTS AND RESULTS

4.1. Dataset

Set of 21 number plate region images are collectively considered as the dataset for the system. The dataset are used for character recognition and thus further accuracy of the system is evaluated. The dataset is collected by taking images using camera and some are downloaded from the web.

4.2. Metrics used

Word error rate (WER) is a common metric of the performance of a speech recognition or machine translation system.

The general difficulty of measuring performance lies in the fact that the recognized word sequence can have a different length from the reference word sequence (supposedly the correct one). The WER is derived from the Levenshtein distance, working at the word level instead of the phoneme level.

Levenshtein distance (LD) is a measure of the similarity between two strings, which we will refer to as the source string (s) and the target string (t). The distance is the number of deletions, insertions, or substitutions required to transform s into t. For example,

- If s is "test" and t is "test", then LD(s,t) = 0, because no transformations are needed. The strings are already identical.
- If s is "test" and t is "tent", then LD(s,t) = 1, because one substitution (change "s" to "n") is sufficient to transform s into t.

The greater the Levenshtein distance, the more different the strings are.

Levenshtein distance is named after the Russian scientist Vladimir Levenshtein, who devised the algorithm in 1965. The metric is also sometimes called edit distance.

The Levenshtein distance algorithm has been used in:

- Spell checking
- Speech recognition
- DNA analysis
- Plagiarism detection

The WER is a valuable tool for comparing different systems as well as for evaluating improvements within one system. This kind of measurement, however, provides no details on the nature of translation errors and further work is therefore required to identify the main source(s) of error and to focus any research effort.

This problem is solved by first aligning the recognized word sequence with the reference (spoken) word sequence using dynamic string alignment. Examination of this issue is seen through a theory called the power law that states the correlation between perplexity and word error rate.

- There are some cases where we get some word errors.
- Word error rate can then be computed as:

$$WER = \frac{S + D + I}{N} = \frac{S + D + I}{S + D + C}$$

where

S is the number of substitutions,
D is the number of deletions,
I is the number of insertions,
C is the number of the corrects,
N is the number of words in the reference (N=S+D+C)

Word Accuracy: (WAcc) is used as :

$$WAcc = 1 - WER = \frac{N - S - D - I}{N} = \frac{H - I}{N}$$

where

H is N-(S+D), the number of correctly recognized words.

These are the metrics used to check the accuracy of the characters identified by OCR and their conversion to standard form by insertion, deletion and substitution techniques.
Accuracy of the system thus obtained by evaluation of above terms is 84.23 %.

4.3. Screenshots

Fig.4.3.1 Set of images used

Fig.4.3.2 Sample Input

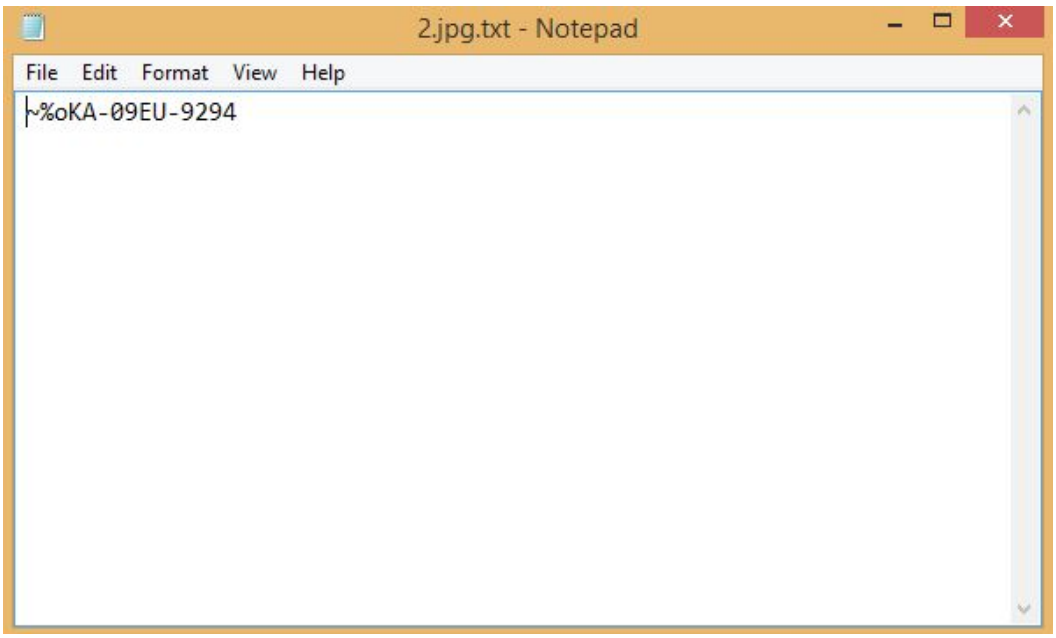

Fig.4.3.3 Output without insertion, substitution or deletion

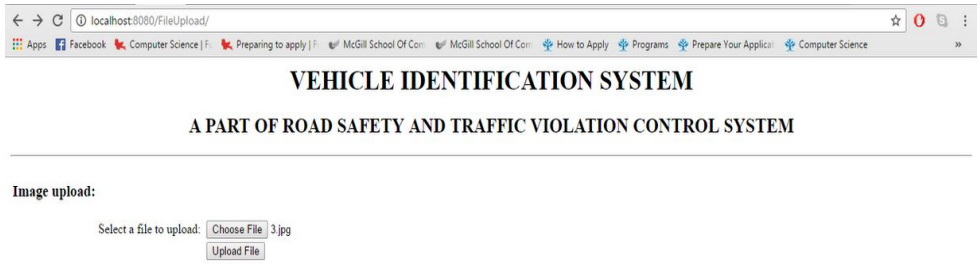

Fig.4.3.4 Web Interface to upload input image

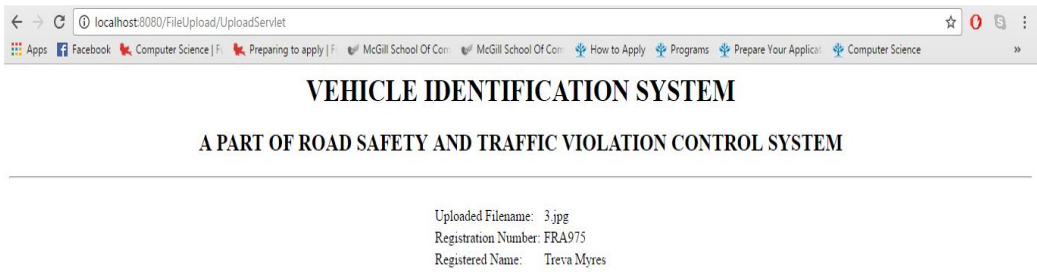

Fig.4.3.5 Output in web interface

Fig.4.3.6 Output in command line along with similarity score and distance

CHAPTER 5

5. CONCLUSION AND FUTURE WORK

5.1 Conclusion

The Proposed Vehicle Identification System is effective to know the identity of vehicle owner when provided with the image of vehicle number plate region. Thus, by all the respective experiments and observations; we conclude the system to 82.3 % accurate.

Its key strength is probably its unusual choice of features. Its key weakness is probably its use of a polygonal approximation as input to the classifier instead of the raw outlines.

5.2 Future Work

For the future related work, we can create a system that captures the image of the vehicles automatically so that they can be used with traffic lights and if at all any person violates traffic rule, the image of vehicle is taken and further processed to know the identity of that person.

With internationalization done, accuracy could n-gram model,probably be improved significantly with the judicious addition of a Hidden-Markov-Model-based character n-gram model, and possibly an improved chopper.

References

- https://en.wikipedia.org/wiki/Connected-component_labeling
- http://ws680.nist.gov/publication/get_pdf.cfm?pub_id=906272
- https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/33418.pdf
- http://research.google.com/pubs/archive/35248.pdf
- http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=620549
- https://en.wikipedia.org/wiki/Word_error_rate
- https://en.wikipedia.org/wiki/Optical_character_recognition
- http://people.cs.pitt.edu/~kirk/cs1501/Pruhs/Fall2006/Assignments/editdistance/Levenshtein%20Distance.htm