CHAPTER TWO

LITERATURE REVIEW

2.1 Overview

This chapter is a brief background of License Plates, Optical Characters Recognition, Open Source Automatic License Plate Recognition library (OpenALPR), Automatic License Plate Recognition algorithm (ALPR), applications of ALPR, SQL Database, Graphics user interface (GUI) and some projects or papers that is related to LPR.

2.2 Background of License Plates

License plates (LP) have different size, base material, character format and color standards in all the world. Generally, license plates are characterized by high contrast between characters and underlying background.

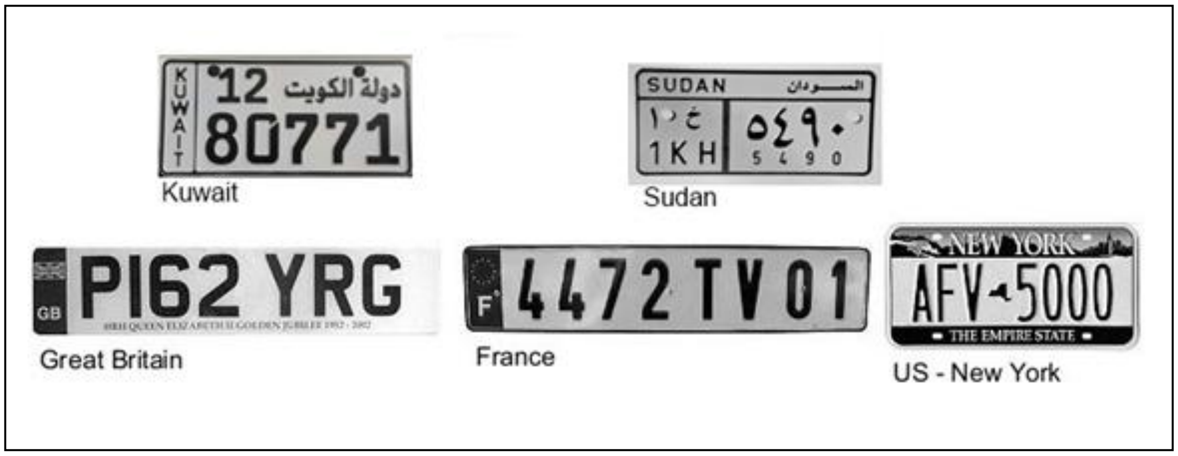

Figure 2-1: Illustrates example plate images from different countries

2.3 License Plate Recognition System (LPR System)

LPR is an image-processing technology used to identify vehicles by their license plates. It uses Optical Character Recognition (OCR) to read the LP. Automatic Number Plate Recognition (ANPR) was invented first time in 1976 at a police station situated in United Kingdom. Prototype systems started working in 1979, and contracts were awarded producing commercial systems. This technology is used in various security and traffic application [1]. In different references this technology is also referred as Automatic Vehicle Identification, Car Plate Recognition, Automatic Number Plate Recognition, Car Plate Reader or optical character recognition.

Most LPR systems focus on the processing of images with only one vehicle others are able to process more vehicles at once [2]. The techniques of the license plate recognition have been developed for many years. With the hardware and software improved, the accuracy of LPR has been improved to some extent. But the current accuracy still cannot satisfy all the requirements of the traffic department concerned. The license plate extraction is not a simple process, as under some situations, it is easy for people to judge but it will be very hard for computer to do it [3].

Early LPR systems suffered from a low recognition rate, lower than required by practical systems. The external effects (sun and headlights, bad plates, wide number of plate’s types) and the limited level of the recognition software and vision hardware yielded low quality systems. Image enhancement technique is very crucial based on filters to remove noise and unwanted effects of the light in order to obtain clear and readable images, are used. Recent improvements in technology like infrared imaging and high resolution cameras, and utilization of high reflective backgrounds in license plate. Manufacturing have improved the accuracy of LPR systems, Sensors and other hardware peripherals are used to improve the image acquisition and remove irrelevant details.

2.3.1 LPR Challenges

Many difficulties occur during the detection and extraction of number plate due to the following reasons:

- What part of the vehicle is actually the license plate.
- The efficiency of extraction is affected by scene complexity.
- Different vehicles have plates located on different position.
- Different countries, states, cities and regions have different standards, dimensions, colors and character sets for license plates.
- Noise can occur during camera capture.
- Weather conditions responsible for noise arrival.
- Time of day affects lighting thus resulting into contras problems.
- Unwanted characters, frames and screws introduce confusion.
- Wrong camera or plate position result into distortion that affect efficiency plate position result into distortion that efficiency of plate extraction [4].

The complexity of each of these subsections of the challenges determines the accuracy of the system. This inconsistency requires algorithms to be inclusive to such extensive criteria. OpenALPR library have been introduced to solve these challenges.

2.4 ALPR System

Automated License Plate Recognition (ALPR) is a technology that uses optical character recognition (OCR) to automatically recognize license plate characters.

ALPR is known by several other names, including Automatic Number Plate Recognition (ANPR), Automatic Vehicle Identification (AVI), Car Plate Recognition (CPR), License Plate Recognition (LPR).

2.4.1 Types of ALPR

There are two types of ALPR:

- Stationary, which uses infrared (IR) cameras at high fixed points.
- Mobile, which uses vehicle-mounted IR cameras.

Stationary cameras can be mounted on signs, street lights, highway overpasses or buildings as a cost-effective way to monitor moving and parked vehicles twenty-four hours a day.

Camera software which is able to identify the pixel patterns that make up a license plate and translate the letters and numbers on the plate to a digital format. The plate data is then sent to a database where it is compared in real-time to a list of plate numbers that belong to "vehicles of interest". If the system detects a match, it sends an alert to the dispatcher or other designated personnel [5].

Mobile ALPR software suites use multiple cameras mounted on a vehicle. As the vehicle moves, it photographs license plates and transmits plate data to a database. The database may be a national database or it may be created at the local level and downloaded into the vehicle's onboard computer at the beginning of each shift. If the system detects a match, the officer receives an alert on his computer. A mobile ALPR can read up to 1,000 plates per hour [5].

Traditionally, mobile ALPR systems require dedicated hardware and specially designed license plate reader cameras to work. Then Smart Plate came along and introduced a mobile ALPR software compatible with any surveillance equipment, meaning you can add license plate reader capability to your existing cameras and VMS systems [5].

2.4.2 Uses of ALPR

Automated License Plate Recognition has a wide range of applications, which use the extracted plate number and optional images to create automated solution for various problems. These include the following sample application:

- Identifying drivers with an open warrant for arrest.

- Catching speeders by comparing the average time it takes to get from stationary camera A to stationary camera B.
- Parking: the recognition of license plates is used to calculate the duration in which the car has been parked. When a vehicle arrives at the entrance to the parking, the registration number is automatically recognized and stored in the database. When the vehicle leaves the parking later and reaches the door, the registration number of the plate is recognized again and compared to the first stored in the database. The time difference is used to calculate the cost of parking. This technology is used in some companies to grant access only to authorized personnel vehicles.
- Access Control: a gate automatically opens for authorized members in a secured area, thus replacing or assisting the security guards. The events are logged on a database and could be used to search the history of events.
- Tolling: the car number is used to calculate the travel fee in a toll-road, or used to double-check the ticket.
- Border Control: the car number is registered in the entry or exits to the Country, and used to monitor the border crossings. It can short the border crossing turnaround time and cut short the typical long lines
- Stolen cars: a list of stolen cars or unpaid fines is used to alert on a passing 'hot' cars. The 'black list' can be updated in real time and provide immediate alarm to the police force. The LPR system is deployed on the roadside or can be attached on traffic police cars, and performs a real-time match between the passing cars and the list. When a match is found a siren or display is activated and the police

officer is notified with the detected car and the reasons for stopping the car.

- Enforcement: the plate number is used to produce a violation fine on speed or red-light systems, the manual process of preparing a violation fine is replaced by an automated process which reduces the overhead and turnaround time. The fines can be viewed and paid on-line
- Traffic control: the vehicles can be directed to different lanes according to their entry permits (such as in University complex projects). The system effectively reduces traffic congestions and the number of attendants
- Marketing Tool: the car plates may be used to compile a list of frequent visitors for marketing purposes, or to build a traffic profile (such as the frequency of entry verses the hour or day).
- Travel: A number of LPR units are installed in different locations in city routes and the passing vehicle plate numbers are matched between the points. The average speed and travel time between these points can be calculated and presented in order to monitor municipal traffic loads. Additionally, the average speed may be used to issue a speeding ticket [6].

2.5 OpenALPR Design

OpenALPR operates as a pipeline. The input is an image, various processing occurs in stages, and the output is the possible plate numbers in the image [7].

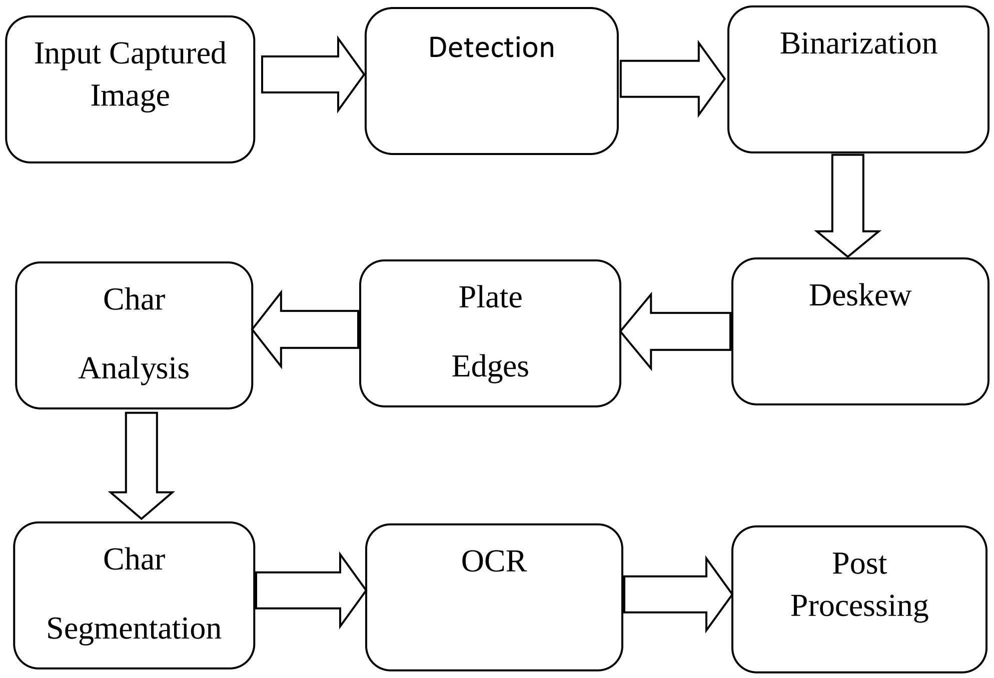

Figure 2-2: Flow Chart of the ALPR System algorithm

There are six primary algorithms that the software requires for identifying a license plate:

1- Detection: Finds potential license plate regions.

2- Binarization: Converts the plate region image into black and white.

3- Char Analysis: Finds character-sized "blobs" in the plate region.

4- Plate Edges: Finds the edges/shape of the license plate.

5- Deskew: Transforms the perspective to a straight-on view based on the ideal license plate size.

6- Character Segmentation: Isolates and cleans up the characters so that they can be processed individually.

7- OCR: Analyzes each character image and provides multiple possible letters/confidences.

8- Post processing: Creates a top N list of plate possibilities based on OCR confidences. Also performs a Region match against region templates if requested.

2.5.1 Detection

The detection phase happens one time for each input image. It uses the Local Binary Pattern (LBP) algorithm to find possible license plate regions (x, y, width, height). Each of these regions is sent to the later pipeline phases for further processing.

By whole capturing image we having license plate covered by background of vehicle’s body. So by this step only plate are is extracted from whole body.

The task now is to identify the region containing the license plate. In this experiment, two features are defined and extracted in order to decide if a candidate region contains a license plate or not.

2.5.1.1 Detector training

The license plate region detector uses the LBP algorithm. In order to train the detector, a set of samples are needed. There are two types of samples: negative and positive. Negative samples correspond to non-object images. Positive samples correspond to images with detected objects. Many positive and negative images are required (hundreds to thousands).

A configuration file has to be changed before starting the training process. The WIDTH, HEIGHT, and COUNTRY variables to match the country that is training for. The width and height should be proportional to the plate size [8].

2.5.2 Binarization

This phase (and all subsequent phases) occur multiple times, once for each possible license plate region.

The binarization phase creates multiple binary images for each plate region. The reason multiple binary images are used is to give the best possible chance of finding all the characters. A single binarized image may miss characters if the image is too dark or too light for example. Binarization uses the Wolf-Jolien method as well as the Sauovola method with various parameters. Each of the binary images are processed in subsequent phases.

2.5.3 Character Analysis

Character analysis attempts to find character-sized regions in the plate region. It does this by first finding all connected blobs in the license plate region. Then it looks for blobs that are roughly the width and height of a license plate character and have tops/bottoms that are in a straight line with other blobs of similar width/height.

This analysis is done multiple times in the region. It starts by looking for small characters, then gradually looks for larger characters.

If nothing is found in the region, then the region is thrown out and no further processing takes place. If it finds some potential characters, then the character region is saved and further processing takes place.

2.5.4 Plate Edges

The next phase is to find the edges of the license plate. The detection phase is only responsible for identifying a possible region where a license plate may exist. It often is going to provide a region that is a little larger or smaller than the actual plate. The plate edges try to find the precise top/bottom/left/right edges of the license plate.

The first step is to find all of the though lines for the license plate region. platelines.cpp processes the plate image and computes a list of horizontal and vertical lines.

Plate corners uses this list as well as the character height (computed in Character Analysis) to find the likeliest plate line edges. It uses a number of configurable weights to determine which edge makes the most sense. It will try using a default edge (based on the ideal width/height of the plate) to see if that makes a good match.

2.5.5 Deskew

Given the plate edges, the deskew stage remaps the plate region to a standard size and orientation. Ideally this will give us a correctly oriented plate image (no rotation or skew).

2.5.6 Character Segmentation

The character segmentation phase tries to isolate all the characters that make up the plate image. It uses a vertical histogram to find gaps in the plate characters. This phase also cleans up the character boxes by removing small, disconnected speckles and disqualifying character regions that are not tall enough. It also tries to remove "edge" regions so that the edge of the license plate doesn't inappropriately get classified as a '1' or an 'I', This step characters on license plate are segmented and identified. This step is the most important step in license plate recognition because all further steps rely on it. This is the second major part of the License Plate detection algorithm.

There are many factors that cause the character segmentation task difficult, such as image noise, plate frame, rivet, space mark, Plate rotation and illumination variance. We here propose the algorithm that is quite robust and gives significantly good results on images having the above mentioned problems. In the segmentation of plate characters, license plate is segmented into its constituent parts obtaining the characters individually. Firstly, image is filtered for enhancing the image and removing the noises and unwanted spots. Then dilation operation is applied to the image for separating the characters from each other if the characters are close to each other. After this operation, horizontal and vertical smearing are applied for finding the character regions.

The next step is to cut the plate characters. It is done by finding starting and end points of characters in horizontal direction.

2.5.7 OCR

The OCR phase analyses each character independently. For each character image, it computes all possible characters and their confidences.

2.5.7.1 OCR’s History

As early as 1929, Tausheck obtained a patent named “Reading Machine” in Germany this patent reflects the basic concept of today’s OCR. The principle of Tausheck’s patent is template matching which is still used in some applications even today.

By 1950 the technological revolution was moving forward at a high speed, and electronic data processing was becoming an important field data entry was performed through punched cards and cost-effective way of handling the increasing amount of data was needed. At the same time technology for machine reading was becoming sufficiently mature for application, and by the middle of the 1950’s OCR machines became commercially available. The first true OCR reading machine was installed at Reader's Digest in 1954. This equipment was used to convert typewritten sales reports into punched cards for input to the computer [9]

2.5.7.2 Tesseract OCR

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, but some of the stages were unusual in their day, and possibly remain so even now. The first step is a connected component analysis in which outlines of the components are stored. This was a computationally expensive design decision at the time, but had a significant advantage: by inspection of the nesting of outlines, and the number of child and grandchild outlines, it is simple to detect inverse text and recognize it as easily as black-on-white text. Tesseract was probably the first OCR engine able to handle white-on-black text so trivially. At this stage, outlines are gathered together, purely by nesting, into Blobs. 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 [10].

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

2.5.7.3 Line and Word Finding

2.5.7.3.1 Line Finding

The line finding algorithm is one of the few parts of Tesseract that has previously been published [11]. The line finding algorithm is designed so that a skewed page can be recognized without having to de-skew, thus saving loss of image quality. The key parts of the process are blob filtering and line construction.

Assuming that page layout analysis has already provided text regions of a roughly uniform text size, a simple percentile height filter removes drop-caps and vertically touching characters. The median height approximates the text size in the region, so it is safe to filter out blobs that are smaller than some fraction of the median height, being most likely punctuation, diacritical marks and noise. The filtered blobs are more likely to fit a model of non-overlapping, parallel, but sloping lines. Sorting and processing the blobs by x-coordinate makes it possible to assign blobs to a unique text line, while tracking the slope across the page, with greatly reduced danger of assigning to an incorrect text line in the presence of skew. Once the filtered blobs have been assigned to lines, a least median of squares fit [12] is used to estimate the baselines, and the filtered-out blobs are fitted back into the appropriate lines.

The final step of the line creation process merges blobs that overlap by at least half horizontally, putting diacritical marks together with the correct base and correctly associating parts of some broken characters.

2.5.7.3.2 Baseline Fitting

Once the text lines have been found, the baselines are fitted more precisely using a quadratic spline. This was another first for an OCR system, and enabled Tesseract to handle pages with curved baselines [13],which are a common artifact in scanning, and not just at book bindings.

The baselines are fitted by partitioning the blobs into groups with a reasonably continuous displacement for the original straight baseline. A quadratic spline is fitted to the most populous partition, (assumed to be the baseline) by a least squares fit. The quadratic spline has the advantage that this calculation is reasonably stable, but the disadvantage that discontinuities can arise when multiple spline segments are required. Amore traditional cubic spline [14] might work better.

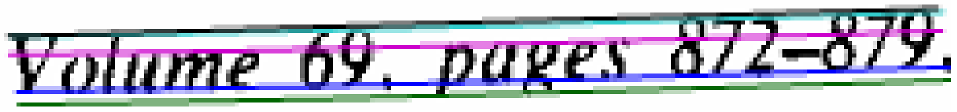

Figure 2-3:.An example of a curved fitted baseline.

Figure 2-3 shows an example of a line of text with a fitted baseline, descender line, meanline and ascender line.

All these lines are "parallel" (the y separation is a constant over the entire length) and slightly curved. The ascender line is cyan (prints as light gray) and the black line above it is actually straight. Close inspection shows that the cyan/gray line is curved relative to the straight black line above it.

2.5.7.3.3 Proportional Word Finding

Non-fixed-pitch or proportional text spacing is a highly non-trivial task. Figure 2-4 illustrates some typical problems. The gap between the tens and units of '11.9%' is a similar size to the general space, and is certainly larger than the kerned space between 'erated' and 'junk'. There is no horizontal gap at all between the bounding boxes of 'of' and 'financial'. Tesseract solves most of these problems by measuring gaps in a limited vertical range between the baseline and meanline. Spaces that are close to the threshold at this stage are made fuzzy, so that a final decision can be made after word recognition.

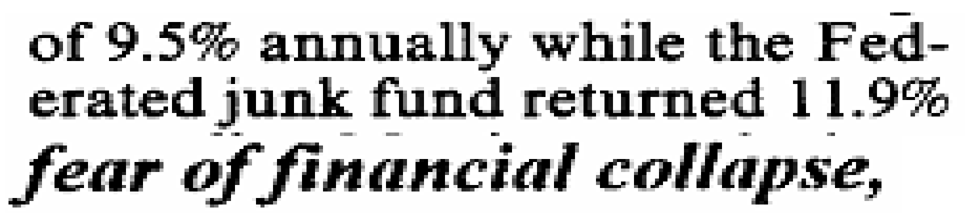
of 9.5% annually while the Fed-
erated junk fund returned 11.9%
fear of financial collapse,

Figure 2-4: Some difficult word spacing.

2.5.7.4 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.

2.5.7.4.1 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.

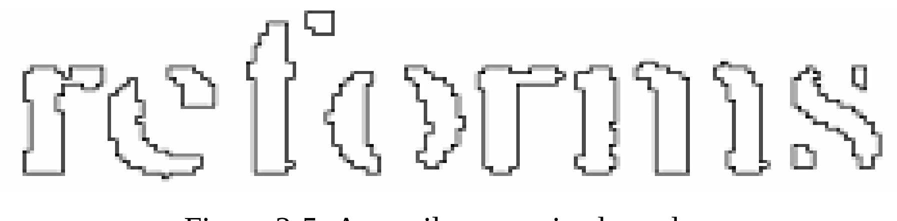

Figure 2-5: An easily recognized word.

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. Figure 2-5 is a typical example. An essential part of that success was the character classifier that could easily recognize broken characters.

2.5.7.5 Static Character Classifier

2.5.7.5.1 Features

An early version of Tesseract used topological features developed from the work of Shillmanet. al. [15-16] Though nicely independent of font and size, these features are not robust to the problems found in real life images, as Bokser [17] describes. 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 Figure 2-6 (a), the right side of the shaft is in two main pieces, but in Figure 2-6 (b) there is just a single piece.

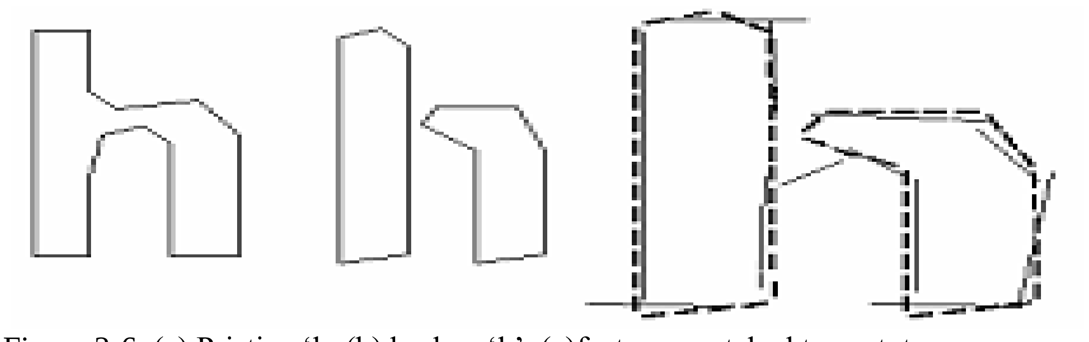
Figure 2-6. (a) Pristine 'h, (b) broken 'h', (c)features matched to prototypes.

The break through solution is the idea that the features in the unknown need not be the same as the features in the training data. 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 Figure 2-6 (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. One prototype bridging the two pieces is completely unmatched. Three features on one side and two on the other are unmatched, but, apart from those, every prototype and every feature is well matched. This example shows that this process of small features matching large prototypes is easily able to cope with recognition of damaged images. Its main problem is that the computational cost of computing the distance between an unknown and a prototype is very high.

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.

2.5.7.5.2 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.

2.5.7.6 Country Plates Font Type

How many fonts in the Country license plates must be known. In the US, for example, many states use very different fonts for their plates. Some countries only use one font. Here is an example of Sudan plates.

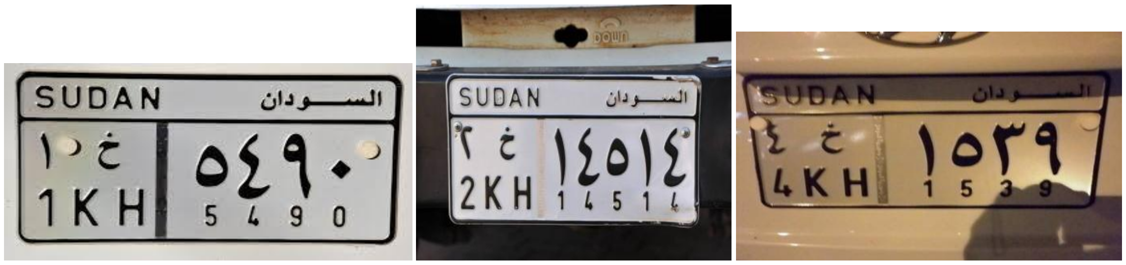

Figure 2-7: example of Sudan plates.

Each font needs to be trained separately. Combining characters across fonts should not happen, this will greatly decrease the accuracy.

2.5.8 Post Processing

Given a list of all possible OCR characters and confidences, post processing determines the best possible plate letter combinations. It is organized as a top N list. Post processing disqualifies all characters below a particular threshold. It also has a "soft" thresholds -- characters that are below this threshold will still be added to the possible list, but they also add a possible blank character -- since it's possible that the low confidence character is not really part of the plate.

The post processing also handles region validation if requested. For example, if I tell OpenALPR that this is a "Missouri" plate, then it will try and match the results against a template that matches the Missouri format
For example, using a pattern against this Czechoslovakian plate results in only one possible match (which is the correct one).
The cz patterns are (@ for characters, # for numbers) [18]:

- cz #@#####
- cz #@@####

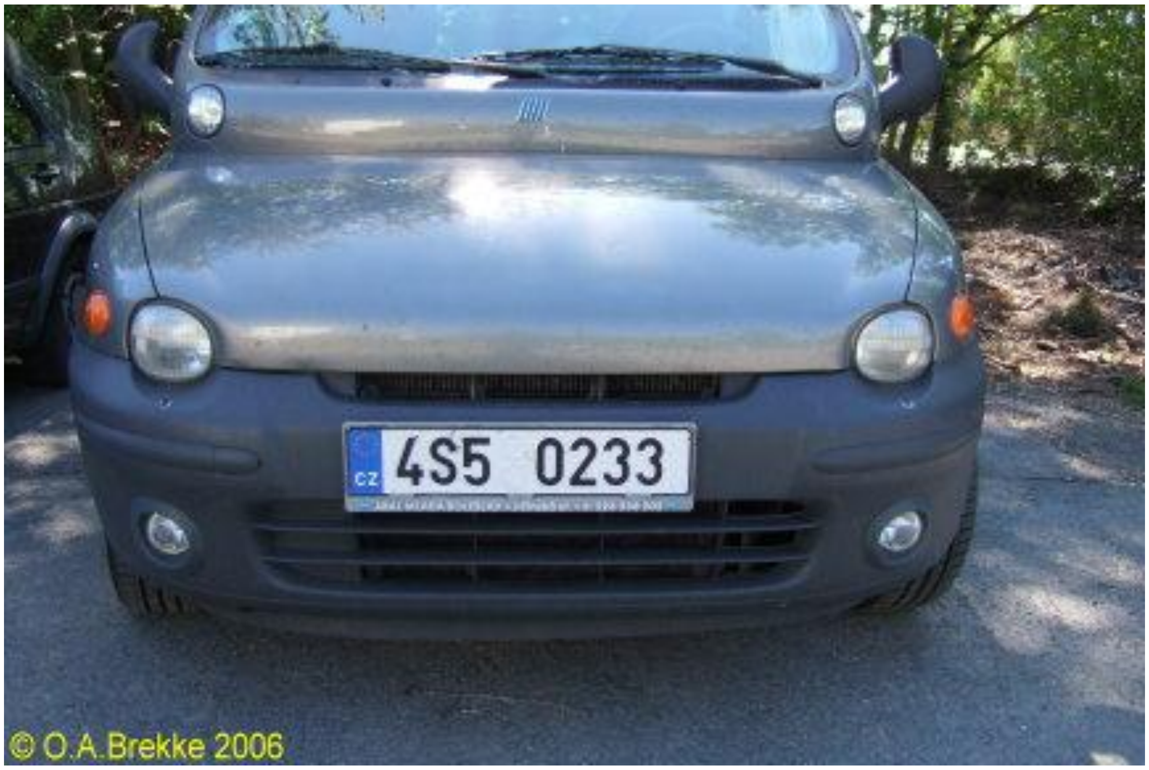

Figure 2-8: Czechoslovakian plate

The ALPR results are:

```
4S5O233     confidence: 90.947      pattern_match: 0
4S5O23      confidence: 87.8683     pattern_match: 0
4S50233     confidence: 83.0457     pattern_match: 1
```

The third entry matches the template, but the other two do not. So, post processing will signal that the third entry is our best match

2.6 Why C++

Using C++ programming language instead of using MATLAB which is widely used in DIP and computer vision application, can give us a lot of advantages in use instead of using MATLAB in licence plate recognition, these advantages can be [19]:

- **Speed:** MATLAB is built on Java, and Java is built upon C. So when you run a MATLAB program, your computer is busy trying to interpret all that MATLAB code. Then it turns it into Java, and then finally executes the code. OpenALPR, on the other hand, is basically a library of functions written in C/C++. You are closer to directly provide machine language code to the computer to get executed. So ultimately you get more image processing done for your computers processing cycles, and not more interpreting. As a result of this, programs written in C++ run much faster than similar programs written in MATLAB. OpenALPR is faster when it comes to speed of execution. For example, we might write a small program to detect people's smiles in a sequence of video frames. In MATLAB, we would typically get 3-4 frames analyzed per second. In C++, we would get at least 30 frames per second, resulting in real-time detection.
- **Resources needed:** Due to the high level nature of MATLAB, it uses a lot of your systems resources. MATLAB code requires over a gigabyte of RAM to run through video. In comparison, typical C++ programs only require ~70mb of RAM to run in real-time. The difference as you can easily see is huge! [5].

- **Cost**: List price for the base (no toolboxes) MATLAB (commercial, single user License) is around USD 2150.
- **Portability:** MATLAB and C++ run equally well on Windows, Linux and MACOS. However, when it comes to C++, any device that can run C, can, in all probability, run C++.

2.7 Hot lists

Once the OCR read is obtained, the information is then compared against a database of vehicles of interest, typically known as a "hot list." Hot list information can come from a variety of sources, the purpose of these lists is to alert the officer that a vehicle displaying a license plate number that is included on a hot list has been identified by the ALPR camera [9].

2.8 Database

In simple words data can be **facts related to** any **object** in consideration. For example, your name, age, height, weight, etc are some data related to you. A picture, image, file, pdf …etc can also be considered data.

A database is a systematic collection of data. Databases **support storage and manipulation of data.** Databases make data management easy [20].

2.8.1 SQL

Structured Query language (SQL) is actually the standard language for dealing with Relational Databases. **SQL can be effectively used to insert, search, update, delete database records**. That doesn't mean SQL cannot do things beyond that. In fact, it can do lot of things including, but not limited to, optimizing and maintenance of databases. Relational databases like MySQL, Oracle, Ms SQL server, Sybase, etc uses SQL. SQL syntaxes used in these databases are almost similar, except the fact that some are using few different syntaxes and even proprietary SQL syntaxes.

2.9 Graphic user interface (GUI)

GUI is a program interface that takes advantage of the computer's graphics capabilities to make the program easier to use. Well-designed graphical user interfaces can free the user from learning complex command languages. On the other hand, many users find that they work more effectively with a command-driven interface, especially if they already know the command language [21].

GUI systems are much easier to learn and use; because commands do not need to be memorized. Additionally, users do not need to know any command line. Because of their ease of use, GUI systems have become the dominant system used by today's end-users

In addition to their visual components, graphical user interfaces also make it easier to move data from one application to another. A true GUI includes

standard formats for representing text and graphics. Because the formats are well-defined, different programs that run under a common GUI can share data. This makes it possible, for example, to copy a graph created by a spreadsheet program into a document created by a word processor.
Many DOS programs include some features of GUIs, such as menus, but are not *graphics based*. Such interfaces are sometimes called *graphical* character-based *user interfaces* to distinguish them from true GUIs.

The first graphical user interface was designed by Xerox Corporation's Palo Alto Research Center in the 1970s, but it was not popular until the 1980s and the emergence of the Apple Macintosh that graphical user interfaces became popular. One reason for their slow acceptance was the fact that they require considerable CPU power and a high-quality monitor, which until recently were prohibitively expensive.

2.10 Related Works

The following table shows some of related works in the last three years.

Table 2-1: Some of related work from 2014 to 2016.

Reference name	**Published at**	**Abstract**

Arabic OCR evaluation tool	2016 7th International Conference on Computer Science and Information Technology (CSIT) (2016)	Performance evaluation of Optical Character Recognition (OCR) systems is an essential task for OCR systems development. However, studies in Arabic OCR suffer from the lack of proper performance evaluation metrics and the availability of evaluation tools. Although the literature provides typical performance metrics, such as character accuracy and word accuracy for OCR performance evaluation, these metrics are not sufficient for evaluating Arabic OCR. This paper presents an open source automated software tool with various metrics for the evaluation of Arabic OCR performance. The developed tool is available for OCR researchers; thus it can be applied for ranking different OCR algorithms.
Automatic license plate recognition: A comparative	2015 IEEE International Symposium on Signal	Automatic license plate recognition (ALPR) is the process of locating and extracting vehicles plate information from images or videos. The extracted

study	Processing and Information Technology (ISSPIT) (2015)	information is essential for several everyday applications, ranging from automated payment services (e.g. parking and toll roads payment collection) to more critical applications, like border crossing security and traffic surveillance systems. Various solutions have been proposed for the ALPR problem, with many available commercial packages. However, amid plate variations from place to place, ALPR systems tend to be region-specific. There is no general solution that works effectively everywhere for every province/state or country. In this paper, we have reviewed a set of state-of-the-art ALPR methods and, compared their respective performances by testing them on a rich database of vehicles from Ontario (Canada).
Vehicle License Plate Recognition	2014 27th SIBGRAPI Conference	Despite decades of research on automatic license plate recognition (ALPR), optical character recognition (OCR) still leaves

with Random Convolutional Networks	on Graphics, Patterns and Images (SIBGRAPI) (2014)	room for improvement in this context, given that a single OCR miss is enough to miss the entire plate. We propose an OCR approach based on convolutional neural networks (CNNs) for feature extraction. The architecture of our CNN is chosen from thousands of random possibilities and its filter weights are set at random and normalized to zero mean and unit norm. By training linear support vector machines (SVMs) on the resulting CNN features, we can achieve recognition rates of over 98% for digits and 96% for letters, something that neither SVMs operating on image pixels nor CNNs trained via back-propagation can achieve. The results are obtained in a dataset that has 182 samples per digit and 28 per letter, and suggest the use of random CNNs as a promising alternative approach to ALPR systems.

2.11 Summary

(ALPR) is a technology that uses optical character recognition (OCR) to automatically read license plate characters, Open ALPR operates as a pipeline. The input is an image, various processing occurs in stages, and the output is the possible plate numbers in the image.

The number of fonts in the license plates must be known because each font needs to be trained separately. Combining characters across fonts should not happen, this will greatly decrease the accuracy. After training the font, it must be combined into one dataset for all license plates.

Structured Query language (SQL) is actually the standard language for dealing with Relational Databases. SQL can be effectively used to insert, search, update and delete database records**.**