Chapter 1

INTRODUCTION

1.1 Introduction Overview

In this chapter, the concepts of Image processing is discussed. After the introduction to the domain, the challenges and opportunities that existed during the implementation of this projected are stated. We then present the problems that have been identified for this domain. For the problems that have been identified, we have selected a few problems for our implementation of this project. We then describe all the problems for which we have attempted to provide a solution.

1.2 Introduction to 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.

It is among rapidly growing technologies today, with its applications in various aspects of a business. Image Processing forms core research area within engineering and computer science disciplines too.

Image processing basically includes the following three steps.

First, importing the image with optical scanner or by digital photography.

Second, Analysing and manipulating the image which includes data compression and image enhancement and spotting patterns that are not to human eyes like satellite photographs.

Last, Output is the last stage in which result can be altered image or report that is based on image analysis.

1.2.1 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. Text Recognition – Distinguish the Text in an image.

1.2.2 What is text recognition?

As its name suggests, Text recognition software recognizes text characters in images, electronic files etc... Usually scanned documents that are saved as images and are not immediately text-searchable.

1.2.3 How image processing is used in text recognition?

Text recognition enhances the processing of scanned images by allowing you to automatically recognize and extract text content from different data fields. For example, when you capture an

image and feed into text recognition software, it will recognition the text within its boundary and extract the text.

1.3 Challenges and opportunities

- Text characters contained in images can be multicolour or any grey-scale value, variable size, low resolution, and embedded in noisy backgrounds.

- Some text images having complex background. Detecting texts in complex images has received lot of attentions and remains a challenge for most practical systems. In this work, an effort is made to build an effective and convenient detection system for texts having variation is style, font, and colour in images. More accurate detection is helps to make character recognition more effective and accurate.

- Text recognition software will simply identify texts, but will not assure grammar corrections for more accuracy of recognition if a word is recognized incorrectly because of varied reasons like different viewing angles.

It is very much challenging in extracting Text from the images where text is too small, fully blurred, faded, over-exposure to light, or having mixed coloured characters.

1.4 Problems identified

After analysing the introduction to the image processing and the challenges and opportunities of implementing such a system, we list a few basic problems which we have identified:

- There is no algorithm that gives absolute result .so, all the algorithms which we have used are giving some approximate values that is absolute to the closer values.

- Optical character recognition (OCR) systems can achieve almost perfect recognition rate on printed text in scanned documents, but cannot accurately recognize text information directly from camera-captured images and videos, and are usually sensitive to font scale changes and background interference which widely exists in scene text.

- If the no of layers are too small, text boundary cannot be effectively extracted from complex background. If the no of layers are too large, the algorithm will lose the tolerance to colour variation within a character or string.

- The frequency of occurrence of text in natural image is very low, and a limited number of text characters are embedded into complex non-text background outliers.

- In natural images, background textures, such as grid, window, and brick, even resemble text characters and strings causes problem.

1.5 Problems for which we are providing Solution.

- The no of layers for boundary clustering depends on the amount of background noises and non-text region in the captured image. So, after referring to some Experimental procedures we are fixing no of layers to 5.

- Background textures, such as grid, window, and brick, even resemble text characters and strings cause the problem which can be removed by model colour difference by a vector of colour Pair, which is obtained by cascading the RGB colours of text and attachment surfaces. Each boundary can be described by a colour-pair, and we cluster the boundaries with similar colour-Pairs into the sample layer.

- It is very much challenging in extracting Text from the images where text is too small, fully blurred, faded, over-exposure to light etc... These problems can be avoided by performing various image filters on input images.

- Currently OCR systems are being used for recognizing text from images.OCR's are efficient in identifying text from images without skew, they fail to recognize text from camera captured images and they are sensitive to variations in fonts and styles. Our Software will be an enhancement to the existing OCRs in terms of non-scanned images.

Chapter 2

PROBLEM DEFINITION

2.1 Problem Statement

Camera Based text information serves as effective tags or clues for many mobile applications associated with media analysis, content retrieval, scene understanding, and assistant navigation. In natural scene images and videos, text characters and strings usually appear in nearby sign boards and hand-held objects and images of vehicle number plates etc. provides significant knowledge and details of surrounding environment and objects.

In our project, as depicted by its title, we are mainly focusing on extracting the text from the images of vehicle number plate. This implementation can be put to use in helping the Traffic Department, Traffic police, where they can capture the images of the moving vehicles and extract the text from the captured image using our model. They can make use of the retrieved text in collecting the registration details of that vehicle like, owner info etc. which will be helpful in identifying stolen vehicles, Hit-and-Run cases and others.

Chapter 3

LITERATURE SURVEY

3.1 Survey Experience

While doing this project, we have done lot of survey about our project. First and foremost thing is understanding the topic. We surveyed a lot and understood the importance of extracting the text from the image. We searched the currently available software and planned how our project should be different from all the existing software. We also thought about the difficulties that we might face once we started to do this project because of unfamiliar with the topic Image processing. We searched about the image processing and its applications etc.

Several papers on image processing and it various techniques were surveyed on going about our topic. After starting this we were in confusion about choosing the tool, language, and libraries etc. which were very much necessary to do this project. We also searched about different algorithms which could be used in our project.at last OpenCV was chosen to be used as the image processing library and surveyed a lot about using this in our project.

3.2 A Focused survey:

A summarization of a few papers and techniques that we have referred are given below.

3.2.1 Grayscale

In photography and computing, a grayscale or grayscale digital image is an image in which the value of each pixel is a single sample, that is, it carries only intensity information. Images of this sort, also known as black-and-white, are composed exclusively of shades of gray, varying from black at the weakest intensity to white at the strongest.

Grayscale images are distinct from one-bit bi-tonal black-and-white images, which in the context of computer imaging are images with only the two colours, black, and white (also called *bi-level* or *binary images*). Grayscale images have many shades of grey in between.

Grayscale images are often the result of measuring the intensity of light at each pixel in a single band of the electromagnetic spectrum (e.g. infrared, visible light, ultraviolet, etc.), and in such cases they are monochromatic proper when only a given frequency is captured. But also they can be synthesized from a full colour image; see the section about converting to grayscale.

3.2.2 RGB Colour spaces

An RGB colour space is any additive colour space based on the RGB colour model. A particular RGB colour space is defined by the three chromaticity's of the red, green, and blue additive

primaries, and can produce any chromaticity that is the triangle defined by those primary colours. The complete specification of an RGB colour space also requires a white point chromaticity and a gamma correction curve. As of 2007, sRGB is by far the most commonly used RGB colour space. RGB is an abbreviation for red–green–blue.

RGB is a convenient colour model for computer graphics because the human visual system works in a way that is similar — though not quite identical — to an RGB colour space. The most commonly used RGB colour spaces are sRGB and Adobe RGB (which has a significantly larger gamut). Adobe has recently developed another colour space called Adobe Wide Gamut RGB, which is even larger, in detriment to gamut density.

3.2.3 Pixel

A pixel is generally thought of as the smallest single component of a digital image. However, the definition is highly context-sensitive. For example, there can be "printed pixels" in a page, or pixels carried by electronic signals, or represented by digital values, or pixels on a display device, or pixels in a digital camera (photo sensor elements). This list is not exhaustive and, depending on context, synonyms include sample, byte, bit, dot, and spot. *Pixels* can be used as a unit of measure such as: 2400 pixels per inch, 640 pixels per line, or spaced 10 pixels apart.

3.2.4 Pixel values

Each of the pixels that represents an image stored inside a computer has a *pixel value* which describes how bright that pixel is, and/or what colour it should be. In the simplest case of binary images, the pixel value is a 1-bit number indicating either foreground or background. For a grayscale images, the pixel value is a single number that represents the brightness of the pixel. The most common *pixel format* is the *byte image*, where this number is stored as an 8-bit integer giving a range of possible values from 0 to 255. Typically zero is taken to be black, and 255 is taken to be white. Values in between make up the different shades of grey.

To represent colour images, separate red, green and blue components must be specified for each pixel (assuming an RGB colour space), and so the pixel `value' is actually a vector of three numbers. Often the three different components are stored as three separate `grayscale' images known as *colour planes* (one for each of red, green and blue), which have to be recombined when displaying or processing.

3.2.5 Pixel Neighbourhood

There are two different ways to define the neighbours of a pixel P located at (x, y):

- **4-neighbours**

  The 4-neighbours of pixel P, denoted by N4(P),are the four pixels located at (x-1,y),(x+1,y),(x,y-1),(x,y+1), they are,

respectively, above(north),below(south),to the left(west), and right(east) of the pixel P.

- **8-neighbours**

  The 8-neighbours of pixel P, denoted by N8(P),include the four 4-neighbours and four pixels along the diagonal direction located at (x-1,y-1)(north-west),(x-1,y+1)(north-east),(x+1,y-1)(south-west),(x+1,y+1)(south-east).

3.2.6 Canny edge detection

The Canny edge detection algorithm is known to many as the optimal edge detector. Canny's intentions were to enhance the many edge detectors already out at the time he started his work. He was very successful in achieving his goal and his ideas and methods can be found in his paper, "*A Computational Approach to Edge Detection*". In his paper, he followed a list of criteria to improve current methods of edge detection. The first and most obvious is low error rate. It is important that edges occurring in images should not be missed and that there be NO responses to non-edges. The second criterion is that the edge points be well localized. In other words, the distance between the edge pixels as found by the detector and the actual edge is to be at a minimum. A third criterion is to have only one response to a single edge. This was implemented because the first 2 were not substantial enough to completely eliminate the possibility of multiple responses to an edge.

3.2.7 Gaussian blur

A 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 and reduce detail. 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.

3.2.8 K-means

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.

3.2.9 Gaussian mixture

In statistics, a mixture model is a probabilistic model for representing the presence of subpopulations within an overall population, without requiring that an observed data set should identify the sub-population to which an individual observation belongs. Formally a mixture model corresponds to the mixture

distribution that represents the probability distribution of observations in the overall population. However, while problems associated with "mixture distributions" relate to deriving the properties of the overall population from those of the sub-populations, "mixture models" are used to make statistical inferences about the properties of the sub-populations given only observations on the pooled population, without sub-population identity information.

3.2.10 EM algorithm

In statistics, an expectation–maximization (EM) algorithm is an iterative method for finding maximum likelihood or maximum a posteriori (MAP) estimates of parameters in statistical models, where the model depends on unobserved latent variables. The EM iteration alternates between performing an expectation (E) step, which creates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters, and a maximization (M) step, which computes parameters maximizing the expected log-likelihood found on the *E* step. These parameter-estimates are then used to determine the distribution of the latent variables in the next E step.

3.2.11 Two-Pass component labelling algorithm

Two-pass algorithm is used to identify the connected components in the image. Connected-component labelling is used in computer

vision to detect connected regions in binary digital images, although colour images and data with higher dimensionality can also be processed. When integrated into an image recognition system or human-computer interaction interface, connected component labelling can operate on a variety of information.

3.2.12 Bag-Of-Words Model

The **bag-of-words model** is a simplifying representation used in natural language processing and information retrieval (IR). In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity. Recently, the bag-of-words model has also been used for computer vision.

The bag-of-words model is commonly used in methods of document classification, where the (frequency of) occurrence of each word is used as a feature for training a classifier.

3.2.13 Support Vector Machine

In machine learning, **support vector machines** (**SVMs**, also **support vector networks**) are supervised learning models with associated learning algorithms that analyse data and recognize patterns, used for classification and regression analysis. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear

gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

3.2.14 OpenCV

OpenCV is released under a BSD license and hence it's free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform. Adopted all around the world, OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 9 million. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics.

Chapter 4

PROJECT REQUIREMENT DEFINITION

4.1 Project Perspective

Currently OCR systems are being used for recognizing text from images. Though OCR's are efficient in identifying text from images without skew, they fail to recognize text from camera captured images and they are sensitive to variations in fonts and styles. Our Software will be an enhancement to the existing OCRs in terms of accuracy on non-scanned images. The software will use two stages to process the image, firstly to detect text regions and then to recognize text from the detected regions.

4.2 Project Function

This project is an application of text recognition. Our product is built to support traffic department in identifying details about vehicles from the text extracted using our software from the image containing the vehicle number plate. Our Software will first recognize text from images of the vehicle number, then this number will be used to retrieve vehicle owner's details instantly from a web service. These functions will help the authorities to identify stolen vehicles, and track offensive drivers indulging in hit and run cases etc.

4.3 User classes and characteristics

Even though we are mainly focusing on Text recognition from the vehicle number plate, there are many other application of our project after recognition the text. Some of the application have been noted down

- General Public can use it for text/business card reading. Contact information on business cards can be directly saved in the address book of the mobile device.
- Visually impaired individuals can benefit from the developed system if the recognized text is passed to a speech converter.
- Students can use the system for automatic reading of books and articles. They can also edit the text, search for keywords on the internet and automatically convert the lecture slide images to notes. The system can also be extended to handwriting recognition allowing student to take images of board notes and convert them to text.
- Non-native speakers of languages based on the Latin alphabet (English, French, and German etc.) can also benefit from the system by translating the text written on sign boards and other places into their desired language.
- Our project could be useful to the traffic police dept. in many ways like they can track the vehicle details during Hit-and-run cases, stolen vehicle case etc.

4.4 Operating Environment

- The system shall provide a simple user interface to interact. The user can capture images or can use previously captured images, and then recognize text information from it. The user interface will be able to guide the user if he makes mistakes.
- The hardware interfaces for this product will be the computer system which does the high computations.
- As the application will be built on Computer System, so it will definitely use the routines and procedures of the

underlying OS. The System must be installed with Java and as we are using OpenCV, OpenCV libraries with Java API should be present. As the implementation of the system will be supported from certain OpenCV libraries, for certain tasks to be done, the application will certainly interact with those libraries.

- As the software is developed to work as two entities –mobile and computer, using mobile we have to capture the image and load it to the Computer System. Communication Interface could be the USB Cable, Bluetooth interface and any general software's that is capable of communicating between mobile and Computer System.

4.5 Assumptions and Dependencies

The project is based on the following assumptions:

- The photo is assumed to be captured and it is available to extract the data.
- The photo should be clear with very less blur.
- Application would run on required computer System without integration of any extra hardware.
- Optional components of the project would only be implemented only if the time and required resources are available.

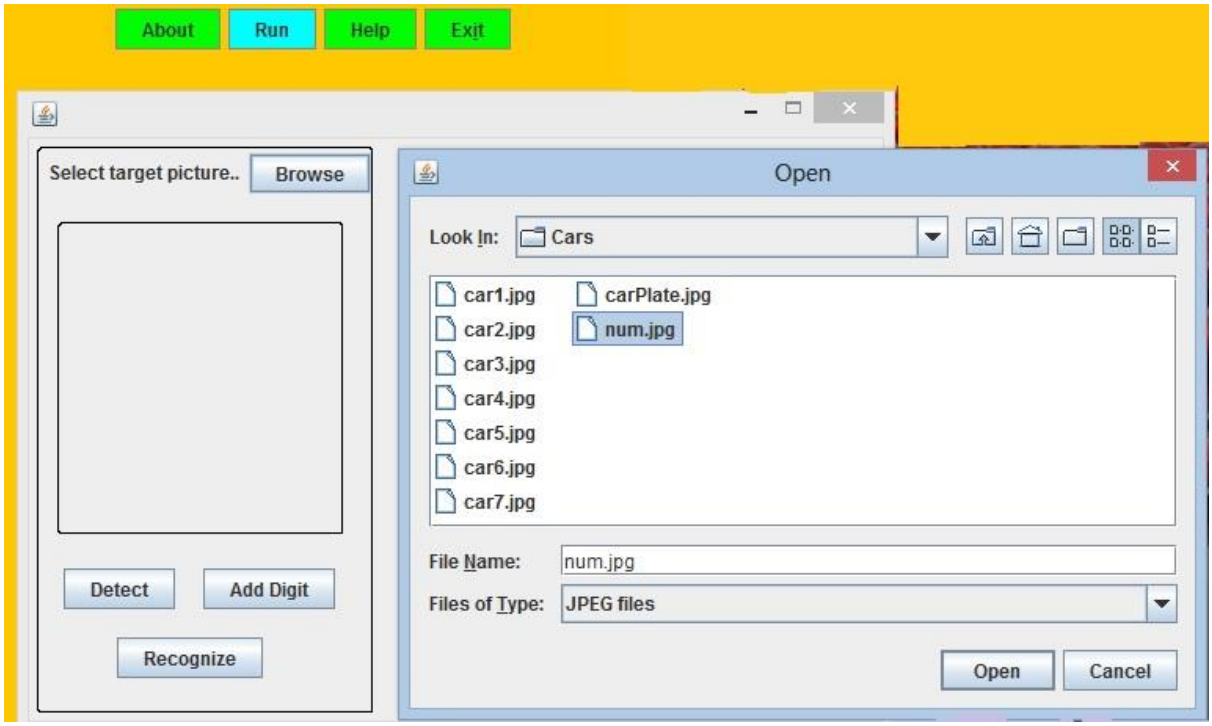
Chapter 5

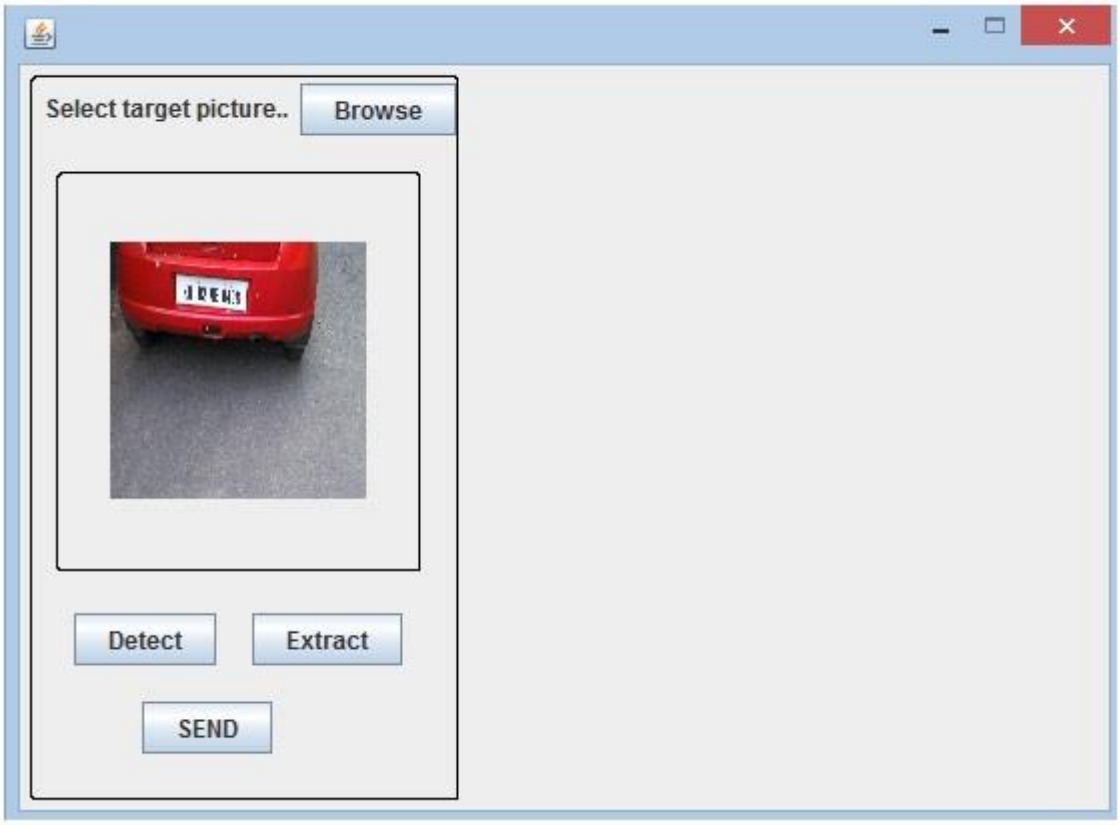
SYSTEM REQUIREMENT SPECIFICATION

5.1 External Interface Requirements

5.1.1 User Interface

The system shall provide a simple user interface to interact. The user can capture images or can use previously captured images, and then recognize text information from it. The user interface will be able to tell the user if he makes mistakes.





5.1.2 Hardware Interface

The hardware interfaces for this product will be the computer system which does the high computations.

5.1.3 Software Interface

As the application will be built on Computer System, so it will definitely use the routines and procedures of the underlying OS. The System must be installed with Java and as we are using OpenCV, OpenCV libraries with Java API should be present. As the implementation of the system will be supported from certain OpenCV libraries, for certain tasks to be done, the application will certainly interact with those libraries.

5.1.4 Communication Interface

As the software is developed to work as two entities –mobile and computer, using mobile we have to capture the image and load it to the Computer System. Communication Interface could be the USB Cable, Bluetooth interface and any general software's that is capable of communicating between mobile and Computer System.

5.2 Functional Requirements

5.2.1 Pre-processing of image

This function will try to reduce noise in the image.

5.2.1.1 Inputs

Image captured.

5.2.1.2 Processing

Reduce noise.

5.2.1.3 Outputs

An image with lesser noise.

5.2.1.4 Error Handling

Retry

5.2.2 Detect text region

This function will detect text regions from image based on the concept that all text inside image will be of same colour.

5.2.2.1 Inputs

Pre-processed Image.

5.2.2.2 Processing

Generate images of prominent colours and localize text regions on the basis of colour pairs.

5.2.2.3 Outputs

Set of images for different prominent colours or layers.

5.2.2.4 Error Handling

Report if text is not found

5.2.3 Recognize text from detected text region

This function will recognize text from the text regions and convert them to readable text codes.

5.2.3.1 Inputs

Detected Text region image.

5.2.3.2 Processing

Recognize text by using character descriptors.

5.2.3.3 Outputs

Text code.

5.2.3.4 Error Handling

Retry for some number of time.

5.3 Non-Functional Requirements

5.3.1 Performance

The software should process the image and retrieve text within a couple of minute once the image is available. And should produce expected result within that time.

5.3.2 Reliability

The system will be designed to provide more reliability.

5.3.3 Availability

The software will be available for processing as long as the System is in working state.

5.3.4 Maintainability

Whenever there is a change in requirement or bug found, the application will be easily maintainable by creating new versions.

5.3.5 Portability

Since the software is developed using Java, it is by default acquires the property of portability and it is naturally adaptable to different platform.

5.3.6 Usability

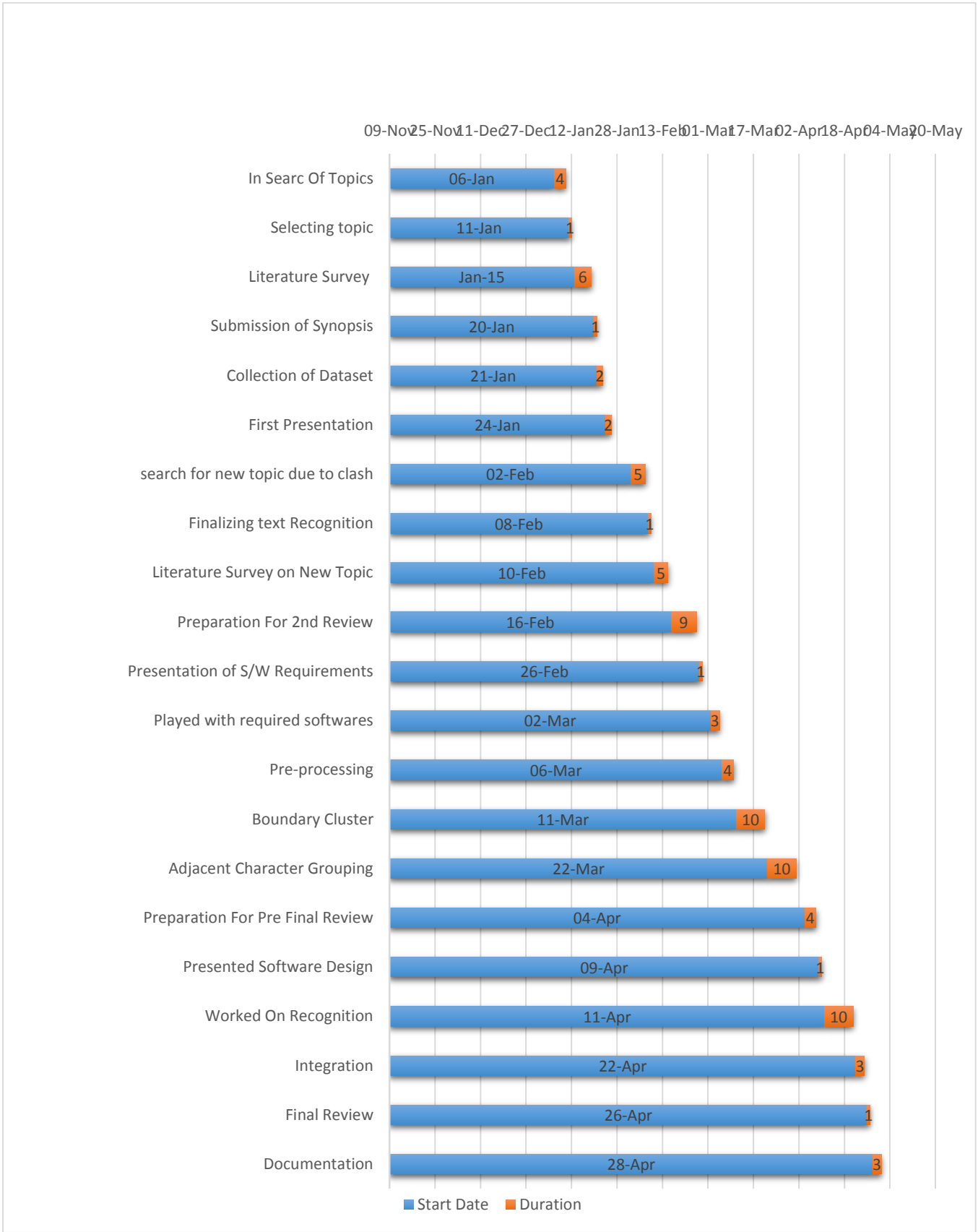
We have provided a simple UI such that it should be understood by each and every one.

5.4 Inverse Requirements

The software will not return text from images that don't contain text. The user shall not be permitted to recognize text before text detection. The software does not recognize text data in other regional languages

Chapter 6

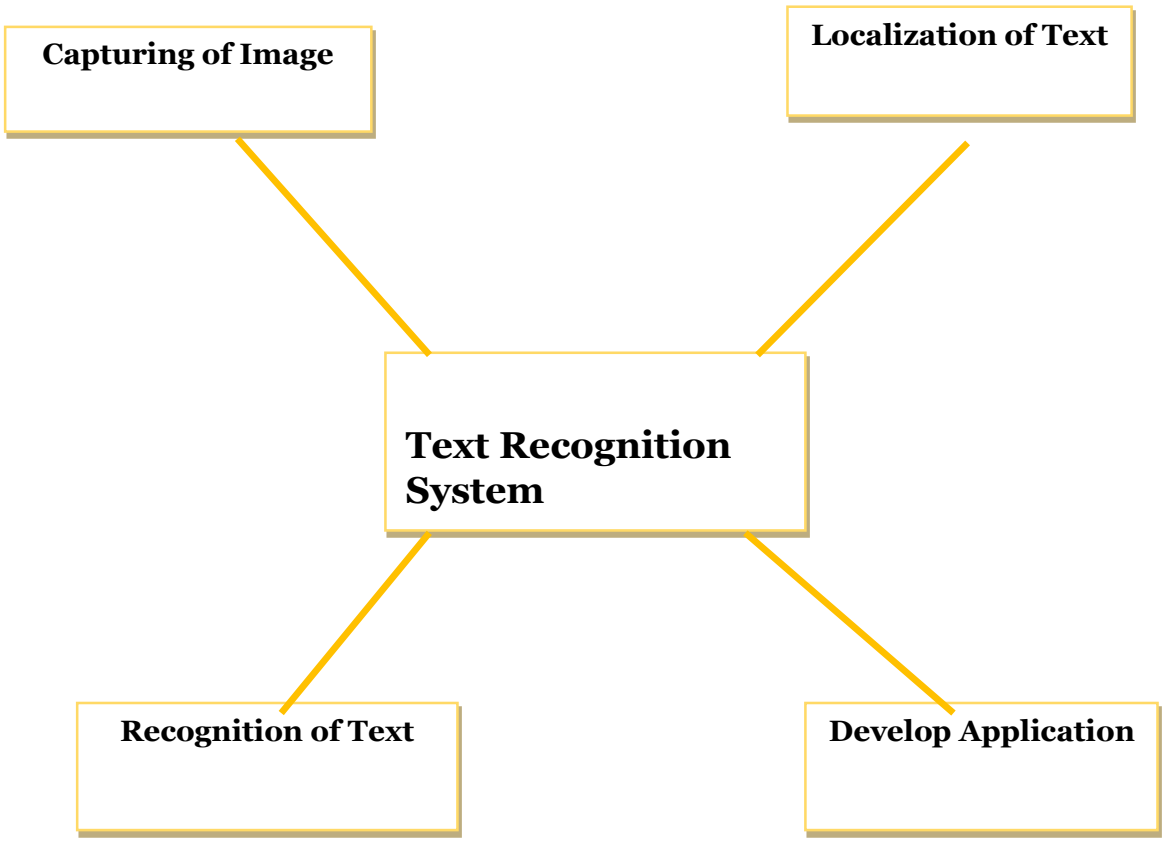
GANTT CHART


09-Nov
25-Nov
11-Dec
27-Dec
12-Jan
28-Jan
13-Feb
01-Mar
17-Mar
02-Apr
18-Apr
04-May
20-May
In Searc Of Topics
06-Jan
4
Selecting topic
11-Jan
1
Literature Survey
Jan-15
6
Submission of Synopsis
20-Jan
1
Collection of Dataset
21-Jan
2
First Presentation
24-Jan
2
search for new topic due to clash
02-Feb
5
Finalizing text Recognition
08-Feb
1
Literature Survey on New Topic
10-Feb
5
Preparation For 2nd Review
16-Feb
9
Presentation of S/W Requirements
26-Feb
1
Played with required softwares
02-Mar
3
Pre-processing
06-Mar
4
Boundary Cluster
11-Mar
10
Adjacent Character Grouping
22-Mar
10
Preparation For Pre Final Review
04-Apr
4
Presented Software Design
09-Apr
1
Worked On Recognition
11-Apr
10
Integration
22-Apr
3
Final Review
26-Apr
1
Documentation
28-Apr
3
Start Date
Duration

Chapter 7

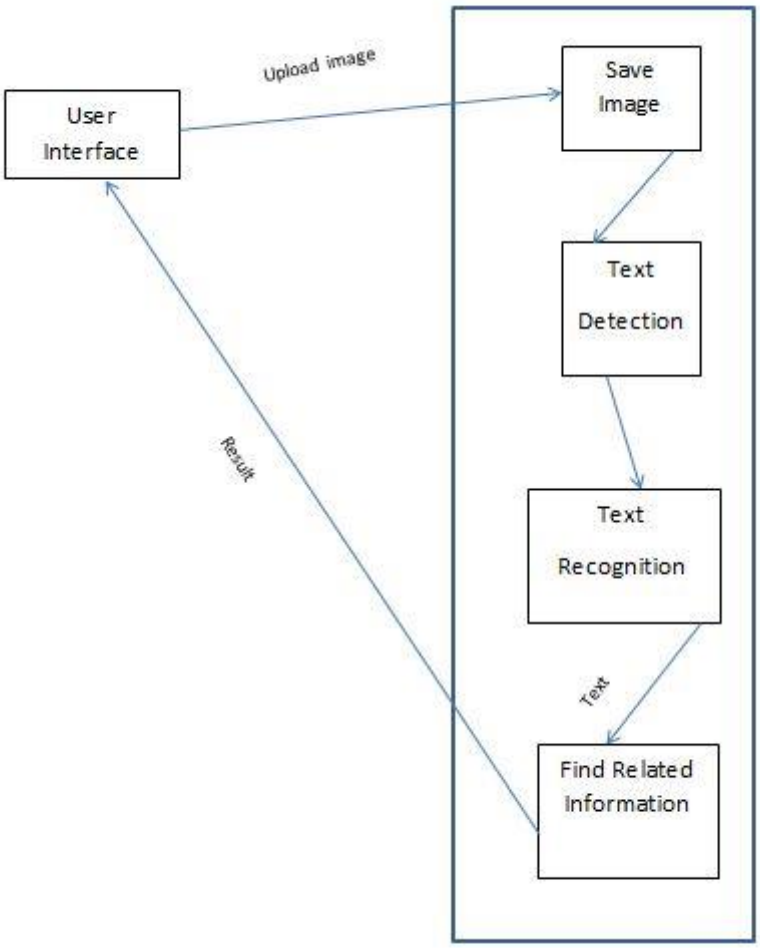
SYSTEM DESIGN

7.1 Block Diagram



1. User Captures Images from his mobile devices.
2. System detects the text.
3. Recognizes the text from the captured image.
4. Retrieved text is used in Application.

7.2 System Architecture



User Interface:

This component is designed to work on the client side and shall run on android mobile devices.

Save Image:

The save image component is a script that runs on the server side to receive the image from the client and save it as file under a specified folder.

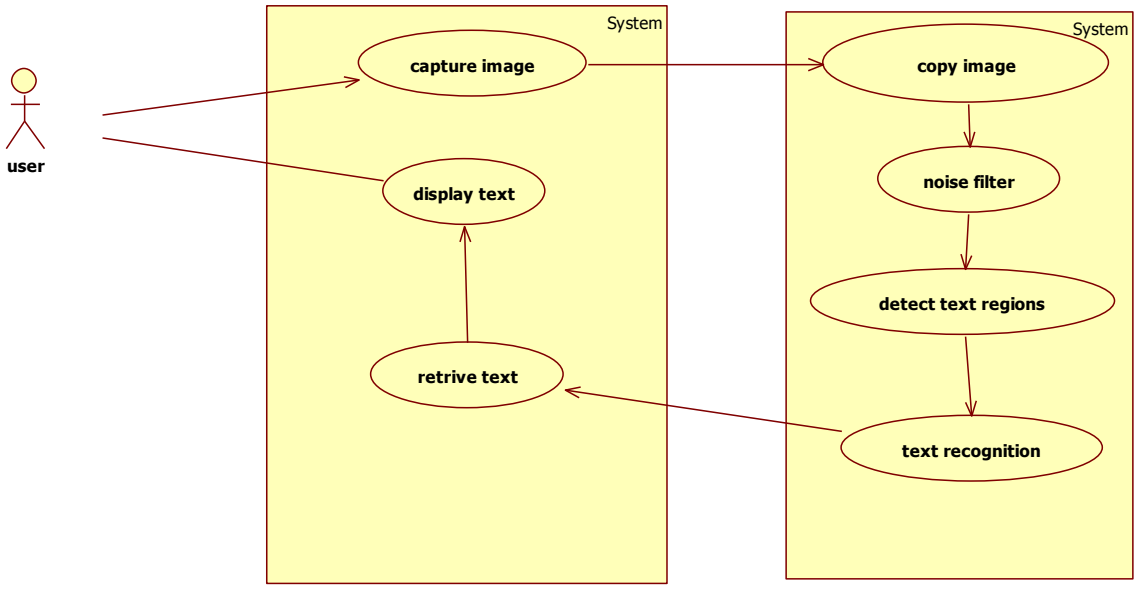
Text Detection:

Text detection component will read image from the folder and perform two algorithms - Boundary Cluster and Adjacent Character Grouping. After this, detected text regions will be marked.

Text Recognition:

Text recognition component will read the detected regions and find features vector for each detected bounding box. Then compare this features to find the exact matching character. Recognised character will be saved as string. The text will be searched in database to find related information like the vehicle details and send it to the client user interface.

7.3 Use Cases



Chapter 8

DETAILED DESIGN

8.1 Data Design

The image data are stored directly as a file in a folder. We haven't used any specific database for storing image instead the images are saved as part of file System. We are selecting the Image and Sending the image path to the program. The program will read the image file from the given path for the image processing.

8.2 Modules in our project

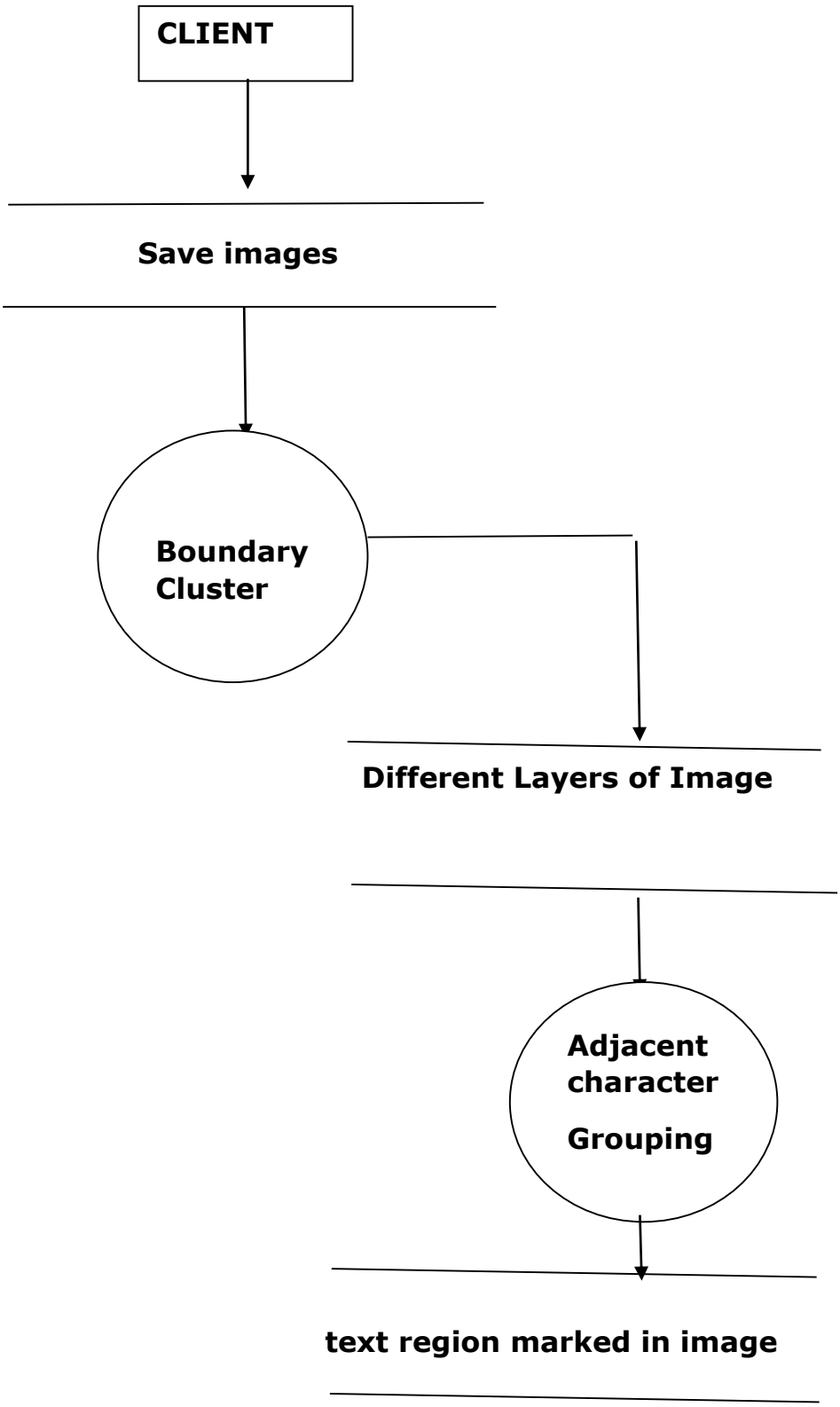
- Boundary Cluster Algorithm
- Two Pass Component Labelling Algorithm
- Adjacent Character Grouping
- Text Recognition by Character Descriptor

Both Boundary Cluster, Two Pass Component Labelling Algorithm, Adjacent Character Grouping will help in detecting the text in the Image and Character Descriptor is used in recognition the text. All these have been explained in detail in Implementation section.

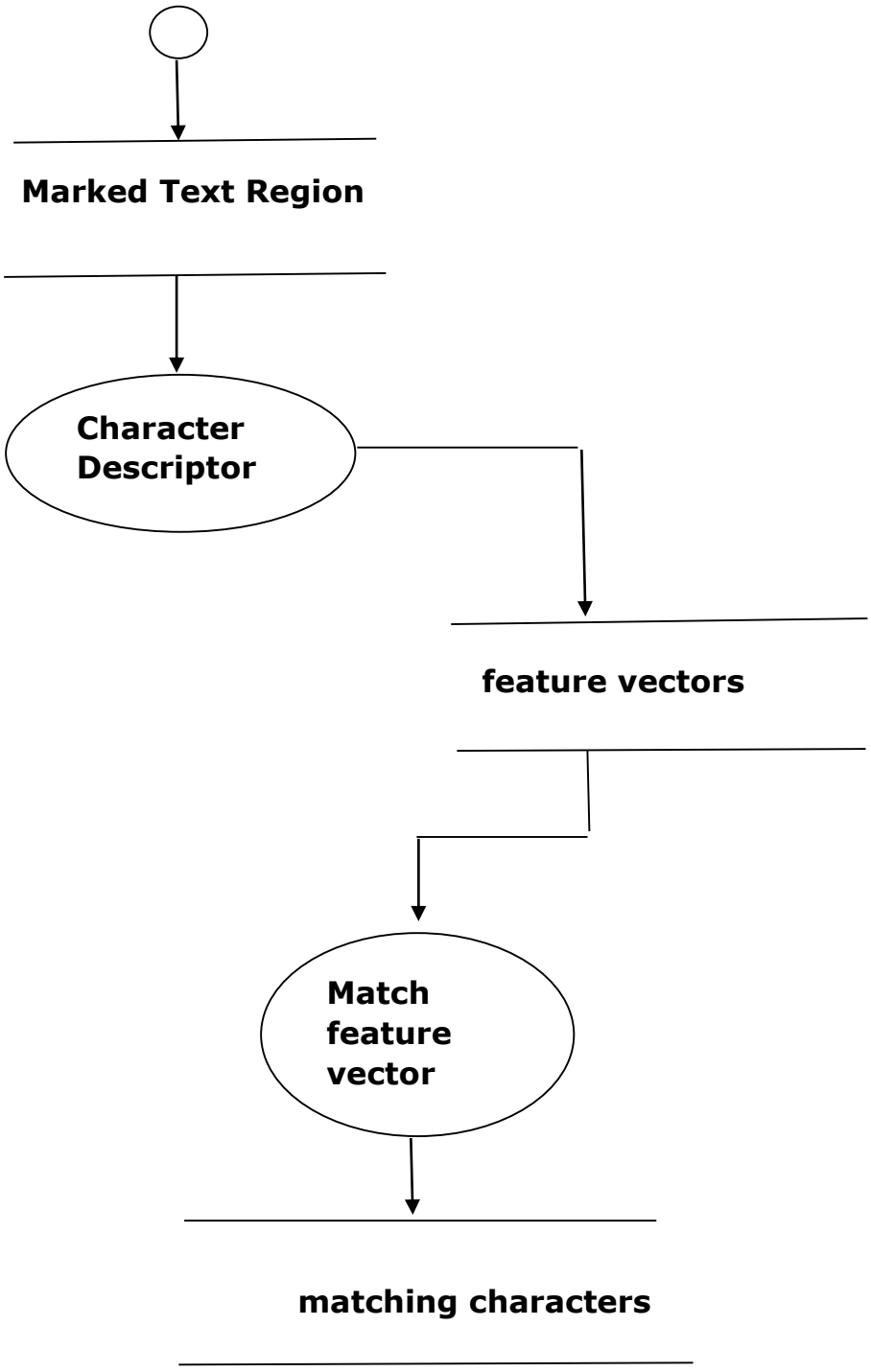
8.3 Design of the flow

In this section, a basic flow of the process is described.it gives the detail idea of our model.

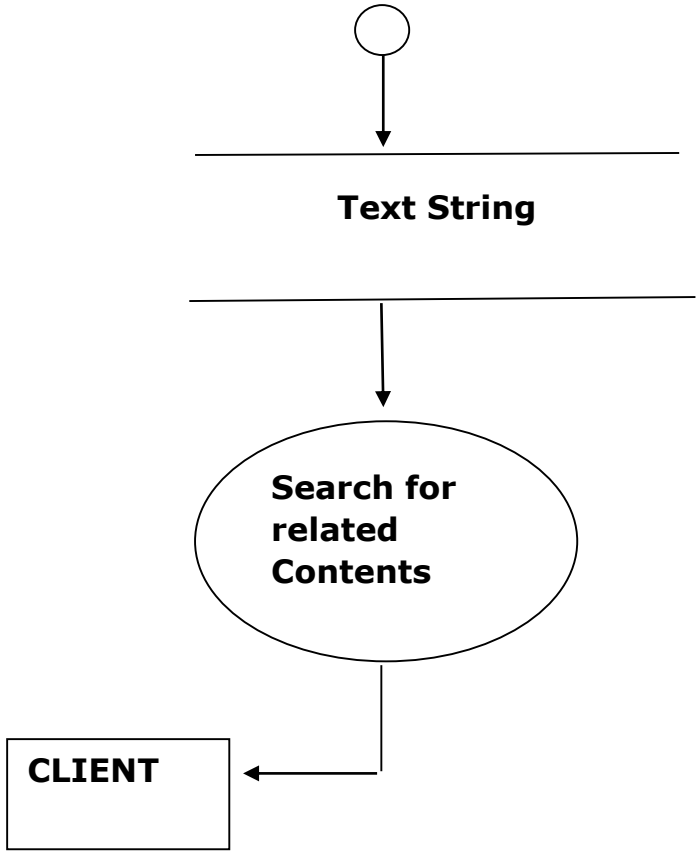
8.3.1 Text Detection DFD



8.3.2 Text Recognition DFD



8.3.3 After recognition DFD



Chapter 9

IMPLEMENTATION

9.1 Introduction to Implementation

Our project Consist of 4 modules, detailed explanation of implementation of these 4 modules are below.

9.2 Pre-processing:

The first step in implementation is to pre-process the input image in order to reduce noise. We remove the blurring noise in the image by sharpening the image using *Gaussian Blur* operation.

9.3 Boundary Cluster:

The first module developed in our system is Boundary Cluster. Boundary Cluster is implemented on the fact that text appears with constant stroke colour over a common background colour. It is intended to separate out the text and background into different layers. It can also separate out text on different lines into different layers. First Canny-Edge-Detection is applied to the pre-processed image to find all the edge pixels, which are above a specified higher threshold and those edge pixels which are connected to the high threshold pixels. For each edge pixel found, we build a vector containing the colour pairs and spatial positions. The colour pairs are those colours with maximum colour difference chosen from the 8-direction neighbourhood. The colour with low intensity indicates text stroke colour and the colour with high intensity indicates background colour.

The vectors are clustered to make different layers. Initially K-means is applied to find centres. These centres are used as initial means for the Gaussian Mixture Model – EM algorithm. EM algorithm will find different Gaussian Distributions, which are

nothing but different clusters for our case. Then we save each cluster of pixels into different layer images.

9.4 Two Pass Component Labelling Algorithm

For each layer generated, we find the connected components and label them by *Two-Pass-Algorithm*. In the first pass we read the image pixels row-wise and start labelling them if the pixel belongs to foreground. If all the neighbouring pixels in 8-directions are unlabelled, then a new label is assigned. Otherwise, if some of the neighbours are labelled then smallest label value among them is used to label the new pixel. As and when we label, we keep track of Equivalent Labels according to neighbouring pixel labels. In the second pass, we again read the image in row-wise and start labelling if pixel belongs to the foreground. For each labelled pixel, we will find the minimum Equivalent Label and use it to label this pixel.

9.5 Adjacent Character Grouping

Once we get all the connected components for each layer, we apply Adjacent Character Grouping to group connected component based on few conditions. This is the second major component developed for our software. Adjacent Character Grouping takes advantage of the fact that text appear in horizontal alignment over the background. For each connected

component we compute a representation vector containing information about area, height, width, and centroid.

For each component, we create two sibling sets – left sibling set to store components on its left side and right sibling set to store components on its right side. A component is part of its sibling set - if the height ratio is within a predefined bound, if the area ration is within predefined limit, if one component exists closer to the other component, if difference in the centroid's y-coordinate is negligible. Then a sibling group is built for each component by merging its left and right sibling set along with the component itself. Two sibling groups are merged if they have common components in them.

When there is no more merge possible, each Sibling group will represent a text string on the image. Some sibling groups could be duplicates and non-text components. Such false groups are filtered out. Bounding Rectangles are marked for the final sibling groups representing detected text region.

9.6 Character Descriptor

Character Descriptor is the final major component in our software. This is responsible for recognising the text characters from detected text regions. The input to this component is an image of a character, or simply the cropped image of individual component found by the previous module.

Four detectors are employed on the image to find *Key Points.* Harris detector will find keypoints around corners and junctions.

Maximally Stable Extremal Regions detector is used to find keypoints at constant coloured strokes. Dense detector will find 64 keypoints taken over the image uniformly. Random detector is designed to find keypoints at pre-specified random positions, in order to maintain the stability for comparison. For each keypoints detected from the four detectors, *Histogram-Of-Gradients* descriptor is computed to get the HOG feature.

Bag-Of-Word model is applied to all the four detectors. First a vocabulary of visual words is built by applying *K-means* clustering algorithm. The size of the vocabulary is 128. Then for each detector, HOG features are mapped to the vocabulary and the frequency of visual words are found. The frequency of visual words from all the four detectors are concatenated to form BOW feature.

Gaussian mixture model is used on dense and random detectors. Both dense detector and random detector find 64 HOG features each. Initially 8 centres are computed using *K-means* and are used as initial means for *EM algorithm*. A likelihood vector representing the probability of belonging to a Gaussian-Distribution is built for each HOG feature. Two likelihood vectors are compared to build binary comparison vector, which is then converted to corresponding decimal value considering the binary comparison vector as bit values. All the decimal values computed for individual detector will be part of GMM feature vector. The GMM feature of dense detector and random detector are appended to form final GMM feature vector. At last the BOW feature and GMM feature are concatenated to get Character Descriptor feature.

We use *Chars74K EnglishImg Dataset* for training *Support Vector Machine* in order to recognise. For every image on the dataset, we compute the Character Descriptor feature and use it as training data for the SVM. Once the SVM is trained, we crop out each component which will be part of a sibling group according to previous module - Adjacent Character Grouping, then find its Character Descriptor feature vector and predict to which character group it belongs by using the trained SVM.

Chapter 10

INTEGRATION

10.1 Choices and Strategies

Our entire code has been divided into 4 modules and each modules has its sub modules, as explained in the design document. Now, these modules and sub modules needed to be integrated.

Individual part or modules of the project were created first. A mix of top down and bottom up integration has been used. As and when each part was ready it was tested with other parts that were available often simulating data when not available.

Our project consist of 4 main modules. They are Boundary Cluster Algorithm, Two Pass Component Labelling Algorithm, Adjacent Character Grouping, and Character Descriptor. All these 4 modules are in MainClass.java. And each one has its own sub modules. When we run MainClass.java these module calls its sub modules and it gets processed and gives expected outputs. And outputs of different modules will be stored in different folders.

Chapter 11

TESTING

11.1 Introduction to Testing

Our product mainly involves a graphical user interface which to upload an image to the system. Owing to this, our testing procedure is largely manual.

11.1.1 Unit testing

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. Unit testing is often automated but it can also be done manually.

Unit tests for various modules are done accordingly. All tests are verified by printing the output onto console or storing the output images in some folder and manually checking them. No testing framework being used, in particular.

11.1.2 Integration testing

Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing.

As and when a module is complete, and its unit tests are complete, we integrated the module along with the previously complete modules and performed integration testing.

11.2 Test Cases covered

Below are a list of test cases that were used. The test cases are listed in the order of functionality that has been coded. For example, test case 1 is the first module that was coded and also includes sub modules followed by test case 2 the second, and so on.

Test Case 1:

Test Case Description: Pre-processing to get a sharpened image by reducing noise.

Input: Camera Captured Image.

Expected Result: A Sharp Image after pre-processing with less noise.

Actual Result: A Clear and Sharpened image is obtained.

Input Image:

Sharpen Image:

Conclusion: Success

Test Case 2:

Test Case Description: Canny-Edge-Detection is applied to the pre-processed image to find all the edge pixels.

Input: Pre-processed image

Expected Result: Greyscale Image with edge pixel.

Actual Result: Greyscale image with edge pixels is obtained.

Conclusion: Success

Test Case 3:

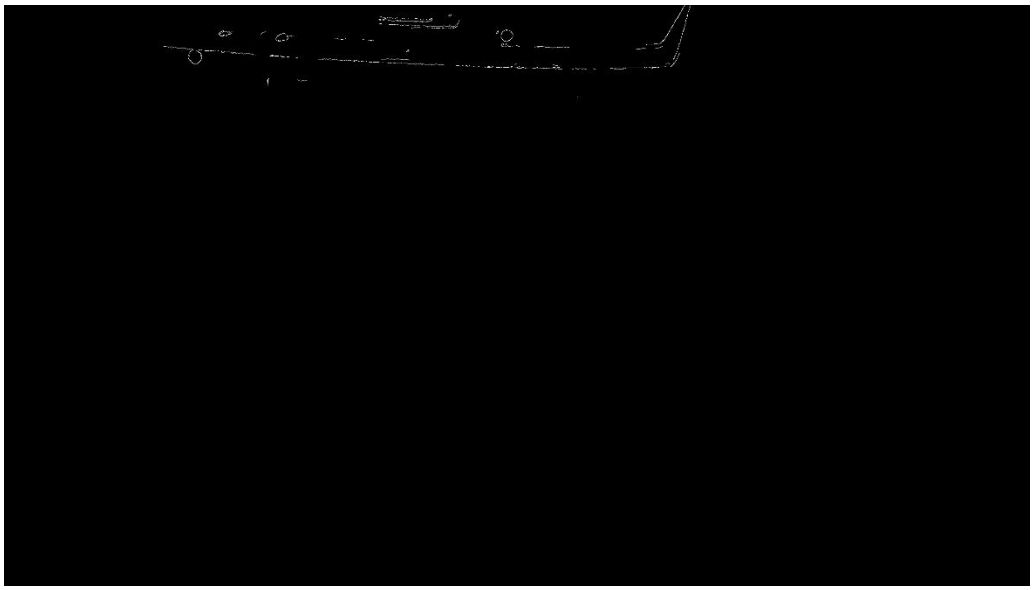
Test Case Description: Clustering the colour-Pairs [vectors] into different layers applying K-Means and Gaussian Mixture Model.

Input: Canny Edge image

Expected Result: Different Cluster are obtained and saved as different layers.

Actual Result: 5 Layers are obtained for k=5 in our case.

Layer0:



Layer1:

Layer2:

Layer3:

Layer4:

Conclusion: Success

Test Case 4:

Test Case Description: Adjacent character grouping for for connected component.

Input: Layer names along with and a Matrix containing set of points of each components.

Expected Result: Group of characters bounded in Rectangle Box

Actual Result: Grouped Character bounded in Rectangle.

Conclusion: success

Test Case 5:

Test Case Description: Character Descriptor recognising the text characters from detected text regions.

Input: The input to this component is an image of a character, or simply the cropped image of individual component found by the Two pass Labelling and adjacent character grouping.

Some of the Input images are below.

Expected Result: Recognised Text in the image.

Actual Result: characters are matching correctly.

Conclusion: Success

Test Case 6:

Test Case Description: A basic integration test of all the features implemented above.

Expected Result: A complete working set of functionalities without any kind of GUI.

Actual Result: A complete working set of functionalities without any kind of GUI.

Conclusion: Success

11.2.1 Manual testing of GUI:

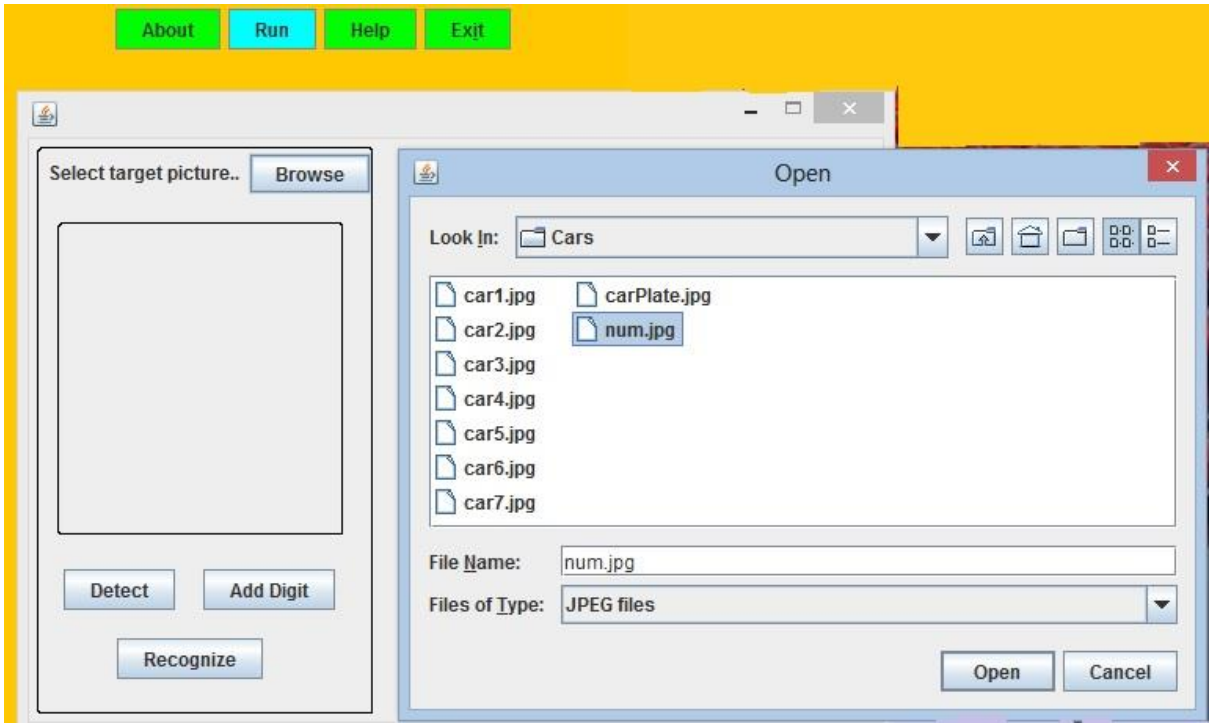
Testing of the overall product through the GUI is done manually. Image gets uploaded correctly from the system and gets executed properly.

Test Case 7:

Test Case Description: Create a basic GUI with all the expected input boxes and fields, without any functionality.

Expected Result: A basic, good looking, simple easy to use GUI.

Actual Result: A basic, good looking, simple easy to use GUI.



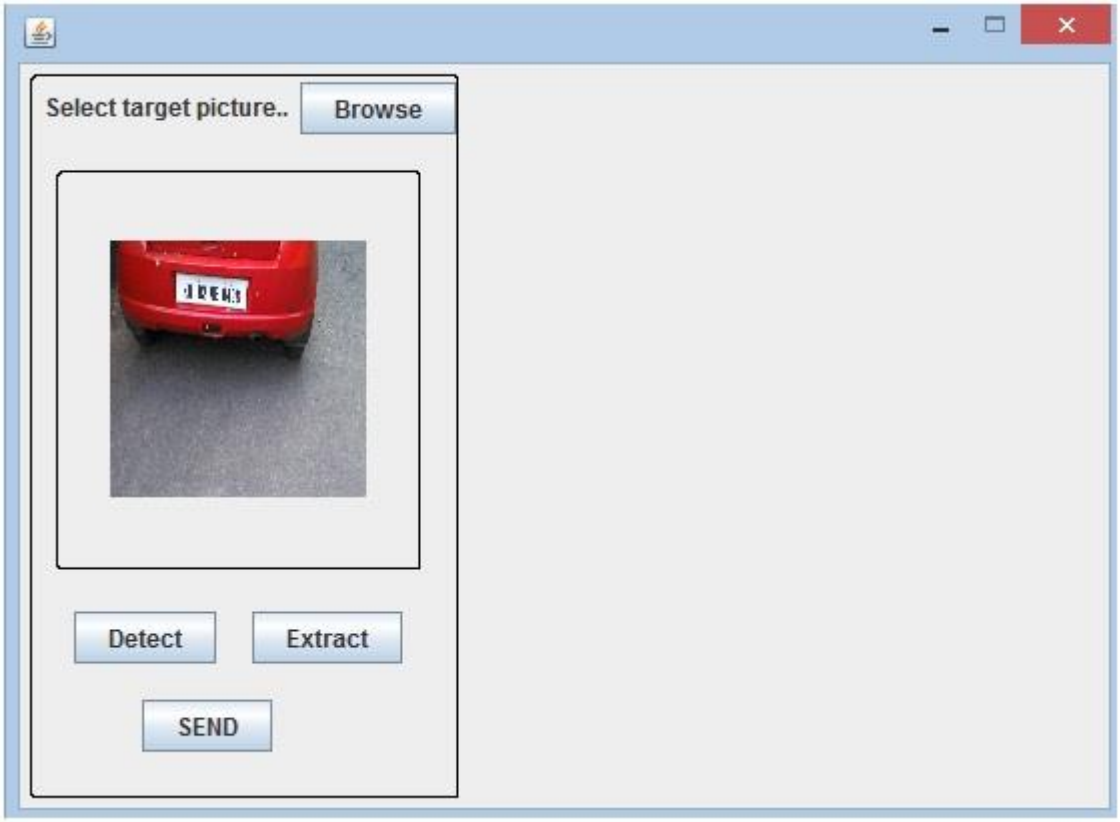
Conclusion: Success

Test Case 8:

Test Case Description: Set up which helps in uploading the image from the system.

Expected Result: Image should get uploaded.

Actual Result: Image uploaded.



Conclusion: Success

Chapter 12

CONCLUSION

12.1 Overall Conclusion

We present a Text recognition from images of vehicle number plate system in this report. Here we Are mainly focusing on recognising the text on images of vehicle number plate, which could help the traffic department in identifying the vehicle details and track them in various cases like hit-and-run, vehicle stolen cases etc. after recognising the text its applications are lot many other than tracking the vehicle details. It can be used in assisting the international tourists to overcome language barriers. The technology can also help a visually handicapped person to increase environmental awareness.

We have proposed a framework for Text Recognition from Images. The framework considers critical challenges in Text Recognition and can recognise robustly under different conditions, such as different image resolutions, camera viewing angles, and image exposer to light. We have successfully applied a user-centred approach in developing a Text recognition system. The system takes advantage of human intelligence if needed and leverages human capabilities by providing Text recognition and translation services. We have developed a prototype system and evaluated the effectiveness of the proposed algorithms. We are further improving the robustness and accuracy of our system. We are particularly interested in enhancing its uses to various other application as mentioned above.

Chapter 13

FUTURE ENHANCEMENT

13.1 Enhancement

Right Now our system focuses only on recognising the text from images of the vehicle number plate. Little more effort on this project could be helpful in enhancing this project to Scene Images. There are lot many application that can be build using the text which is recognised using our system. These applications can be seen as future enhancement of our project.

General Public can use it for text/business card reading. Contact information on business cards can be directly saved in the address book of the mobile device.

Visually impaired individuals can benefit from the developed system if the recognized text is passed to a speech converter.

Students can use the system for automatic reading of books and articles. They can also edit the text, search for keywords on the internet and automatically convert the lecture slide images to notes. The system can also be extended to handwriting recognition allowing student to take images of board notes and convert them to text.

Non-native speakers of languages based on the Latin alphabet (English, French, and German etc.) can also benefit from the system by translating the text written on sign boards and other places into their desired language.

13.1.1 Client Server Architecture

While we currently serves the product as a standalone application, the product would certainly go a long way if it was served in the form of a web app, with a client server architecture.

13.1.2 Android Application

The main reason beyond choosing java as a language in our project is its main feature, Platform Independent.so, a little effort could have been helped us to develop an android app, which could be run on any android application.