1. INTRODUCTION

1.1 Problem Definition: Need of the system

Tollbooths in India generally employ a purely visual system of vehicle classification. However this causes a huge loss of revenue to the firms operating the tollbooths due to rampant malpractices and discrepancies. To keep a tab on the operators some tollbooths employ a system using fibre optic sensors to automatically classify a vehicle in the background and tally the results with the manual entries. However this system is expensive complicated and requires high maintenance. We aim to study the various systems that can be used to replace such a system with a cheaper and efficient alternative

To keep a tab on the operators some tollbooths employ a system using fibre optic sensors to automatically classify a vehicle in the background and tally the results with the manual entries. However this system is expensive complicated and requires high maintenance. We aim to study the various systems that can be used to replace such a system with a cheaper and efficient alternative

However the social scenario in India is significantly different due to problems such as poverty, unemployment as well as a considerably lower respect for rules. This makes it unfeasible to go for a completely automatic tollbooth. The industry requires an automatic vehicle classification system in India not to reduce or eliminate human intervention or labour, but to ensure that human intervention does not cause any financial malpractices. The industry requires a system that runs in the background and merely keeps a cross-check on the manual.

As already stated, the system using fibre optics inherently possesses a large number of problems apart from the main concerns of high cost and maintenance. Although an IR curtain system reduces the cost significantly, it is still quite expensive and cheaper alternatives are desired. As almost all the tollbooths employ cameras for security purposes, it was felt that the feasibility of a system using IP cameras should be tested.

1.2 Scope:

License Plate recognition is one of the techniques used for vehicle identification purposes. The sole intention of this project is to find the most efficient way to recognize the registration information from the digital image (obtained from the camera). This process usually comprises of three steps. First step is the license plate localization, regardless of the license-plate size and orientation. The second step is the segmentation of the characters and last step is the recognition of the characters from the license plate. Thus, this project uncovers the fundamental idea of various algorithms required to accomplish character recognition from the license plate during Template Matching.

This feature of the algorithm mentioned above helped in achieving faster character recognition of the license plate. This process of character recognition consists of steps like Image processing, Defragmentation, Resizing and Character localization that are required to be performed on the image in order for Template Matching to be done.

1.3 Overview of existing system:

The high performance fibre optic sensors are used for detection of moving vehicles. A typical installation consists of an interface device with transmitter (LED), receiver (photo detector), and light guide connection cable (feeder) and fibre optic sensor. As the vehicle passes over the sensors there is a change in the signal levels obtained from the sensors The output signals from the fibre optic sensors are fed into a signal processing and data evaluation unit which comprises of the algorithm, which computes axle count, axle spacing, vehicle lengths and vehicle classes based on time, distance formula, and amount of micro bending.

An IR curtain basically consists of an infra-red transmitter and receiver. These curtains create a clear profile of the vehicle as it passes through it. However the entire profile of the vehicle cannot be obtained by using just one strip of IR curtain due to varying speed of the vehicle that passes the gate. Thus it is important to know the speed of the vehicle. Using the distance between the curtains and the time, we calculate the speed of the vehicle. With the speed of the vehicle known and the frequency of pulses known we can determine the correct profile of the vehicle.

1.4 Proposed System:

Elements of typical LPR systems:

LPR systems normally consist of the following units:

- **Camera(s)** - that take the images of the car (front or rear side)
- **Illumination** - a controlled light that can bright up the plate, and allow day and night operation. In most cases the illumination is Infra-Red (IR) which is invisible to the driver.
- **Computer** - normally a PC running Windows or Linux. It runs the LPR application which controls the system, reads the images, analyzes and identifies the plate, and interfaces with other applications and systems.
- **Software** - the application and the recognition package. Usually the recognition package is supplied as a DLL (Dynamic Link Library).
- **Hardware** - various input/output boards used to interface the external world (such as control boards and networking boards)

There are a number of possible difficulties that the software must be able to cope with. These include:

1. Poor image resolution, usually because the plate is too far away but sometimes resulting from the use of a low-quality camera.

2. Bad images particularly blur.

3. Poor lighting and low contrast due to overexposure, reflection or shadows.

4. An object obscuring (part of) the plate, quite often a tow bar, or dirt on the plate.

5. A different font, popular for vanity plates (some countries do not allow such plates, eliminating the problem).

6. Lack of coordination between countries or states. Two cars from different countries or states can have the same number but different design of the plate.

While some of these problems can be corrected within the software, it is primarily left to the hardware side of the system to work out solutions to these difficulties. Increasing the height of the camera may avoid problems with objects (such as other vehicles) obscuring the plate,but introduces and increases other problems, such as the adjusting for the increased skew of the plate.

On some cars, tow bars may obscure one or two characters of the license plate. Bikes on bike racks can also obscure the number plate, though in some countries and jurisdictions, such as Victoria, Australia, "bike plates" are supposed to be fitted. Some small-scale systems allow for some errors in the license plate. When used for giving specific vehicles access to a barricaded area, the decision may be made to have an acceptable error rate of one character. This is because the likelihood of an unauthorized car having such a similar license plate is seen as quite small. However, this level of inaccuracy would not be acceptable in most applications of an ANPR system.

Core Technology:

If you scan a document into your PC and then open it in a word processor you cannot edit or alter it in any way. This is because it is simply one bitmap made up of thousands of individual pixels. However there is software available, frequently a freebie with scanners that can convert these groups of pixels into characters. This is Optical Character Recognition (OCR), which scans each group of pixels and estimates whether or not it could be a letter and replaces the pixels with the ASCII code for the letter. For instance the ASCII code for the lower case 'a' is 01100001. So, the software scans the whole document and produces a page of letters exactly the same as though you had typed them in, which can be edited or manipulated in any way.

OCR is the fundamental technology used in ANPR and provides the capability to store and sort data. ANPR cameras need to be a special type and set up within certain important parameters as will be described later.

As a vehicle approaches the camera the software takes a series of 'snapshots' and stores them in a file. When the number plate is of sufficient size for the OCR software the frame is scanned and the registration number is converted to ASCII code and held in a list. This continues for a series of images according to the speed and position of the vehicle. The list is

scanned for similarities and a 'favourite' selected to retain. The system would typically scan and compare 10-15 images, with 5 being considered the minimum for high accuracy. Note

Single camera covering one lane:

This could be a pole mounted unit about from 18M to 30M from the vehicle.

Single or multiple cameras covering multiple lanes:

This is a special application requiring input from the ANPR provider.

Town centre cameras already installed:

Usually the cameras will not have been installed with ANPR in mind and so the positioning will not be optimised, they will generally be colour with no infrared illumination and will operating with the shutter speed set to 1/50th.

The first thing to address is the shutter speed if it is adjustable. The best would be if the speed can be set remotely, if not each camera needs to be visited and the speed set manually. The optimum setting is to 1/1000th. Alternative settings may be 1/250th for traffic up to 5 MPH and 1/500th for traffic up to 40 MPH. Note that all these settings will affect the low-light capability of the cameras and a compromise may be required.

Another consideration is that the camera positions and heights would not be at the optimum for ANPR. Particular attention must be paid to the angles of skew and rotation and a guaranty obtained that an acceptable percentage of recognitions will be achieved.

Cameras on motorway bridges:

Again a special application requiring input from the provider.

Congestion charging cameras:

This application requires input from the ANPR provider and local authority before even starting to think of a specification.

Cameras in Police vehicles:

These are normally colour cameras mounted on a swivel mount and can view images to the front or either side of the vehicle. This is another special application requiring input from the provider.

Overview cameras:

It is often necessary to have a conventional colour image of the vehicle especially where prosecution or congestion charging is the application. This would be a separate colour camera mounted alongside or just below the ANPR camera. Saving the overview image is triggered by the ANPR camera registering a number plate. This then adds a colour image to the same file for future reference. It is generally a false economy to attempt to combine the number plate recognition and overview using a single camera for 24/7 operation.

Example flow diagram for ANPR system:

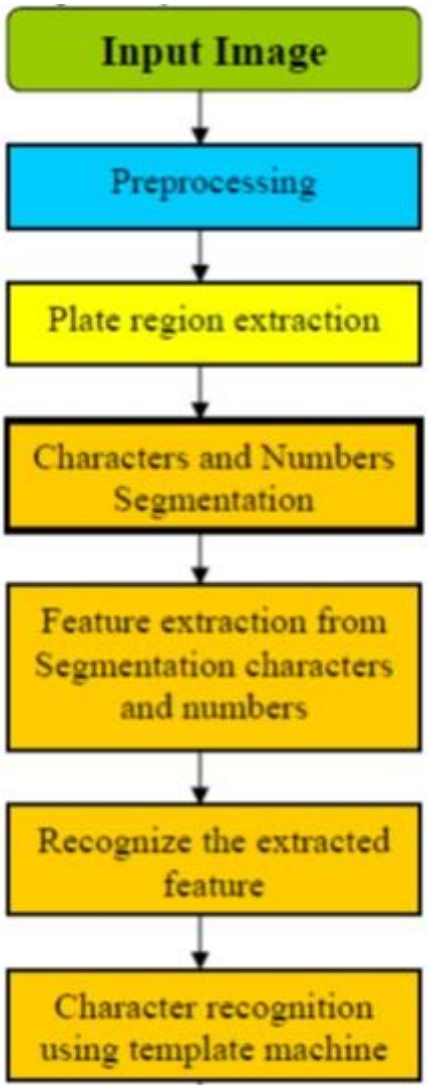


Fig.2
Flow Diagram

Steps followed:

Number plate recognition basically consists of three concrete steps namely:

1. Number Plate Extraction.
2. Character Segmentation.
3. Template Matching.

However, these steps are further divided into a series of other steps whose working is as followed:

1. **Loading an RGB image:**

The image whose number plate recognition is to be done is loaded.

Fig.3
Loading an image

2. Grayscale conversion:

This RGB image is converted to grayscale image using cvCvtColor() function.

Fig.4
Grayscale Image

3. Histogram equalization:

Histogram equalisation is performed on this image using cvEqualizeHist() function.

Fig.5
Histogram Equalized Image

4. Dilation:

This image is dilated using cvDilate() function.

Fig.7
Dilated Image

5. Plate region extraction:

Plate region is found out by passing a rectangular image over the previous using cvMatchTemplate() function.

Fig.9
Extracted Plate Region

6. Character segmentation:

Characters are segmented from the number plate image which is then used for template matching.

Fig.10
Segmented Character

7. Template matching:

Segmented characters are template matched with the templates of each charcter and the number plate is identified as a string.

```
Processing a 1200x1600 image with 1 channels
contour no. = 0          distance from left = 69
contour no. = 1          distance from left = 271
contour no. = 2          distance from left = 218
contour no. = 3          distance from left = 170
contour no. = 4          distance from left = 115
contour no. = 5          distance from left = 369
contour no. = 6          distance from left = 507
contour no. = 7          distance from left = 464
contour no. = 8          distance from left = 418
contour no. = 9          distance from left = 310
number added to the result = M
number added to the result = H
number added to the result = 0
number added to the result = 1
number added to the result = A
number added to the result = V
number added to the result = 6
number added to the result = 2
number added to the result = 7
number added to the result = 5
MH01AV6275
```

Fig.11
Output in string form

Fig.12
Output after Template Matching

OpenCV:

OpenCV is a computer vision library originally developed by Intel and now supported by Willow Garage. It is free for use under the open source BSD license. The library is cross-platform. It focuses mainly on *real-time* image processing. If the library finds Intel's Integrated Performance Primitives on the system, it will use these commercial optimized routines to accelerate it. OpenCV is NOT a piece of software that you run and process images. You need to write code.

You can download Microsoft’s Visual Studio Express Edition (for free). It is one superb IDE. You need to download the Visual C++ 2010 Express.

Also, OpenCV is not some executable file that you double click and it'll start working. It is pure code, library files and DLL files. When you write your own code, you "link" to these library files to access the OpenCV functions.

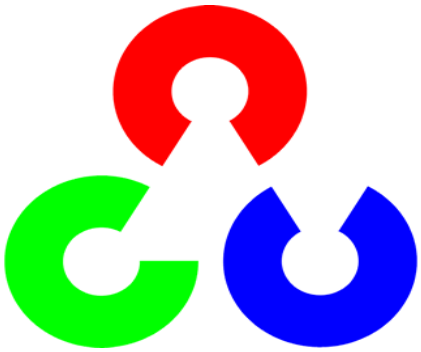


Fig.13

OpenCV

Why OpenCV?

There are a couple of why to prefer OpenCV over Matlab.

Specific

OpenCV was made for image processing. Each function and data structure was designed with the Image Processing coder in mind. Matlab, on the other hand, is quite generic. You get almost anything in the world in the form of toolboxes. All the way from financial tool boxes to highly specialized DNA tool boxes.

Speedy

Matlab is just way too slow. Matlab itself is built upon 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.

If you use C/C++ you don't waste all that time. You directly provide machine language code to the computer, and it gets executed. So ultimately you get more image processing, and not more interpreting.

Sure you pay the price for speed – a more cryptic language to deal with, but it's definitely worth it. You can do a lot more. You could do some really complex mathematics on images with C and still get away with good enough speeds for your application.

Efficient

Matlab uses just way too much system resources. With OpenCV, you can get away with as little as 10mb RAM for a realtime application. But with today's computers, the RAM factor isn't a big thing to be worried about. You do need to take care about memory leaks, but it isn't that difficult. You can read this article about Memory Management in OpenCV if you want.

Requirements for OpenCV:

Operating System:

- 32-bit MS Windows (95/98), 32-bit MS Windows (NT/2000/XP), All 32-bit MS Windows(95/98/NT/2000/XP),All POSIX (Linux/BSD/UNIX-like OSes), OS X, Linux, Win2K, WinXP
- Verified on Windows 7 x86_64; should also be compatible with Windows XP SP3 and newer.
- OpenCV 2.1 is compatible with VC++ 2008 and VC++ 2010.

Programming Language:

- Visual C++

Disk space requirement for OpenCV Package:

- 4 Mb

Supported Architecture:

- x86

- x64 (WOW)

Supported Operating Systems:

- Microsoft® Windows® XP (x86) Service Pack 3
  All editions except Starter Edition
- Microsoft® Windows® Vista (x86 & x64) with Service Pack 2
  All editions except Starter Edition
- Microsoft® Windows® Server 2003 R2 (x86 & x64)
  All editions
- Microsof® Windows® Server 2008 (x86 & x64) with Service Pack 2
  All editions
- Microsoft® Windows® Server 2008 R2 (x64)
  All editions
- Microsoft® Windows® 7
  All editions

Hardware Requirements:

- 1.6 GHz or faster processor.
- 1024 MB RAM (1.5 GB if running on a virtual machine).
- 3 GB of available hard-disk space.
- 5400 RPM hard-disk drive.
- DirectX 9-capable video card running at 1024 x 768 or higher display resolution.
- DVD-ROM drive.

1.5 Assumptions and constraints:

Assumptions:

- Input is an image of a stationary Car.
- Only the most common type of license plates (single line)will be dealt with.
- The license Plate has a yellow background with text written in Black.

Constraints:

- If the image contains too much spoiled license plate or has designs on it, the program can fail to localize the license plate.
- If the license plate happens to be much tilted from horizontal, then again the result of segmentation of the license plate is very poor.

1.6 System Requirements:

- Microsoft Visual Studio 2008.
- OpenCV.
- MFC (Microsoft Foundation Classes) library for the GUI part.

2. SOFTWARE REQUIREMENT SPECIFICATION DESIGN

2.1 Functional Specification:

Software	Hardware		
	Processor	**RAM**	**Disk Space**
Windows XP, Vista or 7.	Pentium VI at 2.6GHz or better.	2 GB or more	3 GB
Microsoft Visual C++ 2008 or 2010.	Pentium VI at 2.6GHz or better.	2 GB or more	3 GB (Excluding data size)
OpenCV 2.1	Pentium VI at 2.6GHz or better.	2GB or more	1 GB

Table.1
Functional Specification

2.2 UML Diagrams:

Use-Case Diagram:

Actor:

- User

Use Case:

- Capture image C1
- Verify Vehicle
- Identify Number

Precondition:

- Vehicle will stop at Toll Booth
- Then take picture.

Post condition:

- If vehicle is not verifiable, inform police immediately.

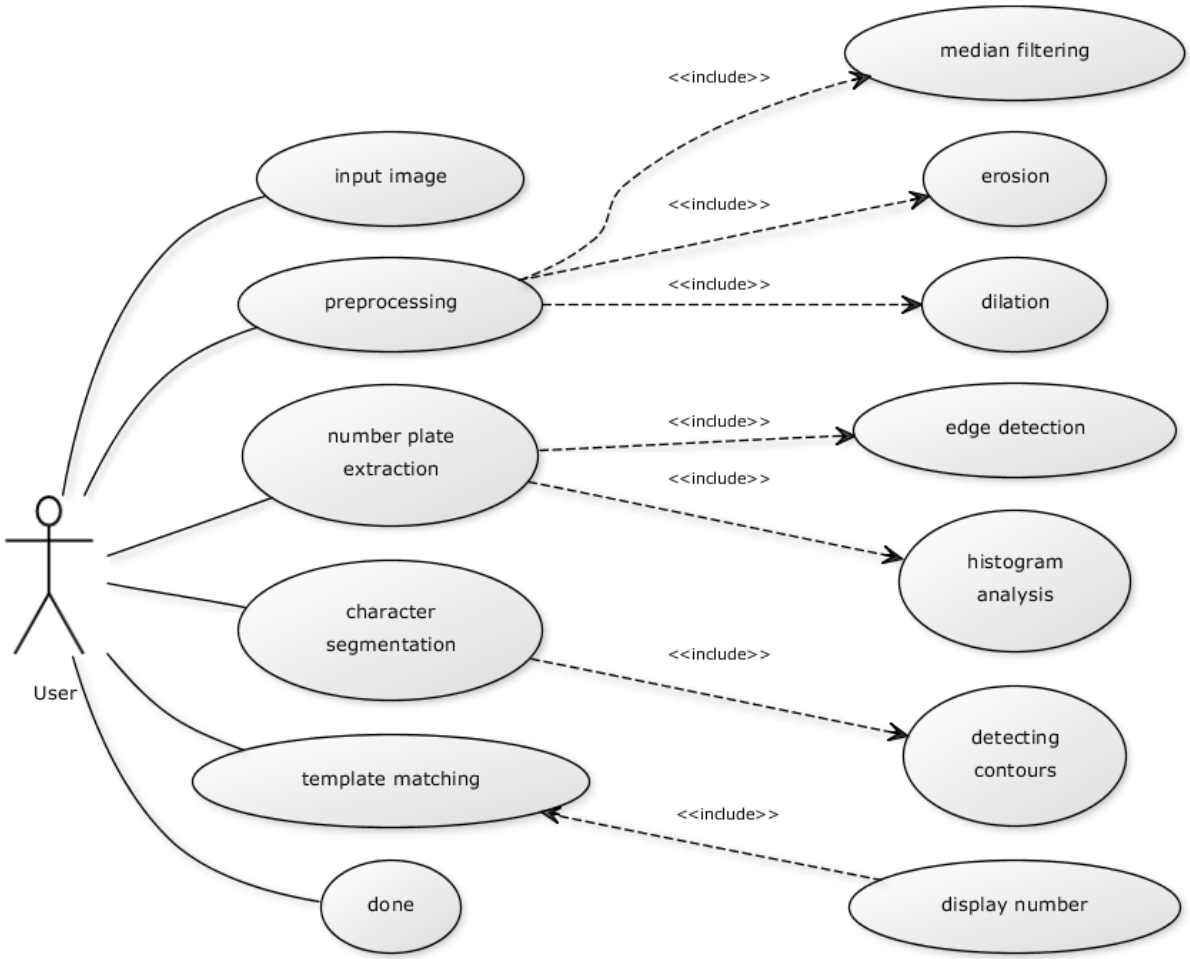


Fig.14

Use Case Diagram

Class Diagram:

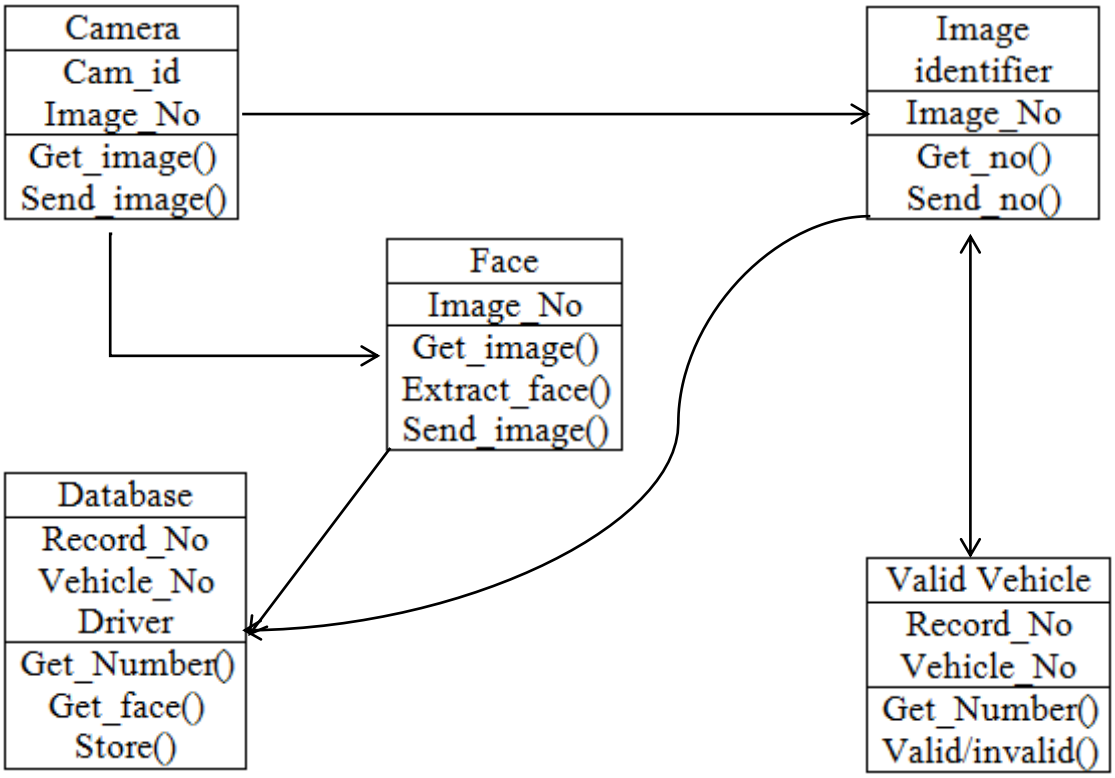


Fig.15
Class Diagram

The entities participating in the use case are camera, Image identifier, Database, Valid vehicle.

The attributes and methods of entities are as follows:

1. Camera:

Attributes:	**Methods:**
Cam_id	Get_image()
Image_No	Send_image

Sequence Diagram:

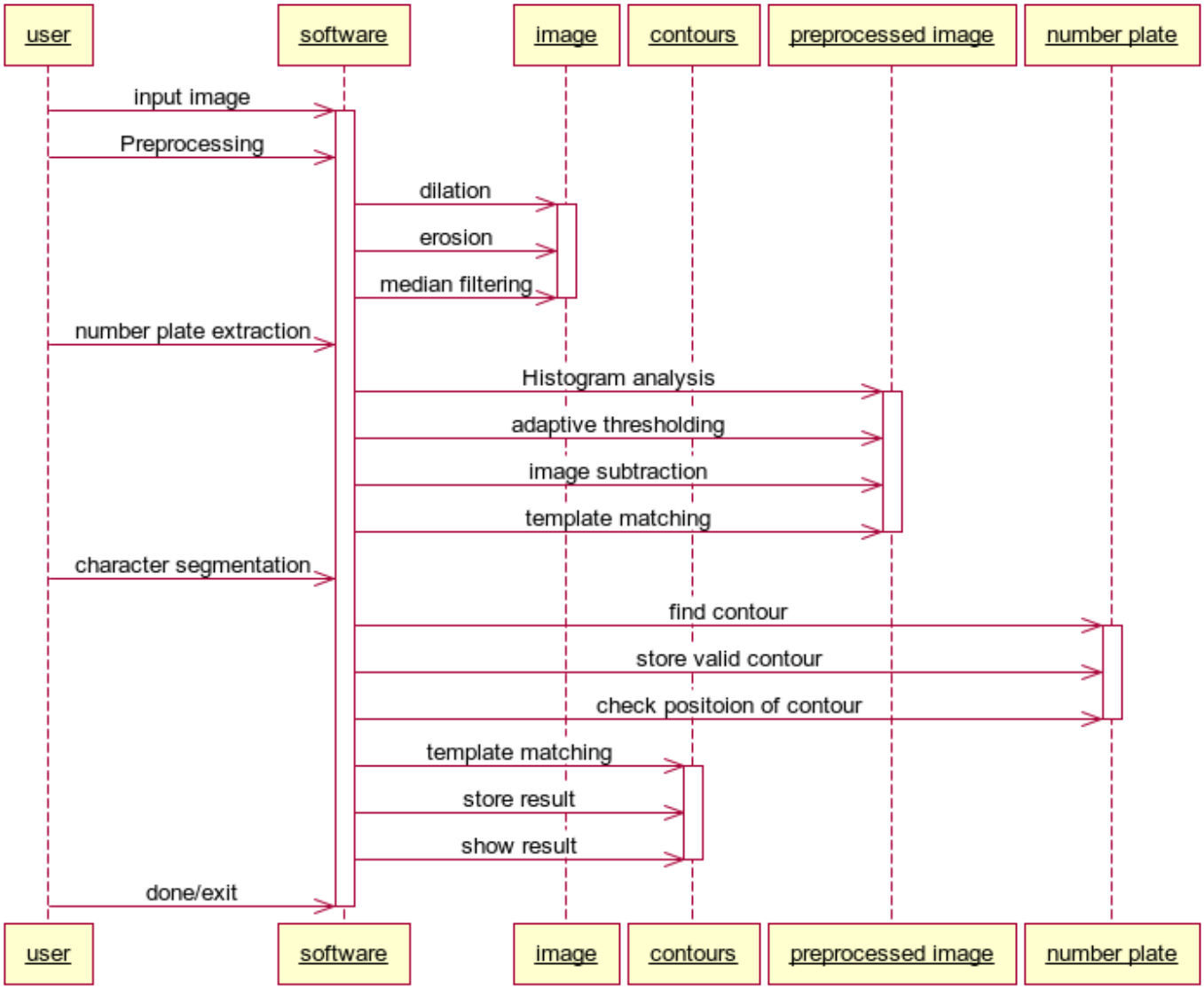


Fig.16

Sequence Diagram

Activity Diagram:

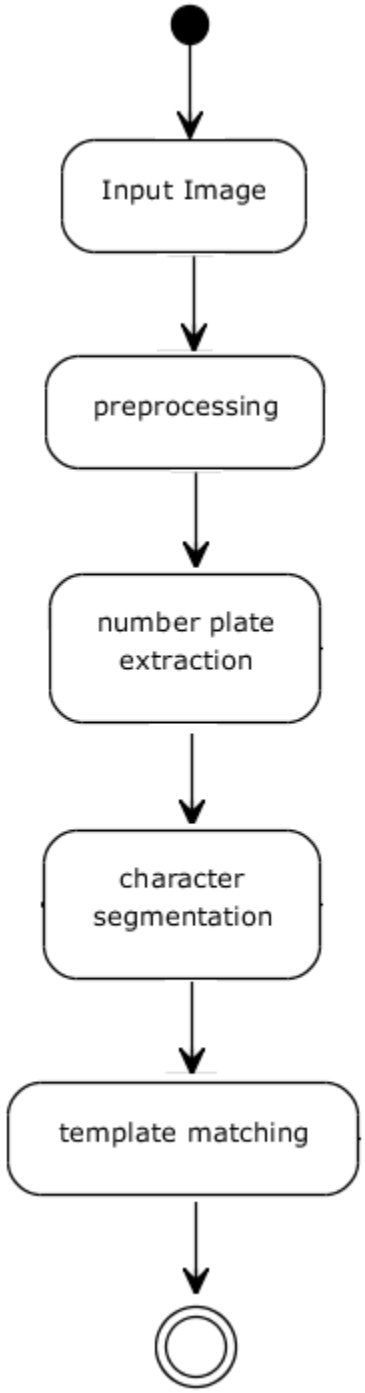


Fig.17
Activity Diagram

Description:

The main aim behind the number plate recognition is the storage of information of vehicles with respect to its number plate characters.The information thus can be used to track the vehicles.

Precondition:

- A camera is placed at 4-5 m away from the vehicle to get the clear view of the number plate.
- The images are to be stored in a repository.

Post condition:

- The repository will consist of all the number of the number plates which have passed the toll booth.
- The police department can map this numbers to check if the number plate is valid.

Normal flow of events:

- The vehicle approaches the toll booth.
- The camera captures the image of the number plate.
- The software identifies and extracts the characters of the number plate of the vehicle.
- It can be checked for its validity.

2.3 Architecture Diagram:

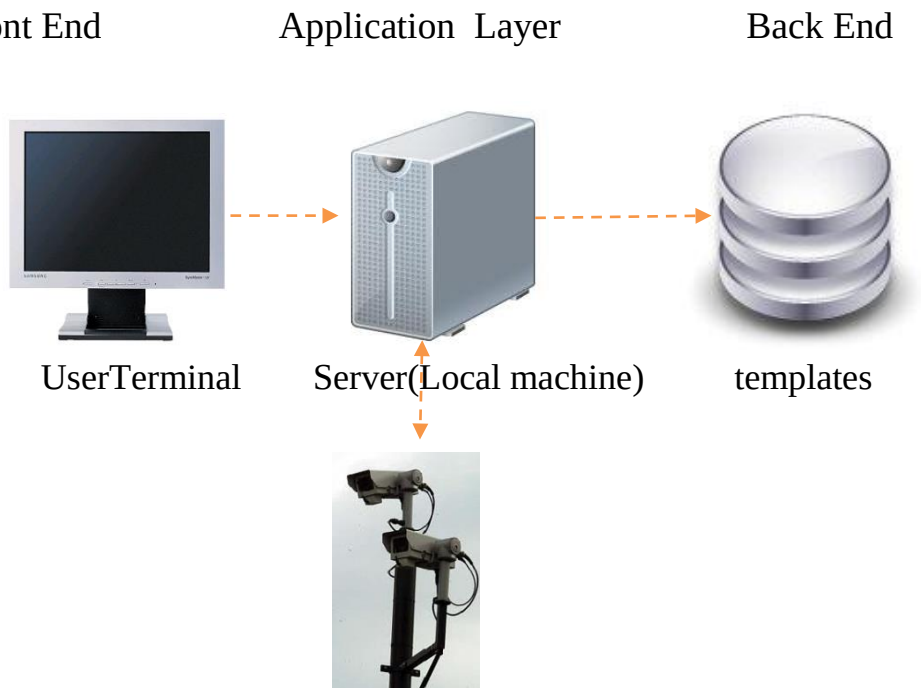


Fig.18
Architecture Diagram

2.4 Interface design:

2.4.1 User Interfaces:

This contains a user input window where in the input image of vehicle whose number plate recognition is to be done is specified. The input image is loaded from the browse option from the list of images. This image will then be processed to generate the number plate in string form as an output after recognition.

2.4.2 Software Interfaces:

The GUI of this application is developed using MFC (Microsoft Foundation Classes) libraries and VC++ code.

2.4.3 Communication Interfaces:

This application could connect to any mysql database server provided all the connection string such as host name, port no, database name and username and password are available. However, database connectivity is out of our scope.

3. PROJECT IMPLEMENTATION:

3.1 Implementation Plan and Gantt Chart:

TASK SHEET:

The task sheet divides the entire project into smaller and manageable components. This helps in planning, organizing and controlling the project and various activities involved in the project. It makes more systematic and the flow of task is determined. It helps us in analysing our project progress as we track tasks in our project. The task sheet for this project is made in Microsoft Project 2003.The snapshot can be seen as shown:

	Task Name	Duration	Start	Finish	Predecessors
1	**Requirement gathering**	**9 days?**	**Wed 2/2/11**	**Mon 2/14/11**	
2	Identify needs	2 days?	Wed 2/2/11	Thu 2/3/11	
3	Define objectives	2 days?	Fri 2/4/11	Mon 2/7/11	2
4	Define detailed problem	3 days?	Tue 2/8/11	Thu 2/10/11	3
5	Define project scope	2 days?	Fri 2/11/11	Mon 2/14/11	4
6	**Requirement analysis**	**10 days?**	**Tue 2/15/11**	**Mon 2/28/11**	**1**
7	Analyse needs	2 days?	Tue 2/15/11	Wed 2/16/11	5
8	Filter out important nee	2 days?	Thu 2/17/11	Fri 2/18/11	7
9	Objective anlysis	1 day?	Mon 2/21/11	Mon 2/21/11	8
10	Study existing systems	3 days?	Tue 2/22/11	Thu 2/24/11	9
11	Software requirement :	2 days?	Fri 2/25/11	Mon 2/28/11	10
12	**Design**	**4 days?**	**Tue 3/1/11**	**Fri 3/4/11**	**6**
13	Draw data flow,uml dia	4 days?	Tue 3/1/11	Fri 3/4/11	11
14	**Coding**	**22 days?**	**Mon 3/7/11**	**Tue 4/5/11**	**12**
15	Browsing and loading i	1 day?	Mon 3/7/11	Mon 3/7/11	13
16	Binarization of image	2 days?	Tue 3/8/11	Wed 3/9/11	15
17	Preprocessing of image	3 days?	Thu 3/10/11	Mon 3/14/11	16
18	Number plate recognitic	2 days?	Tue 3/15/11	Wed 3/16/11	17
19	Character segmentatio	7 days?	Thu 3/17/11	Fri 3/25/11	18
20	Template matching	7 days?	Mon 3/28/11	Tue 4/5/11	19
21	**GUI**	**7 days?**	**Wed 4/6/11**	**Thu 4/14/11**	**14**
22	Designing the gui	5 days?	Wed 4/6/11	Tue 4/12/11	20
23	Testing the gui	2 days?	Wed 4/13/11	Thu 4/14/11	22
24	**Testing**	**8 days?**	**Fri 4/15/11**	**Tue 4/26/11**	**21**
25	Flat file testing	2 days?	Fri 4/15/11	Mon 4/18/11	23
26	Integration testing	2 days?	Tue 4/19/11	Wed 4/20/11	25
27	Mutation testing	2 days?	Thu 4/21/11	Fri 4/22/11	26
28	Validation and verificat	2 days?	Mon 4/25/11	Tue 4/26/11	27
29	**Working on black book**	**6 days?**	**Wed 4/27/11**	**Wed 5/4/11**	**24**
30	Start and end of black l	6 days?	Wed 4/27/11	Wed 5/4/11	28

Fig.19
Task Sheet for Gantt Chart

GANTT CHARTS:

A Gantt chart is a popular type of bar chart that illustrates the project schedule. Gantt charts illustrate the start and finish dates of the terminal elements and summary elements of the project under consideration. The Gantt charts also show the dependency relationship between the activities. Thus a Gantt chart is a graphical representation of the duration of the task against the progression of time. It is a useful tool for planning and scheduling projects. A Gantt chart is helpful when monitoring a projects progress.

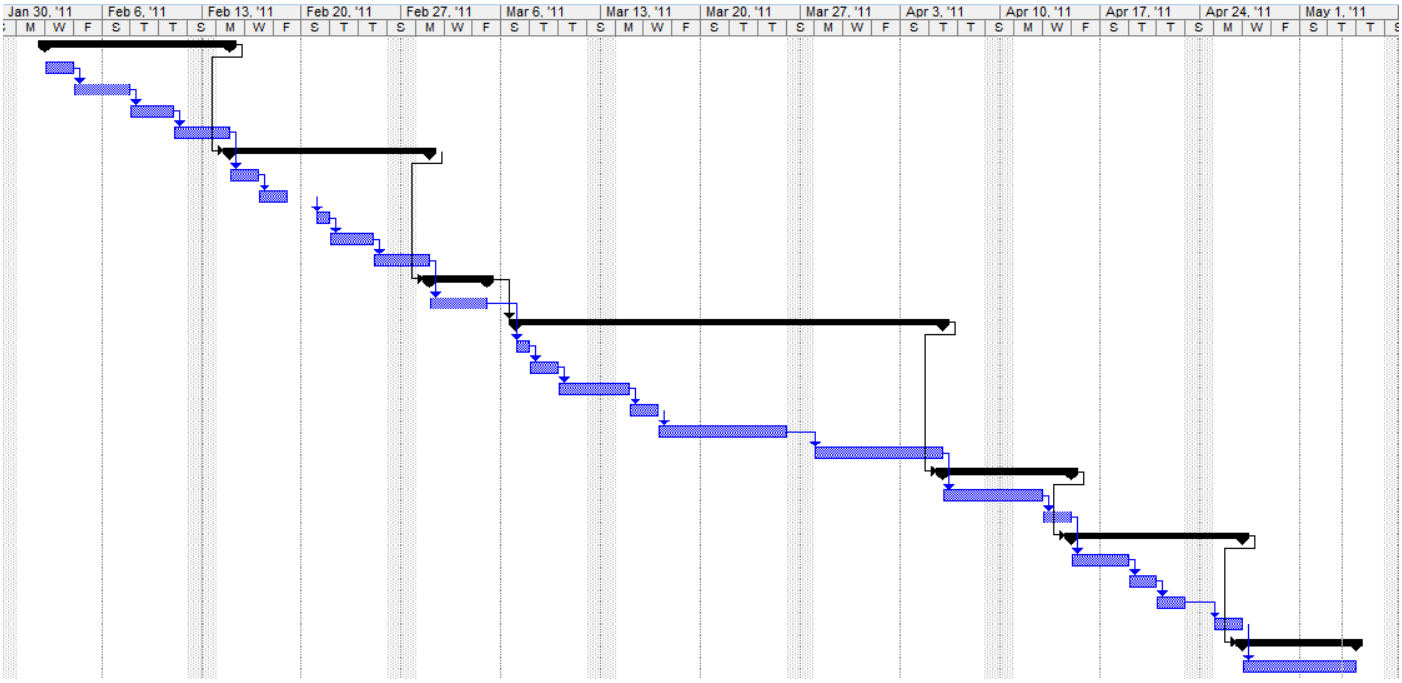


Fig.19
Gantt Chart

3.2 WORK BREAKDOWN STRUCTURE:

A complex project is made manageable by first breaking it into individual components in a hierarchical structure, known as the work breakdown structure, or WBS. Such a structure defines tasks that can be completed independently of the other tasks, facilitating resource allocation, assignment of responsibilities, and measurement and control of the project.

A Work Break down structure is a result oriented family tree that captures all the work of a project in an organised way. It is often portrayed graphically as a hierarchical tree; however, it can also be a tabular list of element categories and task or the intended tasks list that appears in a Gantt chart schedule.

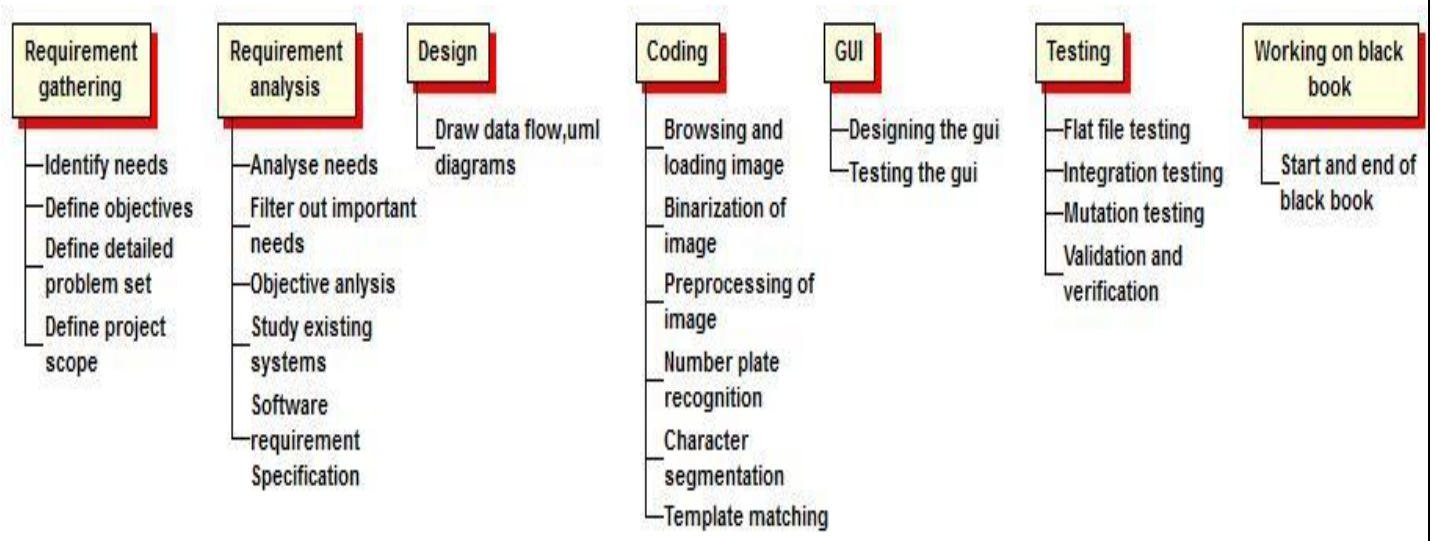


Fig.21
Work Breakdown Structure

3.3 Code with reference to Design with proper comments and brief description:

```
#include<stdio.h>
#include<cv.h>
#include<highgui.h>


        //To draw a histogram

    IplImage* DrawHistogram(CvHistogram *hist, float scaleX=4, float scaleY=3)

{
        float histMax = 0;
        cvGetMinMaxHistValue(hist, 0, &histMax, 0, 0);
        IplImage* imgHist = cvCreateImage(cvSize(256*scaleX, 64*scaleY), 8 ,1);
        cvZero(imgHist);

        for(int i=0;i<255;i++)

         {
                float histValue = cvQueryHistValue_1D(hist, i);
                float nextValue = cvQueryHistValue_1D(hist, i+1);

                CvPoint pt1 = cvPoint(i*scaleX, 64*scaleY);
                CvPoint pt2 = cvPoint(i*scaleX+scaleX, 64*scaleY);
                CvPoint pt3 = cvPoint(i*scaleX+scaleX, (64-
nextValue*64/histMax)*scaleY);
                CvPoint pt4 = cvPoint(i*scaleX, (64-histValue*64/histMax)*scaleY);

                int numPts = 5;
                CvPoint pts[] = {pt1, pt2, pt3, pt4, pt1};

                cvFillConvexPoly(imgHist, pts, numPts, cvScalar(255));
         }

        //Return histogram image
        return imgHist;
}
```

```
                }

                IplImage* img_dilate = cvCloneImage(img);

                //Perform dilation for edge detection which is dilated image-binary image
                cvDilate(img,img_dilate,NULL,1);

        //Create an image to store edge detected result
                IplImage *img_sub = cvCreateImage(cvGetSize(img), img->depth, img->nChannels);
                cvZero(img_sub);

                // load template image to track number plate
                IplImage *tpl = cvLoadImage("C:/pictures/rectangle.jpg",0);

                //Perform edge detection
                cvSub(  img_dilate , img , img_sub, NULL);

                IplImage *imgResult = cvCreateImage(cvSize(img_sub->width - tpl->width + 1 , img_sub->height - tpl->height + 1), IPL_DEPTH_32F, 1);

                cvZero(imgResult);

                //Perform template matching to detect image
                cvMatchTemplate(img_sub, tpl, imgResult, CV_TM_SQDIFF);

                CvPoint    minloc, maxloc;
                double    minval, maxval;

                //Perform minmax function to locate the number plate detected in the image
                cvMinMaxLoc( imgResult, &minval, &maxval, &minloc, &maxloc, 0 );

                //Draw rectangle around the detected number plate
                cvRectangle(cc_color, cvPoint(minloc.x, minloc.y), cvPoint(minloc.x + tpl->width, minloc.y + tpl->height), CV_RGB(255, 0, 0), 1, 0, 0 );
```

```
                //Show detected number plate
                cvShowImage("numberplate",cc_color);

                cvWaitKey(0);

              //Locate number plate in original image
                xxx=(int)((100/(decrement-3))*minloc.x);
                yyy=(int)((100/(decrement-2))*minloc.y);
                www=(int)((100/(decrement-1))*(minloc.x+tpl->width)-xxx);
                hhh=(int)((100/(decrement-1))*(minloc.y+tpl->height)-yyy);

                //Release image
               cvReleaseImage( &img );

        }

  IplImage *img, *cc_color; /*IplImage is an image in OpenCV*/
CvMemStorage *mem;
CvSeq *contours, *ptr;

        //Load original image
img = cvLoadImage("C:/pictures/sample.jpg", 0);

            if( img == 0 )

                {
                        fprintf( stderr, "Cannot load file %s!\n", argv[1] );
                        return 1;
                }

        //Set number plate region as the region of interest
        cvSetImageROI(img, cvRect(xxx,yyy,www,hhh));

        cvShowImage("numberplate",img);

        cvWaitKey(0);

{
   int height,width,step,channels;
```

```
    uchar *data;
    int i,j,k;



    // get the image data
    height    = img->height;
    width     = img->width;
    step      = img->widthStep;
    channels  = img->nChannels;
    data      = (uchar *)img->imageData;
    printf("Processing a %dx%d image with %d channels\n",height,width,channels);


    // invert the image
    for(i=0;i<height;i++) for(j=0;j<width;j++) for(k=0;k<channels;k++)
       data[i*step+j*channels+k]=255-data[i*step+j*channels+k];

    // show the image
    //cvShowImage("image", img );
    //cvWaitKey(0);
  }

  IplImage* tempImg;
  tempImg=cvCloneImage(img);


//Pre Processing on the number plate

  cvMorphologyEx(img,img,tempImg,CV_SHAPE_RECT,CV_MEDIAN,1);
  cvReleaseImage( &tempImg );
  cc_color = cvCreateImage(cvGetSize(img), IPL_DEPTH_8U, 3);
  cvThreshold(img, img, 150, 255, CV_THRESH_OTSU);
  cvMorphologyEx(img,img,tempImg,CV_SHAPE_RECT,CV_MOP_ERODE,1);
  cvMorphologyEx(img,img,tempImg,CV_SHAPE_RECT,CV_MOP_CLOSE,1);
  cvMorphologyEx(img,img,tempImg,CV_SHAPE_RECT,CV_MOP_OPEN,1);
  cvMorphologyEx(img,img,tempImg,CV_SHAPE_RECT,CV_MOP_DILATE,1);
  cvMorphologyEx(img,img,tempImg,CV_SHAPE_RECT,CV_MEDIAN,1);
```

```
mem = cvCreateMemStorage(0);



        //Check total number of contour in the image
int t=cvFindContours(img, mem, &contours, sizeof(CvContour), CV_RETR_CCOMP,
   CV_CHAIN_APPROX_SIMPLE, cvPoint(0,0));



//Assume maximum contour founded are less than 100, create an iplimage array to store
these contours

IplImage *frame_buffer[100];
IplImage *frame_buffer1[100];

int count=0;
char buff[1000];

//track position of the charater in the image
 int distancefromleft[100];

        //to store sorten position of characters
int numberpositions[100];
int x=0;
for (int i = 0; i < 100; i++)
{
   numberpositions[i]=-1;
}

for (ptr = contours; ptr != NULL; ptr = ptr->h_next)
{
   frame_buffer[count]=cvCreateImage( cvGetSize(cc_color), cc_color->depth, 1 );
   cvZero(frame_buffer[count]);
   CvScalar color = CV_RGB( 255, 255, 255 );
```

```
//Draw bounding box on the contour
        CvRect boundbox = cvBoundingRect(ptr);

                templates[7]=cvLoadImage("C:/pictures/7.jpg", 0);
                templates[8]=cvLoadImage("C:/pictures/8.jpg", 0);
                templates[9]=cvLoadImage("C:/pictures/9.jpg", 0);


      char templatename[10]={'7','A','H','0','1','6','5','M','2','V'};
      char result[20];

      for (int i = 0; i < x; i++)
      {
                        result[i]='_';
      }

      for (int i = 0; i < x; i++)
      {
        for(int j = 0;j < 10; j++)
                    {
                        //Check if the template is valid for cvmatchtemplate function
                        if(((templates[j]->width)>(frame_buffer1[numberpositions[i]]-
>width))|| ((templates[j]->height)>(frame_buffer1[numberpositions[i]]->height)))

              {
                 continue;
               }



            //Make iplimage for cvmatchtemplate function result
                                IplImage* imgResult =
cvCreateImage(cvSize(frame_buffer1[numberpositions[i]]->width - templates[j]->width + 1
, frame_buffer1[numberpositions[i]]->height - templates[j]->height + 1), IPL_DEPTH_32F,
1);
             cvZero(imgResult);

                                cvMatchTemplate(frame_buffer1[numberpositions[i]],
templates[j],imgResult, CV_TM_SQDIFF);
```

```
                    double min_val=0, max_val=0,maxx=0, minn=0;

        CvPoint min_loc, max_loc;
        cvMinMaxLoc(imgResult, &min_val, &max_val, &min_loc, &max_loc);

        cvReleaseImage( &imgResult );


                    char temp11[20], temp12[20];
                    sprintf( temp11, "%d", max_val );
                    sprintf( temp12, "%d", min_val );

                    //Check if the result is 0, if not, check for the next template
                    if(temp11[0]!='0')
                    {
                        continue;
                    }
                    if(temp12[0]!='0')
                    {
                        continue;
                    }
                    result[i]=templatename[j];
                    printf("number added to the result = %c \n", templatename[j]);
                    break;

            }
}

          //Print the result
          for(int i=0;i<10;i++)
{

    printf("%c",result[i]);
}
cvWaitKey(0);

/* free memory */
cvDestroyWindow( "image" );
```

```
    cvReleaseImage( &img );
    cvReleaseImage(&cc_color);

  return 0;
  }
```

3.4 Snapshots of UI:

2. Actual Image:

3. Number plate detection:

4. Number plate extracted:

5. Median filtered image:

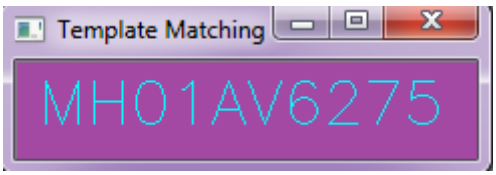
6. Eroded image:

7. Opened image:

8. Closed image:

9. Detect Contours:

10. Display result:



11 Test Cases and Report:

No.	Test Case	Input	Actual Output	Expected Output	Is actual output is same as expected output?
1.	Process without loading an image	nothing	It will show error message – “please load a valid image”	It will show error message – “please load a valid image”	Yes
2.	Load an image	.jpg image	Gray scale image is loaded	Gray scale image	Yes
3.	Load an image without number plate	.jpg image	It will show error message “number plate not present”	It will show error message “number plate not present”	Yes
4.	Load an image with noisy number plate	.jpg image	It will show error message “number plate is noisy”	It will show error message “number plate is noisy”	Yes
5.	Load an image with language other than english	.jpg image	It will show error message “templates cannot be matched”	It will show error message “templates cannot be matched”	Yes
6.	Load an image with number plate of different font	.jpg image	It will show error message “templates cannot be matched”	It will show error message “templates cannot be matched”	Yes

Table.2
Test Cases

4. DOCUMENTATION AND CD (INSTALLABLE):

4.2 System Manual:

Tools:

1. VISUAL STUDIO 2008
2. OPENCV 2.1

1. Installation for Microsoft Visual Studio 2008:

To Install Help for Standard and Professional Editions:

1. Insert the CD or DVD you installed Visual Studio from, or browse to the network share where you installed from.
2. Double click setup.exe
3. In wizard, click **Install Product Documentation**.

To Install Help for Professional Editions:

1. From Start menu, choose Control Panel and then choose Add or Remove Programs.
2. Select the Professional Edition you installed and then click Change or remove.
3. In the setup wizard, choose Add option Products and then click Next.
4. Select Microsoft Visual Studio Professional 2008 and then click Next.
5. Specify whether you have the installation media or intend to download the documentation from the web and then click INSTALL.

2. Installation for OpenCV 2.1:

1. Insert the CD or DVD you installed OpenCV from, or browse to the network share where you installed from.
2. Double-click setup.exe.

4.3 User Manual with Installation Procedure:

Our project is easy to use. By following the below steps the code can be setup with full functionality.

1. Pre-conditions:

1. For implementing our project scenario, Microsoft Visual Studio 2008 professional and Microsoft SQL Server 2008 need to be installed on computer.
2. OpenCV 2.1 needs to be installing on computer.

2. User Manual:

1. Click on Input to load the image whose number plate recognition is to be done.
2. The image is loaded on which preprocessing will be done before getting the plate region.
3. Click on Preprocessing which will take the input image and perform several operations on it like grayscale conversion, histogram equalization, binarization, dilation and edge detection.
4. Click on Number Plate Extraction to find the plate region from the output of the previous step by matching a plain rectangle of the size of number plate with the output image.
5. Click on Character Segmentation to segment the characters that we get as an output to the previous step.
6. Click on Template Matching to match the segmented characters with the templates of alphabets and numbers that we have. This step matches the templates and output of previous step and displays the number plate as a string.
7. Follow the same procedures as in steps 1 to 6 to find the number plate for different images.

8. At any point of time you can click ok or cancel on any form to log out from application.

5. BIBLIOGRAPHY:

Papers:

[1] S. Hamidreza Kasaei, S. Mohammadreza Kasaei, S. Alireza Kasaei
International Journal of Computer Theory and Engineering, Vol. No. 2,
2 April, 2010.

[2] Serkan Ozbay, and Ergun Ercelebi 1793-8201
World Academy of Science, Engineering and Technology 9 2005

[3] Deepak Kumar Gupta-Y6154, Siddhartha Kandoi-Y6472
CS 676: Image Processing and Computer Vision 2009-10 Semester 1

Books:

1. International Journal of Computer Theory and Engineering, Vol. No. 2,
2 April, 2010, Pg. 57.

2. R. Gonzalez, R .Woods , Digital Image Processing, Prentice Hall, New Jersey,
2002, Chapter 5. Operations on an image.

Websites:

1. http://www.anpr-tutorial.com

2. http://www.cctv-information.co.uk/i/An_Introduction_to_ANPR

3. http://www.visl.technion.ac.il/projects/2003w24/

4. http://www.stackoverflow.com

5. http://opencv.willowgarage.com/documentation

6. http://www.aishack.com