AUTOMATIC NUMBERPLATE RECOGNITION

MAIN PROJECT REPORT

Submitted by

ALEENA JOSEPH

CHINJU SHENUS N S

HRIDYA P V

ROSHAN K ANTONY

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY (B.TECH)

in

COMPUTER SCIENCE & ENGINEERING

of

UNIVERSITY OF CALICUT

Under the guidance of

Ms. REMYA RAVEENDRAN

JUNE 2011

Department of Computer Science & Engineering

JYOTHI ENGINEERING COLLEGE, CHERUTHURUTHY

THRISSUR 679 531

AUTOMATIC NUMBERPLATE RECOGNITION

MAIN PROJECT REPORT

Submitted by

ALEENA JOSEPH

CHINJU SHENUS N S

HRIDYA P V

ROSHAN K ANTONY

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY (B.TECH)

in

COMPUTER SCIENCE & ENGINEERING

of

UNIVERSITY OF CALICUT

Under the guidance of

Ms. REMYA RAVEENDRAN

JUNE 2011

Department of Computer Science & Engineering

JYOTHI ENGINEERING COLLEGE, CHERUTHURUTHY

THRISSUR 679 531

Department of Computer Science & Engineering
JYOTHI ENGINEERING COLLEGE, CHERUTHURUTHY
THRISSUR 679 531

JUNE 2011

BONAFIDE CERTIFICATE

Certified that this project report **“ . . . AUTOMATIC NUMBERPLATE RECOGNITION . . . ”** being submitted in partial fulfillment of the requirements for the award of degree of **Bachelor of Technology** of **University of Calicut** is the bonafide work of **“ . . . ALEENA JOSEPH, CHINJU SHENUS, HRIDYA P V, ROSHAN K ANTONY . . . ”**, who carried out the project work under our supervision.

Prof. Muralee Krishnan C	Ms.Remya Ravindran
HOD	**PROJECT GUIDE**
Dept. of CSE	Lecturer
	Dept. of CSE

CONTENTS

ACKNOWLEDGEMENT

We take this opportunity to express our heartfelt gratitude to all respected personalities who had guided, inspired and helped us in the successful completion of this project.

First and foremost, we express our thanks to **The Lord Almighty** for guiding us in this endeavour and making it a success.

We are thankful to our Principal **Dr. U Lazar John** and the Management for providing us with excellent lab and infrastructure facilities.

Our sincere thanks to the Head of the Department of Computer Science & Engineering, Prof. **Muralee Krishnan C** for his valuable guidance and suggestions.

We would like to express our deepest gratitude to **Mr. Remya Ravindran** for his valuable contributions and guidance.

Last but not least, we thank all our teaching and non teaching staffs of Department of Computer Science & Engineering, and also our friends for their immense support and help in all the stages for the development of the project.

ABSTRACT

This thesis develops an algorithm for automatic number plate recognition (ANPR) and implemented it on the Beagle Board. It is an image processing technology which uses number plate to identify the vehicle. The objective is to design an efficient automatic authorized vehicle identification system by using the vehicle number plate. It can be used to implement on the entrance for security control of a highly restricted area like military zones or area around top government offices e.g. Parliament, Supreme Court etc. The algorithm developed here is aimed to be lightweight so that it can be run in real time. With images provided by a USB web camera the system will be able to recognize the number plate under normal condition.

The algorithm is built in three sections; the first is the initial detection of a number plate using edge and intensity features in the image; in the second, the text of the number plate is found; last is the actual character recognition. Optical character recognition technique is used for the character recognition. The resulting data is then used to compare with the records on a database. The major advantages of the system is its real-time capability and that it does not require any additional sensor input (e.g. from infrared sensors) except a video stream. The system is implemented and simulated in Matlab, and its performance is tested on real image. It is observed that the developed system successfully detects and recognize the vehicle number plate on real images.

List of Figures

List of Tables

List of Abbreviations

ANRP	: $AutomaticNumberPlateRecognition$
LPR	: $LicensePlateRecognition$
OCR	: $OpticalCharacterRecognition$

(1)

CHAPTER 1
Introduction

1.1 Overview

Deploying smart cameras for the purpose of video based traffic surveillance has the advantage of allowing direct on site image processing tasks. The information is extracted from the input data and sent in compressed form to a central node, hence decreasing demands on both communication and computation infrastructure. Furthermore, embedded systems are cheaper than general purpose computers and suitable for deployment in harsh enviroThe Automatic Number Plate Recognition (ANPR) was invented in 1976 at the Police Scientific Development Branch in the UK. There is a need for intelligent traffic management systems in order to cope with the constantly increasing traffic on todays roads. Video based traffic surveillance is an important part of such installations. Information about current situations can be automatically extracted by image processing algorithms. Beside vehicle detection and tracking, identification via number plate recognition is important for a variety of applications. These include, e.g. automatic congestion charge systems, access control, tracing of stolen cars, or identification of dangerous drivers.

It is simply the ability to automatically extract and recognition a vehicle number plates characters from an image. In essence it consists of a camera or frame grabber that has the capability to grab an image, find the location of the number in the image and then extract the characters for character recognition tool to translate the pixels into numerically readable character. Furthermore, embedded systems are cheaper than general purpose computers and suitable for deployment in harsh environments due to their physical robustness. Number plate recognition systems in operation today use special hardware like high resolution cameras or infrared sensors to improve the input quality and they operate in controlled settings. A different solution is the continuous analysis and consideration of subsequent frames. The overall system is divided into hardware model and software model. The hardware model consists of a camera to capture the image, a Beagle Board on which the algorithm is executed and for controlling the complete hardware of the system. As the vehicle enters, the camera connected to the Beagle Board through USB port captures the image of the vehicle. The algorithm receives the image and performs the processing, which yields the vehicle number. This number is then compared to the authorized number to confirm it validity and provides signal to control the system hardware. If the inputted plate contains an authorized number then the barrier on the entrance will be raised up using motor, green indication light will be switched on and Access Granted will appears on the display, and if the inputted plate contains an unauthorized number then barrier will not be raised, red indication will be switched on and Access denied will appear on the display.

1.2 Motivation and Technical Relevance

The major advantages of our system over all others are its real-time capabilities in city scenarios and its ability to operate under daytime conditions with sufficient daylight or artificial light from street lamps. The usage of active infrared light is popular because the light is reflected by the license plates only. By using a camera and special filters, the detection of the plates and subsequent character segmentation is relatively easy. However, the usage of alternative light sources comes at additional costs. Thus, one prerequisite during system design was to build a system which operates with conventional off-theshelf video cameras and without additional lighting, in favor of the feasibility to deploy and setup the system within a short time. The major reason why we have focused on cars driving at residential speeds is the lack of a digital camera with a high speed shutter. The drawback of using standard TV cameras is that motion blur is a critical issue, so that reasonable character recognition becomes almost impossible when vehicles are moving at high speeds. Note that this is rather a restriction of the image acquisition device than a deficit of the system. Our system runs fully autonomous and embedded on a smart camera, and thus makes integration into an existing system, e.g. for access control of parking garages, possible

1.3 Member roles and responsibilities

Our team consist of four members. The roles are assigned as follows:- Eventhough

1.1: Team Organization

Name	**Role/Responsibility**
Roshan K Antony	Leader and detection
Chinju Shenus	Detection
Hridya P V	Recognition
Aleena Joseph	Recognitiom

we have assigned specific roles we will be working together during each phase.This would provide each of us with a complete knowledge of the project.

1.4 Layout

If possible present an outline of the contents for the chapters to follow. Start with a brief premise and follow up with chapter-wise outlines as below. Thanks to the hyperref package every citation and reference to labels is hyperlinked and indexed in the final pdf generated. Click the chapter numbers below to have a go at it.

Chapter 2 presents the relevant documents referenced during the initial survey of the project concept. Include algorithms, concepts or theoretical content that have appropriate relevance to the project here. Don't forget to cite each reference made.

Chapter 3 presents the relevant documents referenced during the initial survey of the project concept. Include algorithms, concepts or theoretical content that have appropriate relevance to the project here. Don't forget to cite each reference made.

Chapter 4 includes the hardware and software requirements for the project. You could exploit the graphical properties of the pstricks package here.

Chapter 5 gives an overview of the schedule of the Interim/Term project work. Include member work effort and module allocations to each member here as per his/her responsibility.The section also presents the general architecture of your project concept. Various layers and players involved and the abstraction defined at each level or layer is to be mentioned. If possible, present an abstract diagram an explain each feature or subcomponents in detail. Towards the end; design models, flow diagrams, control flow diagrams or usecase models that comprise the design elements of the project are to be included.

Chapter 6 includes the algorithms or program code elements for the initial model (working implementation) of the project.

Chapter 7 includes the details of the unit tests, integration tests and proposals for future maintenance. You may optionally include the details for modularization of your code, whereby the process of maintenance is rendered to be convenient.

The last chapter, Chapter 8 summarizes the work done in this semester (Interim or Final). Give a concise description of the challenges faced and possible improvements or alternative schemes to implement similar projects.

CHAPTER 2
Literature Survey

2.1 Documentation

Generally, license plate recognition consists of two separate parts. First, plates have to be located within the image, termed license plate detection followed by license plate character recognition, determining the plate number.

2.1.1 Related works

Different methods for the detection of license plates can be applied. Shapiro et al. use a mixture of edge detection and vertical pixel projection for their detection module. In the work of Jia et al. color images were segmented by the MeanShift algorithm into candidate regions and subsequently classified as plate or not. The AdaBoost algorithm was used by Dlagnekov and Belongie for license plate detection on rather low resolution video frames. Similar features like those introduced in were added to the classical set of Haar features, and the located plate was tracked over a sequence of frames. Matas and Zimmermann [13] proposed a different approach for the localization of license plates. Instead of using properties of the plate directly, the algorithm tries to find all character-like regions in the image. This is achieved by using a region-based approach. Regions are enumerated and classified due to a region descriptor using a neural network. If a linear combination of character-like regions is found, the presence of a whole license plate is assumed.

The method described above can be applied to the segmentation- task of character recognition as well. Shapiro et al.use adaptive iterative thresholding and analysis of connected components for segmentation. The classification task is performed with two sets of templates. Rahman et al. used horizontal and vertical intensity projection for segmentation and template matching for classification. Dlagnekov and Belongie [6] use the normalized cross correlation for classification by searching the whole plate, hence skipping segmentation

Those systems described above have not been specifically designed for embedded systems. A lot of commercial mobile systems exist, most of them equipped with special hardware to improve the quality of the input image. Kamat and Ganesan implemented a license plate detection system on a DSP using the Hough transform. Kang et al. implemented a vehicle tracking and license plate recognition system on a PDA. A FPGA was used by Bellas et al. to speed-up parts of their license plate recognition system.

A comparison of different LP detection and recognition systems is difficult as each one is subject to differing prerequisites. Furthermore, the lack of a common evaluation database makes evaluations and comparisons unfair. Our focus was on the architecture and implementation of a complete LPR system on our embedded platform. Thus, we have omitted a detailed discussion of algorithm complexity, power efficiency and accuracy. We have only cited those systems which are somehow related to mobile devices or embedded architectures, or which have recently introduced new techniques to the area of LPR.

2.1.2 Papers

IEEE Papers;

Real-Time License Plate Recognition on an Embedded DSP-Platform Clemens Arth, Florian Limberger, Horst Bischof Graz University of Technology Institute for Computer Graphics and Vision Inffeldgasse 16/2, 8010 Graz, Austria

License Plate Detection Using AdaBoost Louka Dlagnekov Department of Computer Science and Engineering UC San Diego La Jolla, CA 92093-0114

Automatic Number plate Recognition Muhammad Tahir Qadri Department of Electronic Engineering Sir Syed University of Engineering and Technology Karachi,Pakistan Muhammad Asif Department of Electronic Engineering Sir Syed University of Engineering and Technology,Karachi,Pakistan

2.1.3 Advantages of the proposed system

Easy to understand and use.

Platform independent.

Faster Version.

Light size.

Portable.

Algorithm 1 Optical Character Recognition Algorithm

```
1: Create an image model library that contains all the letters A-Z and numbers 0-9.
2: Create templates
a: Read the letters and numbers (characters) from the library
b: Break the image matrix of characters up into a cell array of matrices 42x 24
c: Save as templates
```

3: Read the image(only the number plate portion)
4: Convert to gray scale
5: Convert the gray image to binary format
6: Remove all objects containing fewer than 30 pixels
7: Open a text file to write
8: Compute the number of letters in the template file
a: separate lines in the text(column and raw segmentation)
b: extract letter
c: resize the letter(same size of template)
d: computes the correlation between templates and the letter
e: convert image to text
f: concatenate the letters one by one
9: Open the text file to read

Algorithm 1 Detection Algorithm
1: Read image
2: . Enhance the yellow region to white and others to black ¿convert image from rgb to hsv ¿if Hue ¡0.2, Saturation ¿ 0.4 and Value ¿ 0.6 then it is likely that that region is yellow ¿so make this region to white and all others to black
3: Process this black and white image for large areas with white colour ¿ perform morphological filling to remove small hole like regions ¿ search for large connected regions with minimum 20 x 20 pixels (this can be changed) ¿ separate such regions and store in a structure ¿ Individually process these regions ¿¿ find out the locations in the region with alternate whites and blacks. This can be found by counting number of white pixels in each column ¿¿ identify the points with maximum white pixels, these are the separation between the characters ¿¿ take the regions between two such points as one character. All such characters are stored to a structure
4:Repeat the above steps for white co lour also ¿after converting to hsv format if Hue¡0.2 saturation¡0.1 and value¿0.8 is considered as white colour.

CHAPTER 3

Proposed System

3.1 Overview of the proposed system

Figure below shows the block diagram of the system. The camera captures the image and gives the image to the Beagle Board. Beagle Board is the main part of the system on which the algorithm is executed. For executing the algorithm, the operating system is loaded into the beagle board. Then the camera, keyboard, mouse and display is interfaced into the system. The algorithm is first developed in the Matlab and simulated. Then convert that algorithm to run on the Beagle board. The algorithm takes the image from the camera, cut the number plate portion from the image (detection), and from the number plate characters are recognized. Then the characters are compared with the numbers in the database (authorized numbers). If the number is present on the database Access Granted is displayed on the DVI monitor. Otherwise Access denied will display. There is also an option to add and delete the numbers from the database whenever needed.

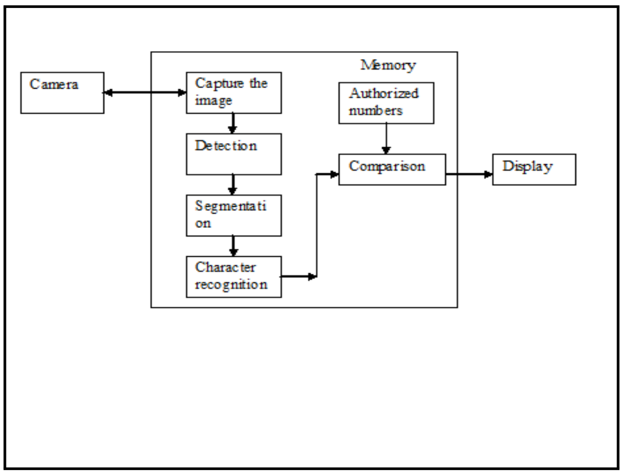

Fig. 3.1: Overall View

3.2 Steps involved in the System with an example

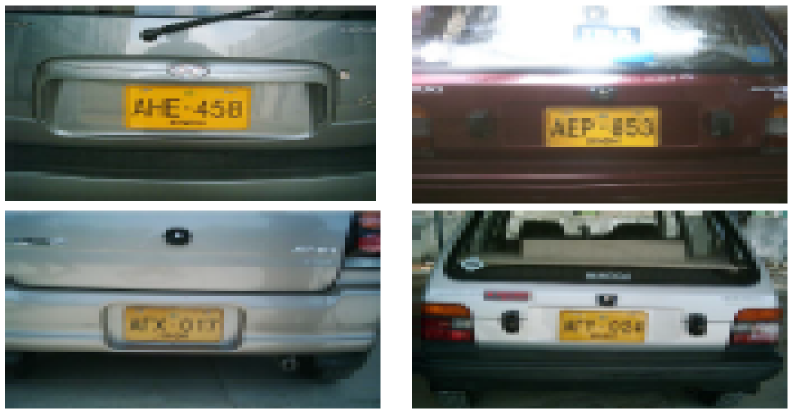

Figure 3 Images taken using USB camera

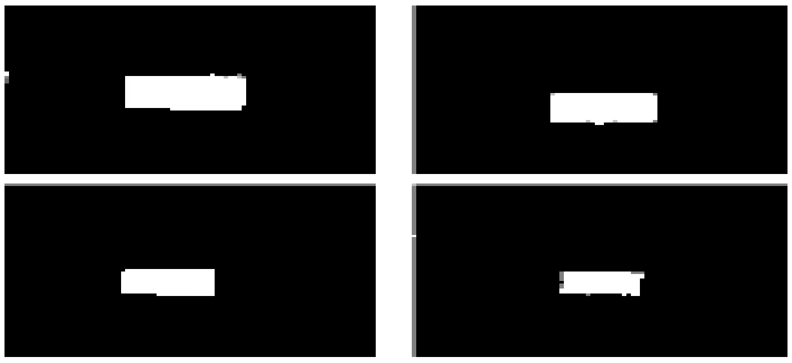

Figure 4 ROI detection using yellow search algorithm

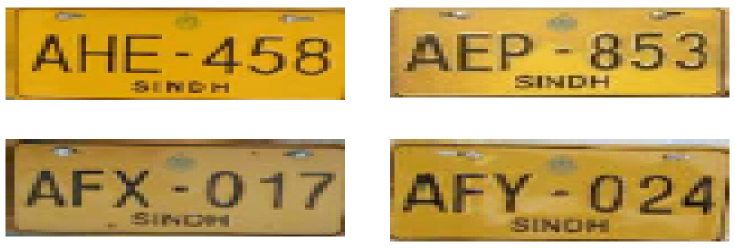

Figure 5 Vehicle number plate extraction using smearing algorithm

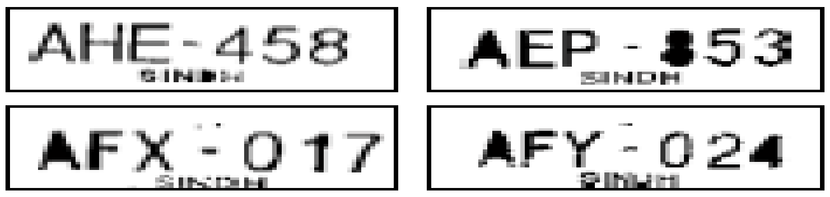

Figure 6 Binary image

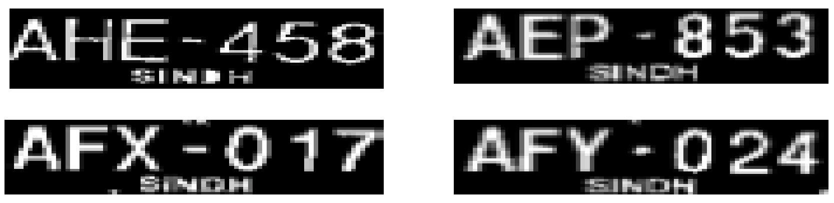

Figure 7 Inverted binary image

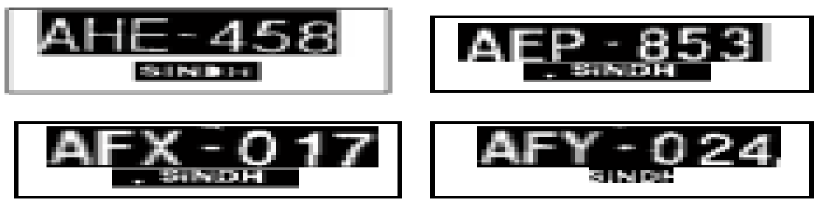

Figure 8 Line separation using row segmentation

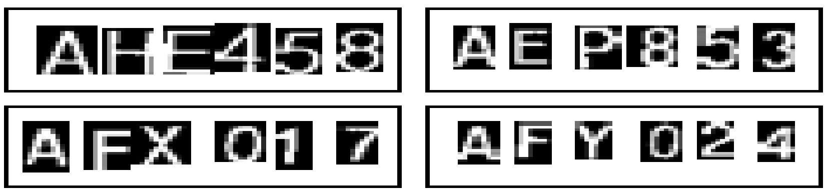

Figure 9 Character separation using column segmentation

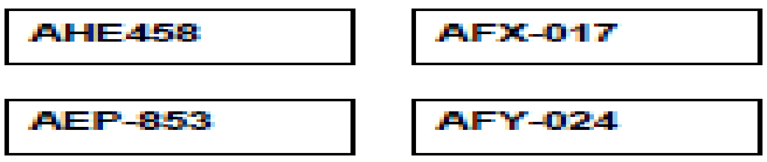

Figure 10 Recognize character using OCR

CHAPTER 4
System Requirements Specification

4.1 Software Requirements

1. Platform: Ubuntu
2. Language: Embedded C
3. OpenCV library

4.2 Hardware Requirements

1. Beagleboard
2. Memory
3. power:Beagle Board
4. RS232 Cable
5. Camera
6. DVI-D monitors

4.3 Beagleboard

The Beagle Board is a low-cost, fan-less single-board computer based on Texas Instruments' OMAP device family, with all of the expandability of today's desktop machines, but without the bulk, expense, or noise.

The Beagle Board is an OMAP3530 platform designed specifically to address the Open Source Community. It has been equipped with a minimum set of features to allow the user to experience the power of the OMAP3530 and is not intended as a full development platform as many of the features and interfaces supplied by the OMAP3530 are not accessible from the Beagle Board. By utilizing standard interfaces, the Beagle Board is highly extensible to add many features and interfaces. It is not intended for use in end products. All of the design information is freely available and can be used as the basis for a product.

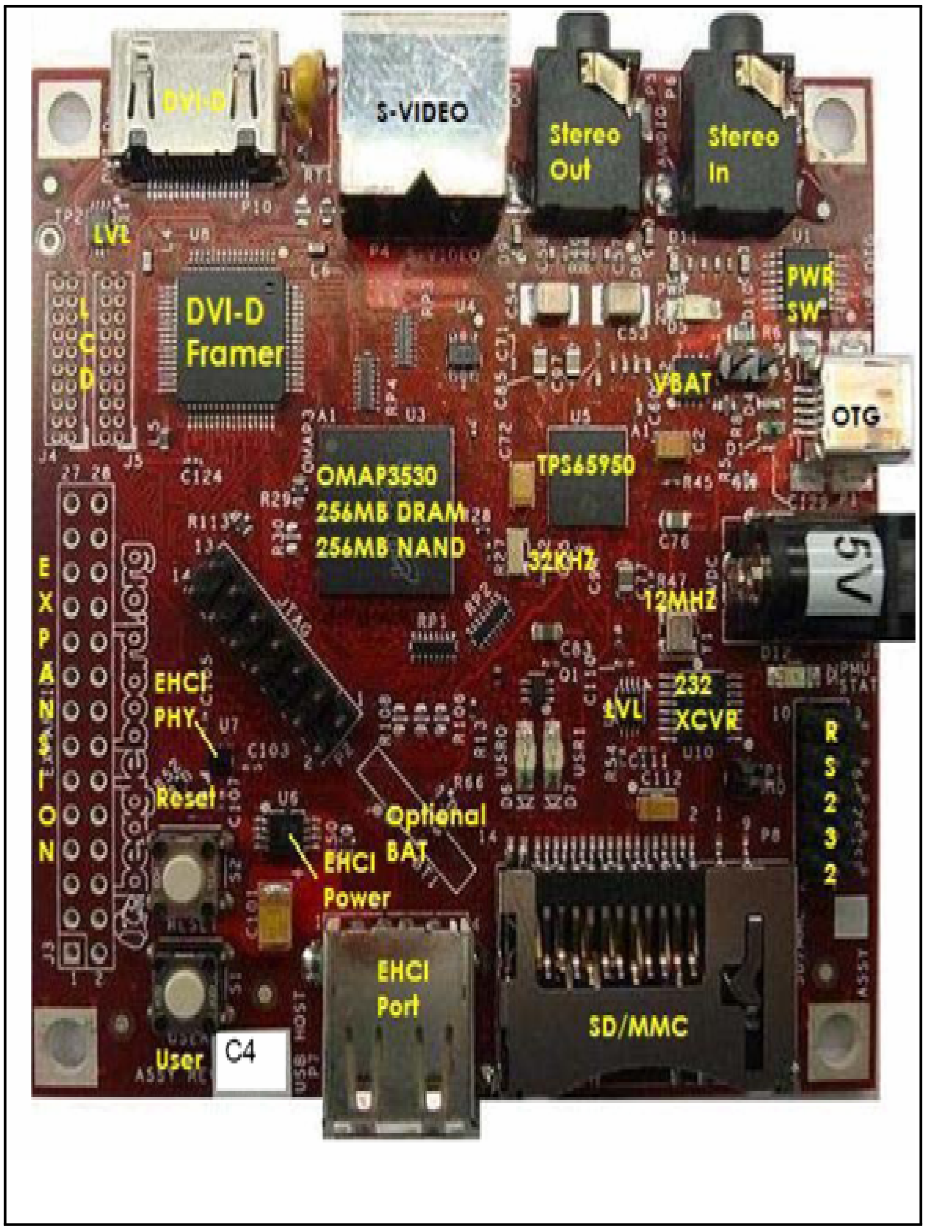

Fig. 4.1: Overall View of Beagleboard

Table provides a list of the Beagle Boards features.

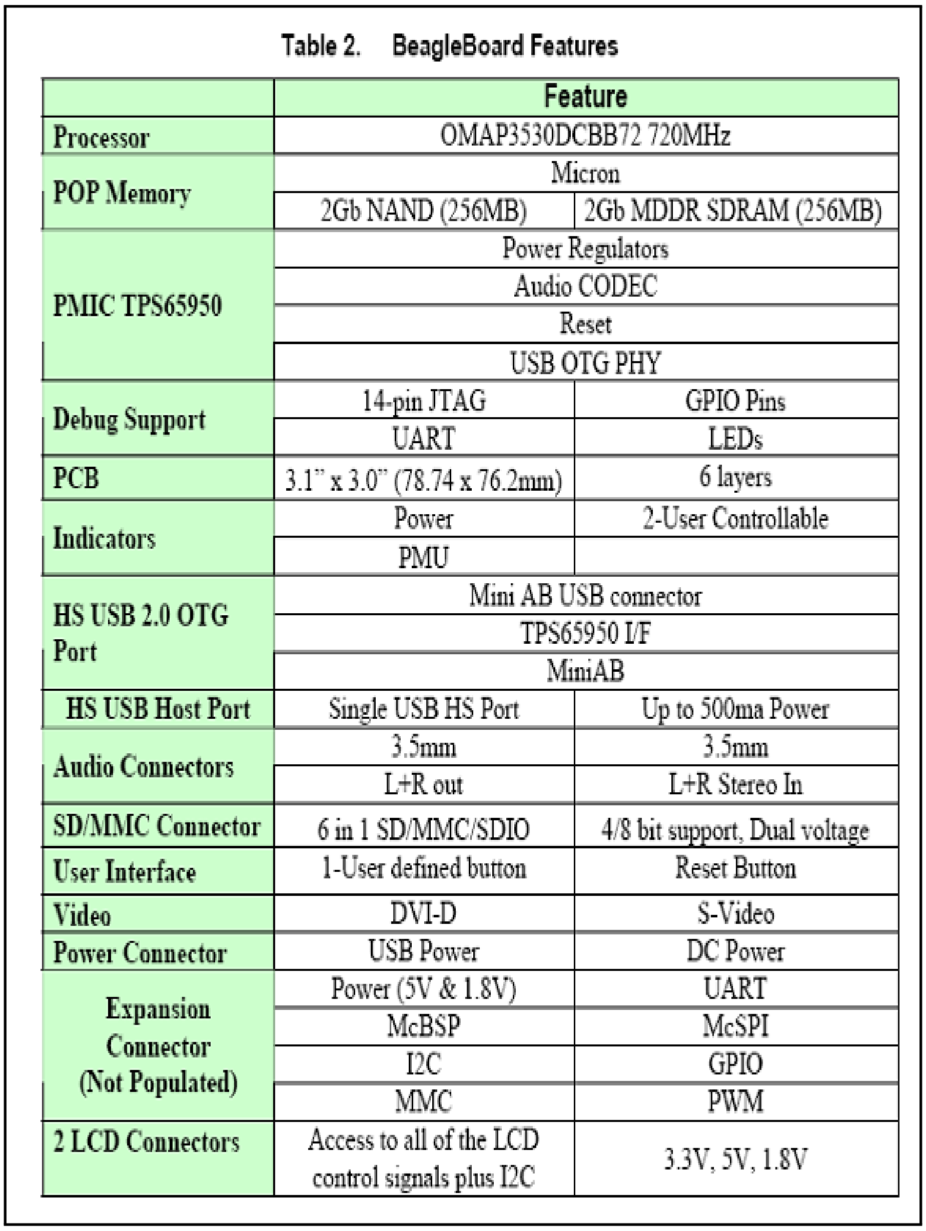

Table 2. BeagleBoard Features

	Feature	
Processor	OMAP3530DCBB72 720MHz	
POP Memory	Micron	
	2Gb NAND (256MB)	2Gb MDDR SDRAM (256MB)
PMIC TPS65950	Power Regulators	
	Audio CODEC	
	Reset	
	USB OTG PHY	
Debug Support	14-pin JTAG	GPIO Pins
	UART	LEDs
PCB	3.1" x 3.0" (78.74 x 76.2mm)	6 layers
Indicators	Power	2-User Controllable
	PMU	
HS USB 2.0 OTG Port	Mini AB USB connector	
	TPS65950 I/F	
	MiniAB	
HS USB Host Port	Single USB HS Port	Up to 500ma Power
Audio Connectors	3.5mm	3.5mm
	L+R out	L+R Stereo In
SD/MMC Connector	6 in 1 SD/MMC/SDIO	4/8 bit support, Dual voltage
User Interface	1-User defined button	Reset Button
Video	DVI-D	S-Video
Power Connector	USB Power	DC Power
Expansion Connector (Not Populated)	Power (5V & 1.8V)	UART
	McBSP	McSPI
	I2C	GPIO
	MMC	PWM
2 LCD Connectors	Access to all of the LCD control signals plus I2C	3.3V, 5V, 1.8V

Fig. 4.2: Specification of beagleboard

OMAP Processor

The Beagle Board uses the OMAP3530DCBB72 720MHZ version and comes in a .4mm pitch POP package. POP (Package on Package) is a technique where the memory, NAND and SDRAM, are mounted on top of the OMAP3530. For this reason, when looking at the Beagle Board, you will not find an actual part labeled OMAP3530.

Memory

The Micron POP memory is used on the Rev C4 Beagle Board and is mounted on top of the processor as mentioned. The key function of the POP memory is to provide: 2Gb NAND x 16 (256MB) , 2Gb MDDR SDRAM x32 (256MB @ 166MHz), No other memory devices are on the Beagle Board. It is possible however, that additional memory can be added to Beagle Board by: Installing a SD or MMC in the SD/MMC slot Use the USB OTG port and a powered USB hub to drive a USB Thumb drive or hard drive. Install a thumb drive into the EHCI USB por Support for this is dependent upon driver support in the OS.

Power Management

The TPS65950 is used on the Rev C4 to provide power to the Beagle Board with the exception of the 3.3V regulator which is used to provide power to the DVI-D encoder and RS232 driver. In addition to the power the TPS65950 also provides: Stereo Audio Out, Stereo Audio in, Power on reset, USB OTG PHY, Status LED

DVI-D Connector

The Beagle Board can drive a LCD panel equipped with a DVI-D digital input. This is the standard LCD panel interface of the OMAP3530 and will support 24b color output. DDC2B (Display Data Channel) or EDID (Enhanced Display ID) support over I2C is provided in order to allow for the identification of the LCD monitor type and the required settings. The Beagle Board is equipped with a DVI-D interface that uses an HDMI connector that was selected for its small size. It does not support the full HDMI interface and is used to provide the DVI-D interface portion only. The user must use a HDMI to DVI-D cable or adapter to connect to a LCD monitor. This cable or adapter is not provided with the Beagle Board. A standard HDMI cable can be used when connecting to a monitor with an HDMI connector.

Do not plug in the DVI-D connector to display with the Beagle Board powered on. Plug in the cable to the display and then power on the Beagle Board

SD/MMC 6 in 1 Connector

A 6 in 1 SD/MMC connector is provided as a means for expansion and can support such devices as: WiFi Cards, Camera, Bluetooth Cards, GPS Modules, SD Memory Cards, MMC Memory Cards, SDIO Cards, MMC Mobile cards, RS-MMC Cards, mini SD Cards

It supports the MMC4.0 (MMC+) standard and can boot from MMC or SD cards. It will support both 4 and 8 bit cards. It will also support most SDHC cards as well.

Reset Button

When pressed and released, causes a power on reset of the Beagle Board.

User/Boot Button

A button is provided on the Beagle Board to provide two functions: Force a change in the boot sequence of the OMAP3530. Used as an application button that can be used by SW as needed.

When used in conjunction with the RESET button, it will force a change to the order in which boot sources are checked as viable boot sources. If the button is pressed while the RESET button is released, the sequence becomes: USB, UART, MMC1, NAND

Even though the NAND may have a program in it, if a card is placed in the MMC slot, it will try to boot from it first. If it is not there, it will boot from NAND. There is also the option to have a serial download application that will program the NAND if connected to the serial or USB ports. In this scenario the internal ROM will stop on either the serial or USB port and start the download process from there. It does require an application to be run on the host PC in order to perform this function. If the user button is not pressed at reset, the sequence in which the internal ROM looks for viable boot sources is as follows: NAND, USB, UART3, MMC1

In this case, NAND overrides every option and will always boot from NAND if there is data in the NAND. If the NAND is empty, then the other sources are available to be used based on the boot order. To force a boot from the SD/MMC card, the reset button must be pushed and the reset button pushed and released.

Indicators

There are three green LEDs on the Beagle Board that can be controlled by the user. o One on the TPS65950 that is programmed via the I2C interface o Two on the OMAP3530 Processor controlled via GPIO pins

There is a fourth LED on the Beagle Board that provides an indication that power is supplied to the board and is not controlled via software.

RS232 Header

Support for RS232 via UART3 is provided by a 10 pin header on the Beagle Board for access to an onboard RS232 transceiver. It does require an IDC to DB9 flat cable, which is not provided, to access the serial port.

Software on the Beagle Board

The board ships with U-Boot and X-Loader flashed onto the Beagle Board.

Connecting USB Host

The Rev C4 Beagle is equipped with a USB Host only connector and can be used to support USB based devices. In order to connect multiple devices a Hub is required. The hub can be powered or un-powered if the total current on the devices connected to the hub do not exceed the available power from the DC power supply that is used. If the board is powered from the OTG connector, then the power available from this port is extremely limited and will not be able to provide sufficient power to run most USB devices. It may be possible to run a USB keyboard or mouse, but that is about all it will have the power to supply. The USB Host port is HS only and does not support LS or FS devices without a hub.

Connecting Optional Power

An optional DC supply can be used to power the Beagle Board by plugging it into the power jack of the Beagle Board. The power supply is not provided with the Beagle Board, but can be obtained from various sources. You need to make sure the supply is a regulated 5V supply.

Connecting Serial Cable

In order to access the serial port of the Beagle Board a flat cable is required to connect to a PC. The adapter will not plug directly into the PC and will require an external Female to Female twisted cable (Null Modem) in order to connect it to the PC. The ribbon cable is not supplied with the Beagle Board but can be obtained from numerous sources.

Connecting DVI-D Cable

In order to connect the DVI-D output to a monitor, a HDMI to DVI-D cable is required.

Button Locations

There are two buttons on the BeagleBoard; the RESET button when pressed will force a board reset and the USER button which can be used by the SW for user interaction. If the user holds the USER button down while pressing and releasing the RESET button, the BeagleBoard will enter the ROM boot loader mode allowing it to boot from other sources than the onboard NAND.

SD/MMC Connection

The SD/MMC connector can be used for Memory or SDIO type cards. This is a full size connector and will support various cards. Whether a particular card is supported or not, is dependent on the available software drivers.

DVI-D monitors

There are many monitors that can be used with the BeagleBoard. With the integrated EDID feature, timing data is collected from the monitor to enable the SW to adjust its timings. Table below shows a short list of the monitors that have been tested to date on the BeagleBoard at the 1024x768 resolution.

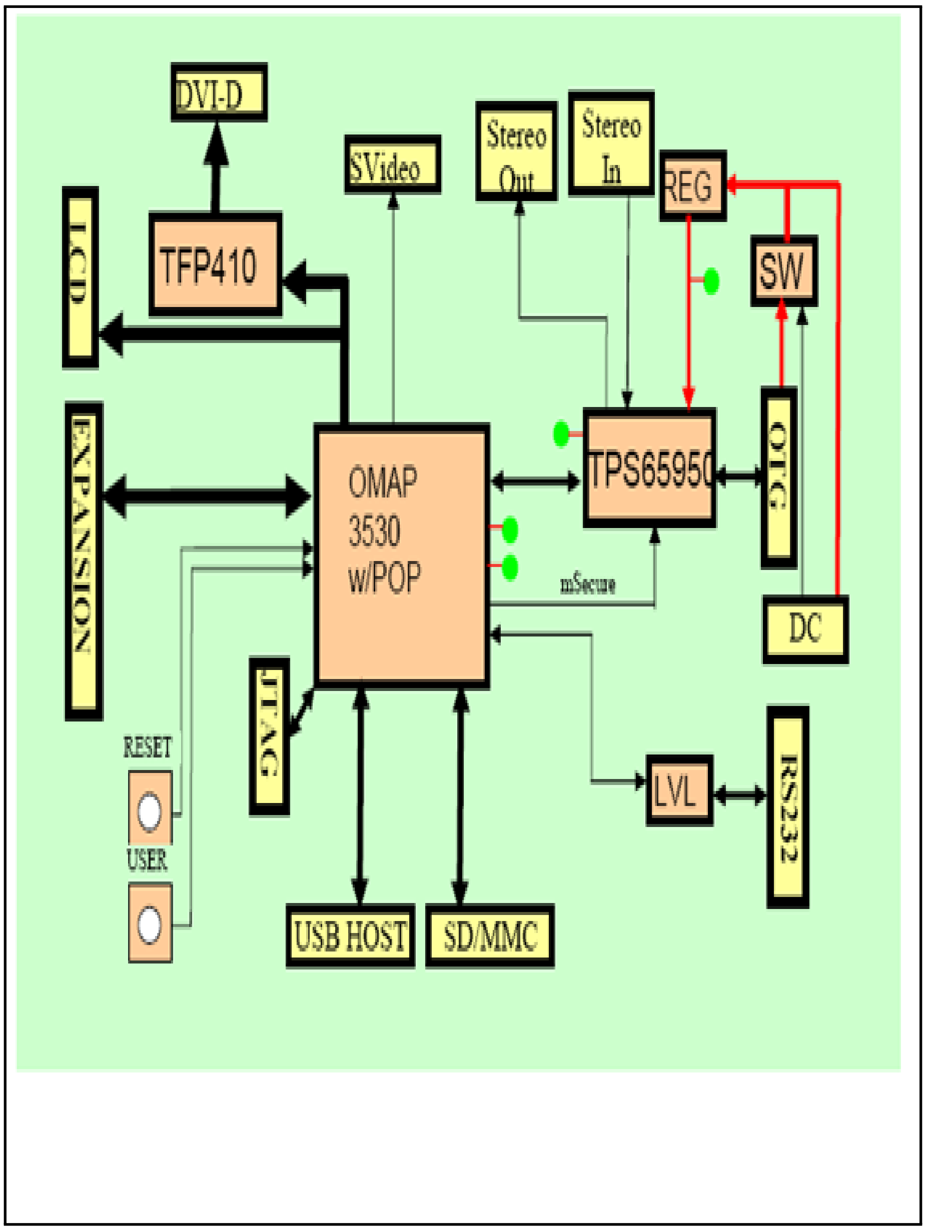

Fig. 4.3: Design of Beagleboard

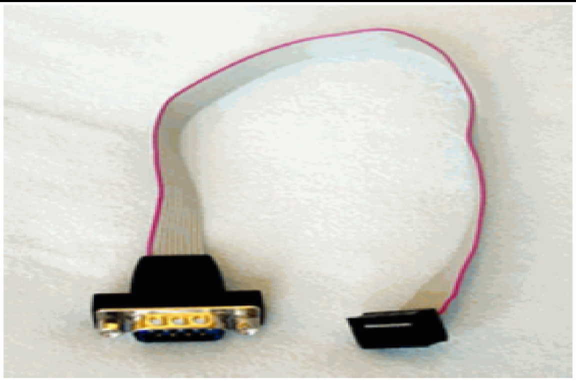

Figure 43. RS232 Cable

Fig. 4.4: RS232 cable

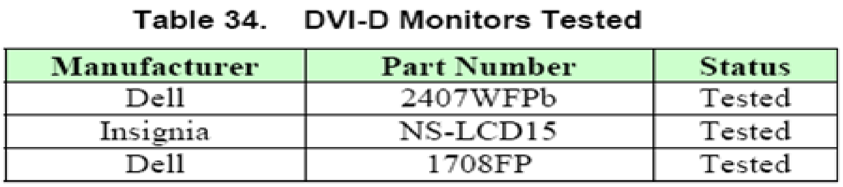

Table 34. DVI-D Monitors Tested

Manufacturer	Part Number	Status
Dell	2407WFPb	Tested
Insignia	NS-LCD15	Tested
Dell	1708FP	Tested

Fig. 4.5: tested monitors

Open Source Computer Vision Library (OpenCV)

The OpenCV Library is mainly aimed at real time computer vision. Some example areas would be Human-Computer Interaction (HCI); Object Identification, Segmentation, and Recognition; Face Recognition; Gesture Recognition; Motion Tracking, Ego Motion, and Motion Understanding; Structure From Motion (SFM); and Mobile Robotics.

Features: Image data manipulation (allocation, release, copying, setting, conversion). Image and video I/O (file and camera based input, image/video file output). Matrix and vector manipulation and linear algebra routines (products, solvers, eigenvalues, SVD). Various dynamic data structures (lists, queues, sets, trees, graphs). Basic image processing (filtering, edge detection, corner detection, sampling and interpolation, color conversion, morphological operations, histograms, image pyramids). Structural analysis (connected components, contour processing, distance transform, various moments, template matching, Hough transform, polygonal approximation, line fitting, ellipse fitting, Delaunay triangulation). Camera calibration (finding and tracking calibration patterns, calibration, fundamental matrix estimation, homography estimation, stereo correspondence). Motion analysis (optical flow, motion segmentation, tracking). Object recognition (eigen-methods, HMM). Basic GUI (display image/video, keyboard and mouse handling, scroll-bars). Image labeling (line, conic, polygon, text drawing)

CHAPTER 5
Design & Analysis

5.1 System Analysis

The system consists of two modules: License Plate Detection and License Plate Character Recognition.

5.1.1 Module breakup

5.1: Module Description

Module	**Description**
Module1: License plate Detection.	detecting,tracker and post processing
Module2: License plate Character Recognition.	segmentation,classification,post processing

The detection module consists of three parts: the detecting, the tracking and the post-processing step.Detector:The license plate detector is based upon the framework proposed by Viola and Jones . Viola and Jones created a fast object detection system by combining the AdaBoost-algorithm with Haar-like features, a classifier cascade and the integral image for fast feature computation. Unlike the original framework, the RealBoost is used for this detector, which provides confidence rated predictions instead of binary class-labels. Furthermore, the classifier cascade is improved by using inter stage feature propagation as proposed by ?Sochman and Matas . The feature-pool is extended with features based on edge orientation histograms as proposed by Levi and Weiss . Post-processing: The exhaustive search technique of the Viola and Jones detector leads to multiple detections for a single license plate. Therefore, post-processing methods have to be applied in order to merge those detections. For this purpose the rather simple non-maximum suppression is used. The non-maximum suppression merges multiple detections, considering the confidence value of each detection. In our case all overlapping detections are substituted by the single detection with the maximum confidence value. If the false positive rate is not too high, this method achieves sufficient results. Nonetheless, the final detection does not solely contain the license plate but parts of the surroundings as well. Tracker: A tracker is integrated into the system in order to limit the search of the detector to certain areas of the input image. This requires a rough calibration of the scene. Figure 2 illustrates a possible setting. Objects are assumed to enter from the top. A static region of interest is defined where the Viola and Jones detector search is performed on each frame. For each new detection a tracker is initialized, which predicts the new objects position on the next frame. This allows for scanning solely within a small area around the estimated position. The update and prediction operations of the tracker require less computation time than a detector scan for the whole image. Therefore, the integration is indeed useful. Furthermore, the tracking information will be important

License Plate Character Recognition Module: License plate character recognition consists of three separate parts .First, characters are isolated with a region-based approach, followed by character classification using a support vector machine (SVM). Finally, during post-processing, a priori knowledge about the expected license plates and information provided by the tracker is used to improve the classification result. Segmentation: A region-based approach is used for character segmentation. Due to dark characters on white background, region-growing seeded with lowest intensity values is applied. In order to determine which regions contain a character two approaches are used. Similar to [13] a region descriptor is computed incrementally, consisting of compactness, entropy of the grayscale histogram, and central invariant statistical moments. Descriptors are classified using support vector classification.nowledge about the expected license plates and information provided by the tracker is used to improve the classification result. Classification :Support vector classification is used for character classification. Segmented regions are scaled to a common patch size. The feature vector consists of direct pixel values. Since character recognition is a multi-class problem a combination of binary classifiers is required. Post-Processing: In this step a priori knowledge about the structure of license plates is exploited i.e. spacing between subsequent characters and furthermore the syntax of the final character string. Needless to mention that this applies only in cases where the nationality of the license plate is known in advance or can be derived beforehand. Additionally, classification results of subsequent frames are combined utilizing the history provided by the tracking module. A simple majority voting individually for each position within the character string is applied. The character achieving the highest count is selected.

5.2 System Design

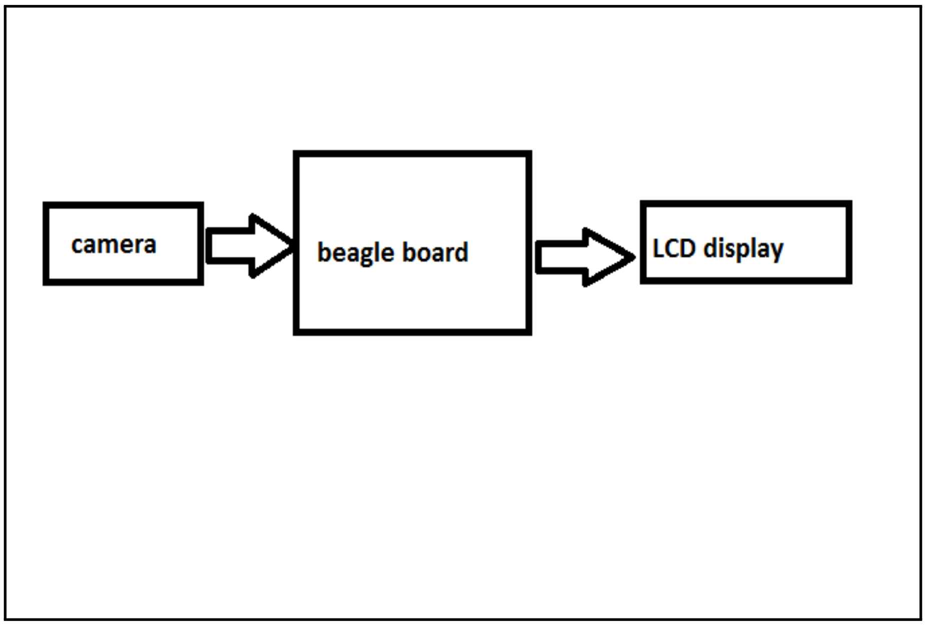

Fig. 5.1: layout

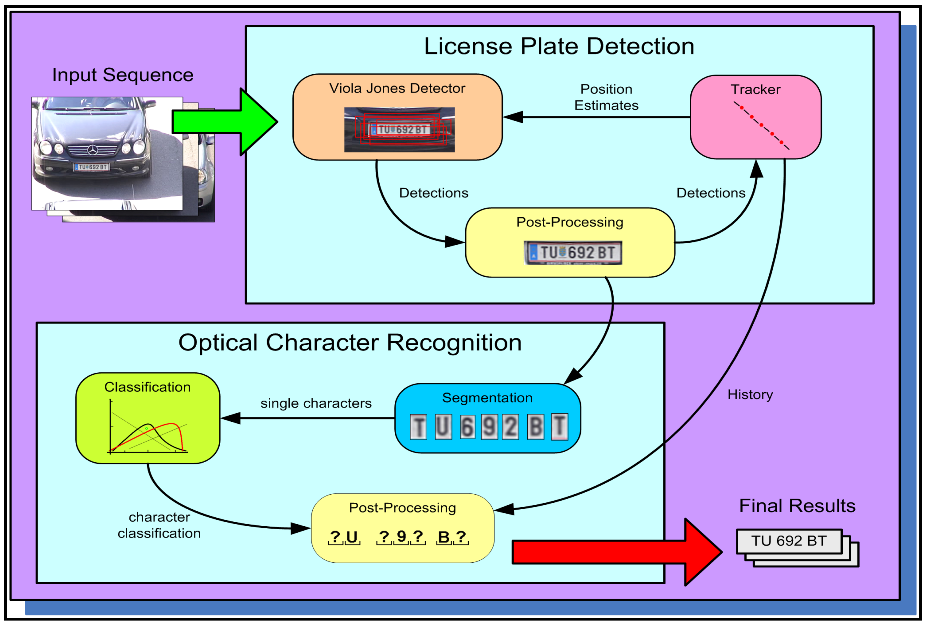

Fig. 5.2: Overall View

5.2.1 Use Case Models / Flow Diagrams

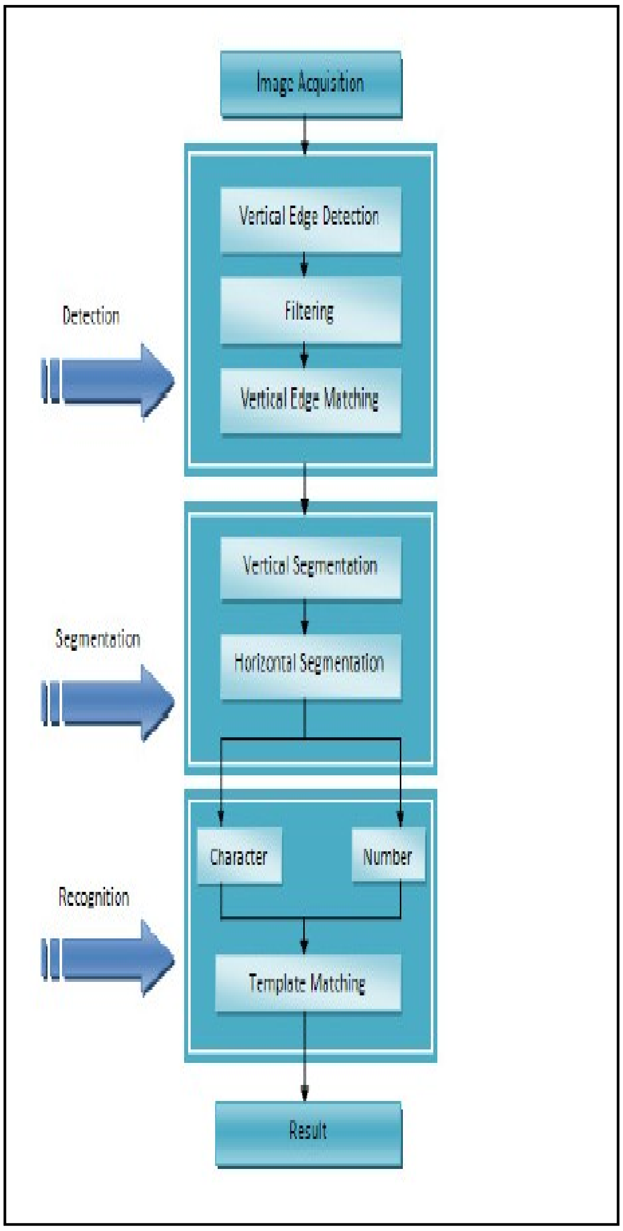

Fig. 5.3: flowchart

CHAPTER 6
Implementation

6.1 Introduction

1. Format the MMC/SD Card for FAT32 File System using the HP USB Disk Storage Format Tool 2.0.6: http://selfdestruct.net/misc/usbboot/SP27213.exe 2. Insert the Card writer/reader into the Windows machine. 3. Insert MMC/SD card into the card reader/writer 4. Open the HP USB Disk Storage Format Tool. 5. Select FAT as File System. Click on Start. 6. After formatting is done Click OK 7. Copy the following files on to MMC in the exact listed. COPY THE MLO FIRST! Make sure you name the file as indicated in the BOLD type. These files can be found at http://code.google.com/p/beagleboard/wiki/BeagleboardRevCValidation MLO as MLO u-boot as u-boot.bin u-boot for flash as u-boot-f.bin ramdisk image as ramdisk.gz Kernel (uImage) as uImage.bin reset.scr as boot.scr x-loader image as x-load.bin.ift Regular script file as normal.scr Setup This step sets up the board for the tests to follow. 1. Make sure Beagle power is in OFF state by removing the 5V supply and the USB host connection. 2. Connect the IDC UART cable the BeagleBoard and using a Null-Modem serial cable connect it to a SERIAL port on a Window/Linux/Mac machine 3. Have terminal program, such as TeraTerm, HyperTerminal, or Minicom, running on the host machine. 4. Configure the terminal program for (BAUD RATE - 115200, DATA - 8 bit, PARITY- none, STOP - 1bit, FLOW CONTROL - none) 5. Insert the MMC/SD card (that is prepared as described above) into MMC/SD slot on Beagle Board. 6. Connect a LCD Monitor to DVI/HDMI port on the Beagle Board.

4. Type root and hit ¡enter¿. You now are in the Linux kernel.

Make sure your terminal settings are correct. o BAUD RATE: 115200 o DATA: 8 bit o PARITY: none o STOP: 1bit o FLOW CONTROL: none (Critical) Make sure that the Flow Control is set to none.

6.2 Pseudo code

```
#include <stdlib.h>
```

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

#define CARS 25 //maximum number of samples expected in humoments tab
int thresh = 50;
int count, imstatus,cam;
IplImage* img = 0;
IplImage* img0 = 0;
IplImage* tpl_Letters = 0;
CvMemStorage* storage = 0;
const char* wndname = "Number plates Detected ";
struct match{
        int num;
        double ccorr;
        int pos;
        int x;
        int y;
        char ch;
        int count;
        } test[20];
char NumPlate[20];
struct cardDB{ // structure to hold the car database
        int id;
        char plate[20];
        }carDB[CARS];

void InitCarDB(void)
{// funciton initializes the carDB database
int n;
strcpy(carDB[0].plate, "KL12AC2134");
carDB[0].id = 101;
strcpy(carDB[1].plate, "KL11AC4111");
carDB[1].id = 102;
strcpy(carDB[2].plate, "KL35A3588");
```

```
carDB[2].id = 103;
strcpy(carDB[3].plate, "KL8H4721");
carDB[3].id = 104;
strcpy(carDB[4].plate, "KL10K3525");
carDB[4].id = 105;
strcpy(carDB[5].plate, "KL11AF3324");
carDB[5].id = 106;

for(n=6;n<CARS;n++)
         {
         strcpy(carDB[n].plate, "KL11AB6865");
         carDB[n].id = n+101;
         }
}
char assignnumber(int x)
{
char ch;
     if(x <   30 ) ch = '1';
else if(x <   80 ) ch = '2';
else if(x <  130 ) ch = '3';
else if(x <  190 ) ch = '4';
else if(x <  240 ) ch = '5';
else if(x <  300 ) ch = '6';
else if(x <  350 ) ch = '7';
else if(x <  400 ) ch = '8';
else if(x <  460 ) ch = '9';
else if(x <  510 ) ch = '0';
else if(x <  570 ) ch = 'A';
else if(x <  625 ) ch = 'B';
else if(x <  690 ) ch = 'C';
else if(x <  745 ) ch = 'D';
else if(x <  800 ) ch = 'E';
else if(x <  850 ) ch = 'F';
else if(x <  905 ) ch = '6';
else if(x <  970 ) ch = 'H';
else if(x < 1000 ) ch = '1';
else if(x < 1035 ) ch = '1';
else if(x < 1090 ) ch = 'K';
else if(x < 1140 ) ch = 'L';
else if(x < 1200 ) ch = 'M';
else if(x < 1265 ) ch = 'N';
else if(x < 1330 ) ch = '0';
else if(x < 1380 ) ch = 'P';
else if(x < 1445 ) ch = 'Q';
else if(x < 1500 ) ch = 'R';
else if(x < 1555 ) ch = '5';
else if(x < 1605 ) ch = 'T';
else if(x < 1660 ) ch = 'U';
else if(x < 1720 ) ch = 'V';
else if(x < 1790 ) ch = 'W';
else if(x < 1850 ) ch = 'X';
else if(x < 1900 ) ch = 'Y';
else if(x < 1950 ) ch = '2';
return ch;
}

int FindNums( const char * fname )
{//function calculates the hu moments of the character images in the
const char *VENTANA="Blob detection";

IplImage* imagen = cvLoadImage(fname,CV_LOAD_IMAGE_GRAYSCALE);
IplImage* imagen_color = cvLoadImage(fname,CV_LOAD_IMAGE_COLOR);
IplImage* smooth;
IplImage* threshold;
IplImage* open_morf;
IplImage* img_contornos;
CvSeq* contour;
CvSeq* contourLow;
int n,m,i,j,k,l;
int sH, sW, ht,wd;
int len1;
double maxval, minval;
CvPoint minloc, maxloc;
IplImage* im_letter;
```

```
IplImage* res=0;
char nums[20];
srand ( time(NULL) );
// convert image to gray
//Create needed images
smooth= cvCreateImage(cvSize(imagen->width , imagen->height), IPL_DEP
threshold= cvCreateImage(cvSize(imagen->width, imagen->height), IPL_D
open_morf= cvCreateImage(cvSize(imagen->width, imagen->height), IPL_D
//Init variables for countours
contour = 0;
contourLow = 0;
//Create storage needed for contour detection
CvMemStorage* storage = cvCreateMemStorage(0);
//Create window
//cvNamedWindow( VENTANA, 0 );
//Smooth image
cvSmooth(imagen, smooth, CV_GAUSSIAN, 7, 0, 0, 0);
CvScalar avg;
CvScalar avgStd;
cvAvgSdv(smooth, &avg, &avgStd, NULL);
//printf("Avg: %f\nStd: %f\n", avg.val[0], avgStd.val[0]);
//threshold image
cvThreshold(smooth, threshold, (int)avg.val[0]-7*(int)(avgStd.val[0]/
//Morfologic filters
cvErode(threshold, open_morf, NULL,1);
cvDilate(open_morf, open_morf, NULL,1);
//Duplicate image for countour
img_contornos=cvCloneImage(open_morf);

//Search countours in preprocesed image
cvFindContours( img_contornos, storage, &contour, sizeof(CvContour),
                CV_RETR_EXTERNAL, CV_CHAIN_APPROX_SIMPLE, cvPoint(0,
//Optimize contours, reduce points
contourLow=cvApproxPoly(contour, sizeof(CvContour), storage,CV_POLY_A

n=0;
ht = tpl_Letters->height; wd=tpl_Letters->width;
```

```
//For each contour found
for( ; contourLow != 0; contourLow = contourLow->h_next )
        {
        CvScalar color = CV_RGB( rand()&200, rand()&200, rand()&200 )
        //Or detect bounding rect of contour
        CvRect rect;
        CvPoint pt1, pt2;
        rect=cvBoundingRect(contourLow, 0);
        //if the letter height is comparable to that of number plate
        if( ( 4 * rect.height > imagen->height) &&
        // also chech for aspect ratio
            ( 10 * rect.width > rect.height)    && (rect.width <
2 * rect.height)  &&
            ( 5 * rect.width <        imagen->width )   )
                {
                //We can draw the contour of object
                cvDrawContours( imagen_color, contourLow, color, colo
                pt1.x = rect.x;
                    pt2.x = (rect.x+rect.width);
                    pt1.y = rect.y;
                    pt2.y = (rect.y+rect.height);
                cvRectangle(imagen_color, pt1,pt2, color, 1, 8, 0);
                //resize the image of character
                sH = 50; sW =(int)(50.0 * rect.width/rect.height);
                im_letter = cvCreateImage(cvSize(sW,sH),IPL_DEPTH_8U,
                cvResetImageROI(open_morf);
                cvSetImageROI(open_morf,rect);
                cvResize(open_morf,im_letter,CV_INTER_NN);
                res = cvCreateImage(cvSize(wd - sW + 1, ht - sH + 1 )
                cvMatchTemplate( tpl_Letters, im_letter, res, CV_TM_C
                cvSmooth(res, res, CV_GAUSSIAN, 3, 0, 0, 0);
                cvMinMaxLoc( res, &minval, &maxval, &minloc, &maxloc,
                if(maxval > 0.3)
                        {
                        test[n].ccorr = maxval; test[n].pos = rect.x;
                        test[n].x = maxloc.x;          test[n].y = ma
                        test[n].ch = assignnumber(maxloc.x);
```

```
                                n+=1;
                                }
                        //Show result.
                        }
                }
if(n > 5)
                {// if there are atleast 7 numbers identified
                for(m=0;m<n;m++)
                        {
                        test[m].num = n;
                        for (k=0;k<n;k++) if(test[m].pos < test[k].pos) test[
                        }
                for(m=0;m<n;m++) { nums[test[m].num-1] = test[m].ch; }//if(is
                if(strcmp(nums,NumPlate))
//                {
   //              len1=strlen(nums);
     //            nums[len1]='\0';
                 strcpy(NumPlate,nums);
        //        }
                }
imstatus = n;
//for(m=0;m<n;m++) printf("Position %d \t cc = %f \t char = %c \n",
//cvShowImage(VENTANA, imagen_color);
//cvWaitKey(0);
//cvDestroyWindow(VENTANA);
cvReleaseImage( &imagen_color );
cvReleaseImage( &res );
cvReleaseImage( &imagen );
cvReleaseImage( &smooth );
cvReleaseImage( &threshold );
cvReleaseImage( &img_contornos );
cvReleaseImage( &open_morf );
return 0;
}

// helper function:
// finds a cosine of angle between vectors
// from pt0->pt1 and from pt0->pt2
double angle( CvPoint* pt1, CvPoint* pt2, CvPoint* pt0 )
{
    double dx1 = pt1->x - pt0->x;
    double dy1 = pt1->y - pt0->y;
    double dx2 = pt2->x - pt0->x;
    double dy2 = pt2->y - pt0->y;
    return (dx1*dx2 + dy1*dy2)/sqrt((dx1*dx1 + dy1*dy1)*(dx2*dx2 + dy
}



int checkblobsize(IplImage* timg, CvArr* blob_pts)
{// check the size of blob to determine whether it is
 // a potential candidate for number plate
CvRect rect;
rect = cvBoundingRect(blob_pts,0);
if ( (rect.width > (int)timg->width/4)  &&
     (rect.height > (int)rect.width/10) &&
     (rect.height < (int)rect.width) )
         return 1;
else
         return 0;
}

/*Take one blob and process it to recognize the numbers */
int BlobProcess(IplImage* imgb, CvPoint* blob_pts)
{
IplImage* blob;
CvRect rect;
blob = cvCloneImage(imgb);
rect = cvBoundingRect(blob_pts,0);
cvSetImageROI(imgb, rect);
cvSaveImage("the_test_blob.jpg",imgb,0);
FindNums("the_test_blob.jpg");// calculates the hu moments of ROI imag
//printf("Blob size x = %d, y = %d, ht= %d, wd= %d \n",rect.x, rect.y
cvReleaseImage( &blob );
return 0;
}

// returns sequence of squares detected on the image.
// the sequence is stored in the specified memory storage
CvSeq* findSquares4( IplImage* img, CvMemStorage* storage )
{
    CvSeq* contours;
    int i, c, l, N = 10;
    CvSize sz = cvSize( img->width & -2, img->height & -2 );
    IplImage* timg = cvCloneImage( img ); // make a copy of input ima
    IplImage* gray = cvCreateImage( sz, 8, 1 );
    IplImage* pyr = cvCreateImage( cvSize(sz.width/2, sz.height/2), 8
    IplImage* tgray;
    CvSeq* result;
    double s, t;
    // create empty sequence that will contain points -
    // 4 points per square (the square's vertices)
    CvSeq* squares = cvCreateSeq( 0, sizeof(CvSeq), sizeof(CvPoint),

    // select the maximum ROI in the image
    // with the width and height divisible by 2
    cvSetImageROI( timg, cvRect( 0, 0, sz.width, sz.height ));

    // down-scale and upscale the image to filter out the noise
    cvPyrDown( timg, pyr, 7 );
    cvPyrUp( pyr, timg, 7 );
    tgray = cvCreateImage( sz, 8, 1 );

    // find squares in every color plane of the image
    for( c = 1; c < 2; c++ )
    {
        // extract the c-th color plane
        cvSetImageCOI( timg, c+1 );
        cvCopy( timg, tgray, 0 );
```

```
// try several threshold levels
for( l = 4; l < N-2; l++ )
{
    cvThreshold( tgray, gray, (l+1)*255/N, 255, CV_THRESH_BIN

    // find contours and store them all as a list
    cvFindContours( gray, storage, &contours, sizeof(CvConto
        CV_RETR_LIST, CV_CHAIN_APPROX_SIMPLE, cvPoint(0,0) );

    // test each contour
    while( contours )
    {
        // approximate contour with accuracy proportional
        // to the contour perimeter
        result = cvApproxPoly( contours, sizeof(CvContour), s
            CV_POLY_APPROX_DP, cvContourPerimeter(contours)*(
        // square contours should have 4 vertices after appro
        // relatively large area (to filter out noisy contou
        // and be convex.
        // Note: absolute value of an area is used because
        // area may be positive or negative - in accordance
        // contour orientation
        if( result->total == 4 &&
            fabs(cvContourArea(result,CV_WHOLE_SEQ,0)) > 1000
            cvCheckContourConvexity(result) &&
            checkblobsize(img, result) )
        {
            s = 0;

            for( i = 0; i < 5; i++ )
            {
                // find minimum angle between joint
                // edges (maximum of cosine)
                if( i >= 2 )
                {
                    t = fabs(angle(
                    (CvPoint*)cvGetSeqElem( result, i ),
                            (CvPoint*)cvGetSeqElem( result, i-2 ),
                            (CvPoint*)cvGetSeqElem( result, i-1 )));
                            s = s > t ? s : t;
                        }
                    }

                    // if cosines of all angles are small
                    // (all angles are ~90 degree) then write quandra
                    // vertices to resultant sequence
                    if( s < 0.15 )
                        {
                        BlobProcess(img, (CvPoint*) result);
                        for( i = 0; i < 4; i++ )
                            cvSeqPush( squares, (CvPoint*)cvGetSeqEle
                        }
                }

                // take the next contour
                contours = contours->h_next;
            }
        }
    }

    // release all the temporary images
    cvReleaseImage( &gray );
    cvReleaseImage( &pyr );
    cvReleaseImage( &tgray );
    cvReleaseImage( &timg );

    return squares;
}


int main(int argc, char *argv[])
{
IplImage* imgm=0;
IplImage* tpl=0;
```

```
CvCapture* camera = cvCreateCameraCapture(-1); // Use the default can
CvSeq*     square_pts;
int height,width,step,channels;
uchar *data;
int m,n,k,l,autherised, id,ml;
float a,b;
FILE* fphu;
imstatus=0; cam = 0;
do          {
        if(argc<2){
                printf("Reading Image from camera... \n\r");
                imgm = cvQueryFrame(camera); //need to capture at lea
                imgm = cvQueryFrame(camera);
                imstatus = 1;
                if (imgm == NULL)
                        {
                        printf("Could not capture image from camera!!
                        imstatus=255; cam = 0;
                        }
                else        {cvSaveImage("cameraframe.jpg",imgm,0); c
                }
        else
                {// load an image file
                imgm=cvLoadImage(argv[1],CV_LOAD_IMAGE_COLOR);
                imstatus=2; cam = 0;
                if(!imgm){
                        printf("Could not load image file: %s \n\r",a
                        imstatus=254;
                        }
                }
        if (imstatus>200) exit(0);
        tpl = cvLoadImage("TemplateNums.jpg",CV_LOAD_IMAGE_GRAYSCALE)
        tpl_Letters = cvCreateImage(cvSize(tpl->width , tpl->height),
        cvThreshold(tpl, tpl_Letters, 100, 255, CV_THRESH_BINARY_INV)
        InitCarDB();
        // cvErode(tpl_Letters, tpl, NULL,1);
        // cvDilate(tpl, tpl_Letters, NULL,1);
        // get the image data
        img0=cvCloneImage(imgm);//
        height = imgm->height;
        width = imgm->width;
        step = imgm->widthStep;
        channels = imgm->nChannels;

        printf("Processing a %d x %d image with %d channels  \n\r",he
        // create a window
        //cvNamedWindow("mainWin", CV_WINDOW_AUTOSIZE);
        //cvMoveWindow("mainWin", 100, 100);

        // create memory storage that will contain all the dynamic da
        storage = cvCreateMemStorage(0);
        img = cvCloneImage( img0 );
        strcpy(NumPlate,"  ");
        // list of corner points of the detected squares
        square_pts = findSquares4( img, storage );
        for(m1=10;m1<(strlen(NumPlate));m1++)
        //len1=strlen(NumPlate);
        NumPlate[m1]='\0';
        if (imstatus > 5)
                {
                printf("\n the number plate is %s \n", NumPlate);
                autherised = 0;
                for(m=0;m<CARS;m++)
                        {
                        if(strcmp(carDB[m].plate ,NumPlate)==0)
                                {
                                autherised=1;
                                id = carDB[m].id;
                                }//
                        }
                if(autherised==1) printf("Access granted... Your ID r
                else printf("Unauthorized vehicle... Access denied...
                }
        else printf("\n No number plates found  \n");
```

```
          cvWaitKey(1000);
          }while(cam);
// release both images
cvReleaseImage( &img );
cvReleaseImage( &img0 );
// clear memory storage - reset free space position
cvClearMemStorage( storage );
//cvDestroyWindow( wndname );
cvReleaseImage( &tpl );
cvReleaseImage( &tpl_Letters );
// release the image
cvReleaseImage(&imgm );
if (imstatus == 1) cvReleaseCapture(&camera);
return 0;
}
```

CHAPTER 7
Testing & Maintenance

7.1 Tests

7.1: Unit test chart

No	**Unit Name**	**Test Status**
1	Detection module	Complete
2	Recgnition Module	Partial

7.2 Maintenance

The implementation works quite well however, there is still room for improvement. The camera used in this project is sensitive to vibration and fast changing targets due to the long shutter time. The system robustness and speed can be increase if high resolution camera is used. The OCR methods used in this project for the recognition is sensitive to misalignment and to different sizes, the affine transformation can be used to improve the OCR recognition from different size and angles. The statistical analysis can also be used to define the probability of detection and recognition of the vehicle number plate.

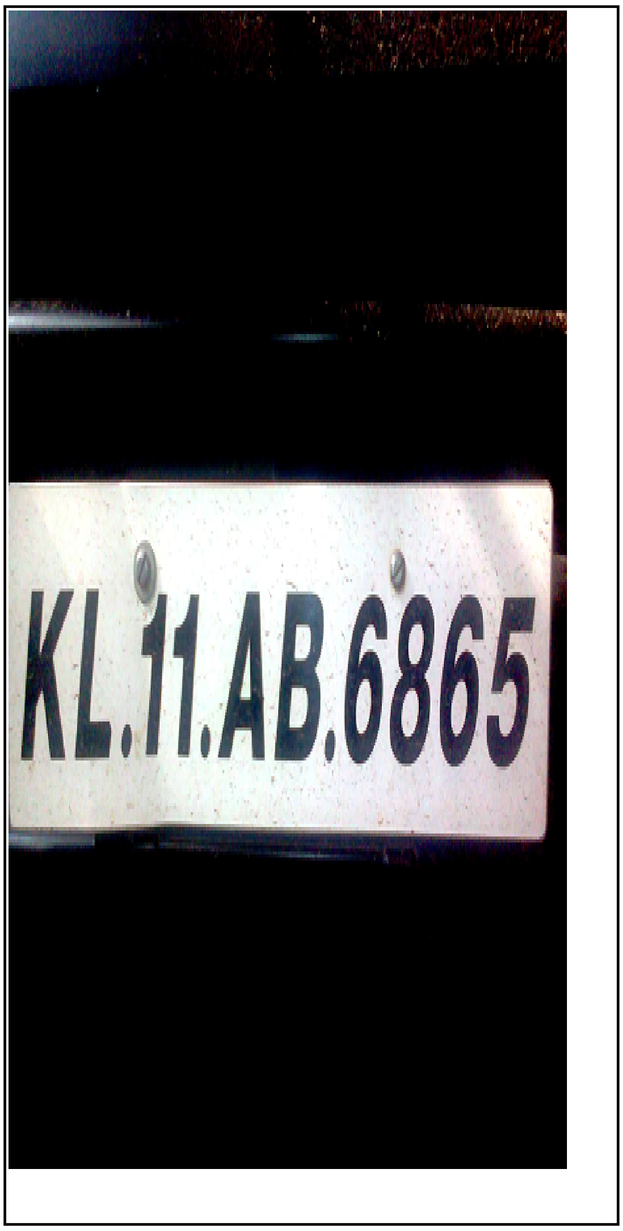

Fig. 7.1: inputdata

CHAPTER 8
Conclusion

In this paper we have presented a real-time license plate detection and recognition system. The system operates on image frames acquired with standard video equipment without any additional sensor input. The high performance of the system allows for compensating the low image resolution by considering the classification results of subsequent frames. Due to the complete integration on an embedded device, the system operates autonomously, reporting only the final classification results to connected operators. Self-evidently, all advantages and capabilities of distributed systems apply here as well. We are currently working on an optimization to the detection algorithm implementation; our goal is to adapt the algorithm to better fit to the architectural characteristics of our platform. In the future we will also have a look at special region segmentation methods as the one proposed in [13] and examine their applicability on our hardware.

[?]

REFERENCES

[1] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(11):14751490, 2002.

[2] C. Arth, C. Leistner, and H. Bischof. TRIcam: An Embedded Platform for Remote Traffic Surveillance. In Proceedings of the IEEE Conference on Computer Vision and Pattern

[3]V. Kasmat, and S. Ganesan, An efficient implementation of the Hough transform for detecting vehicle license plates using DSPs, IEEE International Conference on Real-Time Technology and Application Symposium, Chicago, USA, pp. 58-59, 2005.

[4]S.H. Park, K.I. kim, K. Jung and H.J. Kim, Locating car license plate using Neural Network, Electronic Letters, Vol. 35, No. 17, pp. 1474 1477, 1999.

[5]K.K. KIM, K.I., KIM, J.B. KIM, and H.J. KIM, Learning-Based Apporach for License Plate Recognition Proceeding of IEEE Signal Processing Society Workshop, Vol. 2, pp.614-623, 2000.

[6]S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(11):14751490, 2002.

[7]C. Arth, C. Leistner, and H. Bischof. TRIcam: An Embedded Platform for Remote Traffic Surveillance. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (Embedded Computer Vision Workshop), 2006.

[8]N. Bellas, S. M. Chai, M. Dwyer, and D. Linzmeier. FPGA implementation of a license plate recognition soc using automatically generated streaming accelerators. In 20th International Parallel and Distributed Processing Symposium (IPDPS), page 8 pp., 2006.

[9]C.-C. Chang and C.-J. Lin. LIBSVM: a library for support vector machines, 2001. Software available at http://www.csie.ntu.edu.tw/cjlin/libsvm.