VIDEO/IMAGE SEGMENTATION USING MUTIMODAL ANALYSIS

A PROJECTREPORT

Submitted by

SAKTHI NIVEDITHA.R	**REG NO.:13BEC126**
SOWBARNIKA.L.S	**REG NO.:13BEC146**
SOWNDARYA.R	**REG NO.:13BEC147**
VISHALAKSHI.M	**REG NO.:13BEC173**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION

KUMARAGURU COLLEGE OF TECHNOLOGY

ANNA UNIVERSITY : CHENNAI

APRIL 2017

BONAFIDE CERTIFICATE

Certified that this project report **“VIDEO/IMAGE SEGMENTATION USING MULTIMODAL ANALYSIS” is the bonafide work of Ms SAKTHI NIVEDITHA.R [13BEC126], Ms SOWBARNIKA.L.S [13BEC146], Ms SOWNDARYA.R [13BEC147] and Ms VISHALAKSHI.M [13BEC173]**who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other project or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
Mr. Ram Prakash ,M.E	
PROJECT SUPERVISOR	**HEAD OF THE DEPARTMENT**
Department of ECE	Department of ECE
Kumaraguru College of Technology	Kumaraguru College of Technology
Coimbatore-641049	Coimbatore- 641049

The candidates with Register No: 13BEC126, 13BEC146, 13BEC147, 13BEC173 are examined by us in the project viva-voce examination held on …………………….

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We express our sincere thanks to the Management of Kumaraguru College of Technology and Joint Correspondent **Shri.ShankarVanavarayar** for the kind support and for providing necessary facilities to carry out the project work.

We would like to express our sincere thanks to our beloved Principal **Dr.R.S.Kumar, Ph.D.,**Kumaraguru College of Technology, who encouraged us in each and every step of the project.

We would like to thank   Head of the Department, Electronics and Communication Engineering, for her kind support and for providing necessary facilities to carry out the project work.

We wish to thank with everlasting gratitude to our Project Coordinator **Mrs.Rama Latha, M.E.,** Department of Electronics and Communication Engineering for her consistent support throughout the course of this project work.

We are greatly privileged to express our deep sense of gratitude and heartfelt thanks to our Project Guide **Mr.Ram Prakash, M.E.,** Department of Electronics and Communication Engineering for his expert counseling and guidance to make this project to a great deal of success and also we wish to convey our regards to all teaching and non-teaching staff of ECE Department for their help and cooperation.

Finally, we thank our parents and our family members for giving us the moral support and abundant blessings in all of our activities and our dear friends who helped us to endure our difficult times with their unfailing support and warm wishes.

ABSTRACT

Surveillance is one of the most promising applications for video processing, stimulated by a confluence of simultaneous advances in key disciples :computer vision, image segmentation, embedded computing. However, computer vision typically requires notable amounts of computing performance , a considerable memory footprint and high power consumption. To overcome this Video processing algorithm is used .Video processing requires a stream processing architecture , in which video frames from a continuous stream are processed one (or more) at a time. Computer vision system toolbox supports a stream processing architecture through system objects for use in MATLAB.Finally simulation results show how perpetual work can be achieved in an indoor scenario within a typical video surveillance application dealing with abandoned removed object detection.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATION

GSM	**Global System For Mobile Communication**
BMP	**Bit Map Image**
JPEG	**Joint Photographic Experts Group**
RGB	**Red Green Blue**
PCX	**Picture Exchange**

1. INTRODUCTION

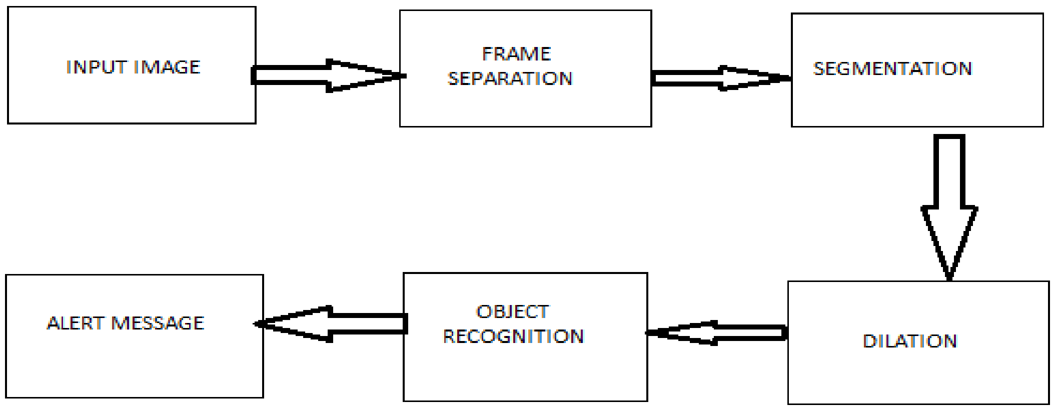
Computer Vision System Toolbox provides algorithms, functions, and apps for designing and simulating computer vision and video processing systems. You can perform feature detection, extraction, and matching; object detection and tracking; motion estimation; and video processing. For 3-D computer vision, the system toolbox supports camera calibration, stereo vision, 3-D reconstruction, and 3-D point cloud processing. With machine learning based frameworks, you can train object detection, object recognition, and image retrieval systems. Algorithms are available as MATLAB functions, System objects, and Simulinkblocks.

For rapid prototyping and embedded system design, the system toolbox supports fixed-point arithmetic and C-code generation.

Features of proposed system:

- Compatible for any computers
- Self dependent OS
- Time dependent and reliable system
- High accuracy and convenience
- High response speed

1.1 BLOCK DIAGRAM:



1.2 PROJECT EXECUTION:

Most real-time signal processing applications use stream processing, a memory-efficient technique for handling large amounts of data. Stream processing divides incoming data into frames and fully processes each frame before the next one arrives. Examples of applications that use stream processing include audio enhancement, wireless baseband processing, object tracking, and radar beam forming.

The just-in-time and memory-sensitive nature of stream processing presents special challenges. Streaming algorithms must be efficient and keep up with the rate of data updates. To handle large data sets, the algorithms must also manage memory and state information, store previous data buffers only as needed, and update each buffer and state frame-by-frame.

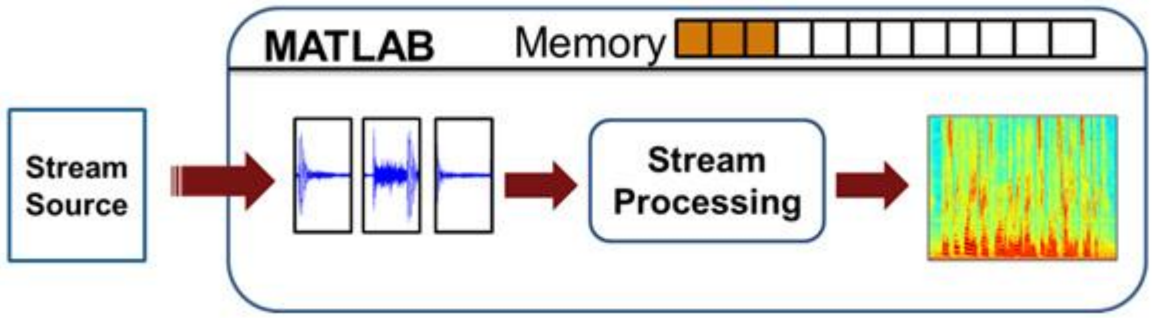


Fig 1.2 MATLAB Process

Object Detection and Recognition

Object detection andrecognition are used to locate, identify, and categorize objects in images and video. Computer Vision System Toolbox provides a comprehensive suite of algorithms and tools for object detection and recognition.

1.3 REQUIREMENTS:

1.3.1 Hardware Requirements

- Arduino
- GSM-800A
- RS-232 Cable
- LCD
- Keypad
- Power supply

1.3.2 Software Requirements

- MATLAB
- Arduino-1.6.5-r2

2.HARDWARE DESCRIPTION:

2.1 BLOCK DIAGRAM:

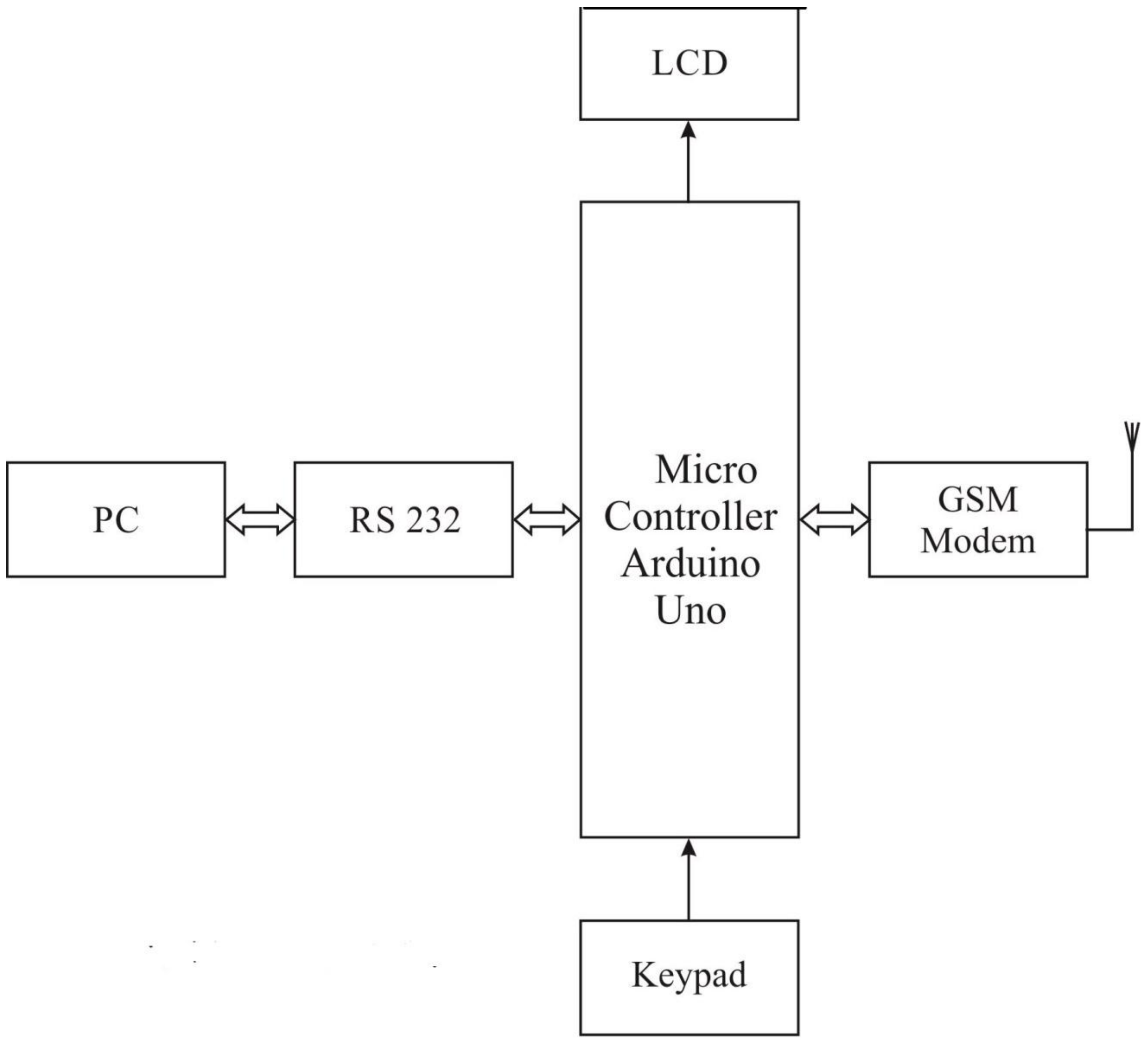


Fig 2.1:Block Diagram

2.2 ARDUINO:

2.2.1 INTRODUCTION

The ATmega328P is a low-power CMOS 8-bit microcontroller based on theAVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, theATmega328P achieves throughputs approaching 1 MIPS per MHz allowingthe system designer to optimize power consumption versus processing speed.

Fig 2.2:ARDUINO UNO

2.2.2:PIN DETAILS AND DESCRIPTION

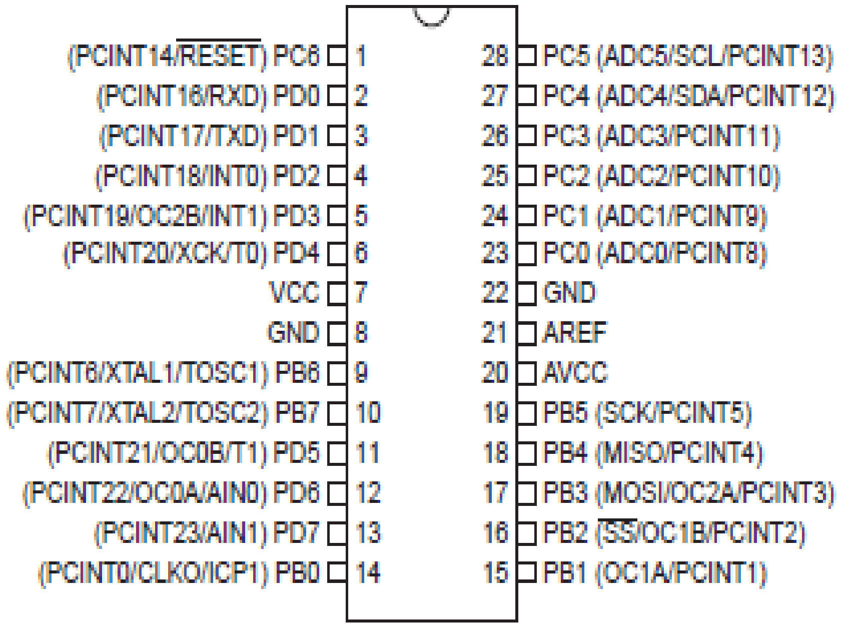


Fig 2.3:Pin Details

Pin Descriptions:

VCC

Digital supply voltage.

GND

Ground.

Port B (PB7:0) XTAL1/XTAL2/TOSC1/TOSC2

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Depending on the clock selection fuse settings, PB6 can be used as input to the inverting Oscillator amplifier and input to the internal clock operating circuit. Depending on the clock selection fuse settings, PB7 can be used as output from the inverting Oscillator amplifier.

If the Internal Calibrated RC Oscillator is used as chip clock source, PB7..6 is used as TOSC2..1 input for the Asynchronous Timer/Counter2 if the AS2 bit in ASSR is set.

Port C (PC5:0)

Port C is a 7-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The PC5..0 output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

PC6/RESET

If the RSTDISBL Fuse is programmed, PC6 is used as an I/O pin. Note that the electrical characteristics of PC6 differ from those of the other pins of Port C. If the RSTDISBL Fuse is unprogrammed, PC6 is used as a Reset input. A low level on this pin for longer than the minimum pulse length will generate a Reset, even if the clock is not running.

Port D (PD7:0)

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

AVCC

AVCC is the supply voltage pin for the A/D Converter, PC3:0, and ADC7:6. It should be externally connected to VCC, even if the ADC is not used. If the ADC is used, it should be connected to VCC through a low-pass filter. Note that PC6..4 use digital supply voltage, VCC.

AREF

AREF is the analog reference pin for the A/D Converter.

2.2.3 ATMEGA 328 FEATURES:

4K/8K bytes of In-SystemProgrammable Flash with Read-While-Write capabilities, 1K bytes EEPROM, 2K bytes SRAM, 23 general purpose I/O lines, 32 general purpose working registers,three flexible Timer/Counters with compare modes, internal and external interrupts, a serial programmableUSART, a byte-oriented 2-wire Serial Interface, an SPI serial port, a 6-channel 10-bitADC (8 channels in TQFP and QFN/MLF packages), a programmable Watchdog Timer withinternal Oscillator, and five software selectable power saving modes. The Idle mode stops theCPU while allowing the SRAM, Timer/Counters, USART, 2-wire Serial Interface, SPI port, andinterrupt system to continue functioning. The Power-down mode saves the register contents butfreezes the Oscillator, disabling all other chip functions until the next interrupt or hardware reset.In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain atimer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops theCPU and all I/O modules except asynchronous timer and ADC, to minimize switching noise duringADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the restof the device is sleeping. This allows very fast start-up combined with low power consumption.

The device is manufactured using Atmel's high density non-volatile memory technology. TheOn-chip ISP Flash allows the program memory to be reprogrammed In-System through an SPIserial interface, by a conventional non-volatile memory programmer, or by an On-chip Boot programrunning on the AVR core. The Boot program can use any interface to download theapplication program in the Application Flash memory. Software in the Boot Flash section willcontinue to run while the Application Flash section is updated, providing true Read-While Writeoperation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega328P is a powerful microcontroller thatprovides a highly flexible and cost effective solution to many embedded control applications.

The ATmega328P AVR is supported with a full suite of program and systemdevelopment tools including: C Compilers, Macro Assemblers, ProgramDebugger/Simulators,In-Circuit Emulators, and Evaluation kits.

2.3 GSM:

A GSM modem is a specialized type of modem which accepts a SIM card, and operates over a subscription to a mobile operator, just like a mobile phone. From the mobile operator perspective, a GSM modem looks just like a mobile phone.

A GSM modem exposes an interface that allows applications such as nowSMS to send and receive messages over the modem interface. The mobile operator charges for this message sending and receiving as if it was performed directly on a mobile phone. To perform these tasks, a GSM modem must support an “extended AT command set” for sending/receiving SMS messages, as defined in the ETSI GSM 07.05 and and 3GPP TS 27.005 specifications.

GSM modems can be a quick and efficient way to get started with SMS, because a special subscription to an SMS service provider is not required. In most parts of the world, GSM modems are a cost effective solution for receiving SMS messages, because the sender is paying for the message delivery.

A GSM modem can be a dedicated modem device with a serial, USB or Bluetooth connection such as RS 232.A GSM modem could also be a standard GSM mobile phone with the appropriate cable and software driver to connect to a serial port or USB port on your computer. Any phone that supports the “extended AT command set” for sending/receiving SMS messages.

2.3.1:FEATURES:

- Quad Band GSM/GPRS
- 850/900/1800/1900 MHz
- GPRS multi-slot class 10/8
- GPRS Mobile station class B
- Compliant to GSM Phase 2/2+
- Class 4 (2W@850/900Mhz)
- Class 1(1W@1800/1900Mhz)
- Control via AT commands(GSM 07.07, 07.05 and enhanced AT commands)
- Operation Temperature(-20 deg C to +55 deg C)

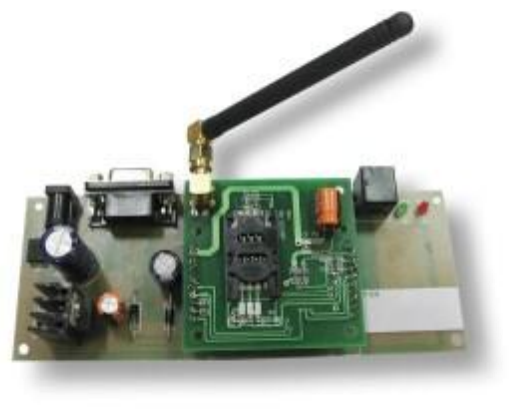


Fig 2.4:GSM Module

For sending message, a GSM Module named SIMCOM_300 with RS232, power supply, buzzer and audio interface are used. This can be connected to PC by using a USB to

Serial Adaptor. Terminal programs such as Real term are used to send & receive data. The interface between GSM Module and microcontroller can also be done directly with the help of wires.

2.3.2 AT COMMANDS:

GSM Module works with AT COMMANDS where AT stands for Application Terminal.

Some useful AT Commands are:

1. AT
2. AT+CMGS
3. AT+CMGR
4. AT+CMGD
5. AT+CSQ?

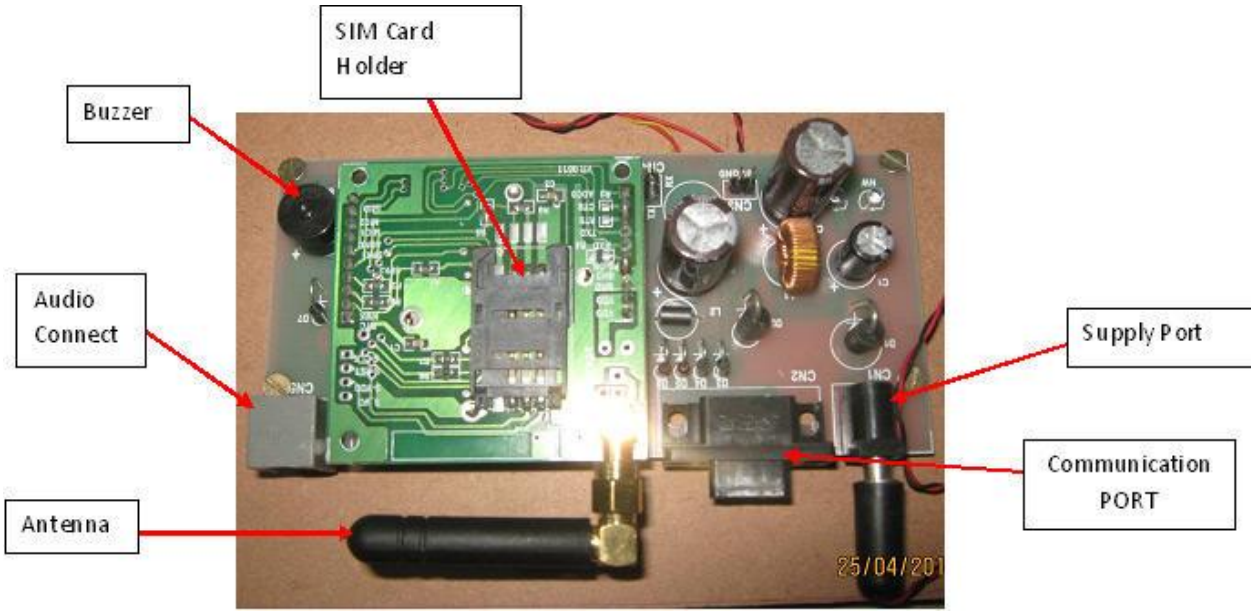


Fig 2.5:GSM PORTS

2.3.3 CONNECTION BETWEEN MICROCONTROLLER AND GSM MODULE:

For connection, Receiver Pin (Rx) of Microcontroller is connected to the Transmitter Pin (Tx) of GSM Module and Transmitter Pin (Tx) of Microcontroller is connected to the Receiver Pin (Rx) of GSM Module. Also Ground Pin (GND) of both are connected.

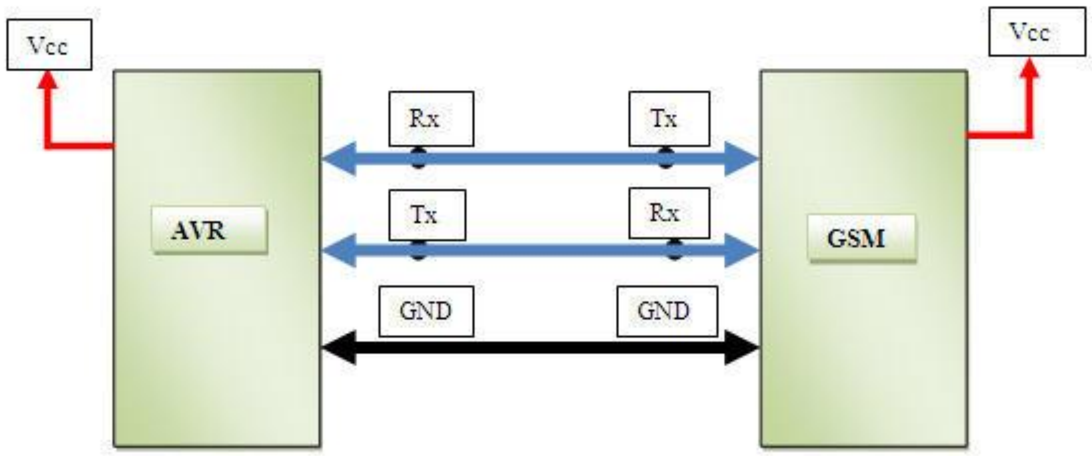


Fig 2.6:Connection between Microcontroller and GSM module

2.4 RS- 232 CABLE:

In telecommunications, **RS-232** is a standard for serial binary data interconnection between a *DTE* (Data terminal equipment) and a *DCE* (Data Circuit-terminating Equipment). It is commonly used in computer serial ports.

2.4.1 Scope of the Standard:

The Electronic Industries Alliance (EIA) standard RS-232-C [3] as of 1969 defines:

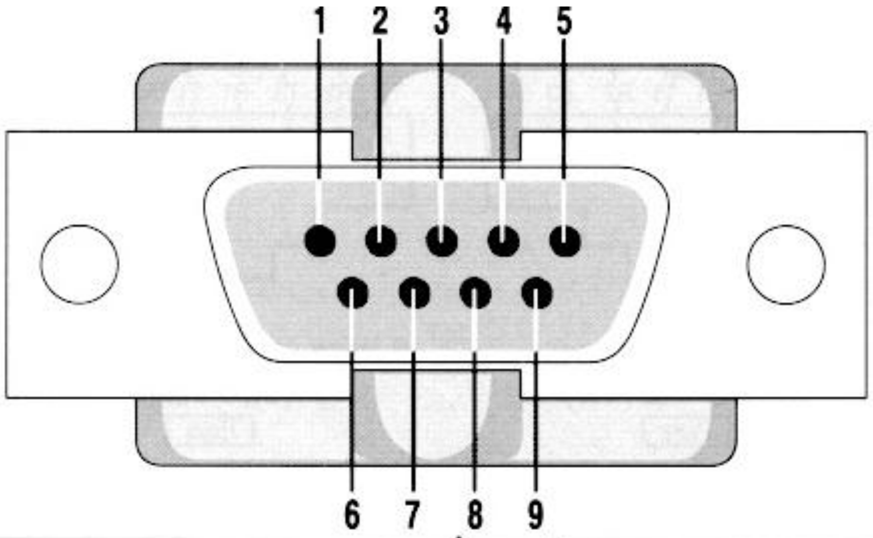
- Electrical signal characteristics such as voltage levels, signaling rate, timing and slew-rate of signals, voltage withstand level, short-circuit behavior, maximum stray capacitance and cable length
- Interface mechanical characteristics, pluggable connectors and pin identification
- Functions of each circuit in the interface connector
- Standard subsets of interface circuits for selected telecom applications

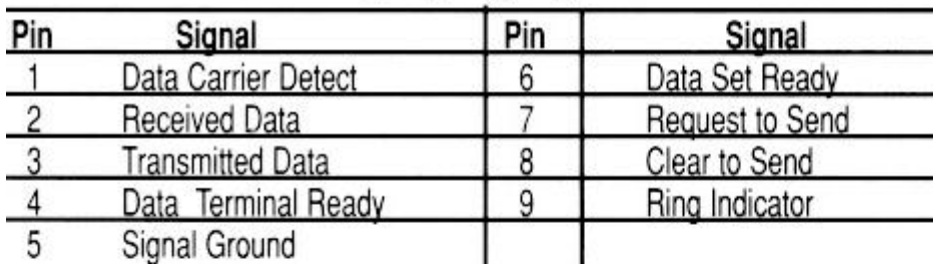
The standard does not define such elements as character encoding (for example, ASCII, Baudot or EBCDIC), or the framing of characters in the data stream (bits per character, start/stop bits, parity). The standard does not define protocols for error detection or algorithms for data compression.

The standard does not define bit rates for transmission, although the standard says it is intended for bit rates lower than 20,000 bits per second. Many modern devices can exceed this speed (38,400 and 57,600 bit/s being common, and 115,200 and 230,400 bit/s making occasional appearances) while still using RS-232 compatible signal levels.

Details of character format and transmission bit rate are controlled by the serial port hardware, often a single integrated circuit called a UART that converts data from parallel to serial form. A typical serial port includes specialized driver and receiver integrated circuits to convert between internal logic levels and RS-232 compatible signal levels.

2.4.2 RS-232 SIGNALLING:





Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		

TABLE 2.1:PINS AND SIGNALS

2.5 LCD DISPLAY:

LCD (Liquid Crystal Display) screen is an electronic display module and find a wide range of applications. A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits. These modules are preferred over seven segments and other multi segment LEDs. The reasons being: LCDs are economical; easily programmable; have no limitation of displaying special & even custom characters (unlike in seven segments), animations and so on.

A 16x2 means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. This LCD has two registers, namely, Command and Data.

The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD.

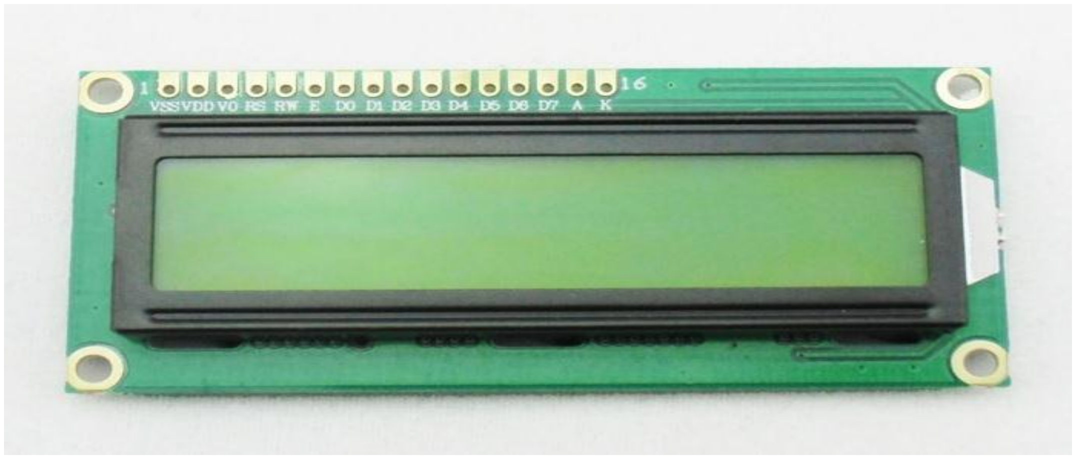


Fig 2.7: LCD

2.5.1: PIN DESCRIPTION :

PIN NUMBER	SYMBOL	FUNCTION
1	Vss	GND
2	Vdd	+ 3V or + 5V
3	Vo	Contrast Adjustment
4	RS	H/L Register Select Signal
5	R/$\overline{W}$	H/L Read/Write Signal
6	E	H ®L Enable Signal
7	DB0	H/L Data Bus Line
8	DB1	H/L Data Bus Line
9	DB2	H/L Data Bus Line
10	DB3	H/L Data Bus Line
11	DB4	H/L Data Bus Line
12	DB5	H/L Data Bus Line
13	DB6	H/L Data Bus Line
14	DB7	H/L Data Bus Line
15	A/Vee	+ 4.2V for LED/Negative Voltage

		Output
16	K	Power Supply for B/L (OV)

TABLE 2.2: PIN DESCRIPTION

MECHANICAL DATA		
ITEM	**STANDARD VALUE**	**UNIT**
Module Dimension	80.0 x 36.0	mm
Viewing Area	66.0 x 16.0	mm
Dot Size	0.56 x 0.66	mm
Character Size	2.96 x 5.56	mm

TABLE 2.3: MECHANICAL DATA

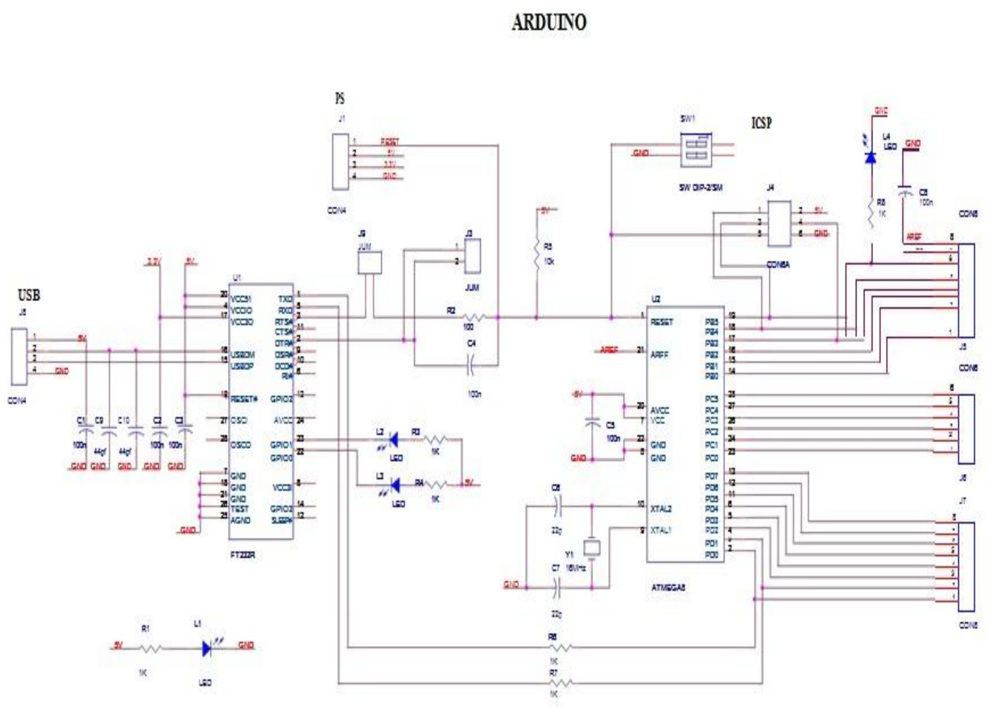


Fig 2.8: ARDUINO INTERFACED WITH LCD

2.6 KEYPAD

A group of keys in a single printed circuit board is call key pad. These key pads are classified into two types.

1) Key pad
2) Matrix keypad

KEYPAD

In a key pad it has a one or more then one keys are placed in a PCB. And all the keys are commonly grounded. This is the main difference to compared to matrix keypad. This key pads having maximum 8 numbers of keys. morethen 8 keys are can not be connected because its not a efficient one. If we need more then 8 kays means, then only we can operate it a matrix keypad.

2.6.1 SCHEMATIC EXPLANATION:

There are many methods depending on how you connect your keypad with your controller, but the basic logic is same. We make the columns as i/p and we drive the rows making them o/p, this whole procedure of reading the keyboard is called scanning. In order to detect which key is pressed from the matrix, we make row lines low one by one and read the columns. Lets say we first make Row1 low, then read the columns. If any of the key in row1 is pressed will make the corresponding column as low i.e. if second key is pressed in Row1, then column2 will give low. So we come to know that key 2 of Row1 is pressed. This is how scanning is done. So to scan the keypad completely, we need to make rows low one by one and read the columns. If any of the button is pressed in a row, it will take the corresponding column to a low state which tells us that a key is pressed in that row. If button 1 of a row is pressed then Column 1 will become low, if button 2 then column2 and so on.

2.6.2 APPLICATION:

Basically key pad is a number of buttons compiled in such a manner so that forms formation of numeral button and some other menus. Following is example configuration of key pad. Keypad needed to interaction with system, for example we make setting with set-point would a control feedback at the time of program still run. Actually every programmer has different way interaction to with system. Even for keypad in hardware every programmer can differ in. This thing is more because of different requirement.

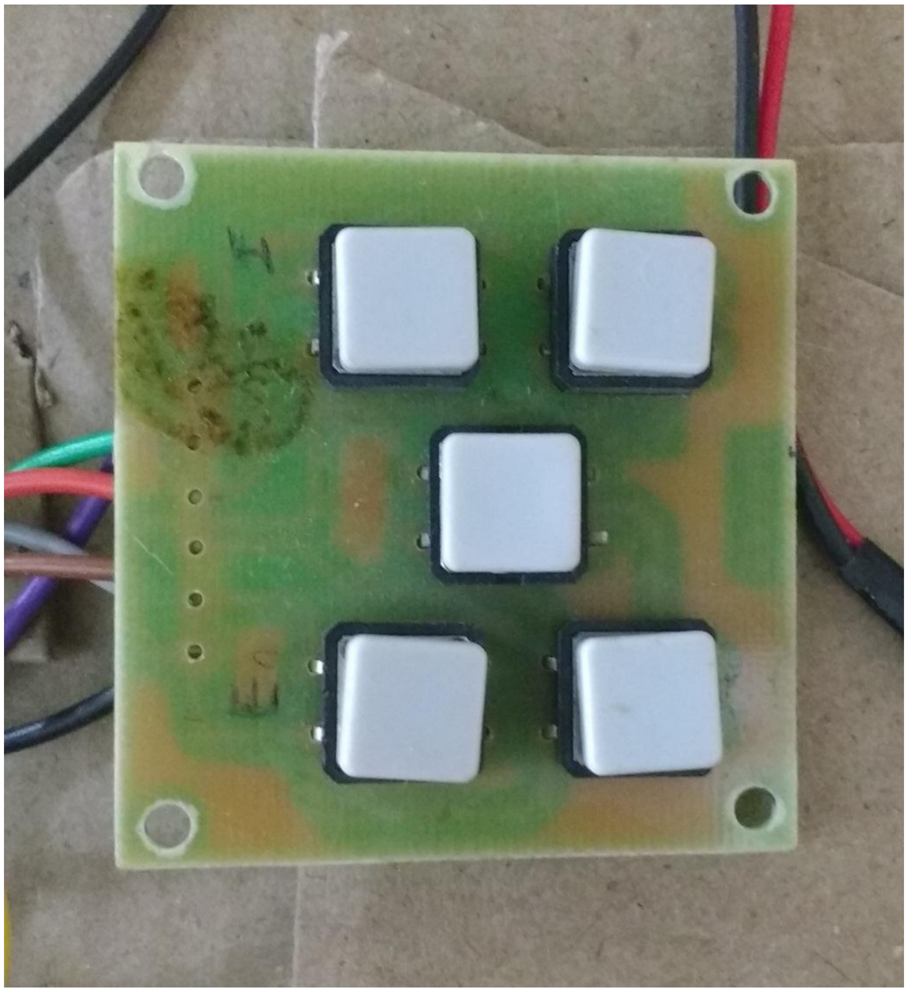


Fig 2.9:

2.7 POWER SUPPLY:

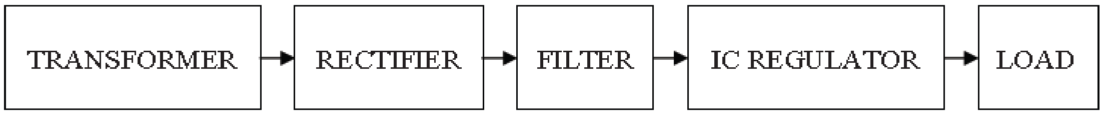
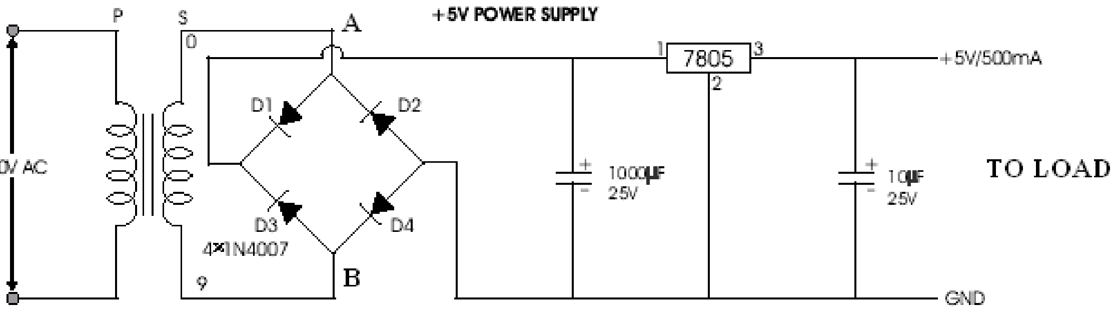


Fig 2.10: Block diagram (Power supply)

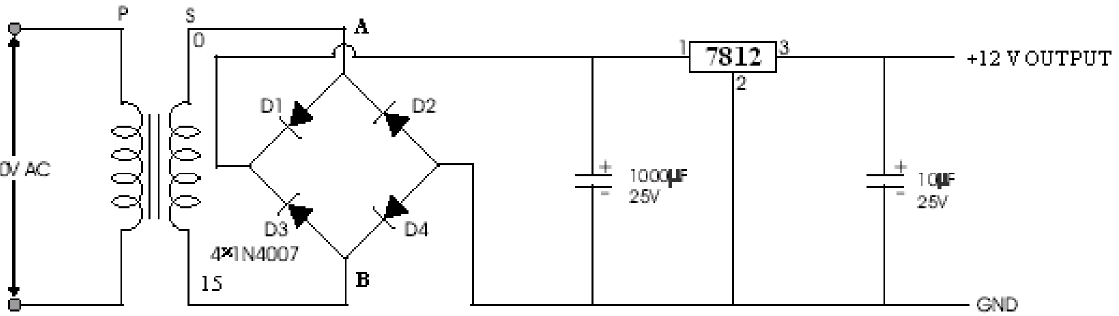
The ac voltage, typically 220V rms, is connected to a transformer, which steps that ac voltage down to the level of the desired dc output. A diode rectifier then provides a full-wave rectified voltage that is initially filtered by a simple capacitor filter to produce a dc voltage. This resulting dc voltage usually has some ripple or ac voltage variation. A regulator circuit removes the ripples and also remains the same dc value even if the input dc voltage varies. This voltage regulation is usually obtained using one of the popular voltage regulator IC units.

2.7.1 SCHEMATIC DIAGRAM :

Fig 2.11: Power Supply



2.7.2 Working principle



Transformer

The potential transformer will step down the power supply voltage (0-230V) to (0-15V and 0-9V) a level. If the secondary has less turns in the coil then the primary, the secondary coil's voltage will decrease and the current or AMPS will increase or decreased depend upon the wire gauge. This is called a STEP-DOWN transformer.Then the secondary of the potential transformer will be connected to the rectifier.

2.7.3 Bridge rectifier

When four diodes are connected as shown in figure, the circuit is called as bridge rectifier. The input to the circuit is applied to the diagonally opposite corners of the network, and the output is taken from the remaining two corners.

Let us assume that the transformer is working properly and there is a positive potential, at point A and a negative potential at point B. the positive potential at point A will forward bias D3 and reverse bias D4.

The negative potential at point B will forward bias D1 and reverse D2. At this time D3 and D1 are forward biased and will allow current flow to pass through them; D4 and D2 are reverse biased and will block current flow. The path for current flow is from point B through D1, up through Load, through D3, through the secondary of the transformer back to point B.

One-half cycle later the polarity across the secondary of the transformer reverse, forward biasing D2 and D4 and reverse biasing D1 and D3. Current flow will now be from point A through D4, up through Load, through D2, through the secondary of transformer, and back to point A. Across D2 and D4. The current flow through Load is always in the same direction. In flowing through Load this current develops a voltage corresponding to that. Since current flows through the load during both half cycles of the applied voltage, this bridge rectifier is a full-wave rectifier.

2.7.4 IC voltage regulators

Voltage regulators comprise a class of widely used ICs. Regulator IC units contain the circuitry for reference source, comparator amplifier, control device, and overload protection all in a single IC. IC units provide regulation of either a fixed positive voltage, a fixed negative voltage, or an adjustably set voltage.

A fixed three-terminal voltage regulator has an unregulated dc input voltage, it is applied to one input terminal, a regulated dc output voltage from a third terminal, with the second terminal connected to ground.

The series 78 regulators provide fixed positive regulated voltages from 5 to 24 volts. Similarly, the series 79 regulators provide fixed negative regulated voltages from 5 to 24 volts.

This is a regulated power supply circuit using the 78xx IC series. These regulators can deliver current around 1A to 1.5A at a fix voltage levels. The common regulated voltages are 5V, 6V, 8V, 9V, 10V, 12V, 15V, 18V, and 24V. It is important to add capacitors across the input and output of the regulator IC to improve the regulation.

In this circuit we are using 7805 and 7812 regulator so it converts variable dc into constant positive 5V and 12V power supply respectively.

2.7.5 OVERALL CIRCUIT:

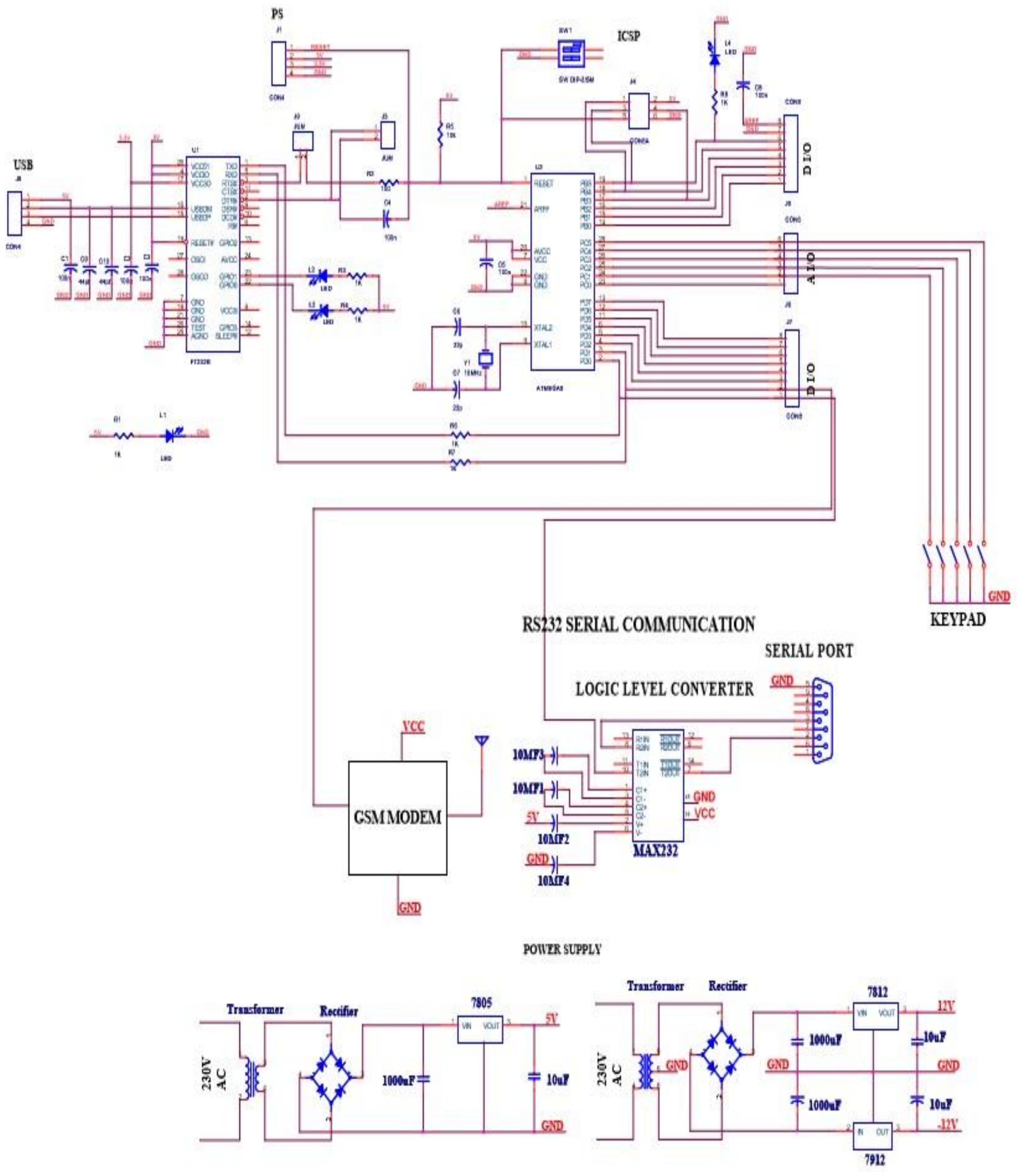


Fig 2.12: Overall circuit diagram

3.SOFTWARE DESCRIPTION:

3.1 IMAGE PROCESSING

Imageprocessing is a method to convert an image into digital form and perform some operations on it, in order to get an enhanced image or to extract some useful information from it. It is a type of signal dispensation in which input is image, like video frame or photograph and output may be image or characteristics associated with that image. Usually Image Processing system includes treating images as two dimensional signals while applying already set signal processing methods to them.

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

Image processing basically includes the following three steps:

- Importing the image with optical scanner or by digital photography.
- Analyzing and manipulating the image which includes data compression and image enhancement and spotting patterns that are not to human eyes like satellite photographs.
- Output is the last stage in which result can be altered image or report that is based on image analysis.

The purpose of image processing is divided into 5 groups. They are:

- Visualization - Observe the objects that are not visible.
- Image sharpening and restoration - To create a better image.
- Image retrieval - Seek for the image of interest.

- Measurement of pattern – Measures various objects in an image.
- Image Recognition – Distinguish the objects in an image.

BLOCK DIAGRAM :

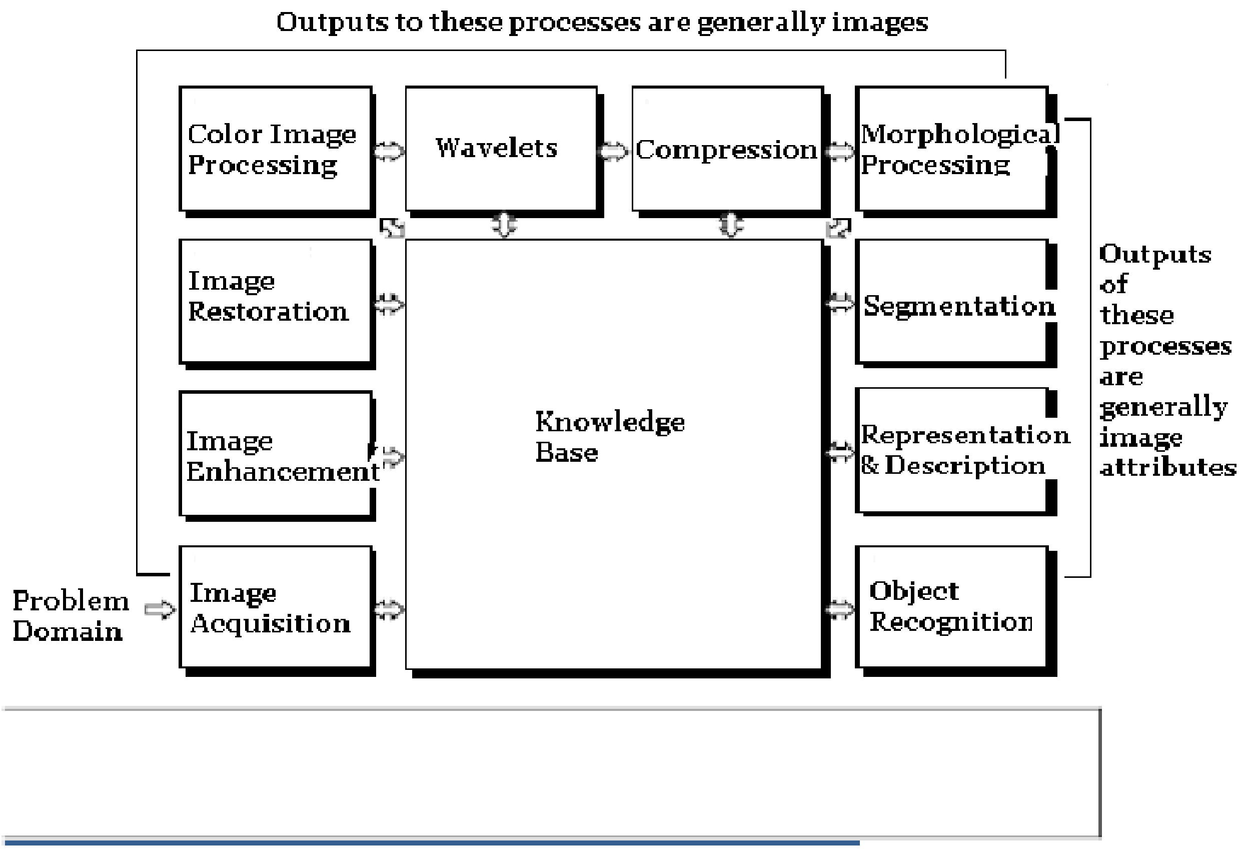


Fig.3.1.Block Diagram

3.1.1 IMAGE ENHANCEMENT:

Image enhancement is the improvement of digital image quality (wanted e.g. for visual inspection or for machine analysis), without knowledge about the source of degradation. If the source of degradation is known, one calls the process image restoration. Both are ironical processes, viz. input and output are images.

Many different, often elementary and heuristic methods are used to improve images in somesense. The problem is, of course, not well defined, as there is no objective measure for image quality. Here, we discuss a few recipes that have shown to be useful both for the human observer and/or for machine recognition. These methods are very problem-oriented: a method that works fine in one case may be completely inadequate for another problem.

Apart from geometrical transformations some preliminary grey level adjustments may be indicated, to take into account imperfections in the acquisition system. This can be done pixel by pixel, calibrating with the output of an image with constant brightness. Frequently space-invariant grey value transformations are also done for contrast stretching, range compression, etc. The critical distribution is the relative frequency of each grey value, the gray value histogram.

Examples are:

Grey values can also be modified such that their histogram has any desired shape, e.gflat(very grey value has the same probability). All examples assume pointprocessing, viz. each output pixel is the function of one input pixel; usually, the transformation is implemented with a look-up table:

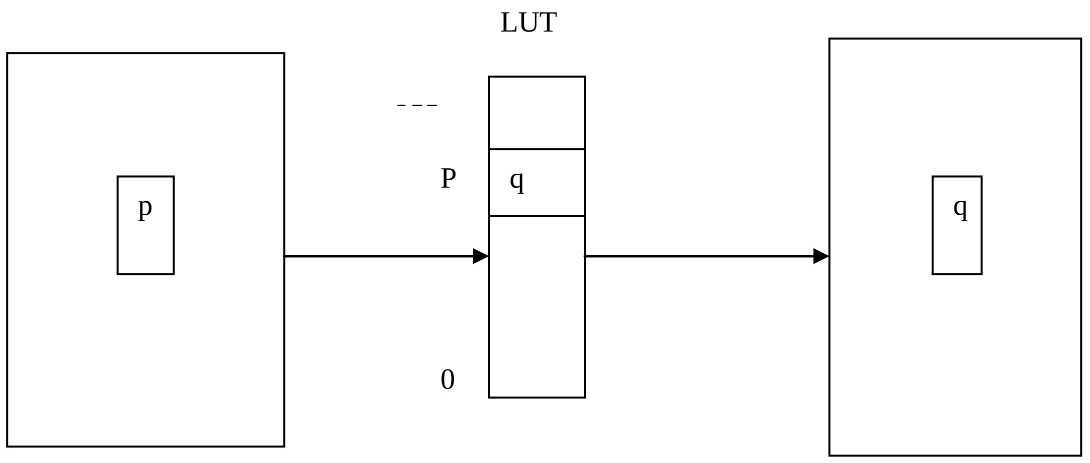


Table 3.1: Look-up table

Physiological experiments have shown that very small changes in luminance are recognized by the human visual system in regions of continuous grey value, and not at all seen in regions of some discontinuities. Therefore, a design goal for image enhancement often is to smooth images in more uniform regions, but to preserve edges. On the other hand, it has also been shown that somehow degraded images with enhancement of certain features, e.g. edges, can simplify image interpretation both for a human observer and for machine recognition. A second design goal, therefore, is image sharpening. All these operations need neighbourhood processing, viz. the output pixel is a function of some neighbourhood of the input pixels:

These operations could be performed using linear operations in either the frequency or the spatial domain. We could, e.g. design, in the frequency domain, one-dimensional low or high pass filters (Filtering), and transform them according to the two-dimensional case.

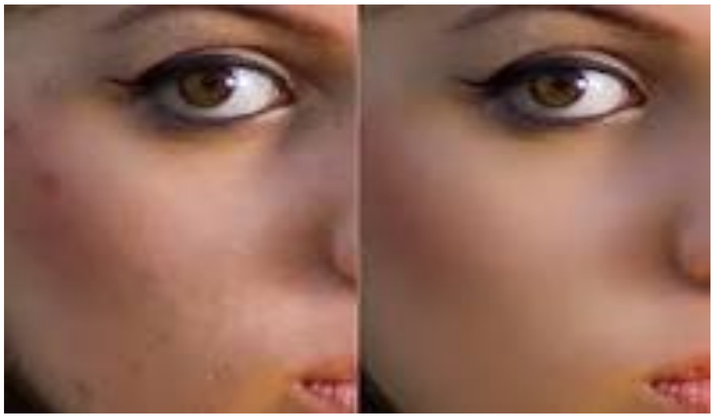
Unfortunately, linear filter operations do not really satisfy the above two design goals; in this book, we limit ourselves to discussing separately only (and superficially) Smoothing and Sharpening.

Here is a trick that can speed up operations substantially, and serves as an example for both point and neighbourhood processing in a binary image: we number the pixels in a 3x3 neighbourhood like:

5	6	7
4	8	0
3	2	1

And note the binary values (0,1) by b_i (i = 0,8); we then concatenate the bits into a 9-bit word, like $b_8b_7b_6b_5b_4b_3b_2b_1b_0$. This leaves us with a 9-bit grey value for each pixel, hence a new image (an 8-bit image with b_8 taken from the original binary image will also do). The new image corresponds to the result of a convolution of the binary image, with a 3 x 3 matrix containing as coefficients the powers of two. This neighbour image can then be passed through a look-up table to perform erosions, dilations, noise cleaning, etc.

Apart from point and neighbourhood processing, there are also global processing techniques, i.e. methods where every pixel depends on all pixels of the whole image. Histogram methods are usually global, but they can also be used in a neighbourhood.



(a) (b)

Figure 3.2:Imageenhancenent(a)original image (b) enhanced image

3.1.2 IMAGE COMPRESSION

Compressing an image is significantly different than compressing raw binary data. Of course, general purpose compression programs can be used to compress images, but the result is less than optimal. This is because images have certain statistical properties which can be exploited by encoders specifically designed for them. Also, some of the finer details in the image can be sacrificed for the sake of saving a little more bandwidth or storage space.

The objective of image compression is to reduce irrelevance and redundancy of the image data in order to be able to store or transmit data in an efficient form.

LOSSY AND LOSSLESS COMPRESSION

Image compression may be lossyor lossless. Lossless compression involves with compressing data which, when decompressed, will be an exact replica of the original data. This is the case when binary data such as executables, documents etc. are compressed. They need to be exactly reproduced when decompressed. On the other hand, images (and music too) need not be reproduced 'exactly'. An approximation of the original image is enough for most purposes, as long as the error between the original and the compressed image is tolerable.

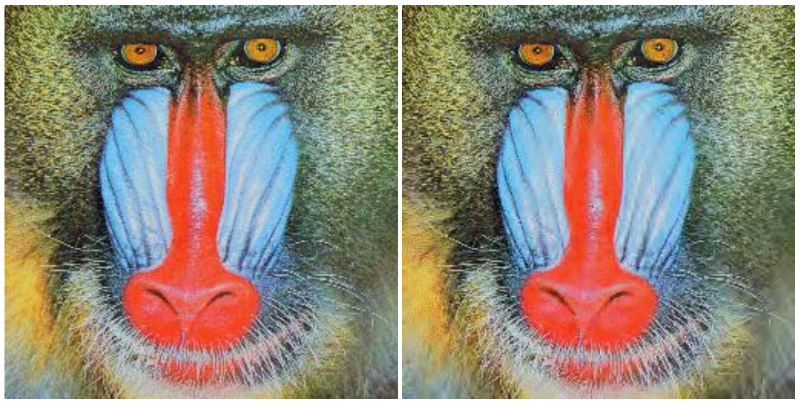
Lossless compression is preferred for archival purposes and often for medical imaging, technical drawings, clip art, or comics. This is because lossy compression methods, especially when used at low bit rates, introduce compression artifacts. Lossy methods are especially suitable for natural images such as photographs in applications where minor (sometimes imperceptible) loss of fidelity is acceptable to achieve a substantial reduction in bit rate. The lossy compression that produces imperceptible differences may be called visually lossless.

Methods for lossless image compression

- Run-length encoding – used as default method in PCX and as one of possible in BMP, TGA, TIFF
- DPCM and Predictive Coding
- Entropy encoding
- Adaptive dictionary algorithms such as LZW – used in GIF and TIFF
- Deflation – used in PNG, MNG, and TIFF
- Chain codes

Methods for lossy compression

- Reducing the color space to the most common colors in the image. The selected colors are specified in the color palette in the header of the compressed image. Each pixel just references the index of a color in the color palette. This method can be combined with dithering to avoid posterization.
- Chroma subsampling. This takes advantage of the fact that the human eye perceives spatial changes of brightness more sharply than those of color, by averaging or dropping some of the chrominance information in the image.
- Transform coding. This is the most commonly used method. A Fourier-related transform such as DCT or the wavelet transform are applied, followed by quantization and entropy coding.



(a) (b)

Figure 3.3: Image compression(a) original image (b) compressed image

Other properties

The best image quality at a given bit rate (or compression rate) is the main goal of image compression, however, there are other important properties of image compression schemes:

Scalability

Scalability generally refers to a quality reduction achieved by manipulation of the bit stream or file (without decompression and re-compression). Other names for scalability are progressive coding or embedded bit streams. Despite its contrary nature, scalability also may be found in lossless codes, usually in form of coarse-to-fine pixel scans. Scalability is especially useful for previewing images while downloading them (e.g., in a web browser) or for providing variable quality access to e.g., databases.

There are several types of scalability:

- Quality progressive or layer progressive: The bit stream successively refines the reconstructed image.
- Resolution progressive: First encode a lower image resolution; then encode the difference to higher resolutions
- Component progressive: First encode grey; then color.

Region of interest coding. Certain parts of the image are encoded with higher quality than others. This may be combined with scalability (encode these parts first, others later).

Meta information. Compressed data may contain information about the image which may be used to categorize, search, or browse images. Such information may include colour and texture statistics, small preview images, and author or copyright information.

Processing power. Compression algorithms require different amounts of processing power to encode and decode. Some high compression algorithms require high processing power.

The quality of a compression method often is measured by the Peak signal-to-noise ratio. It measures the amount of noise introduced through a lossy compression of the image, however, the subjective judgment of the viewer also is regarded as an important measure, perhaps, being the most important measure.

ERROR METRICS

Two of the error metrics used to compare the various image compression techniques are the Mean Square Error (MSE) and the Peak Signal to Noise Ratio (PSNR). The MSE is the cumulative squared error between the compressed and the original image, whereas PSNR is a measure of the peak error. The mathematical formulae for the two are

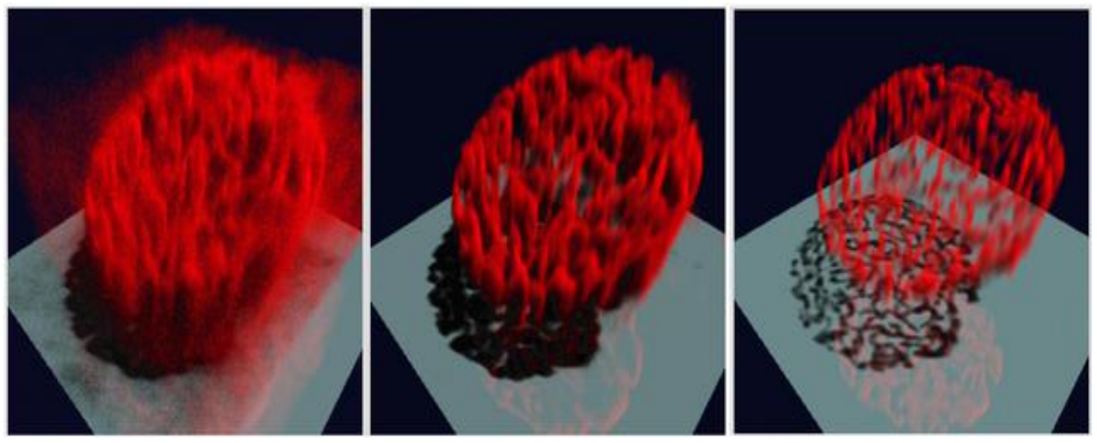
$$\text{MSE} = \frac{1}{MN}\sum_{y=1}^{M}\sum_{x=1}^{N}\left[I(x,y) - I'(x,y)\right]^2$$

PSNR = 20 * log10 (255 / sqrt(MSE))

where I(x,y) is the original image, I'(x,y) is the approximated version (which is actually the decompressed image) and M,N are the dimensions of the images. A lower value for MSE means lesser error, and as seen from the inverse relation between the MSE and PSNR, this translates to a high value of PSNR. Logically, a higher value of PSNR is good because it means that the ratio of Signal to Noise is higher. Here, the 'signal' is the original image, and the 'noise' is the error in reconstruction.

3.1.3 IMAGE RESTORATION

The purpose of image restoration is to "compensate for" or "undo" defects which degrade an image. Degradation comes in many forms such as motion blur, noise, and camera misfocus. In cases like motion blur, it is possible to come up with an very good estimate of the actual blurring function and "undo" the blur to restore the original image. In cases where the image is corrupted by noise, the best we may hope to do is to compensate for the degradation it caused. In this project, we will introduce and implement several of the methods used in the image processing world to restore images.



(b) (b) (c)

Figure 3.4:Image restoration(a) original image (b) partially restored image (c) fully restored image

3.2 Image Processing Toolbox:

The Image Processing Toolbox is a collection of functions that extend the capability of the MATLAB numeric computing environment. The toolbox supports a wide range of image processing operations, including:

- Geometric operations
- Neighborhood and block operations
- Linear filtering and filter design
- Transforms
- Image analysis and enhancement
- Binary image operations
- Region of interest operations

Many of the toolbox functions are MATLAB M-files, series of MATLAB statements that implement specialized image processing algorithms. You can view the MATLAB code for these functions using the statement :

Typefunction name

You can extend the capabilities of the Image Processing Toolbox by writing your own M-files, or by using the toolbox in combination with with other toolboxes, such as the Signal Processing Toolbox and the Wavelet Toolbox.

3.2.1 Images in MATLAB and the Image Processing Toolbox :

The basic data structure in MATLAB is the array, an ordered set of real or complex elements. This object is naturally suited to the representation of images, real-valued, ordered sets of color or intensity data. (MATLAB does not support complex-valued images.)

MATLAB stores most images as two-dimensional arrays (i.e., matrices), in which each element of the matrix corresponds to a single pixel in the displayed image. (Pixel is derived from picture element and usually denotes a single dot on a computer display.) For example, an image composed of 200 rows and 300 columns of different colored dots would be stored in MATLAB as a 200-by-300 matrix.

This convention makes working with images in MATLAB similar to working with any other type of matrix data, and makes the full power of MATLAB available for image processing applications. For example, you can select a single pixel from an image matrix using normal matrix subscripting:I(2,15)

3.2.2Data Types :

By default, MATLAB stores most data in arrays of class double. The data in these arrays is stored as double precision (64-bit) floating-point numbers. All of MATLAB’s functions and capabilities work with these arrays. For image processing, however, this data representation is not always ideal.

The number of pixels in an image may be very large; for example, a 1000-by-1000 image has a million pixels. Since each pixel is represented by at least one array element, this image would require about 8 megabytes of memory.

In order to reduce memory requirements, MATLAB supports storing image data in arrays of class uint8. The data in these arrays is stored as 8-bit unsigned integers.

Data stored in uint8 arrays requires one eighth as much memory as data in double arrays.

Because the types of values that can be stored in uint8 arrays and double arrays differ, the Image Processing Toolbox uses different conventions for interpreting the values in these arrays. (Noninteger values cannot be stored in uint8 arrays, for example, but they can be stored in double arrays.) The next section discusses how the toolbox interprets image data, depending on the class of the data array.

In addition to differences in the types of data values they store, uint8 arrays and double arrays differ in the operations that MATLAB supports. See page 1-13 for information about the operations MATLAB supports for uint8 arrays.

3.2.3 Image Types in the Toolbox :

The Image Processing Toolbox supports four basic types of images:

- Indexed images
- Intensity images
- Binary images
- RGB images

This section discusses how MATLAB and the Image Processing Toolbox represent each of these image types.

Indexed Images

An indexed image consists of two arrays, an image matrix and a colormap. The colormap is an ordered set of values that represent the colors in the image. For each image pixel, the image matrix contains a value that is an index into the colormap.

The colormap is an m-by-3 matrix of class double. Each row of the colormap matrix specifies the red, green, and blue (RGB) values for a single color:

color = [R G B]

R, G, and B are real scalars that range from 0 (black) to 1.0 (full intensity).

The figure below illustrates the structure of an indexed image.

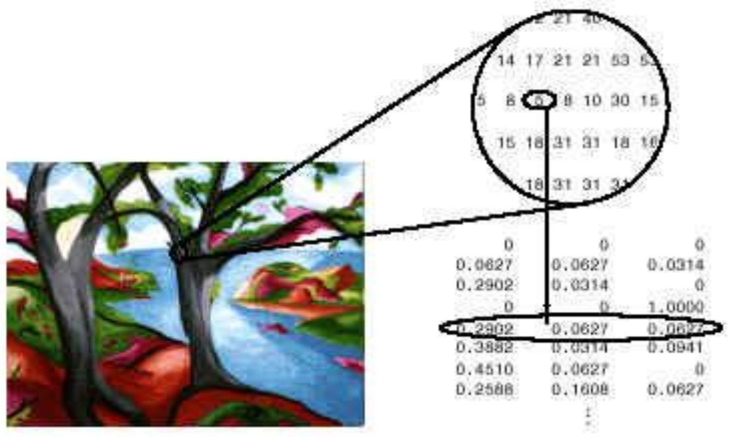


Fig 3.5: Indexed Image

The pixels in the image are represented by integers, which are pointers (indices) to color values stored in the colormap. The relationship between the values in the image matrix and the colormap depends on whether the image matrix is of class double or uint8. If the image matrix is of class double, the value 1 points to the first row in the colormap, the value 2 points to the second row, and so on. If the image matrix is of class uint8, there is an offset; the value 0 points to the first row in the colormap, the value 1 points to the second row, and so on.

The uint8 convention is also used in graphics file formats, and enables 8-bit indexed images to support up to 256 colors. In the image above, the image matrix is of class double, so there is no offset. For example, the value 5 points to the fifth row of the colormap.

Intensity Images

MATLAB stores an intensity image as a single matrix, with each element of the matrix corresponding to one image pixel. The matrix can be of class double, in which case it contains values in the range [0,1], or of class uint8, in which case the data range is [0,255]. The elements in the intensity matrix represent various intensities, or gray levels, where the intensity 0 represents black and the intensity 1 (or 255) represents full intensity, or white.

This figure depicts an intensity image of class double.

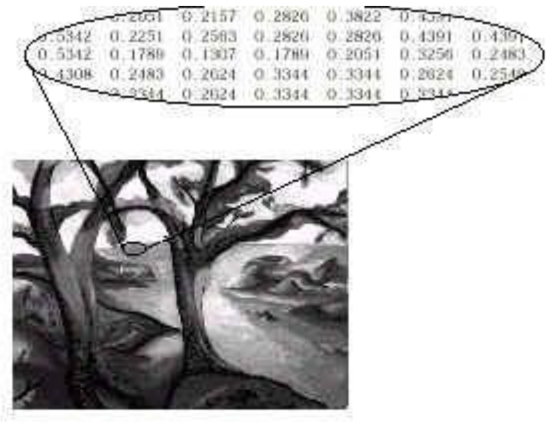


Fig.3.6 : Intensity image

Binary Images

In a binary image, each pixel assumes one of only two discrete values. Essentially, these two values correspond to on and off. A binary image is stored as a two-dimensional matrix of 0's (off pixels) and 1's (on pixels).

A binary image can be considered a special kind of intensity image, containing only black and white. Other interpretations are possible, however; you can also think of a binary image as an indexed image with only two colors.

A binary image can be stored in an array of class double or uint8. However, a uint8 array is preferable, because it uses far less memory. In the Image Processing Toolbox, any function that returns a binary image returns it as a uint8 logical array. The toolbox uses the presence of the logical flag to signify that the data range is [0,1]. (If the logical flag is off, the toolbox assumes the data range is [0,255].)

This figure shows an example of a binary image.

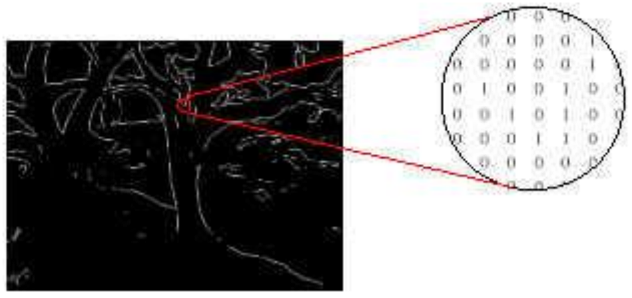


Fig.3.7 : Binary image

RGB Images

Like an indexed image, an RGB image represents each pixel color as a set of three values, representing the red, green, and blue intensities that make up the color. Unlike an indexed image, however, these intensity values are stored directly in the image array, not indirectly in a colormap.

In MATLAB, the red, green, and blue components of an RGB image reside in a single m-by-n-by-3 array. m and n are the numbers of rows and columns of pixels in the image, and the third dimension consists of three planes, containing red, green, and blue intensity values. For each pixel in the image, the red, green, and blue elements combine to create the pixel's actual color.

For example, to determine the color of the pixel (112,86), look at the RGB triplet stored in (112,86,1:3). Suppose (112,86,1) contains the value 0.1238, (112,86,2) contains 0.9874, and (112,86,3) contains 0.2543. The color for the pixel at (112,86) is:

0.1238 0.9874 0.2543

An RGB array can be of class double, in which case it contains values in the range [0,1], or of class uint8, in which case the data range is [0,255]. The figure below shows an RGB image of class double:

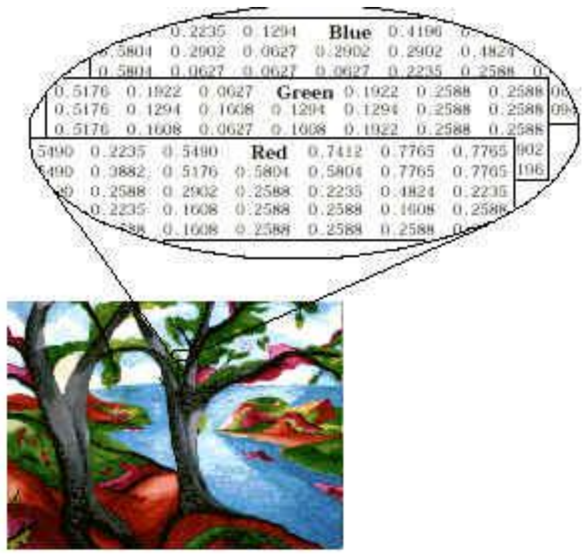


Fig.3.8: RGB image

Working with Image Data

This section discusses ways of working with the data arrays that represent images, including:

- Reading in image data from files, and writing image data out to files
- Converting images to other image types
- Working with uint8 arrays in MATLAB and the Image Processing Toolbox

3.2.4 Reading and Writing Images :

You can use the MATLAB imread function to read image data from files. imread can read these graphics file formats:

- BMP
- HDF

- JPEG
- PCX
- TIFF
- XWD

To write image data from MATLAB to a file, use
the imwrite function. imwrite can write the same file formats that imread reads.

In addition, you can use the imfinfo function to return information about the image data in a file.

3.2.5 Converting Images to Other Types :

For certain operations, it is helpful to convert an image to a different image type. For example, if you want to filter a color image that is stored as an indexed image, you should first convert it to RGB format. When you apply the filter to the RGB image, MATLAB filters the intensity values in the image, as is appropriate. If you attempt to filter the indexed image, MATLAB simply applies the filter to the indices in the indexed image matrix, and the results may not be meaningful.

The Image Processing Toolbox provides several functions that enable you to convert any image to another image type. These functions have mnemonic names; for example, ind2gray converts an indexed image to a grayscale intensity format.

Note that when you convert an image from one format to another, the resulting image may look different from the original. For example, if you convert a color indexed image to an intensity image, the resulting image is grayscale, not color.

The table below summarizes these image conversion functions:

Function	**Purpose**
Dither	Create a binary image from a grayscale intensity image by dithering; create an indexed image from an RGB image by dithering
gray2ind	Create an indexed image from a grayscale intensity image
grayslice	Create an indexed image from a grayscale intensity image by thresholding
im2bw	Create a binary image from an intensity image, indexed image, or RGB image, based on a luminance threshold
ind2gray	Create a grayscale intensity image from an indexed image
ind2rgb	Create an RGB image from an indexed image
mat2gray	Create a grayscale intensity image from data in a matrix, by scaling the data
rgb2gray	Create a grayscale intensity image from an RGB image

Table 3.2:Converting image to other types

3.2.6 Color Space Conversions:

The Image Processing Toolbox represents colors as RGB values, either directly (in an RGB image) or indirectly (in an indexed image). However, there are other methods for representing colors. For example, a color can be represented by its hue, saturation, and value components (HSV). Different methods for representing colors are called color spaces. The toolbox provides a set of routines for converting between RGB and other color spaces. The image processing functions themselves assume all color data is RGB, but you can process an image that uses a different color space by first converting it to RGB, and then converting the processed image back to the original color space.

3.2.7. Coordinate Systems:

Locations in an image can be expressed in various coordinate systems, depending on context. This section discusses the two main coordinate systems used in the Image Processing Toolbox, and the relationship between them. These systems are:

- The pixel coordinate system
- The spatial coordinate system

Pixel Coordinates

Generally, the most convenient method for expressing locations in an image is to use pixel coordinates. In this coordinate system, the image is treated as a grid of discrete elements, ordered from top to bottom and left to right. For example:

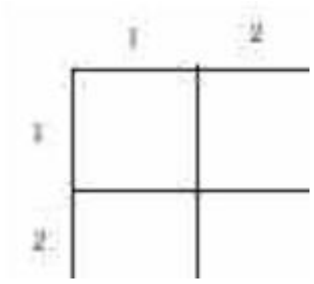


Fig .3.9: Pixel coordinates

For pixel coordinates, the first component r (the row) increases downward, while the second component c (the column) increases to the right. Pixel coordinates are integer values and range between 1 and the length of the row or column.

There is a one-to-one correspondence between pixel coordinates and the coordinates MATLAB uses for matrix subscripting. This correspondence makes the relationship between an image's data matrix and the way the image displays easy to understand. For example, the data for the pixel in the fifth row, second column is stored in the matrix element (5, 2).

Spatial Coordinates

In the pixel coordinate system, a pixel is treated as a discrete unit, uniquely identified by a single coordinate pair, such as (5,2). From this perspective, a location such as (5.3,2.2) is not meaningful.

At times, however, it is useful to think of a pixel as a square patch, having area. From this perspective, a location such as (5.3,2.2) is meaningful, and is distinct from (5,2). In this spatial coordinate system, locations in an image are positions on a plane, and they are described in terms of x and y.

This figure illustrates the spatial coordinate system used for images. Notice that y increases downward:

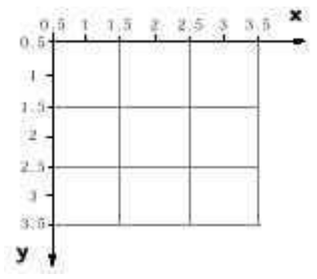
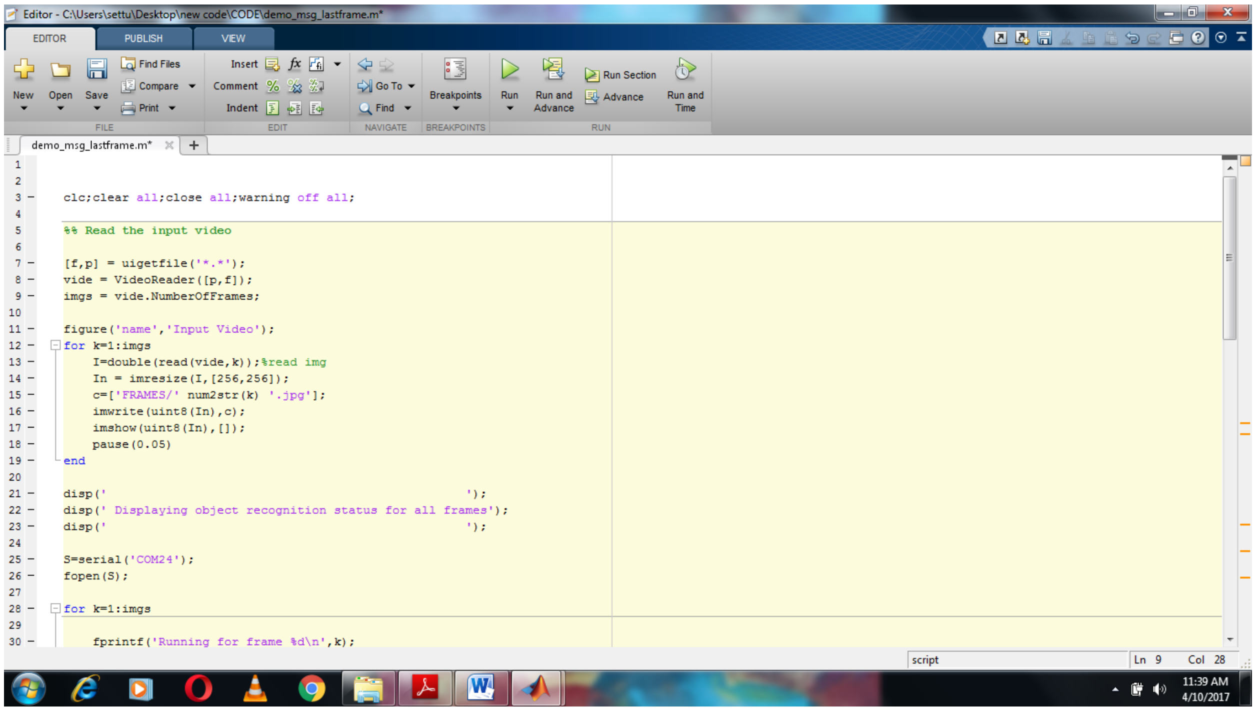


Fig.3.10: Spatial Coordinates

This spatial coordinate system corresponds quite closely to the pixel coordinate system in many ways. For example, the spatial coordinates of the center point of any pixel are identical to the pixel coordinates for that pixel.

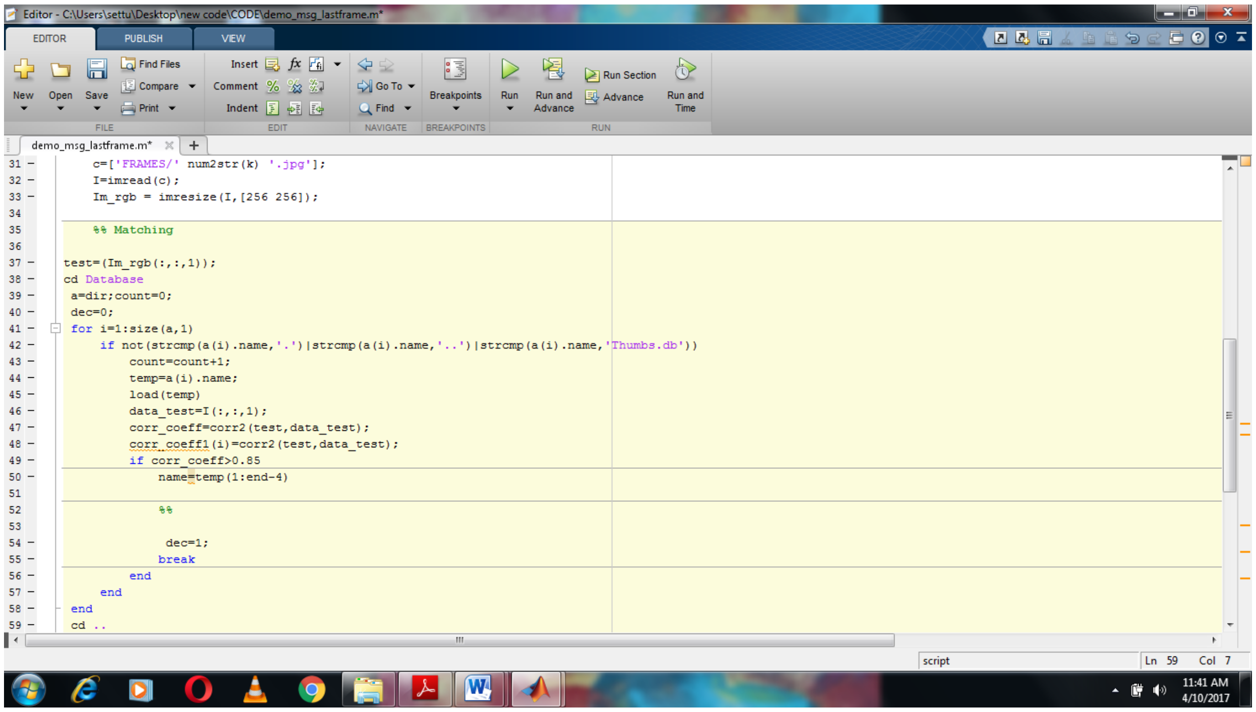
There are some important differences, however. In pixel coordinates, the upper-left corner of an image is (1, 1), while in spatial coordinates, this location by default is (0.5, 0.5). This difference is due to the pixel coordinate system being discrete, while the spatial coordinate system is continuous. Also, the upper-left corner is always (1, 1) in pixel coordinates, but you can specify a no default origin for the spatial coordinate system.

3.3. MATLAB CODE:



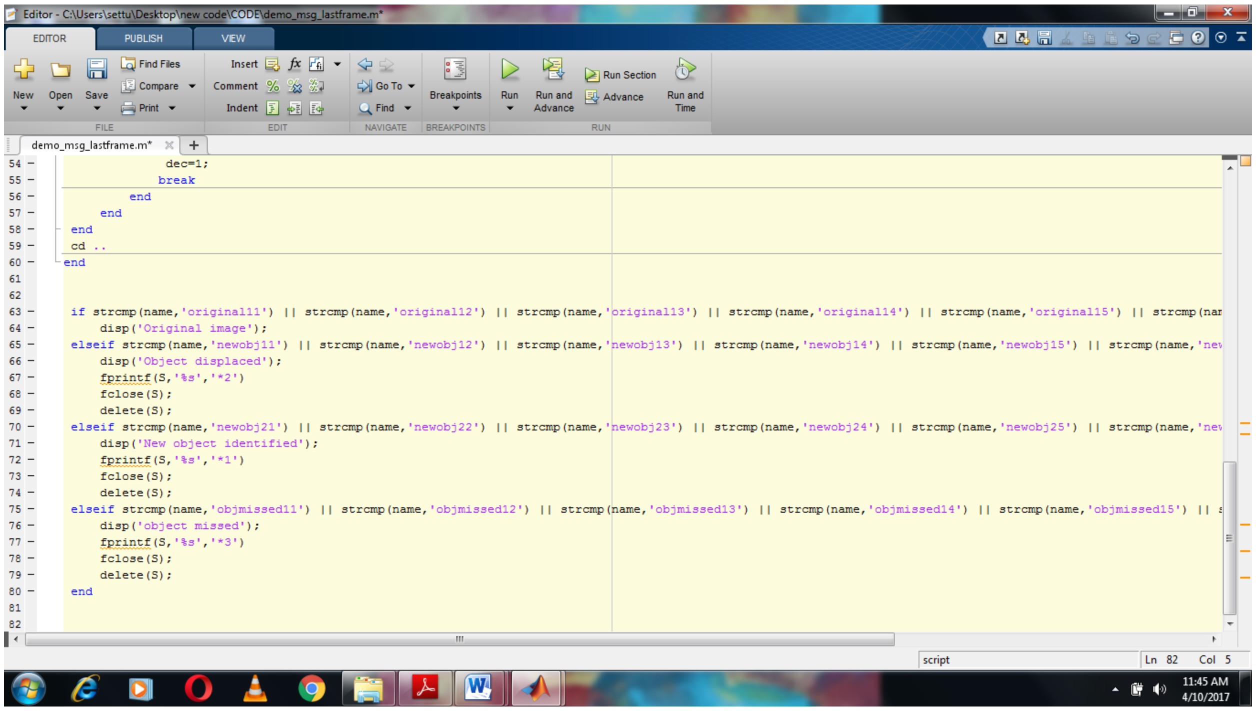
```matlab


clc;clear all;close all;warning off all;

%% Read the input video

[f,p] = uigetfile('*.*');
vide = VideoReader([p,f]);
imgs = vide.NumberOfFrames;

figure('name','Input Video');
for k=1:imgs
    I=double(read(vide,k));%read img
    In = imresize(I,[256,256]);
    c=['FRAMES/' num2str(k) '.jpg'];
    imwrite(uint8(In),c);
    imshow(uint8(In),[]);
    pause(0.05)
end

disp('                                              ');
disp(' Displaying object recognition status for all frames');
disp('                                              ');

S=serial('COM24');
fopen(S);

for k=1:imgs

    fprintf('Running for frame %d\n',k);
```

Fig.3.10(a)


Editor - C:\Users\settu\Desktop\new code\CODE\demo_msg_lastframe.m*
demo_msg_lastframe.m*
script
Ln 59 Col 7

```
    c=['FRAMES/' num2str(k) '.jpg'];
    I=imread(c);
    Im_rgb = imresize(I,[256 256]);

    %% Matching

test=(Im_rgb(:,:,1));
cd Database
 a=dir;count=0;
 dec=0;
 for i=1:size(a,1)
     if not(strcmp(a(i).name,'.')|strcmp(a(i).name,'..')|strcmp(a(i).name,'Thumbs.db'))
         count=count+1;
         temp=a(i).name;
         load(temp)
         data_test=I(:,:,1);
         corr_coeff=corr2(test,data_test);
         corr_coeff1(i)=corr2(test,data_test);
         if corr_coeff>0.85
             name=temp(1:end-4)

             %%

              dec=1;
             break
         end
     end
 end
 cd ..
```

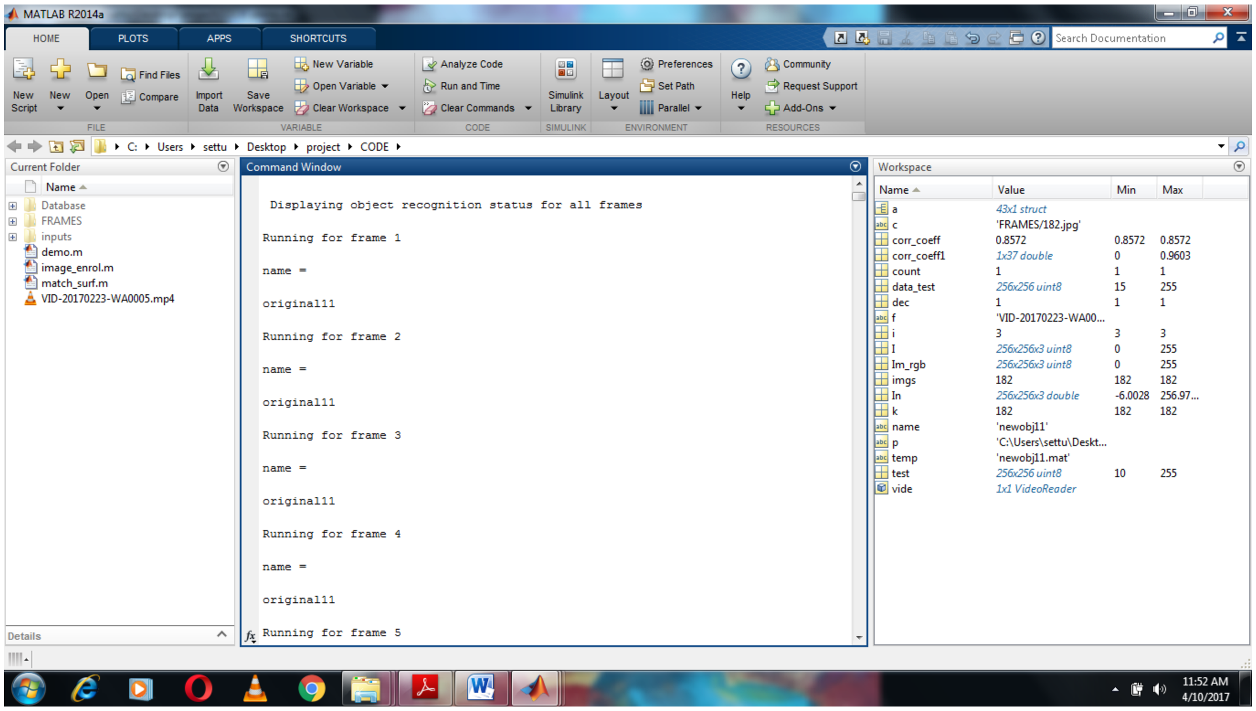
Fig.3.10 (a)


Editor - C:\Users\settu\Desktop\new code\CODE\demo_msg_lastframe.m*
demo_msg_lastframe.m*
script
Ln 82 Col 5

```
              dec=1;
             break
         end
     end
 end
 cd ..
end


 if strcmp(name,'original11') || strcmp(name,'original12') || strcmp(name,'original13') || strcmp(name,'original14') || strcmp(name,'original15') || strcmp(nar
     disp('Original image');
 elseif strcmp(name,'newobj11') || strcmp(name,'newobj12') || strcmp(name,'newobj13') || strcmp(name,'newobj14') || strcmp(name,'newobj15') || strcmp(name,'new
     disp('Object displaced');
     fprintf(S,'%s','*2')
     fclose(S);
     delete(S);
 elseif strcmp(name,'newobj21') || strcmp(name,'newobj22') || strcmp(name,'newobj23') || strcmp(name,'newobj24') || strcmp(name,'newobj25') || strcmp(name,'new
     disp('New object identified');
     fprintf(S,'%s','*1')
     fclose(S);
     delete(S);
 elseif strcmp(name,'objmissed11') || strcmp(name,'objmissed12') || strcmp(name,'objmissed13') || strcmp(name,'objmissed14') || strcmp(name,'objmissed15') || s
     disp('object missed');
     fprintf(S,'%s','*3')
     fclose(S);
     delete(S);
 end

```

Fig.3.10 (b)

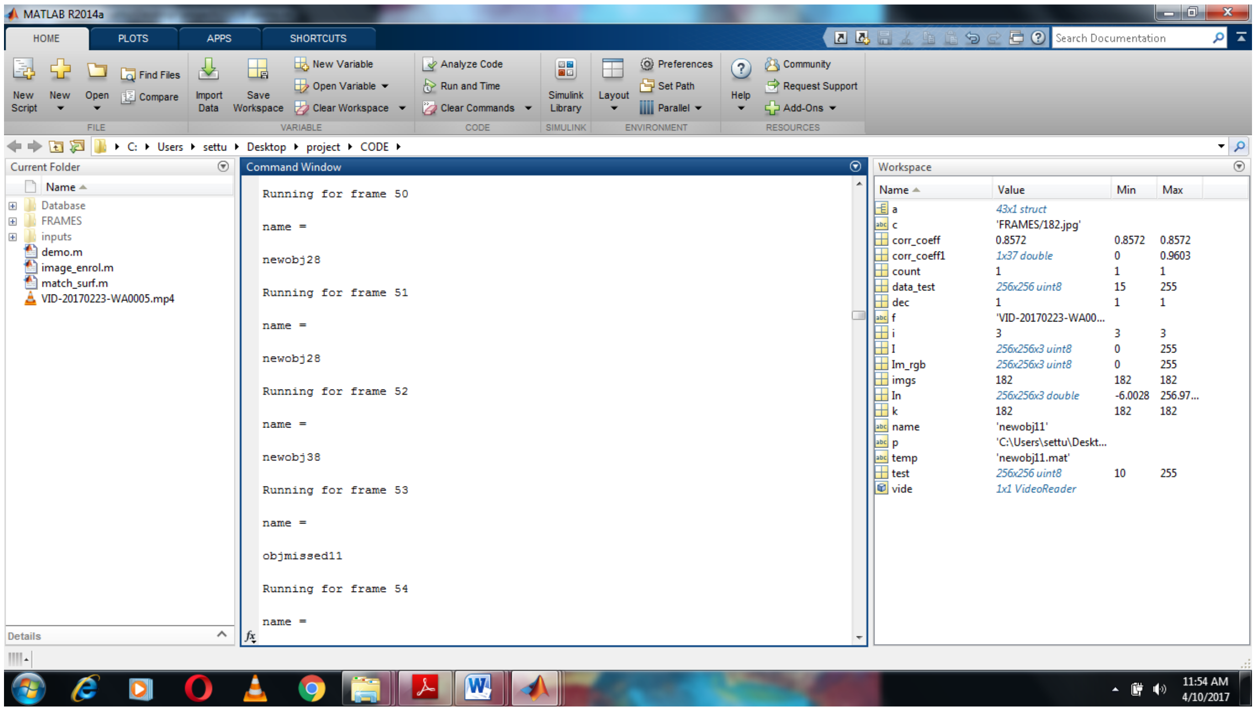
3.4. PROGRAM OUTPUT:



```
 Displaying object recognition status for all frames

Running for frame 1

name =

original11

Running for frame 2

name =

original11

Running for frame 3

name =

original11

Running for frame 4

name =

original11

Running for frame 5
```

Name	Value	Min	Max
a	43x1 struct		
c	'FRAMES/182.jpg'		
corr_coeff	0.8572	0.8572	0.8572
corr_coeff1	1x37 double	0	0.9603
count	1	1	1
data_test	256x256 uint8	15	255
dec	1	1	1
f	'VID-20170223-WA00...		
i	3	3	3
I	256x256x3 uint8	0	255
Im_rgb	256x256x3 uint8	0	255
imgs	182	182	182
In	256x256x3 double	-6.0028	256.97...
k	182	182	182
name	'newobj11'		
p	'C:\Users\settu\Deskt...		
temp	'newobj11.mat'		
test	256x256 uint8	10	255
vide	1x1 VideoReader		

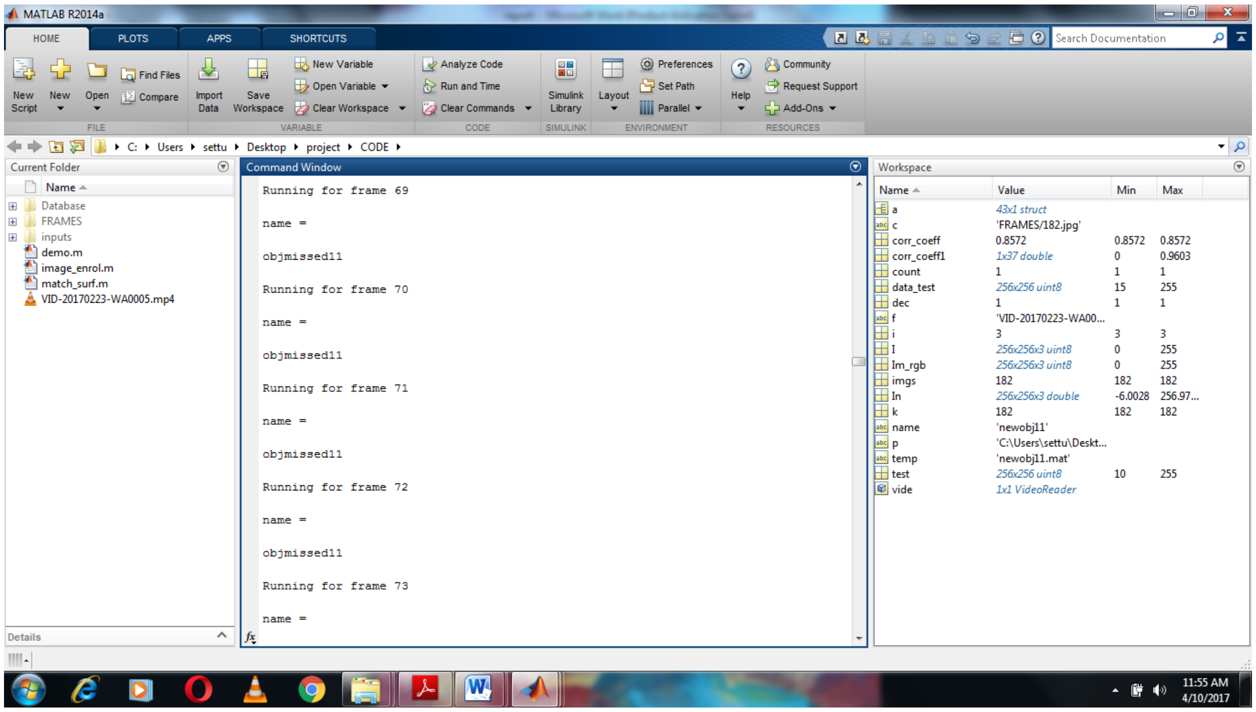
Fig.3.11 (a)



```
Running for frame 50

name =

newobj28

Running for frame 51

name =

newobj28

Running for frame 52

name =

newobj38

Running for frame 53

name =

objmissed11

Running for frame 54

name =
```

Name	Value	Min	Max
a	43x1 struct		
c	'FRAMES/182.jpg'		
corr_coeff	0.8572	0.8572	0.8572
corr_coeff1	1x37 double	0	0.9603
count	1	1	1
data_test	256x256 uint8	15	255
dec	1	1	1
f	'VID-20170223-WA00...		
i	3	3	3
I	256x256x3 uint8	0	255
Im_rgb	256x256x3 uint8	0	255
imgs	182	182	182
In	256x256x3 double	-6.0028	256.97...
k	182	182	182
name	'newobj11'		
p	'C:\Users\settu\Deskt...		
temp	'newobj11.mat'		
test	256x256 uint8	10	255
vide	1x1 VideoReader		

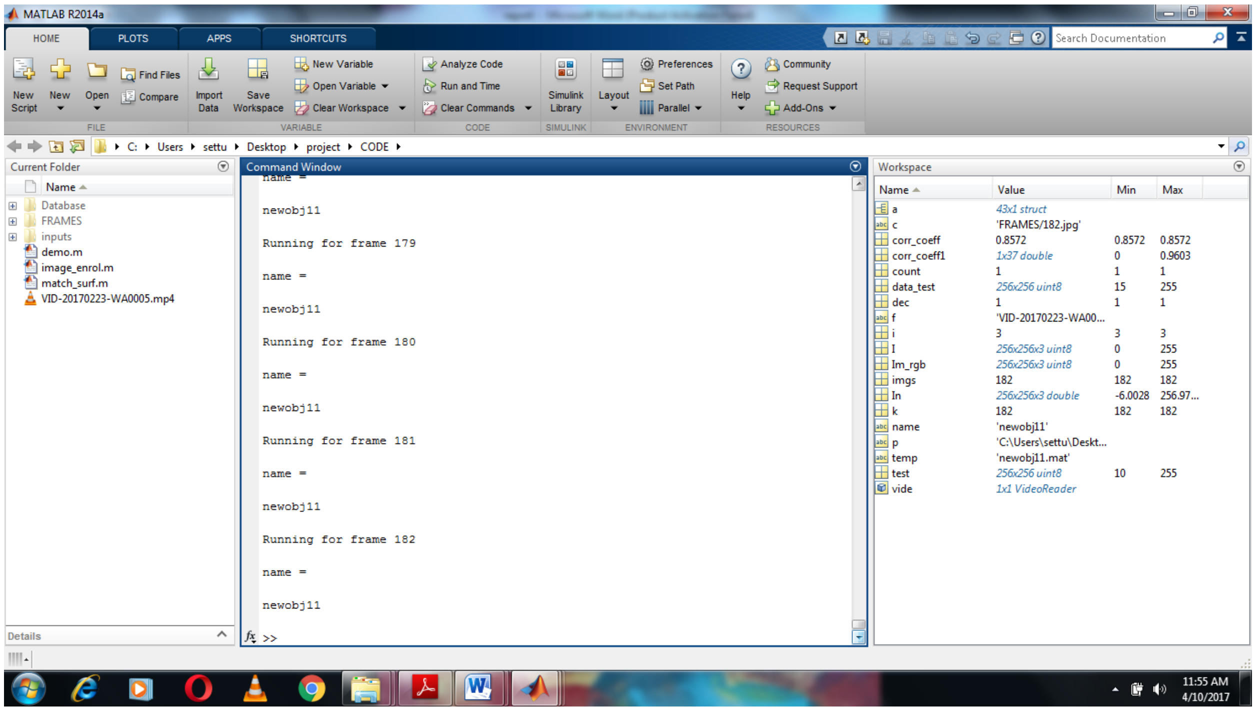
Fig.3.11 (b)



```
Running for frame 69

name =

objmissed11

Running for frame 70

name =

objmissed11

Running for frame 71

name =

objmissed11

Running for frame 72

name =

objmissed11

Running for frame 73

name =
```

Name	Value	Min	Max
a	43x1 struct		
c	'FRAMES/182.jpg'		
corr_coeff	0.8572	0.8572	0.8572
corr_coeff1	1x37 double	0	0.9603
count	1	1	1
data_test	256x256 uint8	15	255
dec	1	1	1
f	'VID-20170223-WA00...		
i	3	3	3
I	256x256x3 uint8	0	255
Im_rgb	256x256x3 uint8	0	255
imgs	182	182	182
In	256x256x3 double	-6.0028	256.97...
k	182	182	182
name	'newobj11'		
p	'C:\Users\settu\Deskt...		
temp	'newobj11.mat'		
test	256x256 uint8	10	255
vide	1x1 VideoReader		

Fig.3.11 (c)

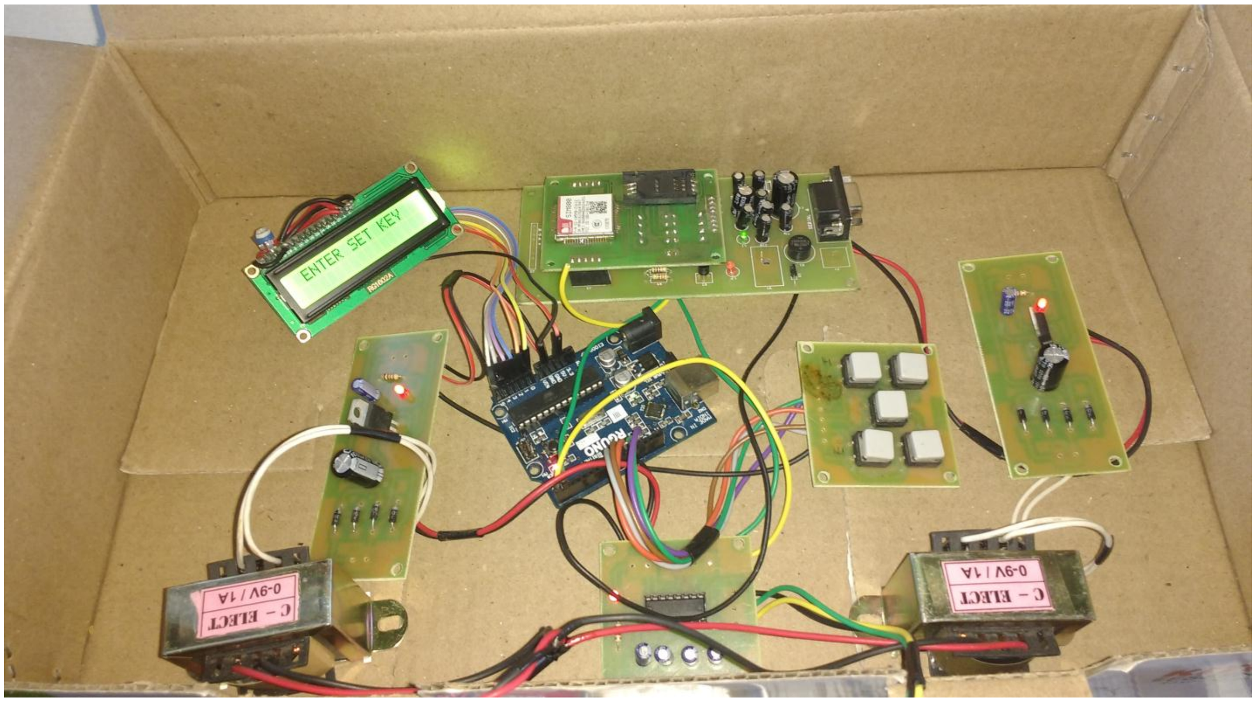


```
name =

newobj11

Running for frame 179

name =

newobj11

Running for frame 180

name =

newobj11

Running for frame 181

name =

newobj11

Running for frame 182

name =

newobj11

>>
```

Name	Value	Min	Max
a	43x1 struct		
c	'FRAMES/182.jpg'		
corr_coeff	0.8572	0.8572	0.8572
corr_coeff1	1x37 double	0	0.9603
count	1	1	1
data_test	256x256 uint8	15	255
dec	1	1	1
f	'VID-20170223-WA00...		
i	3	3	3
I	256x256x3 uint8	0	255
Im_rgb	256x256x3 uint8	0	255
imgs	182	182	182
In	256x256x3 double	-6.0028	256.97...
k	182	182	182
name	'newobj11'		
p	'C:\Users\settu\Deskt...		
temp	'newobj11.mat'		
test	256x256 uint8	10	255
vide	1x1 VideoReader		

Fig.3.11 (d)

4.WORKING MODEL:



5. CONCLUSION:

Thus, a system for video processing has been proposed in this project. The proposed approach has shown that combining hardware and software solutions makes possible high accuracy and perpetual work in high accuracy . The key advantage is that this can avoid the need of manpower for safety control.

5.1 FUTURE APPLICATION

The project is aimed at working with several scenarios, including surveillance, target acquisition ,situation awareness,chemical,biological,radiological and nuclear early warning.To support these capabilities,it is now necessary to develop new architectures and design concepts that offer multimodal sensing without sacrificing the attractive low size,weight and power capability.

REFERENCES:

1. Fast 2D convolutions and cross-correlations using scalable architechtures- Ganggang Dong ; Gangyokuang ; Na Wang ;Wei wang

2. ViBe: A Universal background subtraction algorithm for video sequences- Olivier barnich;Marc Van Droogenbroeck(Publication year:2011)

3. Image denoising with edge preserving and segmentation based on mask NHA- Fumitaka Hosotani;Yuyainuzuka.