# ACKNOWLEDGMENT

We are pleased to present **“Information hidding in image enhancement using steganography analysis”** project and take this opportunity to express our profound gratitude to all those people who helped us in the completion of this project.

We thank our college for providing us with excellent facilities that helped us to complete and present this project. We would also like to thank the staff members and lab assistants for permitting us to use computers in the lab as and when required.

We would like to extend my deepest gratitude of our guide **Mr.Naveen Chandra Sir** Dept. of MCA , AVANTHI PG COLLEGwithout whom this project would have been incomplete. Her understanding, encouraging and personal guidance have provided a good basis for this project report. We would also like to thank her for providing us with all the proper facilities and support as the project guide. We would like to thank her for support, patience, and faith in our capabilities and for giving us flexibility in terms of working and reporting schedules.

**INDEX**

ABSTRACT

LIST OF FIG

[CHAPTER-1 INTRODUCTION 1](#_TOC_250045)

[CHAPTER-2 LITERATURE SURVEY 2](#_TOC_250044)

* 1. [EFFICIENT REVERSIBLE DATA HIDING BASED ON MULTIPLE HISTOGRAMS MODIFICATION **3**](#_TOC_250043)
  2. REMOVAL OF HIGHDENSITY SALT & PEPPER NOISE THROUGH SUPER MEAN FILTER FOR **5**

NATURAL IMAGES

* 1. A DATABASE OF HUMAN SEGMENTED NATURAL IMAGES AND ITS APPLICATIONS **8**

TO EVALUATING SEGMENTATION ALGORITHMS AND MEASURING ECOLOGICAL STATISTICS

* 1. IMAGE QUALITY ASSESSMENT FROM ERROR VISIBILITY TO STRUCTURAL **9**

SIMILARITY

CHAPTER 3 INTRODUCTION TO IMAGE PROCESSING 11

* 1. [IMAGE 11](#_TOC_250042)
  2. [IMAGE FILE SIZES **13**](#_TOC_250041)
  3. [IMAGE FILE FORMATS **13**](#_TOC_250040)
     1. [RASTER FORMATS **14**](#_TOC_250039)
     2. [VECTOR FORMATS **16**](#_TOC_250038)
  4. [IMAGE PROCESSING **16**](#_TOC_250037)
  5. [FUNDAMENTAL STEPS IN DIGITAL IMAGE PROCESSING **17**](#_TOC_250036)
     1. [IMAGE ACQUISITION **17**](#_TOC_250035)
     2. [IMAGE ENHANCEMENT **18**](#_TOC_250034)
     3. [IMAGE RESTORATION **19**](#_TOC_250033)
     4. [COLOR IMAGE PROCESSING **20**](#_TOC_250032)
     5. [WAVELETS AND MULTIRESOLUTION PROCESSING **20**](#_TOC_250031)
     6. [COMPRESSION **21**](#_TOC_250030)
     7. [MORPHOLOGICAL PROCESSING **21**](#_TOC_250029)
     8. [SEGMENTATION **22**](#_TOC_250028)
     9. [REPRESENTATION AND DESCRIPTION **22**](#_TOC_250027)
     10. [OBJECT RECOGNITION **23**](#_TOC_250026)
     11. [KNOWLEDGEBASE **23**](#_TOC_250025)
  6. [COMPONENTS OF AN IMAGE PROCESSING SYSTEM **24**](#_TOC_250024)

CHAPTER-4 DIGITAL IMAGE PROCESSING 40

CHAPT ER-5 INFORMATION HIDDING IN IMAGE ENHANCEMENT 50

**USING STEGANOGRAPHY TECHNIQUE**

5.1[REVISITING MEDIAN FILTERING](#_TOC_250005)  **50**

5.2DATA EMBEDDINNG 50

5.3[DATA EXTRACTION 51](#_TOC_250004)

[CHAPTER 6 SIMULATION RESULTS](#_TOC_250002)  57

[CHAPTER 7 CONCLUSION AND FUTURE SCOPE 60](#_TOC_250001)

**APPENDIX**

**1. SOFTWARE INTRODUCTION**  62

1.1[INTRODUCTION TO MATLAB 62](#_TOC_250023)

1.2[THE MATLAB SYSTEM](#_TOC_250022)  **64**

1.3[GRAPHICAL USER INTERFACE(GUI) 65](#_TOC_250021)

1.4[GETTING STARTED](#_TOC_250020)  **67**

1.4.1INTRODUCTION 67

1.4.2DEVELOPMENT ENVIRONMENT 67

1.4.3MANIPULATING MATRICES 67

1.4.4GRAPHICS 67

1.4.5PROGRAMMING WITH MATLAB 67

1.5DEVELOPMENT ENVIRONMENT 67

1.5.1INTRODUCTION 67

1.5.2STARTING MATLAB 68

1.5.3[QUITTING MATLAB 6](#_TOC_250019)**8**

1.5.4[MATLAB DESKTOP 6](#_TOC_250018)**8**

1.5.5[DESKTOP TOOLS 6](#_TOC_250017)**9**

1.6[MANIPULATING MATRICES 7](#_TOC_250016)**2**

1.6.1[ENTERING MATRICES 7](#_TOC_250015)**2**

1.6.2[EXPRESSIONS 7](#_TOC_250014)**3**

1.6.3[OPERATORS 7](#_TOC_250013)**4**

1.6.4[FUNCTIONS 7](#_TOC_250012)**5**

1.7[GUI](#_TOC_250011)  76

1.7.1[CREATING GUIS WITH GUIDE 76](#_TOC_250010)

1.7.2[GUI DEVELOPMENT ENVIRONMENT 77](#_TOC_250009)

1.7.3FEATURES OF THE GUIDE GENERATED APPLICATION M-FILE 78

1.7.4[BEGINNING THE IMPLEMENTATION PROCESS 79](#_TOC_250008)

1.7.5 USER INTERFACE CONTROL 79

[2. SOURCE CODE 96](#_TOC_250003)

**LIST OF FIGURES**

Fig-1 General image 11

Fig- 3.1 Image pixel 12

Fig-3.2 Transparency image 12

Fig-3.3 Resolution image 13

Fig-3.5 Image fundamental 17

Fig-3.5.2 Digital camera roll 18

Fig-3.5.4 Image restoration 19

Fig-3.5.5 Color and gray scale image 20

Fig-3.5.6 RGB Histogram image 20

Fig-3.5.7 Blur to deblur image 21

Fig-3.5.8 Image segmentation 22

Fig-3.6 Component of image processing 24

Fig-6.1 Reference image 57

Fig-6.2 Secret image 57

Fig-6.3 Encrypted image 58

Fig-6.4 First decoded image 58

Fig-6.5 Second decoded image 59

# Abstract

This project proposes an information hiding method to embed data while executing image enhancement steps. The 2D Median Filter is adapted and re-engineered to demonstrate the feasibility of this concept. In particular, the filtering embedding steps are performed for each pixel in a sliding window manner. Pixels enclosed within the predefined window (neighborhood) are gathered, linearized and sorted. Then, the linearized pixels are divided into partitions, in which each partition is assigned to represent a certain sequence of bits. The performance of the proposed method is evaluated by using the BSD300 dataset for various settings. The embedding capacity, image quality, data extraction error rate are reported and analyzed. Besides, the robustness of the proposed method against brute force attack is also discussed. In the best case scenario, when the window size is 7 × 7, ∼ 0.97 bpp is achieved with acceptable image quality while having ∼ 3.5% data extraction error rate.

Index Terms— Median filter, data embedding, hiding capacity, image quality, data extraction error rate

# Chapter 1 INTRODUCTION

Image enhancement, as its name implies, improves the quality of image by means of emphasizing the desired details or removing/suppressing the undesired noisy/irrelevant information. Image enhancement plays a crucial role in digital image processing, and it has become one of the processes that takes place in any smart device after an image is captured. Some commonly deployed image enhancement techniques include contrast enhancement, image sharpening, smoothing, noise removal, to name a few. Each enhancement technique can either act as a standalone process to enhance the appearance of an image, or utilized as a pre-processing step before further analysis is carried out. In fact, image enhancement (commonly known as filter) is often utilized to enhance image quality before it is shared online on social media or communicated for consumption by a wider audience. However, image enhancement is particularly important when the received original/source image (e.g., from field reporter, amateur photographer) contains noise. While the noisy image is enhanced, the processes involved are not published or shared. One can also imagine the need to share specific tone-mapping operators as well as the parameters used when shrinking the dynamic range of a noisy high dynamic range image into the standard dynamic range. To accommodate this additional data, information hiding appears to be one of the commonly implemented solutions. There are many conventional information hiding methods, including Least Significant Bit (LSB) insertion, Histogram Shifting (HS) and Difference Expansion (DE), to name a few. These techniques are studied extensively and many of their variants are proposed [1, 2, 3] to achieve a balance trade-off among embedding capacity, image quality, robustness against attacks, etc. However, these techniques are not widely adopted into the usual operations performed by the users, and often they are implemented as an additional step after the image is processed. In most cases, the user has to explicitly install or develop the data embedding algorithm to enable data embedding into the image of interest. Therefore, in this paper, we design an information hiding method as part of the image enhancement process. In other words, data can be inserted into the image while executing the image enhancement steps. As a proof of concept, the proposed method is demonstrated by using the Median Filter.

# CHAPTER 2 LITERATURE SURVEY

There are many conventional information hiding methods, including Least Significant Bit (LSB) insertion, Histogram Shifting (HS) and Difference Expansion (DE), to name a few. These techniques are studied extensively and many of their variants are proposed [1, 2, 3] to achieve a balance trade-off among embedding capacity, image quality, robustness against attacks, etc. However, these techniques are not widely adopted into the usual operations performed by the users, and often they are implemented as an additional step after the image is processed. In most cases, the user has to explicitly install or develop the data embedding algorithm to enable data embedding into the image of interest. Therefore, in this paper, we design an information hiding method as part of the image enhancement process. In other words, data can be inserted into the image while executing the image enhancement steps. As a proof of concept, the proposed method is demonstrated by using the Median Filter.

There are many conventional information hiding methods, including Least Significant Bit (LSB) insertion, Histogram Shifting (HS) and Difference Expansion (DE), to name a few. These techniques are studied extensively and many of their variants are proposed [1, 2, 3] to achieve a balance trade-off among embedding capacity, image quality, robustness against attacks, etc. However, these techniques are not widely adopted into the usual operations performed by the users, and often they are implemented as an additional step after the image is processed. In most cases, the user has to explicitly install or develop the data embedding algorithm to enable data embedding into the image of interest. Therefore, in this paper, we design an information hiding method as part of the image enhancement process. In other words, data can be inserted into the image while executing the image enhancement steps. As a proof of concept, the proposed method is demonstrated by using the Median Filter.

## Efficient Reversible Data Hiding Based on Multiple Histograms Modification:

Reversible data hiding (RDH) aims to embed secret message into a cover image by slightly modifying its pixels, and more importantly, the original image as well as the embedded message should be completely restored from the marked image. In the last decade, RDH has received much attention from the information hiding community and this technique has also been applied in some applications, such as image authentication, medical image processing multimedia archive management, image trans-coding, and data coloring in the cloud, etc. In general, RDH is a fragile technique and the marked image cannot undergo any degradation. In this light, a RDH method is usually evaluated by its capacity-distortion performance, i.e., for a given embedding capacity (EC), one expects to minimize the embedding distortion measured by PSNR of the marked image versus the original one. Early RDH methods are mainly based on lossless compression. The idea behind these methods is to losslessly compress a feature set of cover image and utilize the saved space for reversible embedding.

In, Celik et al. proposed a generalized least significant bit (LSB) compression method to improve the compression efficiency by using unaltered bit-planes as side information. However, the lossless compression-based methods cannot yield satisfactory performance, since the correlation within a bit-plane is too weak to provide a high EC. As EC increases, one needs to compress more bit-planes, thus the distortion increases dramatically.

Later on, more efficient RDH methods based on histogram modification and expansion technique have been devised. The histogram-modification-based method is firstly proposed by Ni et al. This method focuses on high visual quality with quite limited EC, in which the peak point of image histogram is utilized for data embedding. In this method, each pixel value is modified at most by 1, and thus the marked image quality is well guaranteed. Ni et al.’s method is improved by Lee et al. by using the histogram of difference image. The spatial correlation of natural images is exploited in by considering the difference of adjacent pixels. Thus, a regular- shaped histogram is utilized in Lee et al.’s method. This histogram is centered at origin and has rapid two-sided decay which is more suitable for RDH. The expansion technique is firstly proposed by Tian .

This method is performed on pixel pairs, and one data bit is embedded into each selected pixel pair by expanding its difference. Compared with the lossless-compression based RDH, Tian’s difference expansion (DE) based method can provide a higher EC with an improved PSNR. The DE approach has attracted considerable attention and it makes an important progress in RDH. Afterwards, the expansion technique has been widely investigated and developed, mainly in the aspects of integer-to-integer such as image authentication, medical image processing, multimedia archive management, image trans-coding [9], and data coloring in the cloud, etc. In general, RDH is a fragile technique and the marked image cannot undergo any degradation. In this light, a RDH method is usually evaluated by its capacity-distortion performance, i.e., for a given embedding capacity (EC), one expects to minimize the embedding distortion measured by PSNR of the marked image versus the original one.

Early RDH methods are mainly based on lossless compression. The idea behind these methods is to losslessly compress a feature set of cover image and utilize the saved space for reversible embedding. In, Fridrich et al. proposed to compress a proper bit-plane with the minimum redundancy.

In, Celik et al. proposed a generalized least significant bit (LSB) compression method to improve the compression efficiency by using unaltered bit-planes as side information. However, the lossless compression-based methods cannot yield satisfactory performance, since the correlation within a bit-plane is too weak to provide a high EC. As EC increases, one needs to compress more bit-planes, thus the distortion increases dramatically. Later on, more efficient RDH methods based on histogram modification and expansion technique have been devised. The histogram-modification-based method is firstly proposed by Ni et al.

This method focuses on high visual quality with quite limited EC, in which the peak point of image histogram is utilized for data embedding. In this method, each pixel value is modified at most by 1, and thus the marked image quality is well guaranteed. Ni et al.’s method is improved by Lee et al. by using the histogram of difference image. The spatial correlation of natural images is exploited in by considering the difference of adjacent pixels. Thus, a regular-shaped histogram is utilized in Lee et al.’s method.

This histogram is centered at origin and has rapid two-sided decay which is more suitable for RDH. The expansion technique is firstly proposed by Tian. This method is performed on pixel pairs, and one data bit is embedded into each selected pixel pair by expanding its difference. Compared with the lossless-compression based RDH, Tian’s difference expansion (DE) based method can provide a higher EC with an improved PSNR. The DE approach has attracted considerable attention and it makes an important progress in RDH. Afterwards, the expansion technique has been widely investigated and developed, mainly in the aspects of integer-to-integer transformation, location map reduction, and prediction-error expansion (PEE). Besides the histogram modification and the expansion technique, the analysis about theoretical capacity limit subjected to admissible distortion has also been studied in some recent works .

Nowadays, the most effective and extensively exploited RDH technique is the PEE technique which is firstly proposed by Thodi and Rodriguez. Instead of the difference value in DE, the prediction-error is utilized in PEE for expansion embedding. Thus, unlike DE where only the correlation of two adjacent pixels is considered, the local correlation of a larger neighborhood is exploited in PEE. As a result, compared with DE, better performance can be derived by PEE.

Following Thodi and Rodriguez’s work, many RDH techniques related to PEE have been proposed in recent years, for example, double-layered embedding adaptive embedding, context modification, optimal expansion bins selection and two-dimensional histogram modification etc. On the other hand, some PEE-based method exploit advanced prediction techniques to generate a more sharply distributed prediction-error histogram (PEH), and this is also helpful for enhancing the embedding performance. Notice that, most previous PEE-based methods are based on one- or two-dimensional PEH modification.

## Removal of High Density Salt & Pepper Noise Through Super Mean Filter for Natural Images:

The transmission and acquisition of digital images through sensors or communication channels are often interfered by impulse noise. It is important to eliminate the impulse noise from the image before some subsequent processing such as edge detection, image segmentation

and object reorganization. During last one decade various algorithms have been proposed for removal of impulse noise. The salt & pepper noise is a special type of impulse noise in which some portion of image pixel values are replaced by either minimum or maximum pixel values.

The main objective of salt & pepper noise removal is that it removes the noise from the image by preserving the other image details. The linear filters used for impulse noise removal works much better for low noise density as compared to high noise density. For high noise density, the output images are blurred and edges are not preserved accurately by the linear filters. Therefore the non- linear filters have been used to provide better filtering performance in terms of impulse noise removal and preservation of other details of the images. In this context various non-linear filters have been proposed by various researchers for removing salt & pepper noise. During last one decade, median based filters have attracted very much attention due to their simplicity and information preservation capabilities.

The main drawback of the median filter is that it also modifies non noisy pixels thus removing some fine details of the image. Therefore it is only suitable for very low level noise density. At high noise density it shows the blurring for the larger template sizes and not able to suppress the noise completely for smaller template sizes.

Therefore, contemporary switching filters split the denoising process in two steps. First one is detection of noise and second one is the replacement of the noisy pixel value with estimated median value. These are weighted median filter, adaptive impulse detection using centre weighted median, rank order filtering algorithm. The performance of the centre weighted median filter (CWMF), standard adaptive median filter (AMF) and progressive switching median filter (PSMF) algorithms are good at the lower noise density due to less numbers of the noisy pixels which are replaced with the median values. But at high noise density, there are a large number of the noisy pixels which are need to be replaced.

Therefore the size of the template will be larger to provide the better performance; however, the values of the noisy pixel and its replacement as median values are less correlated which results in information loss. The main disadvantage of the switching median filter and decision based filter is that it is based on the predefined threshold, due to this some details and edges are also removed particularly in case of high noise density. Ideally the filtering should be

applied only to the values of the noisy pixel while keeping the values of the noise free pixels. In order to overcome the disadvantages of these mentioned filtering techniques a two stage algorithm has been proposed.

In this algorithm an adaptive median filter is used in first stage to classify the values of the noisy and noise free pixels and detail preserving regularization technique is used in second stage to preserve the details and edges as much as possible. Due to large template size, processing time is too large and more complexity is involved in its implementation. In order to avoid this drawback, open-close sequence filter (OCSF) has been proposed. This algorithm is based on mathematical morphology, which is suitable only for high density impulse noise (noise density ranging from 50% to 80%). The main drawback of this algorithm is that its performance is not good in very low noise density as well as in very high noise density.

To overcome this drawback, decision based algorithm (DBA) is proposed. In this algorithm, image is denoised by using a 3X3 window. The image is denoised for pixel value ‘0’ or ‘255’ else it is left unchanged. At high noise density the median value will be ‘0’ or ‘255’ which is noisy. In such case, neighboring pixel is used for replacement. This repeated replacement of neighboring pixel produces streaking effect. In order to avoid this drawback, decision based un symmetric trimmed median filter (DBUTMF) is proposed.

At high noise densities, if the selected window contains all ‘0’s or ‘255’s or both then, trimmed median value cannot be obtained. To avoid the major drawback of decision based un symmetric trimmed median filter, modified decision based un symmetric trimmed median filter (MDBUTMF) is proposed. In this filter the noisy image is denoised by using 3X3 window elements which are arranged in increasing or decreasing order. Then the pixel values ‘0’s and ‘255’s in the image (i.e. the pixel values responsible for the salt & pepper noise) are removed from the image.

Then the median value of the remaining pixels is taken. This median value is used to replace the pixel. This algorithm does not give better results at high noise density ranging from 70% to 95%. Therefore to avoid the drawback of modified decision based un symmetric trimmed median filter, a new & efficient algorithm is proposed in this paper. This is suitable for elimination of high density impulse noise ranging from 60% to 95%. In this filter the values of

the noisy pixels are replaced with the mean value of noise free pixel in selected window. In addition, the proposed filter (SUMF) uses simple fixed length window of size 2X2 which results in easy implementation. The rest of the paper is organized as follows. The proposed filter is described in section 2, where its implementation steps are also discussed. Section 3 reports a simulation and experimental results to demonstrate the performance of the proposed filter. Finally, conclusion is drawn in section 4.

## A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics :

Two central problems in vision are image segmentation and recognition. Both problems are hard, and we do not yet have any general purpose solution approaching human level competence for either one. While it is unreasonable to expect quick solutions to either problem, there is one dimension on which research in recognition is on much more solid grounds–it is considerably easier to quantify the performance of computer vision algorithms at recognition than at segmentation. Recognition is classification, and one can empirically estimate the probability of misclassification by simply counting classification errors on a test set.

The ready availability of test sets – two of most significant ones are the MNIST handwritten digit dataset and the FERET face data set–has meant that different algorithms can be compared directly using the same quantitative error measures. It is well accepted that one cannot evaluate a recognition algorithm by showing a few images of correct classification. In contrast, image segmentation performance evaluation remains subjective. Typically, researchers will show their results on a few images and point out why the results ‘look good’. We never know from such studies whether the results are best examples or typical examples, whether the technique will work only on images that have no texture, and so on. The major challenge is that the question “What is a correct segmentation” is a subtler question than “Is this digit a 5”.

This has led researchers e.g. Borra and Sarkar to argue that segmentation or grouping performance can be evaluated only in the context of a task such as object recognition. We don’t wish to deny the importance of evaluating segmentations in the context of a task. However, the

thesis of this paper is that segmentations can also be evaluated purely as segmentations by comparing them to those produced by multiple human observers and that there is considerable consistency among different human segmentations of the same image so as to make such a comparison reliable. Figure 1 shows some example images from the database and 3 different segmentations for each image.

The images are of complex, natural scenes. In such images, multiple cues are available for segmentation by a human or a computer program–low level cues such as coherence of brightness, texture or continuity of contour, intermediate level cues such as symmetry and convexity, as well as high level cues based on recognition of familiar objects. The instructions to the human observers made no attempt to restrict or encourage the use of any particular type of cues. For instance, it is perfectly reasonable for observers to use their familiarity with faces to guide their segmentation of the image in the second row of Figure 1.

We realize that this implies that a computational approach based purely on, say, low-level coherence of color and texture, would find it difficult to attain perfect performance. In our view, this is perfectly fine. We wish to define a ‘gold standard’ for segmentation results without any prior biases on what cues and algorithms are to be exploited to obtain those results. We expect that as segmentation and perceptual organization algorithms evolve to make richer use of multiple cues, their performance could continue to be evaluated on the same dataset.

Note that the segmentations produced by different humans for a given image in Figure 1 are not identical. But, are they consistent.

## Image Quality Assessment From Error Visibility to Structural Similarity:

Digital images are subject to a wide variety of distortions during acquisition, processing, compression, storage, transmission and reproduction, any of which may result in a degradation of visual quality. For applications in which images are ultimately to be viewed by human beings, the only “correct” method of quantifying visual image quality is through subjective evaluation. In practice, however, subjective evaluation is usually too inconvenient, time-consuming and expensive.

The goal of research in objective image quality assessment is to develop quantitative measures that can automatically predict perceived image quality. An objective image quality metric can play a variety of roles in image processing applications. First, it can be used to dynamically monitor and adjust image quality. For example, a network digital video server can examine the quality of video being transmitted in order to control and allocate streaming resources. Second, it can be used to optimize algorithms and parameter settings of image processing systems.

Objective image quality metrics can be classified according to the availability of an original (distortion-free) image, with which the distorted image is to be compared. Most existing approaches are known as full-reference, meaning that a complete reference image is assumed to be known. In many practical applications, however, the reference image is not available, and a no-reference or “blind” quality assessment approach is desirable.

In a third type of method, the reference image is only partially available, in the form of a set of extracted features made available as side information to help evaluate the quality of the distorted image. This is referred to as reduced-reference quality assessment. This paper focuses on full-reference image quality assessment. The simplest and most widely used full-reference quality metric is the mean squared error (MSE), computed by averaging the squared intensity differences of distorted and reference image pixels, along with the related quantity of peak signal-to-noise ratio (PSNR).

# CHAPTER 3

**INTRODUCTION TO IMAGE PROCESSING**

# INTRODUCTION

## 3.1.IMAGE:

An image is a two-dimensional picture, which has a similar appearance to some subject usually a physical object or a person.

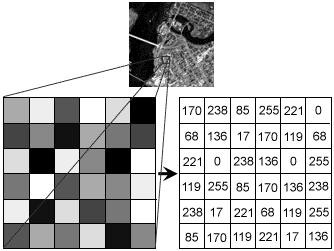
Image is a two-dimensional, such as a photograph, screen display, and as well as a three- dimensional, such as a statue. They may be captured by optical devices—such as cameras, mirrors, lenses, telescopes, microscopes, etc. and natural objects and phenomena, such as the human eye or water surfaces.

The word image is also used in the broader sense of any two-dimensional figure such as a map, a graph, a pie chart, or an abstract painting. In this wider sense, images can also be rendered manually, such as by drawing, painting, carving, rendered automatically by printing or computer graphics technology, or developed by a combination of methods, especially in a pseudo-photograph.



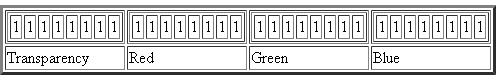
### Fig 1 General image

An image is a rectangular grid of pixels. It has a definite height and a definite width counted in pixels. Each pixel is square and has a fixed size on a given display. However different computer monitors may use different sized pixels. The pixels that constitute an image are ordered as a grid (columns and rows); each pixel consists of numbers representing magnitudes of brightness and color.



### Fig 3.1 Image pixel

Each pixel has a color. The color is a 32-bit integer. The first eight bits determine the redness of the pixel, the next eight bits the greenness, the next eight bits the blueness, and the remaining eight bits the transparency of the pixel.



**Fig3.2 Transparency image**

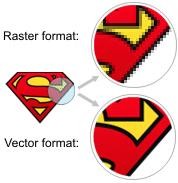
## 3.1IMAGE FILE SIZES:

Image file size is expressed as the number of bytes that increases with the number of pixels composing an image, and the color depth of the pixels. The greater the number of rows and columns, the greater the image resolution, and the larger the file. Also, each pixel of an image increases in size when its color depth increases, an 8-bit pixel (1 byte) stores 256 colors, a 24-bit pixel (3 bytes) stores 16 million colors, the latter known as true color.

Image compression uses algorithms to decrease the size of a file. High resolution cameras produce large image files, ranging from hundreds of kilobytes to megabytes, per the camera's resolution and the image-storage format capacity. High resolution digital cameras record 12 megapixel (1MP = 1,000,000 pixels / 1 million) images, or more, in true color. For example, an image recorded by a 12 MP camera; since each pixel uses 3 bytes to record true color, the uncompressed image would occupy 36,000,000 bytes of memory, a great amount of digital storage for one image, given that cameras must record and store many images to be practical. Faced with large file sizes, both within the camera and a storage disc, image file formats were developed to store such large images.

## IMAGE FILE FORMATS:

Image file formats are standardized means of organizing and storing images. This entry is about digital image formats used to store photographic and other images. Image files are composed of either pixel or vector (geometric) data that are rasterized to pixels when displayed (with few exceptions) in a vector graphic display. Including proprietary types, there are hundreds of image file types. The PNG, JPEG, and GIF formats are most often used to display images on the Internet.



### Fig3.3 Resolution image

In addition to straight image formats, Metafile formats are portable formats which can include both raster and vector information. The metafile format is an intermediate format. Most Windows applications open metafiles and then save them in their own native format.

### RASTER FORMATS:

These formats store images as bitmaps (also known as pixmaps)

### JPEG/JFIF:

JPEG (Joint Photographic Experts Group) is a compression method. JPEG compressed images are usually stored in the JFIF (JPEG File Interchange Format) file format. JPEG compression is lossy compression. Nearly every digital camera can save images in the JPEG/JFIF format, which supports 8 bits per color (red, green, blue) for a 24-bit total, producing relatively small files. Photographic images may be better stored in a lossless non-JPEG format if they will be re-edited, or if small "artifacts" are unacceptable. The JPEG/JFIF format also is used as the image compression algorithm in many Adobe PDF files.

### EXIF:

The EXIF (Exchangeable image file format) format is a file standard similar to the JFIF format with TIFF extensions. It is incorporated in the JPEG writing software used in most cameras. Its purpose is to record and to standardize the exchange of images with image metadata between digital cameras and editing and viewing software. The metadata are recorded for individual images and include such things as camera settings, time and date, shutter speed, exposure, image size, compression, name of camera, color information, etc. When images are viewed or edited by image editing software, all of this image information can be displayed.

### TIFF:

The TIFF (Tagged Image File Format) format is a flexible format that normally saves 8 bits or 16 bits per color (red, green, blue) for 24-bit and 48-bit totals, respectively, usually using either the TIFF or TIF filename extension. TIFFs are lossy and lossless. Some offer relatively good lossless compression for bi-level (black & white) images. Some digital cameras can save in TIFF format, using the LZW compression algorithm for lossless storage. TIFF image format is

not widely supported by web browsers. TIFF remains widely accepted as a photograph file standard in the printing business. TIFF can handle device-specific color spaces, such as the CMYK defined by a particular set of printing press inks.

### PNG:

The PNG (Portable Network Graphics) file format was created as the free, open-source successor to the GIF. The PNG file format supports true color (16 million colors) while the GIF supports only 256 colors. The PNG file excels when the image has large, uniformly colored areas. The lossless PNG format is best suited for editing pictures, and the lossy formats, like JPG, are best for the final distribution of photographic images, because JPG files are smaller than PNG files. PNG, an extensible file format for the lossless, portable, well-compressed storage of raster images. PNG provides a patent-free replacement for GIF and can also replace many common uses of TIFF. Indexed-color, grayscale, and true color images are supported, plus an optional alpha channel. PNG is designed to work well in online viewing applications, such as the World Wide Web. PNG is robust, providing both full file integrity checking and simple detection of common transmission errors.

### GIF:

GIF (Graphics Interchange Format) is limited to an 8-bit palette, or 256 colors. This makes the GIF format suitable for storing graphics with relatively few colors such as simple diagrams, shapes, logos and cartoon style images. The GIF format supports animation and is still widely used to provide image animation effects. It also uses a lossless compression that is more effective when large areas have a single color, and ineffective for detailed images or dithered images.

### BMP:

The BMP file format (Windows bitmap) handles graphics files within the Microsoft Windows OS. Typically, BMP files are uncompressed, hence they are large. The advantage is their simplicity and wide acceptance in Windows programs.

### VECTOR FORMATS:

As opposed to the raster image formats above (where the data describes the characteristics of each individual pixel), vector image formats contain a geometric description which can be rendered smoothly at any desired display size.

At some point, all vector graphics must be rasterized in order to be displayed on digital monitors. However, vector images can be displayed with analog CRT technology such as that used in some electronic test equipment, medical monitors, radar displays, laser shows and early video games. Plotters are printers that use vector data rather than pixel data to draw graphics.

### CGM:

CGM (Computer Graphics Metafile) is a file format for 2D vector graphics, raster graphics, and text. All graphical elements can be specified in a textual source file that can be compiled into a binary file or one of two text representations. CGM provides a means of graphics data interchange for computer representation of 2D graphical information independent from any particular application, system, platform, or device.

### SVG:

SVG (Scalable Vector Graphics) is an open standard created and developed by the World Wide Web Consortium to address the need for a versatile, scriptable and all purpose vector format for the web and otherwise. The SVG format does not have a compression scheme of its own, but due to the textual nature of XML, an SVG graphic can be compressed using a program such as gzip.

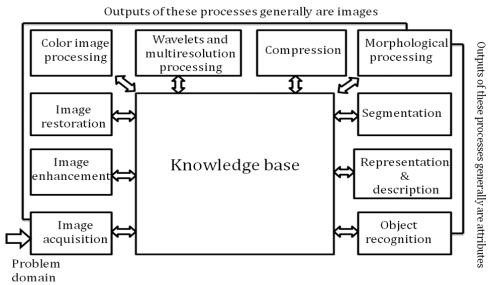
## IMAGE PROCESSING:

Digital image processing, the manipulation of images by computer, is relatively recent development in terms of man’s ancient fascination with visual stimuli. In its short history, it has been applied to practically every type of images with varying degree of success. The inherent subjective appeal of pictorial displays attracts perhaps a disproportionate amount of attention from the scientists and also from the layman. Digital image processing like other glamour fields, suffers from myths, mis-connect ions, mis-understandings and mis-information. It is vast

umbrella under which fall diverse aspect of optics, electronics, mathematics, photography graphics and computer technology. It is truly multidisciplinary endeavor ploughed with imprecise jargon.

Several factor combine to indicate a lively future for digital image processing. A major factor is the declining cost of computer equipment. Several new technological trends promise to further promote digital image processing. These include parallel processing mode practical by low cost microprocessors, and the use of charge coupled devices (CCDs) for digitizing, storage during processing and display and large low cost of image storage arrays.

## FUNDAMENTAL STEPS IN DIGITAL IMAGE PROCESSING:



### Fig 3.5 Image fundamental

* + 1. **Image Acquisition:**

**Image Acquisition** is to acquire a digital image. To do so requires an image sensor and the capability to digitize the signal produced by the sensor. The sensor could be monochrome or color TV camera that produces an entire image of the problem domain every 1/30 sec. the image sensor could also be line scan camera that produces a single image line at a time. In this case, the objects motion past the line.

### Fig 3.5.1 Digital camera image

Scanner produces a two-dimensional image. If the output of the camera or other imaging sensor is not in digital form, an analog to digital converter digitizes it. The nature of the sensor and the image it produces are determined by the application.



### Fig 3.5.2 digital camera cell

* + 1. **Image Enhancement:**

**Image enhancement** is among the simplest and most appealing areas of digital image processing. Basically, the idea behind enhancement techniques is to bring out detail that is obscured, or simply to highlight certain features of interesting an image. A familiar example of enhancement is when we increase the contrast of an image because “it looks better.” It is important to keep in mind that enhancement is a very subjective area of image processing.

* + 1. **Image restoration:**

**Image restoration** is an area that also deals with improving the appearance of an image. However, unlike enhancement, which is subjective, image restoration is objective, in the sense that restoration techniques tend to be based on mathematical or probabilistic models of image degradation.



### Fig 3.5.4 Image restoration

Enhancement, on the other hand, is based on human subjective preferences regarding what constitutes a “good” enhancement result. For example, contrast stretching is considered an enhancement technique because it is based primarily on the pleasing aspects it might present to the viewer, where as removal of image blur by applying a deblurring function is considered a restoration technique.

### Color image processing:

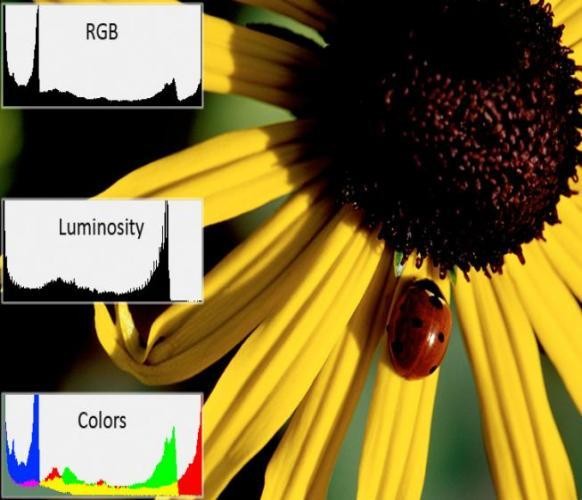
The use of color in image processing is motivated by two principal factors. First, color is a powerful descriptor that often simplifies object identification and extraction from a scene. Second, humans can discern thousands of color shades and intensities, compared to about only two dozen shades of gray. This second factor is particularly important in manual image analysis.



### Fig 3.5.5 Color & Gray scale image

* + 1. **Wavelets and multiresolution processing:**

**Wavelets** are the formation for representing images in various degrees of resolution. Although the Fourier transform has been the mainstay of transform based image processing since the late1950’s, a more recent transformation, called the wavelet transform, and is now making it even easier to compress, transmit, and analyze many images. Unlike the Fourier transform, whose basis functions are sinusoids, wavelet transforms are based on small values, called Wavelets, of varying frequency and limited duration.



### Fig 3.5.6 rgb histogram image

Wavelets were first shown to be the foundation of a powerful new approach to signal processing and analysis called **Multiresolution** theory. Multiresolution theory incorporates and unifies techniques from a variety of disciplines, including sub band coding from signal processing, quadrature mirror filtering from digital speech recognition, and pyramidal image processing.

### Compression:

**Compression**, as the name implies, deals with techniques for reducing the storage required saving an image, or the bandwidth required for transmitting it. Although storage technology has improved significantly over the past decade, the same cannot be said for transmission capacity. This is true particularly in uses of the Internet, which are characterized by significant pictorial content. Image compression is familiar to most users of computers in the form of image file extensions, such as the jpg file extension used in the JPEG (Joint Photographic Experts Group) image compression standard.

### Morphological processing:

**Morphological processing** deals with tools for extracting image components that are useful in the representation and description of shape. The language of mathematical morphology is set theory. As such, morphology offers a unified and powerful approach to numerous image processing problems. Sets in mathematical morphology represent objects in an image. For example, the set of all black pixels in a binary image is a complete morphological description of the image.

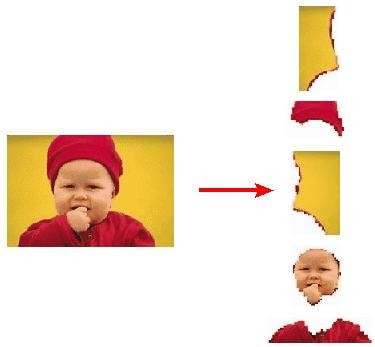


### Fig 3.5.7 blur to deblur image

In binary images, the sets in question are members of the 2-D integer space Z2, where each element of a set is a 2-D vector whose coordinates are the (x,y) coordinates of a black(or white) pixel in the image. Gray-scale digital images can be represented as sets whose components are in Z3. In this case, two components of each element of the set refer to the coordinates of a pixel, and the third corresponds to its discrete gray-level value.

### Segmentation:

**Segmentation** procedures partition an image into its constituent parts or objects. In general, autonomous segmentation is one of the most difficult tasks in digital image processing. A rugged segmentation procedure brings the process a long way toward successful solution of imaging problems that require objects to be identified individually.



### Fig 3.5.8 Image segmentation

On the other hand, weak or erratic segmentation algorithms almost always guarantee eventual failure. In general, the more accurate the segmentation, the more likely recognition is to succeed.

### Representation and description:

Representation and description almost always follow the output of a segmentation stage, which usually is raw pixel data, constituting either the boundary of a region (i.e., the set of pixels separating one image region from another) or all the points in the region itself. In either case,

converting the data to a form suitable for computer processing is necessary. The first decision that must be made is whether the data should be represented as a boundary or as a complete region. Boundary representation is appropriate when the focus is on external shape characteristics, such as corners and inflections.

Regional representation is appropriate when the focus is on internal properties, such as texture or skeletal shape. In some applications, these representations complement each other. Choosing a representation is only part of the solution for transforming raw data into a form suitable for subsequent computer processing. A method must also be specified for describing the data so that features of interest are highlighted. Description, also called feature selection, deals with extracting attributes that result in some quantitative information of interest or are basic for differentiating one class of objects from another.

### Object recognition:

The last stage involves recognition and interpretation. Recognition is the process that assigns a label to an object based on the information provided by its descriptors. Interpretation involves assigning meaning to an ensemble of recognized objects.

### Knowledge base:

Knowledge about a problem domain is coded into image processing system in the form of a knowledge database. This knowledge may be as simple as detailing regions of an image when the information of interests is known to be located, thus limiting the search that has to be conducted in seeking that information. The knowledge base also can be quite complex, such as an inter related to list of all major possible defects in a materials inspection problem or an image data base containing high resolution satellite images of a region in connection with change deletion application. In addition to guiding the operation of each processing module, the knowledge base also controls the interaction between modules. The system must be endowed with the knowledge to recognize the significance of the location of the string with respect to other components of an address field. This knowledge glides not only the operation of each module, but it also aids in feedback operations between modules through the knowledge base. We implemented preprocessing techniques using MATLAB.

## COMPONENTS OF AN IMAGE PROCESSING SYSTEM:

As recently as the mid-1980s, numerous models of image processing systems being sold throughout the world were rather substantial peripheral devices that attached to equally substantial host computers. Late in the 1980s and early in the 1990s, the market shifted to image processing hardware in the form of single boards designed to be compatible with industry standard buses and to fit into engineering workstation cabinets and personal computers. In addition to lowering costs, this market shift also served as a catalyst for a significant number of new companies whose specialty is the development of software written specifically for image processing.

Network

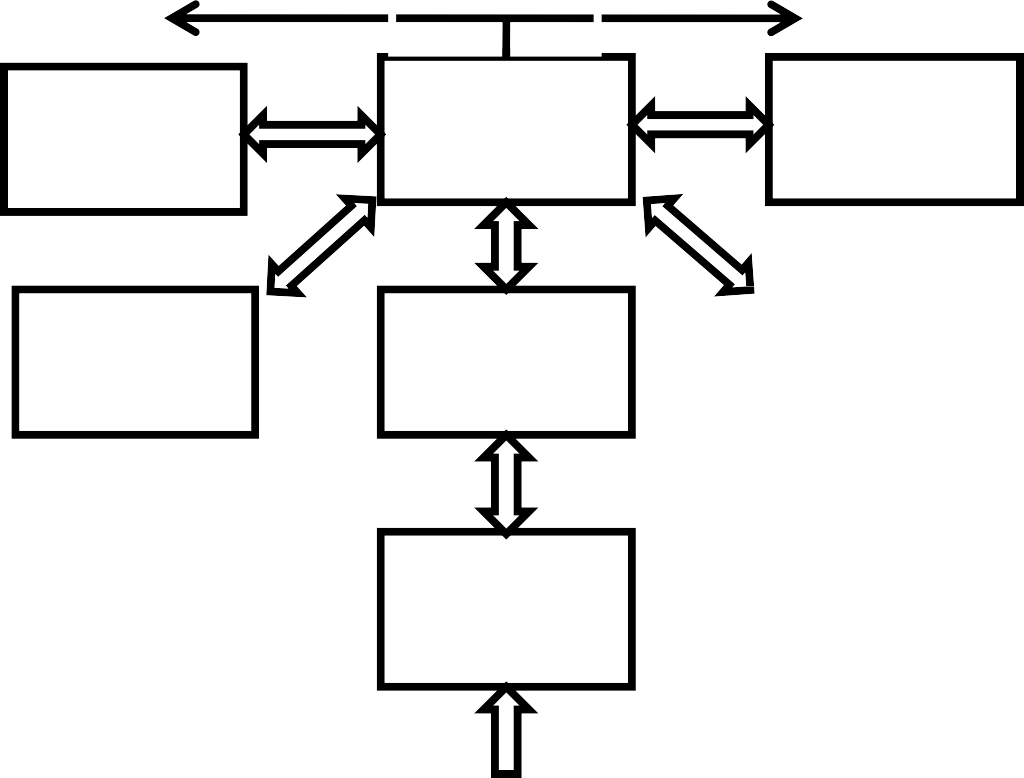


Image displays

computer

Mass storage

Hard copy

Specialized

image processing

Image

processing software

Image sensor

Problem domain

### Fig 3.6 Component of image processing

Although large-scale image processing systems still are being sold for massive imaging applications, such as processing of satellite images, the trend continues toward miniaturizing and blending of general-purpose small computers with specialized image processing hardware. Figure 1.24 shows the basic components comprising a typical general-purpose system used for digital image processing. The function of each component is discussed in the following paragraphs, starting with image sensing.

### Image sensors:

With reference to sensing, two elements are required to acquire digital images. The first is a physical device that is sensitive to the energy radiated by the object we wish to image. The second, called a digitizer, is a device for converting the output of the physical sensing device into digital form. For instance, in a digital video camera, the sensors produce an electrical output proportional to light intensity. The digitizer converts these outputs to digital data.

### Specialized image processing hardware:

Specialized image processing hardware usually consists of the digitizer just mentioned, plus hardware that performs other primitive operations, such as an arithmetic logic unit (ALU), which performs arithmetic and logical operations in parallel on entire images. One example of how an ALU is used is in averaging images as quickly as they are digitized, for the purpose of noise reduction. This type of hardware sometimes is called a front-end subsystem, and its most distinguishing characteristic is speed. In other words, this unit performs functions that require fast data throughputs (e.g., digitizing and averaging video images at 30 frames) that the typical main computer cannot handle.

### Computer:

The computer in an image processing system is a general-purpose computer and can range from a PC to a supercomputer. In dedicated applications, sometimes specially designed computers are used to achieve a required level of performance, but our interest here is on general-purpose image processing systems. In these systems, almost any well-equipped PC-type machine is suitable for offline image processing tasks.

### Image processing software:

Software for image processing consists of specialized modules that perform specific tasks. A well-designed package also includes the capability for the user to write code that, as a

minimum, utilizes the specialized modules. More sophisticated software packages allow the integration of those modules and general-purpose software commands from at least one computer language.

### Mass storage:

Mass storage capability is a must in image processing applications. An image of size 1024\*1024 pixels, in which the intensity of each pixel is an 8-bit quantity, requires one megabyte of storage space if the image is not compressed. When dealing with thousands, or even millions, of images, providing adequate storage in an image processing system can be a challenge. Digital storage forimage processing applications fall into three principal categories: (1) short-term storage for use during processing, (2) on-line storage for relatively fast recall, and (3) archival storage, characterized by infrequent access. Storage is measured in bytes (eight bits), Kbytes (one thousand bytes), Mbytes (one million bytes), Gbytes (meaning giga, or one billion, bytes), and Tbytes (meaning tera, or one trillion, bytes)

One method of providing short-term storage is computer memory. Another is by specialized boards, called frame buffers that store one or more images and can be accessed rapidly, usually at video rates. The latter method allows virtually instantaneous image zoom, as well as scroll (vertical shifts) and pan (horizontal shifts). Frame buffers usually are housed in the specialized image processing hardware unit shown in Fig. 1.24. Online storage generally takes the form of magnetic disks or optical-media storage. The key factor characterizing on-line storage is frequent access to the stored data. Finally, archival storage is characterized by massive storage requirements but infrequent need for access. Magnetic tapes and optical disks housed in “jukeboxes” are the usual media for archival applications.

### Image displays:

Image displays in use today are mainly color (preferably flat screen) TV monitors. Monitors are driven by the outputs of image and graphics display cards that are an integral part of the computer system. Seldom are there requirements for image display applications that cannot be met by display cards available commercially as part of the computer system. In some

cases, it is necessary to have stereo displays, and these are implemented in the form of headgear containing two small displays embedded in goggles worn by the user.

### Hardcopy:

Hardcopy devices for recording images include laser printers, film cameras, heat- sensitive devices, inkjet units, and digital units, such as optical and CD-ROM disks. Film provides the highest possible resolution, but paper is the obvious medium of choice for written material. For presentations, images are displayed on film transparencies or in a digital medium if image projection equipment is used. The latter approach is gaining acceptance as the standard for image presentations.

### Network:

Networking is almost a default function in any computer system in use today. Because of the large amount of data inherent in image processing applications, the key consideration in image transmission is bandwidth. In dedicated networks, this typically is not a problem, but communications with remote sites via the Internet are not always as efficient. Fortunately, this situation is improving quickly as a result of optical fiber and other broadband technologies.

Color and texture are two low-level features widely used for image classification, indexing and retrieval. Color is usually represented as a histogram, which is a first order statistical measure that captures global distribution of color in an image One of the main drawbacks of the histogram- based approaches is that the spatial distribution and local variations in color are ignored. Local spatial variation of pixel intensity is commonly used to capture texture information in an image. Grayscale Co-occurrence Matrix (GCM) is a well-known method for texture extraction in the spatial domain. A GCM stores the number of pixel neighborhoods in an image that have a particular grayscale combination. Let I be an image and let p and Np respectively denote any arbitrary pixel and its neighbor in a given direction. If GL denotes the total number of quantized gray levels and gl denotes the individual gray levels, where, gl 

{0, . . .,GL \_ 1}, then each component of GCM can be written as follows:

gcm(i, j) is the number of times the gray level of a pixel p denoted by glp equals i, and the gray level of its neighbor Np denoted by glNp equals j, as a fraction of the total number of pixels in the image. Thus, it estimates the probability that the gray level of an arbitrary pixel in an image is i, and that of its neighbor is j. One GCM matrix is generated for each possible neighborhood direction, namely, 0, 45, 90 and 135.Average and range of 14 features like Angular Second Moment, Contrast, Correlation, etc., are generated by combining all the four matrices to get a total of 28 features. In the GCM approach for texture extraction, color information is completely lost since only pixel gray levels are considered.

To incorporate spatial information along with the color of image pixels, a feature called color correlogram has recently been proposed. It is a three dimensional matrix that represents the probability of finding pixels of any two given colors at a distance ‘d’ apart Auto correlogram is a variation of correlogram, which represents the probability of finding two pixels with the same color at a distance ‘d’ apart. This atproach can effectively represent color distribution in an image. However, correlogram features do not capture intensity variation Many image databases often contain both color as well as gray scale images. The color correlogram method does not constitute a good descriptor in such databases.

Another method called Color Co-occurrence Matrix (CCM) has been proposed to capture color variation in an image. CCM is represented as a three-dimensional matrix, where color pair of the pixels p and Np are captured in the first two dimensions of the matrix and the spatial distance ‘d’ between these two pixels is captured in the third dimension. This approach is a generalization of the color correlogram and reduces to the pure color

correlogram for d = 1. CCM is generated using only the Hue plane of the HSV (Hue, Saturation and Intensity Value) color space. The Hue axis is quantized into HL number of levels. If individual hue values are denoted by hl, where  , then each component of

CCM can be written as follows:



Four matrices representing neighbors at angles 0, 90, 180 and 270 are considered. This approach was further extended by separating the diagonal and the non-diagonal components of CCM to generate a Modified Color Co-occurrence Matrix (MCCM). MCCM, thus, may be written as follows: MCCM = (CCMD;CCMND)

Here, CCMD and CCMND correspond to the diagonal and off-diagonal components of CCM. The main drawback of this approach is that, like correlogram, it also captures only color information and intensity information is completely ignored.

An alternative approach is to capture intensity variation as a texture feature from an image and combine it with color features like histograms using suitable weights . One of the challenges of this approach is to determine suitable weights since these are highly application-dependent. In certain applications like Content-based Image Retrieval (CBIR), weights are often estimated from relevance feedback given by users.

While relevance feedback is sometimes effective, it makes the process of image retrieval user-dependent and iterative. There is also no guarantee on the convergence of

the weight-learning algorithms. In order to overcome these problems, researchers have tried to combine color and texture features together during extraction.

proposed two approaches for capturing color and intensity variations from an image using the LUV color space. In the Single-channel Co-occurrence Matrix (SCM), variations for each color channel, namely, L, U and V are considered independently. In the Multi channel Co-occurrence Matrix (MCM), variations are captured taking two channels at a time – UV, LU and LV. Since the LUV color space separates out chrominance (L and U) from luminance (V), SCM in effect, generates one GCM and two CCMs from each image independently. As a result, correlation between the color channels is lost

However, in MCM, the count of pair wise occurrences of the values of different channels of the color space is captured. Thus, each component of MCM can be written as follows:

mcmUV(i; j) = Pr((up; vNp) = (i; j))

mcmLU(i; j) = Pr((lp; uNp) = (i; j))

mcmLV(i; j) = Pr((lp; vNp) = (i; j))

Here, mcmUV(i, j) is the number of times the U chromaticity value of a pixel p denoted by up equals i, and the V chromaticity value of its neighbor Np denoted by vNp equals j, as a fraction of the total number of pixels in the image. Similarly, mcmLU(i, j) and mcmLV(i, j) are defined. One MCM matrix is generated for each of the four neighborhood directions, namely, 0, 45, 90 and 135.

Deng and Manjunath (2001) proposed a two-stage method called JSEG, which combines color and texture after image segmentation. In the first stage, colors are quantized to the required levels for differentiating between various regions of an image. Pixel values of the regions are then replaced by their quantized color levels to form a color map. Spatial variation of color levels between different regions in the map is viewed as a type of texture composition of the image.

Yu et al. (2002) suggested the use of color texture moments to represent both color and texture of an image. This approach is based on the calculation of Local Fourier Transformation (LFT) coefficients. Eight templates equivalent to LFT are operated over an image to generate a characteristic map of the image. Each template is a 3 · 3 filter that considers eight neighbors of the current pixel for LFT calculation. First and second order moments of the characteristic map are then used to generate a set of features.

In this paper, we propose an integrated approach for capturing spatial variation of both color and intensity levels in the neighborhood of each pixel using the HSV color space. In contrast to the other methods, for each pixel and its neighbor, the amount of color and intensity variation between them is estimated using a weight function. Suitable constraints are satisfied while choosing the weight function for effectively relating visual perception of color and the HSV color space properties. The color and intensity variations are represented in a single composite feature known as Integrated Color and Intensity Co-occurrence Matrix (ICICM). While the existing schemes generally treat color and intensity separately, the proposed method provides a composite view to both color and intensity variations in the same feature. The main advantage of using ICICM is that it avoids the use of weights to combine individual color and

texture features. We use ICICM feature in an image retrieval application from large image databases.

Early result on this work was reported in (Vadivel et al., 2004a). In the next section, we describe the proposed feature extraction technique after introducing some of the properties of the HSV color space. Choice of quantization levels for color and intensity axes, selection of parameter values and a brief overview of the image retrieval application

## Integrated color and intensity co-occurrence matrix:

We propose to capture color and intensity variation around each pixel in a two-dimensional matrix called Integrated Color and Intensity Co-occurrence Matrix (ICICM). This is a generalization of the Grayscale Co-occurrence Matrix and the Color Co-occurrence Matrix techniques. For each pair of neighboring pixels, we consider

their contribution to both color perception as well as gray level perception to the human eye. Some of the useful properties of the HSV color space and their relationship to human color perception are utilized for extracting this feature. In the next sub-section, we briefly explain relevant properties of the HSV color space. In the subsequent subsection, we describe how the properties can be effectively used for generating ICICM.

# HSV color space:

HSV Color space: Basically there are three properties or three dimensions of color that being hue, saturation and value HSV means Hue, Saturation and Value. It is important to look at because it describes the color based on three properties. It can create the

full spectrum of colors by editing the HSV values. The first dimension is the Hue. Hue is the other name for the color or the complicated variation in the color. The quality of color as determined by its dominant wavelength. This Hue is broadly classified into three categories. They are primary Hue, Secondary Hue and Teritiary Hue. The first and the foremost is the primary Hue it consists of three colors they are red, yellow and blue. The secondary Hue is formed by the combination of the equal amount of colors of the primary Hue and the colors of the secondary Hue which was formed by the primary Hue are Orange, Green and violet. The remaining one is the teritiary Hue is formed by the combination of the primary Hue and the secondary Hue. The limitless number of colors are produced by mixing the colors of the primary Hue in different amounts. Saturation is the degree or the purity of color. Then the second dimension is the saturation. Saturation just gives the intensity to the colors. The saturation and intensity drops just by mixing the colors or by adding black to the color. By adding the white to the color in spite of more intense the color becomes lighter. Then finally the third dimension is the Value. The value is the brightness of the color. When the value is zero the color space is totally black with the increase in the color there is also increase in the brightness and shows the various colors. The value describes the contrast of the color. That means it describes the lightness and darkness of the color. As similar to the saturation this value consists of the tints and shades. Tints are the colors with the added white and shades are the colors with the added black.

### Properties of the HSV color space:

Sensing of light from an image in the layers of human retina is a complex process with rod cells contributing to scotopic or dim-light vision and cone cells to photopic or bright-light vision (Gonzalez and Woods, 2002). At low levels of illumination, only the rod cells are excited so that only gray shades are perceived. As the illumination level increases, more and

more cone cells are excited, resulting in increased color perception. Various color spaces have been introduced to represent and specify colors in a way suitable for storage, processing or transmission of color information in images. Out of these, HSV is one of the models that separate out the luminance component (Intensity) of a pixel color from its chrominance components (Hue and Saturation). Hue represents pure color, which is perceived when incident light is of sufficient illumination and contains a single wavelength. Saturation gives a measure of the degree by which a pure color is diluted by white light. For light with low illumination, corresponding intensity value in the HSV color space is also low.

The HSV color space can be represented as a Hexa cone, with the central vertical axis denoting the luminance component, I (often denoted by V for Intensity Value). Hue, is a chrominance component defined as an angle in the range [0,2p] relative to the red axis with red at angle 0, green at 2p/3, blue at 4p/3 and red again at 2p. Saturation, S, is the other chrominance component, measured as a radial distance from the central axis of the hexacone with value between 0 at the center to 1 at the outer surface. For zero saturation, as the intensity is increased, we move from black to white through various shades of gray. On the other hand, for a given intensity and hue, if the saturation is changed from 0 to 1, the perceived color changes from a shade of gray to the most pure form of the color represented by its hue. When saturation is near 0, all the pixels in an image look alike even though their hue values are different.

As we increase saturation towards 1, the colors get separated out and are visually perceived as the true colors represented by their hues. Low saturation implies presence of a large number of spectral components in the incident light, causing loss of color information even though the illumination level is sufficiently high. Thus, for low values of saturation or intensity, we can approximate a pixel color by a gray level while for higher saturation and

intensity, the pixel color can be approximated by its hue. For low intensities, even for a high saturation, a pixel color is close to its gray value. Similarly, for low saturation even for a high value of intensity, a pixel is perceived as gray. We use these properties to estimate the degree by which a pixel contributes to color perception and gray level perception.

One possible way of capturing color perception of a pixel is to choose suitable thresholds on the intensity and saturation. If the saturation and the intensity are above their respective thresholds, we may consider the pixel to have color dominance; else, it has gray level dominance. However, such a hard thresholding does not properly capture color perception near the threshold values. This is due to the fact that there is no fixed level of illumination above which the cone cells get excited. Instead, there is a gradual transition from scotopic to photopic vision. Similarly, there is no fixed threshold for the saturation of cone cells that leads to loss of chromatic information at higher levels of illumination caused by color dilution. We, therefore, use suitable weights that vary smoothly with saturation and intensity to represent both color and gray scale perception for each pixel.

**NON INTERVAL QUANTIZATION:**

Due to the large range for each component by directly calculating the characteristics for the retrieval then the computation will be very difficult to ensure rapid retrieval. It is essential to quantify HSV space component to reduce computation and improve efficiency. At the same time, because the human eye to distinguish colors is limited, do not need to calculate all segments. Unequal interval quantization according the human color perception has been applied on H , S ,V components.

Based on the color model of substantial analysis, we divide color into eight parts. Saturation and intensity is divided into three parts separately in accordance with the human eyes to distinguish. In accordance with the different colors and subjective color perception quantification, quantified hue(H), saturation(S) and value(V)

In accordance with the quantization level above, the H, S, V three- dimensional feature vector for different values of with different weights to form one dimensional feature vector and is given by the following equation:

G = Qs\*Qv\*H+Qv\*s+V

Where Qs is the quantized series of S and Qv is the quantized series of V. And now by setting Qs = Qv = 3, Then G = 9H+3S+V

In this way three component vector of the HSV from one dimensional vector, Which quantize the whole color space for the 72 kinds of the main colors. So we can handle 72 bins of one dimensional histogram. This qualification is effective in reducing the images by the effect of the light intensity, but also reducing the computational time and complexity.

**IMAGE RETRIEVAL:**

Image retrieval is nothing but a computer system used for browsing searching and retrieving images from a large database of digital images. Most traditional and common methods of image retrieval use some method of adding metadata by captioning, Keywords or the descriptions to the images so that the retrieval can be performed. Manual image annotation is time consuming, expensive and laborious. For addressing this there has been a large

amount of research done on automatic image annotation. It is crucial to understand the scope and nature of the image data in order to determine the complexity of the image search system design. The design is also largely dependent on the factors. And some of the factors include archives, Domain specific collection, Enterprise collection, Personal collection and web etc..,

Invention of the digital camera has given the common man the privilege to capture his world in pictures, and conveniently share them with others. one can today generate volumes of images with content as diverse as family get-togethers and national park visits. Low-cost storage and easy Web hosting has fueled the metamorphosis of common man from a passive consumer of photography in the past to a current-day active producer. Today, searchable image data exists with extremely diverse visual and semantic content, spanning geographically disparate locations, and is rapidly growing in size. All these factors have created innumerable possibilities and hence considerations for real-world image search system designers.

As far as technological advances are concerned, growth in Content-based image retrieval has been unquestionably rapid. In recent years, there has been significant effort put into understanding the real world implications, applications, and constraints of the technology. Yet, real-world application of the technology is currently limited. We devote this section to understanding image retrieval in the real world and discuss user expectations, system constraints and requirements, and the research effort to make image retrieval a reality in the not-too-distant future.

An image retrieval system designed to serve a personal collection should focus on features such as personalization, flexibility of browsing, and display methodology. For example, Google’s Picasa system [Picasa 2004] provides a chronological display of images taking a user

on a journey down memory lane. Domain specific collections may impose specific standards for presentation of results. Searching an archive for content discovery could involve long user search sessions. Good visualization and a rich query support system should be the design goals. A system designed for the Web should be able to support massive user traffic. One way to supplement software approaches for this purpose is to provide hardware support to the system architecture. Unfortunately, very little has been explored in this direction, partly due to the lack of agreed-upon indexing and retrieval methods. The notable few applications include an FPGA implementation of a color-histogram-based image retrieval system [Kotoulas and Andreadis 2003], an FPGA implementation for sub image retrieval within an image database [Nakano and Takamichi 2003], and a method for efficient retrieval in a network of imaging devices [Woodrow and Heinzelman 2002].

Discussion*.* Regardless of the nature of the collection, as the expected user-base grows, factors such as concurrent query support, efficient caching, and parallel and distributed processing of requests become critical. For future real-world image retrieval systems, both software and hardware approaches to address these issues are essential. More realistically, dedicated specialized servers, optimized memory and storage support, and highly parallelizable image search algorithms to exploit cluster computing powers are where the future of large-scale image search hardware support lies.

**OVERVIEW OF TEXTURE:**

We all know about the term Texture but for defining it is a hard time. One can differentiate the two different Textures by recognizing the similarities and differences. Commonly there are three ways for the usage of the Textures:

Based on the Textures the images can be segmented To differentiate between already segmented regions or to classify them.We can reproduce Textures by producing the descriptions. The texture can be analyzed in three different ways. They are Spectral, Structural and Statistical:

# CHAPTER-4

**DIGITAL IMAGE PROCESSING**

### Digital image processing Background:

Digital image processing is an area characterized by the need for extensive experimental work to establish the viability of proposed solutions to a given problem. An important characteristic underlying the design of image processing systems is the significant level of testing & experimentation that normally is required before arriving at an acceptable solution. This characteristic implies that the ability to formulate approaches &quickly prototype candidate solutions generally plays a major role in reducing the cost & time required to arrive at a viable system implementation.

### What is DIP

An image may be defined as a two-dimensional function f(x, y), where x & y are spatial coordinates, & the amplitude of f at any pair of coordinates (x, y) is called the intensity or gray level of the image at that point. When x, y & the amplitude values of f are all finite discrete quantities, we call the image a digital image. The field of DIP refers to processing digital image by means of digital computer. Digital image is composed of a finite number of elements, each of which has a particular location & value. The elements are called pixels.

Vision is the most advanced of our sensor, so it is not surprising that image play the single most important role in human perception. However, unlike humans, who are limited to the visual band of the EM spectrum imaging machines cover almost the entire EM spectrum, ranging from gamma to radio waves. They can operate also on images generated by sources that humans are not accustomed to associating with image.

There is no general agreement among authors regarding where image processing stops & other related areas such as image analysis& computer vision start. Sometimes a distinction is made by defining image processing as a discipline in which both the input & output at a process

are images. This is limiting & somewhat artificial boundary. The area of image analysis (image understanding) is in between image processing & computer vision.

There are no clear-cut boundaries in the continuum from image processing at one end to complete vision at the other. However, one useful paradigm is to consider three types of computerized processes in this continuum: low-, mid-, & high-level processes. Low-level process involves primitive operations such as image processing to reduce noise, contrast enhancement & image sharpening. A low- level process is characterized by the fact that both its inputs & outputs are images.

Mid-level process on images involves tasks such as segmentation, description of that object to reduce them to a form suitable for computer processing & classification of individual objects. A mid-level process is characterized by the fact that its inputs generally are images but its outputs are attributes extracted from those images. Finally higher- level processing involves “Making sense” of an ensemble of recognized objects, as in image analysis & at the far end of the continuum performing the cognitive functions normally associated with human vision.

Digital image processing, as already defined is used successfully in a broad range of areas of exceptional social & economic value.

### What is an image?

An image is represented as a two dimensional function f(x, y) where x and y are spatial co-ordinates and the amplitude of ‘f’ at any pair of coordinates (x, y) is called the intensity of the image at that point.

### Gray scale image:

A grayscale image is a function I (xylem) of the two spatial coordinates of the image

plane.

I(x, y) is the intensity of the image at the point (x, y) on the image plane.

I (xylem) takes non-negative values assume the image is bounded by a rectangle [0, a] [0, b]I: [0, a]  [0, b]  [0, info)

### Color image:

**It** can be represented by three functions, R (xylem) for red, G (xylem) for green *and* B (xylem) for blue.

An image may be continuous with respect to the x and y coordinates and also in amplitude. Converting such an image to digital form requires that the coordinates as well as the amplitude to be digitized. Digitizing the coordinate’s values is called sampling. Digitizing the amplitude values is called quantization.

### Coordinate convention:

The result of sampling and quantization is a matrix of real numbers. We use two principal ways to represent digital images. Assume that an image f(x, y) is sampled so that the resulting image has M rows and N columns. We say that the image is of size M X N. The values of the coordinates (xylem) are discrete quantities. For notational clarity and convenience, we use integer values for these discrete coordinates.

In many image processing books, the image origin is defined to be at (xylem)=(0,0).The next coordinate values along the first row of the image are (xylem)=(0,1).It is important to keep in mind that the notation (0,1) is used to signify the second sample along the first row. It does not mean that these are the actual values of physical coordinates when the image was sampled. Following figure shows the coordinate convention. Note that x ranges from 0 to M-1 and y from 0 to N-1 in integer increments.

The coordinate convention used in the toolbox to denote arrays is different from the preceding paragraph in two minor ways. First, instead of using (xylem) the toolbox uses the notation (race) to indicate rows and columns. Note, however, that the order of coordinates is the same as the order discussed in the previous paragraph, in the sense that the first element of a coordinate topples, (alb), refers to a row and the second to a column. The other difference is that the origin of the coordinate system is at (r, c) = (1, 1); thus, r ranges from 1 to M and c from 1 to N in integer increments. IPT documentation refers to the coordinates. Less frequently the toolbox also employs another coordinate convention called spatial coordinates which uses x to refer to columns and y to refers to rows. This is the opposite of our use of variables x and y.

### Image as Matrices:

The preceding discussion leads to the following representation for a digitized image function:

|  |  |  |
| --- | --- | --- |
| f (0,0) | f(0,1) | ……….. f(0,N-1) |
| f (1,0) | f(1,1) | ………… f(1,N-1) |

f (xylem)= f (M-1,0) f(M-1,1) f(M-1,N-1)

The right side of this equation is a digital image by definition. Each element of this array is called an image element, picture element, pixel or pel. The terms image and pixel are used throughout the rest of our discussions to denote a digital image and its elements.

A digital image can be represented naturally as a MATLAB matrix:

f (1,1) f(1,2) f(1,N)

f (2,1) f(2,2) …….. f (2,N) .. . f = . f (M,1) f(M,2) f(M,N)

Where f (1,1) = f(0,0) (note the use of a monoscope font to denote MATLAB quantities). Clearly the two representations are identical, except for the shift in origin. The notation f(p ,q) denotes the element located in row p and the column q. For example f(6,2) is the element in the sixth row and second column of the matrix f. Typically we use the letters M and N respectively to denote the number of rows and columns in a matrix. A 1xN matrix is called a row vector whereas an Mx1 matrix is called a column vector. A 1x1 matrix is a scalar.

Matrices in MATLAB are stored in variables with names such as A, a, RGB, real array and so on. Variables must begin with a letter and contain only letters, numerals and underscores. As noted in the previous paragraph, all MATLAB quantities are written using mono-scope characters. We use conventional Roman, italic notation such as f(x ,y), for mathematical expressions

### Reading Images:

Images are read into the MATLAB environment using function imread whose syntax is Imread (‘filename’)

|  |  |  |
| --- | --- | --- |
| **Format name** | **Description** | **recognized extension** |
| TIFF | Tagged Image File Format | .tif, .tiff |
| JPEG | Joint Photograph Experts Group | .jpg, .jpeg |
| GIF | Graphics Interchange Format | .gif |
| BMP | Windows Bitmap | .bmp |

PNG Portable Network Graphics .png

XWD X Window Dump .xwd

Here filename is a spring containing the complete of the image file(including any applicable extension).For example the command line

>> f = imread (‘8. jpg’);

Reads the JPEG (above table) image chestxray into image array f. Note the use of single quotes (‘) to delimit the string filename. The semicolon at the end of a command line is used by MATLAB for suppressing output If a semicolon is not included. MATLAB displays the results of the operation(s) specified in that line. The prompt symbol (>>) designates the beginning of a command line, as it appears in the MATLAB command window.

### Data Classes:

Although we work with integers coordinates the values of pixels themselves are not restricted to be integers in MATLAB. Table above list various data classes supported by MATLAB and IPT are representing pixels values. The first eight entries in the table are refers to as numeric data classes. The ninth entry is the char class and, as shown, the last entry is referred to as logical data class.

All numeric computations in MATLAB are done in double quantities, so this is also a frequent data class encounter in image processing applications. Class unit 8 also is encountered frequently, especially when reading data from storages devices, as 8 bit images are most common representations found in practice. These two data classes, classes logical, and, to a lesser degree, class unit 16 constitute the primary data classes on which we focus. Many ipt functions however support all the data classes listed in table. Data class double requires 8 bytes to represent a number uint8 and int 8 require one byte each, uint16 and int16 requires 2bytes and unit 32.

### Name Description

Double Double \_ precision, floating\_ point numbers the Approximate.

Uint8 unsigned 8\_bit integers in the range [0,255] (1byte per

Element).

Uint16 unsigned 16\_bit integers in the range [0, 65535] (2byte per element).

Uint 32 unsigned 32\_bit integers in the range [0, 4294967295](4 bytes per element). Int8 signed 8\_bit integers in the range [-128,127] 1 byte per element)

Int 16 signed 16\_byte integers in the range [32768, 32767] (2 bytes per

element).

Int 32 Signed 32\_byte integers in the range [-2147483648, 21474833647] (4 byte per element).

Single single \_precision floating \_point numbers with values

In the approximate range (4 bytes per elements) Char characters (2 bytes per elements).

Logical values are 0 to 1 (1byte per element).

Int 32 and single required 4 bytes each. The char data class holds characters in Unicode representation. A character string is merely a 1\*n array of characters logical array contains only the values 0 to 1,with each element being stored in memory using function logical or by using relational operators.

### Image Types:

The toolbox supports four types of images:

1 .Intensity images;

1. Binary images;
2. Indexed images;
3. R G B images.

Most monochrome image processing operations are carried out using binary or intensity images, so our initial focus is on these two image types. Indexed and RGB colour images.

### Intensity Images:

An intensity image is a data matrix whose values have been scaled to represent intentions. When the elements of an intensity image are of class unit8, or class unit 16, they have integer values in the range [0,255] and [0, 65535], respectively. If the image is of class double, the values are floating point numbers. Values of scaled, double intensity images are in the range [0, 1] by convention.

### Binary Images:

Binary images have a very specific meaning in MATLAB.A binary image is a logical array 0s and1s.Thus, an array of 0s and 1s whose values are of data class, say unit8, is not considered as a binary image in MATLAB .A numeric array is converted to binary using

function logical. Thus, if A is a numeric array consisting of 0s and 1s, we create an array B using the statement.

B=logical (A)

If A contains elements other than 0s and 1s.Use of the logical function converts all nonzero quantities to logical 1s and all entries with value 0 to logical 0s.

Using relational and logical operators also creates logical arrays.

To test if an array is logical we use the I logical function: islogical(c).

If c is a logical array, this function returns a 1.Otherwise returns a 0. Logical array can be converted to numeric arrays using the data class conversion functions.

### Indexed Images:

An indexed image has two components: A data matrix integer, x

A color map matrix, map

Matrix map is an m\*3 arrays of class double containing floating point values in the range [0, 1].The length m of the map are equal to the number of colors it defines. Each row of map specifies the red, green and blue components of a single color. An indexed images uses “direct mapping” of pixel intensity values color map values. The color of each pixel is determined by using the corresponding value the integer matrix x as a pointer in to map. If x is of class double ,then all of its components with values less than or equal to 1 point to the first row in map, all components with value 2 point to the second row and so on. If x is of class units or unit 16, then all components value 0 point to the first row in map, all components with value 1 point to the second and so on.

### RGB Image:

An RGB color image is an M\*N\*3 array of color pixels where each color pixel is triplet corresponding to the red, green and blue components of an RGB image, at a specific spatial

location. An RGB image may be viewed as “stack” of three gray scale images that when fed in to the red, green and blue inputs of a color monitor

Produce a color image on the screen. Convention the three images forming an RGB color image are referred to as the red, green and blue components images. The data class of the components images determines their range of values. If an RGB image is of class double the range of values is [0, 1].

Similarly the range of values is [0,255] or [0, 65535].For RGB images of class units or unit 16 respectively. The number of bits use to represents the pixel values of the component images determines the bit depth of an RGB image. For example, if each component image is an 8bit image, the corresponding RGB image is said to be 24 bits deep.

Generally, the number of bits in all component images is the same. In this case the number of possible color in an RGB image is (2^b) ^3, where b is a number of bits in each component image. For the 8bit case the number is 16,777,216 colors.



**5.1Revisiting Median Filtering**

**CHAPTER 5**

Median filter is commonly utilized for noise removal in image enhancement. Specifically, median filter focuses on a target pixel X, and considers the w × w-neighborhood pixels surrounding X. These pixels are sorted, and the middle value is selected to replace the target pixel. The process is repeated for all pixels in the image. This technique is feasible because noises are abrupt changes in an image and most natural images have more smooth areas in comparison to edges. Median filter is particularly effective for salt-and-pepper and speckle noises [4]. Figure 1 shows an example of median filter with w = 3, i.e., 3 × 3 neighborhood. Here, the neighborhood N = {67, 32, 31; 45, 21, 22; 83, 42, 46}. After the sorting step, the pixels are

arranged in ascending order to produce N = {21, 22, 31, 32, 42, 45, 46, 67, 83}. Next, the middle value 42 (i.e., 5-th pixel value in this case) is selected to replace the initial target pixel, i.e., 21.

### 5.2Data Embedding

We re-engineer the conventional Median Filter algorithm to hide data expressed in binary representation. To ease the presentation, let M and N be the number of rows and columns in an 8-bit gray scale image, G. Then, let G(m, n) denote the (m, n)-th pixel in G, for m ∈ [1, M] and n

∈ [1, N]. Without loss of generality, let the to-be-embedded data Φ = {φj} be as sequence of 0’s and 1’s, where φj refers to the j-th bit of Φ. Next, for the target pixel X, the pixels within the w × w-neighborhood NX are gathered, sorted, stored in a temporary array Γ. Let γi denote the i-th pixel in the sorted array, where γi ≤ γi+1 for i = 1, 2, ··· , w2. Next, Γ is divided into r partitions to represent φj . When r = 2, two partitions are formed, where Γ0 consists of the first pixel to the w2/r- th pixel, and the second partition Γ1 has the rest of the pixels from Γ. Here, z refers to an integer less than or equal to z. Using the same example in Fig. 1, Γ0 = {21, 22, 31, 32} and Γ1 =

{42, 45, 46, 67, 83}. If φj = ‘0 , any pixel in Γ0 can be selected to replace the target pixel X. Else when φj = ‘1 , a pixel from Γ1 will be selected. These steps are repeated to hide data φj in every pixel of the image. Finally, the output image with embedded data, G is generated.

### 5.3Data Extraction

During data extraction, G is processed one pixel at a time. Specifically, for each target pixel X = G (m, n), the same w × w-neighborhood, i.e., NX is considered. Similar to the embedding process, the pixels in NX are sorted and stored into a temporary array Γ . However, the pixel located in the middle of Γ , namely γ w2/2+1, is used as the threshold for data extraction. Specifically, if the target pixel X < γ w2/2+1, then ‘0’ is extracted. Otherwise, if the target pixel X > γ w2/2+1, then ‘1’ is extracted. The steps are repeated for each pixel in G to extract all the embedded data φj . It is noteworthy that, the pixel values within the neighborhood NX in G may not be the same as those within NX in G although both X and X are the (m, n)-th pixel in the respective images. Therefore, error could occur during decoding.

Image enhancement, as its name implies, improves the quality of image by means of emphasizing the desired details or removing/suppressing the undesired noisy/irrelevant information. Image enhancement plays a crucial role in digital image processing and it has become one of the processes that takes place in any smart device after an image is captured. Some commonly deployed image enhancement techniques include contrast enhancement, image sharpening, smoothing, noise removal, to name a few.

Each enhancement technique can either act as a standalone process to enhance the appearance of an image, or utilized as a pre-processing step before further analysis is carried out. In fact, image enhancement (commonly known as filter) is often utilized to enhance image quality before it is shared online on social media or communicated for consumption by a wider audience. However, image enhancement is particularly important when the received original/source image (e.g., from field reporter, amateur photographer) contains noise. While the noisy image is enhanced, the processes involved are not published or shared.

One can also imagine the need to share specific tone-mapping operators as well as the parameters used when shrinking the dynamic range of a noisy high dynamic range image into the standard dynamic range. To accommodate this additional data, information hiding appears to be one of the commonly implemented solutions. There are many conventional information hiding methods, including Least Significant Bit (LSB) insertion, Histogram Shifting (HS) and Difference Expansion (DE), to name a few.

These techniques are studied extensively and many of their variants are proposed [1, 2, 3] to achieve a balance trade-off among embedding capacity, image quality, robustness against attacks, etc. However, these techniques are not widely adopted into the usual operations performed by the users, and often they are implemented as an additional step after the image is processed. In most cases, the user has to explicitly install or develop the data embedding algorithm to enable data embedding into the image of interest.

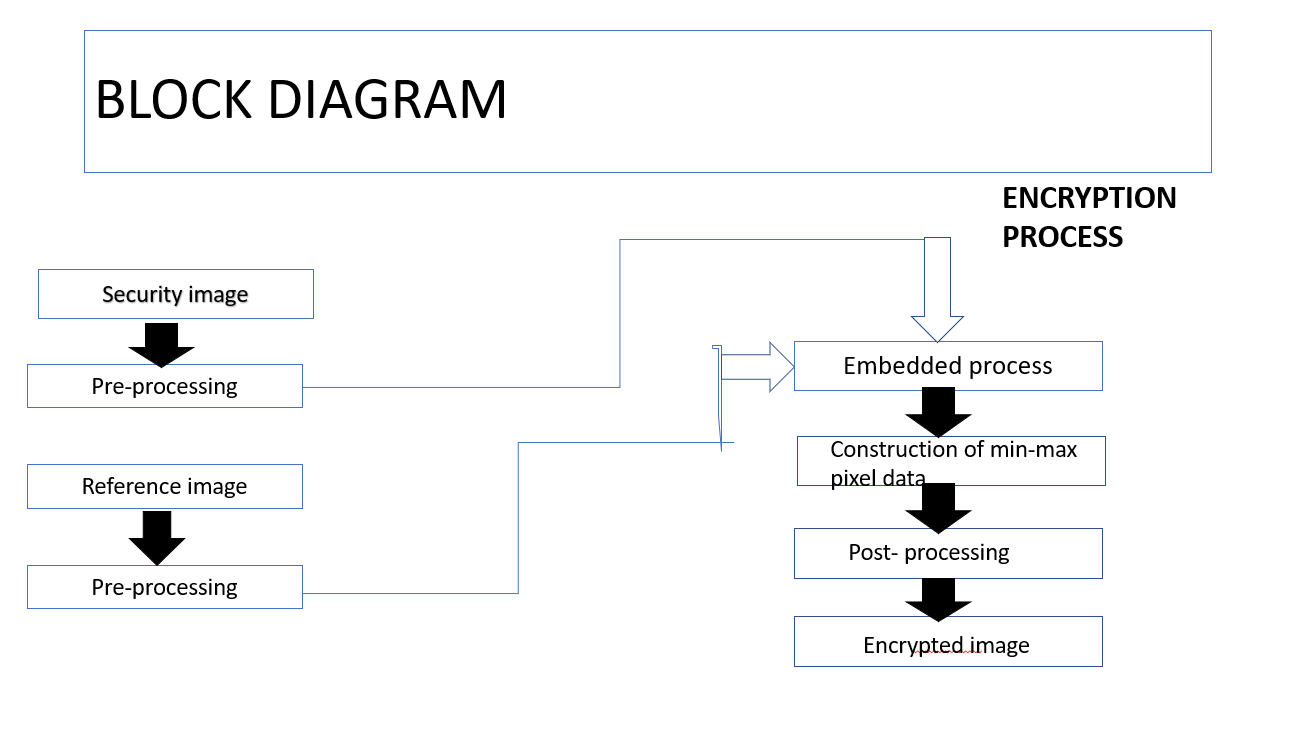
Therefore, in this paper, we design an information hiding method as part of the image enhancement process. In other words, data can be inserted into the image while executing the image enhancement steps. As a proof of concept, the proposed method is demonstrated by using the Median Filter.

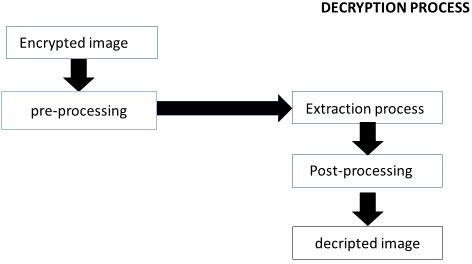
The need of integrity and confidentiality of data has increased along with the significant development of the information technology. Data must be protected before being sent to other parties in a network without being worried about illegal access of unauthorized users. In order to achieve this secure condition, some methods have been introduced, such as data hiding. In the implementation level, this can be combined with other securing methods, for example cryptography. That combination step are taken with aim of providing more information protection in terms of confidentiality and reliability.

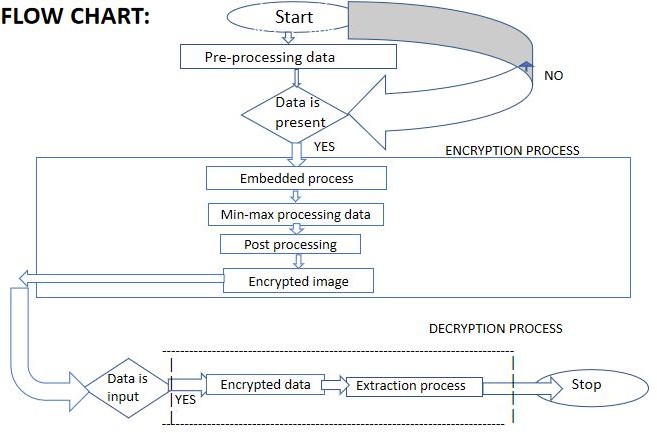
Digital images are subject to a wide variety of distortions during acquisition, processing, compression, storage, transmission and reproduction, any of which may result in a degradation of visual quality. For applications in which images are ultimately to be viewed by human beings, the only “correct” method of quantifying visual image quality is through subjective evaluation. In practice, however, subjective evaluation is usually too inconvenient, time-consuming and expensive. The goal of research in objective image quality assessment is to develop quantitative measures that can automatically predict perceived image quality.

Image enhancement, as its name implies, improves the quality of image by means of emphasizing the desired details or removing/suppressing the undesired noisy/irrelevant information. Image enhancement plays a crucial role in digital image processing, and it has become one of the processes that takes place in any smart device after an image is captured. Some commonly deployed image enhancement techniques include contrast enhancement, image

sharpening, smoothing, noise removal, to name a few. Each enhancement technique can either act as a standalone process to enhance the appearance of an image, or utilized as a pre-processing step before further analysis is carried out. In fact, image enhancement (commonly known as filter) is often utilized to enhance image quality before it is shared online on social media or communicated for consumption by a wider audience.



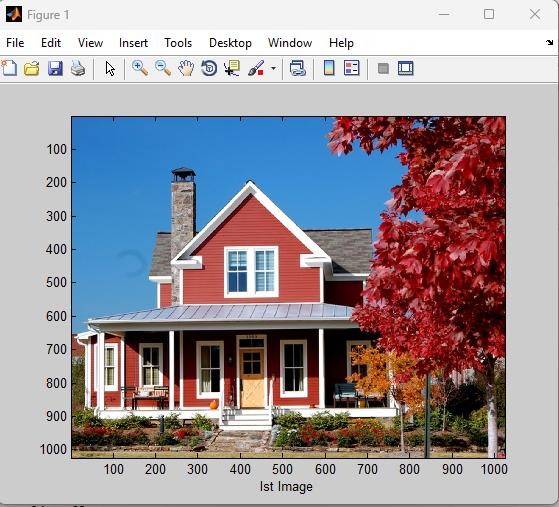




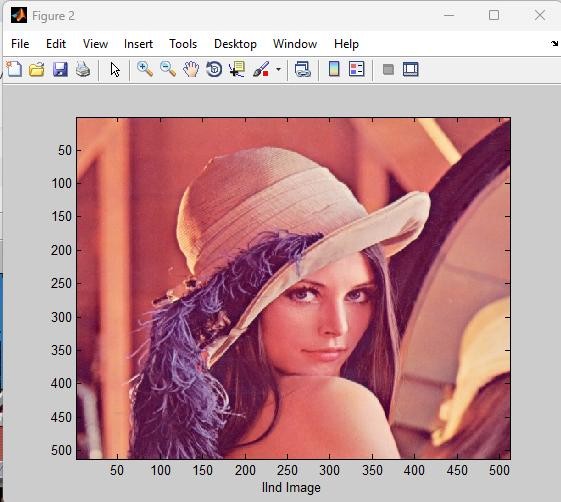
### CHAPTER 6 SIMULATION RESULTS

**ENCRYPTION PROCESS:**

* Select one reference image which is in the form of JPG format.

 Fig:6.1 reference image

* Select one secret image which in the format of JPG.

Fig:6.2 Secret image

* Combination of reference image and secret image which is encrypted image as shown in fig-6.3

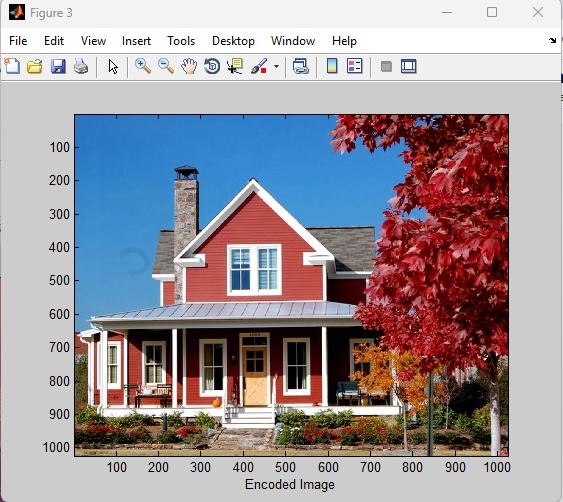


Fig:6.3 Encrypted image

**DECRYPTION PROCESS:**

* After encryption process, encrypted image which is in bmp format taken as input in decryption process
* After decryption process we get decoded image of reference image and secret image.

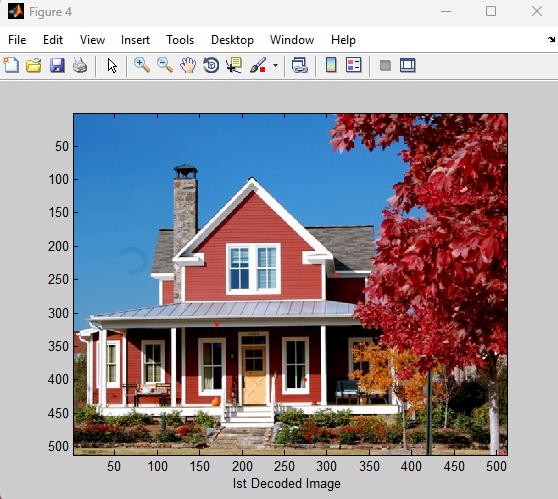


Fig:6.4 First decoded image

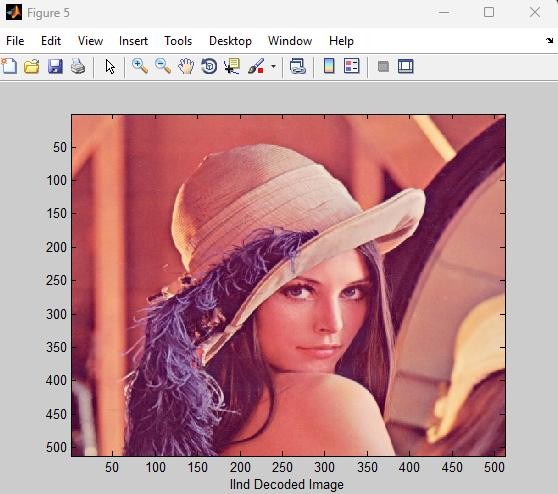


Fig:6.5 second decoded image

# CHAPTER 7

**CONCLUSION AND FUTURE SCOPE**

In this project, data is embedded while executing the image enhancement steps performed on image. In particular, pixels within the neighbour for a target pixel is collected and sorted in ascending order. This pixels list is then partitioned and each partition is assigned to represent data in bit(s). Performance of the proposed method is evaluated to identify trends and patterns. Trade-offs among embedding capacity, image quality and data extraction error rate are further discussed. Suggestions are provided to reduce the data extraction error rate and to enhance the robustness of the proposed method against unauthorized viewing. Under various settings, the proposed method can embed 0.99bpp when w = 3, and the lowest data extraction error rate is ∼ 3.5% when w = 7. As future work, we will consider color image and investigate into the correlation among the RGB color channels for deploying the proposed filtering-embedding method

An ideal steganographic algorithm should have high precision, a higher level of security with good embedded capacity. Simplicity and cost effectiveness should also be considered. Thus, it is necessary to investigate steganographic problems and solve with different approaches using different domains. Compressing the secret message file enables us to have bigger one with the same pay load capacity and therefore achieve better results. Encrypting the stegno file before transmitting it through the communication channel will ensure more security to the steganography approach.

Irrespective of these limitations and future scope, these thesis will be beneficial to the researches who are perusing researches in the field of computer science, engineering and information technology. The scheme developed here or better then other existing scheme with respect to the payload, visual quality, reversibility and security. This work would give the prospective researches an excellent insight in to the various new concepts used in steganography.

# REFERENCES

1. X. Li, W. Zhang, X. Gui, and B. Yang, “Efficient reversible data hiding based on multiple histograms modification,” IEEE Transactions on Information Forensics and Security, vol. 10, no. 9, pp. 2016–2027, Sep. 2015.
2. B. Ou, X. Li, Y. Zhao, R. Ni, and Y. Shi, “Pairwise prediction-error expansion for efficient reversible data hiding,” IEEE Transactions on Image Processing, vol. 22, no. 12, pp. 5010–5021, Dec 2013.
3. T. Zhang, X. Li, W. Qi, and Z. Guo, “Location-based pvo and adaptive pairwise modification for efficient reversible data hiding,” IEEE Transactions on Information Forensics and Security, pp. 1–1, 2020.
4. S. Esakkirajan, T. Veerakumar, A. N. Subramanyam, and C. H. PremChand, “Removal of high density salt and pepper noise through modified decision based unsymmetric trimmed median filter,” IEEE Signal Processing Letters, vol. 18, no. 5, pp. 287–290, May 2011.
5. D. Martin, C. Fowlkes, D. Tal, and J. Malik, “A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics,” in Proc. 8th Int’l Conf. Computer Vision, July 2001, vol. 2, pp. 416–423.

Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600–612, April 2004.

**APPENDIX**

**1.Software Introduction**:

### Introduction to MATLAB

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include

* + - Math and computation
    - Algorithm development
    - Data acquisition
    - Modeling, simulation, and prototyping
    - Data analysis, exploration, and visualization
    - Scientific and engineering graphics
    - Application development, including graphical user interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar non interactive language such as C or FORTRAN.

The name MATLAB stands for matrix laboratory. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects. Today, MATLAB engines incorporate the LAPACK and BLAS libraries, embedding the state of the art in software for matrix computation.

MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high- productivity research, development, and analysis.

MATLAB features a family of add-on application-specific solutions called toolboxes. Very important to most uses of MATLAB, toolboxes allow you to learn and apply specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M – files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

### The MATLAB system:

The MATLAB system consists of five main parts

### Development Environment:

This is the set of tools and facilities that help you use MATLAB functions and files. Many of these tools are graphical user interfaces. It includes the MATLAB desktop and command window, a command history, an editor and debugger, and browsers for viewing help, the workspace, files, and the search path.

### The MATLAB Mathematical Function Library:

This is a vast collection of computational algorithms ranging from elementary functions, like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix Eigen values, Bessel functions, and fast Fourier transforms.

### The MATLAB Language:

This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both “programming in the small” to rapidly create quick and dirty throw-away programs, and “programming in the large” to create large and complex application programs.

### Graphics:

MATLAB has extensive facilities for displaying vectors and matrices as graphs, as well as annotating and printing these graphs. It includes high-level functions for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level functions that allow you to fully customize the appearance of graphics as well as to build complete graphical user interfaces on your MATLAB applications.

### The MATLAB Application Program Interface (API):

This is a library that allows you to write C and FORTRAN programs that interact with MATLAB. It includes facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

Various toolboxes are there in MATLAB for computing recognition techniques, but we are using **IMAGE PROCESSING** toolbox.

### GRAPHICAL USER INTERFACE (GUI):

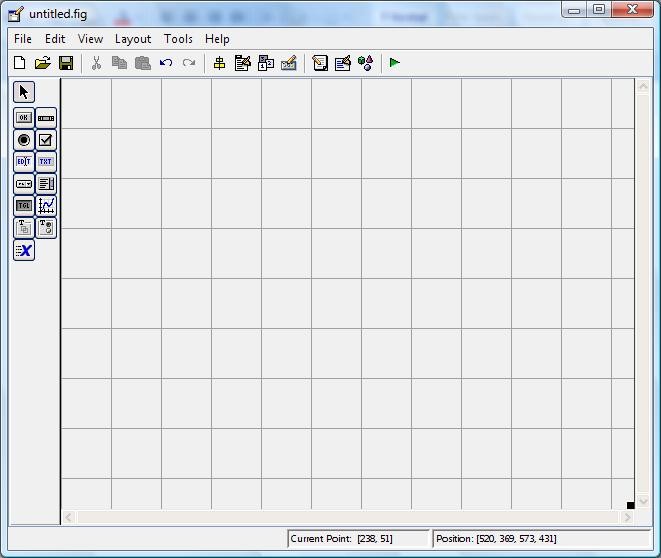
MATLAB’s Graphical User Interface Development Environment (GUIDE) provides a rich set of tools for incorporating graphical user interfaces (GUIs) in M-functions. Using GUIDE, the processes of laying out a GUI (i.e., its buttons, pop-up menus, etc.)and programming the operation of the GUI are divided conveniently into two easily managed and relatively independent tasks. The resulting graphical M-function is composed of two identically named (ignoring extensions) files:

* A file with extension .fig, called a FIG-file that contains a complete graphical description of all the function’s GUI objects or elements and their spatial arrangement. A FIG-file contains binary data that does not need to be parsed when he associated GUI-based M-function is executed.A file with extension .m, called a GUI M-file, which contains the code that controls the GUI operation. This file includes functions that are called when the GUI is launched and exited, and callback functions that are executed when a user interacts with GUI objects for example, when a button is pushed.

To launch GUIDE from the MATLAB command window, type guide filename

Where filename is the name of an existing FIG-file on the current path. If filename is omitted,

GUIDE opens a new (i.e., blank) window.



A graphical user interface (GUI) is a graphical display in one or more windows containing controls, called components that enable a user to perform interactive tasks. The user of the GUI does not have to create a script or type commands at the command line to accomplish the tasks. Unlike coding programs to accomplish tasks, the user of a GUI need not understand the details of how the tasks are performed.

GUI components can include menus, toolbars, push buttons, radio buttons, list boxes, and sliders just to name a few. GUIs created using MATLAB tools can also perform any type of

computation, read and write data files, communicate with other GUIs, and display data as tables or as plots.

### Getting Started

If you are new to MATLAB, you should start by reading Manipulating Matrices. The most important things to learn are how to enter matrices, how to use the: (colon) operator, and how to invoke functions. After you master the basics, you should read the rest of the sections below and run the demos.

At the heart of MATLAB is a new language you must learn before you can fully exploit its power. You can learn the basics of MATLAB quickly, and mastery comes shortly after. You will be rewarded with high productivity, high-creativity computing power that will change the way you work.

* + 1. **Introduction** - describes the components of the MATLAB system.

**1.4.2 Development Environment** - introduces the MATLAB development environment, including information about tools and the MATLAB desktop.

**1.4.3 Manipulating Matrices** - introduces how to use MATLAB to generate matrices and perform mathematical operations on matrices.

**1.4.4 Graphics** - introduces MATLAB graphic capabilities, including information about plotting data, annotating graphs, and working with images.

**1.4.5 Programming with MATLAB** - describes how to use the MATLAB language to create scripts and functions, and manipulate data structures, such as cell arrays and multidimensional arrays.

### 1.5DEVELOPMENT ENVIRONMENT 1.5.1Introduction

This chapter provides a brief introduction to starting and quitting MATLAB, and the tools and functions that help you to work with MATLAB variables and files. For more information

about the topics covered here, see the corresponding topics under Development Environment in the MATLAB documentation, which is available online as well as in print.

### Starting and Quitting MATLAB

**1.5.2 Starting MATLAB**

On a Microsoft Windows platform, to start MATLAB, double-click the MATLAB shortcut icon on your Windows desktop.On a UNIX platform, to start MATLAB, type matlab at the operating system prompt. After starting MATLAB, the MATLAB desktop opens - see MATLAB Desktop.

You can change the directory in which MATLAB starts, define startup options including running a script upon startup, and reduce startup time in some situations.

### 1.5.3 Quitting MATLAB

To end your MATLAB session, select Exit MATLAB from the File menu in the desktop, or type quit in the Command Window. To execute specified functions each time MATLAB quits, such as saving the workspace, you can create and run a finish.m script.

### 1.5.4 MATLAB Desktop

When you start MATLAB, the MATLAB desktop appears, containing tools (graphical user interfaces) for managing files, variables, and applications associated with MATLAB.The first time MATLAB starts, the desktop appears as shown in the following illustration, although your Launch Pad may contain different entries.

You can change the way your desktop looks by opening, closing, moving, and resizing the tools in it. You can also move tools outside of the desktop or return them back inside the desktop (docking). All the desktop tools provide common features such as context menus and keyboard shortcuts.

You can specify certain characteristics for the desktop tools by selecting Preferences from the File menu. For example, you can specify the font characteristics for Command Window text. For more information, click the Help button in the Preferences dialog box.

### 1.5.5 Desktop Tools

This section provides an introduction to MATLAB's desktop tools. You can also use MATLAB functions to perform most of the features found in the desktop tools. The tools are:

* Current Directory Browser
* Workspace Browser
* Array Editor
* Editor/Debugger
* Command Window
* Command History
* Launch Pad
* Help Browser

### Command Window

Use the Command Window to enter variables and run functions and M-files.

### Command History

Lines you enter in the Command Window are logged in the Command History window. In the Command History, you can view previously used functions, and copy and execute selected lines. To save the input and output from a MATLAB session to a file, use the diary function.

### Running External Programs

You can run external programs from the MATLAB Command Window. The exclamation point character! is a shell escape and indicates that the rest of the input line is a command to the operating system. This is useful for invoking utilities or running other programs without quitting MATLAB. On Linux, for example,!emacs magik.m invokes an editor called emacs for a file named magik.m. When you quit the external program, the operating system returns control to MATLAB.

### Launch Pad

MATLAB's Launch Pad provides easy access to tools, demos, and documentation.

### Help Browser

Use the Help browser to search and view documentation for all your Math Works products. The Help browser is a Web browser integrated into the MATLAB desktop that displays HTML documents.

To open the Help browser, click the help button in the toolbar, or type helpbrowser in the Command Window. The Help browser consists of two panes, the Help Navigator, which you use to find information, and the display pane, where you view the information.

### Help Navigator

Use to Help Navigator to find information. It includes:

**Product filter** - Set the filter to show documentation only for the products you specify.

**Contents tab** - View the titles and tables of contents of documentation for your products.

**Index tab** - Find specific index entries (selected keywords) in the MathWorks documentation for your products.

**Search tab** - Look for a specific phrase in the documentation. To get help for a specific function, set the Search type to Function Name.

**Favorites tab** - View a list of documents you previously designated as favorites.

### Display Pane

After finding documentation using the Help Navigator, view it in the display pane. While viewing the documentation, you can:

**Browse to other pages** - Use the arrows at the tops and bottoms of the pages, or use the back and forward buttons in the toolbar.

**Bookmark pages** - Click the Add to Favorites button in the toolbar.

**Print pages** - Click the print button in the toolbar.

**Find a term in the page** - Type a term in the Find in page field in the toolbar and click Go.

Other features available in the display pane are: copying information, evaluating a selection, and viewing Web pages.

### Current Directory Browser

MATLAB file operations use the current directory and the search path as reference points.

Any file you want to run must either be in the current directory or on the search path.

### Search Path

To determine how to execute functions you call, MATLAB uses a search path to find M-files and other MATLAB-related files, which are organized in directories on your file system. Any file you want to run in MATLAB must reside in the current directory or in a directory that is on the search path. By default, the files supplied with MATLAB and MathWorks toolboxes are included in the search path.

### Workspace Browser

The MATLAB workspace consists of the set of variables (named arrays) built up during a MATLAB session and stored in memory. You add variables to the workspace by using functions, running M-files, and loading saved workspaces.

To view the workspace and information about each variable, use the Workspace browser, or use the functions who and whos.

To delete variables from the workspace, select the variable and select Delete from the Edit menu. Alternatively, use the clear function.

The workspace is not maintained after you end the MATLAB session. To save the workspace to a file that can be read during a later MATLAB session, select Save Workspace As from the File menu, or use the save function. This saves the workspace to a binary file called a MAT-file, which has a .mat extension. There are options for saving to different formats. To read in a MAT- file, select Import Data from the File menu, or use the load function.

### Array Editor

Double-click on a variable in the Workspace browser to see it in the Array Editor. Use the Array Editor to view and edit a visual representation of one- or two-dimensional numeric arrays, strings, and cell arrays of strings that are in the workspace.

### Editor/Debugger

Use the Editor/Debugger to create and debug M-files, which are programs you write to runMATLAB functions. The Editor/Debugger provides a graphical user interface for basic text editing, as well as for M-file debugging.

You can use any text editor to create M-files, such as Emacs, and can use preferences (accessible from the desktop File menu) to specify that editor as the default. If you use another editor, you can still use the MATLAB Editor/Debugger for debugging, or you can use debugging functions, such as dbstop, which sets a breakpoint.

If you just need to view the contents of an M-file, you can display it in the Command Window by using the type function.

### MANIPULATING MATRICES

* + 1. **Entering Matrices**

The best way for you to get started with MATLAB is to learn how to handle matrices. Start MATLAB and follow along with each example.

You can enter matrices into MATLAB in several different ways:

* + - 1. Enter an explicit list of elements.
      2. Load matrices from external data files.
      3. Generate matrices using built-in functions.
      4. Create matrices with your own functions in M-files.

Start by entering Dürer's matrix as a list of its elements. You have only to follow a few basic conventions:

* + - 1. Separate the elements of a row with blanks or commas.
      2. Use a semicolon, ; , to indicate the end of each row.
      3. Surround the entire list of elements with square brackets, [ ]. To enter Dürer's matrix, simply type in the Command Window

A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]

MATLAB displays the matrix you just entered.

A =

|  |  |  |  |
| --- | --- | --- | --- |
| 16 | 3 | 2 | 13 |
| 5 | 10 | 11 | 8 |
| 9 | 6 | 7 | 12 |
| 4 | 15 | 14 | 1 |

This exactly matches the numbers in the engraving. Once you have entered the matrix, it is automatically remembered in the MATLAB workspace. You can refer to it simply as A.

### Expressions

Like most other programming languages, MATLAB provides mathematical expressions, but unlike most programming languages, these expressions involve entire matrices. The building blocks of expressions are:

* + - 1. Variables
      2. Numbers
      3. Operators
      4. Functions

### Variables

MATLAB does not require any type declarations or dimension statements. When MATLAB encounters a new variable name, it automatically creates the variable and allocates the appropriate amount of storage. If the variable already exists, MATLAB changes its contents and, if necessary, allocates new storage. For example,

num\_students = 25

Creates a 1-by-1 matrix named num\_students and stores the value 25 in its single element.

Variable names consist of a letter, followed by any number of letters, digits, or underscores. MATLAB uses only the first 31 characters of a variable name. MATLAB is case sensitive; it distinguishes between uppercase and lowercase letters. A and a are not the same variable. To view the matrix assigned to any variable, simply enter the variable name.

### Numbers

MATLAB uses conventional decimal notation, with an optional decimal point and leading plus or minus sign, for numbers. Scientific notation uses the letter e to specify a power-of-ten scale factor. Imaginary numbers use either i or j as a suffix. Some examples of legal numbers are

|  |  |  |
| --- | --- | --- |
| 3 | -99 | 0.0001 |
| 9.6397238 | 1.60210e-20 | 6.02252e23 |
| 1i | -3.14159j | 3e5i |

All numbers are stored internally using the long format specified by the IEEE floating-point standard. Floating-point numbers have a finite precision of roughly 16 significant decimal digits and a finite range of roughly 10-308 to 10+308.

### Operators

Expressions use familiar arithmetic operators and precedence rules.

|  |  |
| --- | --- |
| + | Addition |
| - | Subtraction |
| \* | Multiplication |
| / | Division |
| \ | Left division (described in "Matrices and Linear Algebra" in Using MATLAB) |
| ^ | Power |
| ' | Complex conjugate transpose |
| ( ) | Specify evaluation order |

### Functions

MATLAB provides a large number of standard elementary mathematical functions, including

abs, sqrt, exp, and

|  |  |
| --- | --- |
| Pi | 3.14159265... |
| I | Imaginary unit, √-1 |
| I | Same as i |
| Eps | Floating-point relative precision, 2-52 |
| Realmin | Smallest floating-point number, 2-1022 |
| Realmax | Largest floating-point number, (2- **ε**)21023 |
| Inf | Infinity |
| NaN | Not-a-number |

sin. Taking

the square

root or logarith m of a negative number is not an

error; the appropriate complex result is produced automatically. MATLAB also provides many more

advanced mathematical functions, including Bessel and gamma functions. Most of these functions accept complex arguments. For a list of the elementary mathematical functions, type help elfun, For a list of more advanced mathematical and matrix functions, type help specfun help elmat

Some of the functions, like sqrt and sin, are built-in. They are part of the MATLAB core so they are very efficient, but the computational details are not readily accessible. Other functions, like gamma and sinh, are implemented in M-files. You can see the code and even modify it if you want. Several special functions provide values of useful constants.

### GUI

A graphical user interface (GUI) is a user interface built with graphical objects, such as buttons, text fields, sliders, and menus. In general, these objects already have meanings to most computer users. For example, when you move a slider, a value changes; when you press an OK button, your settings are applied and the dialog box is dismissed. Of course, to leverage this built-in familiarity, you must be consistent in how you use the various GUI-building components.

Applications that provide GUIs are generally easier to learn and use since the person using the application does not need to know what commands are available or how they work. The action that results from a particular user action can be made clear by the design of the interface.

The sections that follow describe how to create GUIs with MATLAB. This includes laying out the components, programming them to do specific things in response to user actions, and saving and launching the GUI; in other words, the mechanics of creating GUIs. This documentation does not attempt to cover the "art" of good user interface design, which is an entire field unto itself. Topics covered in this section include:

### Creating GUIs with GUIDE

MATLAB implements GUIs as figure windows containing various styles of uicontrol objects. You must program each object to perform the intended action when activated by the user of the

GUI. In addition, you must be able to save and launch your GUI. All of these tasks are simplified by GUIDE, MATLAB's graphical user interface development environment.

### GUI Development Environment

The process of implementing a GUI involves two basic task.

* + - 1. Laying out the GUI components
      2. Programming the GUI components

GUIDE primarily is a set of layout tools. However, GUIDE also generates an M-file that contains code to handle the initialization and launching of the GUI. This M-file provides a framework for the implementation of the callbacks - the functions that execute when users activate components in the GUI.

### The Implementation of a GUI

While it is possible to write an M-file that contains all the commands to lay out a GUI, it is easier to use GUIDE to lay out the components interactively and to generate two files that save and launch the GUI:

**A FIG-file** - contains a complete description of the GUI figure and all of its children (uicontrols and axes), as well as the values of all object properties. **An M-file** - contains the functions that launch and control the GUI and the callbacks, which are defined as subfunctions. This M-file is referred to as the application M-file in this documentation.

Note that the application M-file does not contain the code that lays out the uicontrols; this information is saved in the FIG-file.

The following diagram illustrates the parts of a GUI implementation.

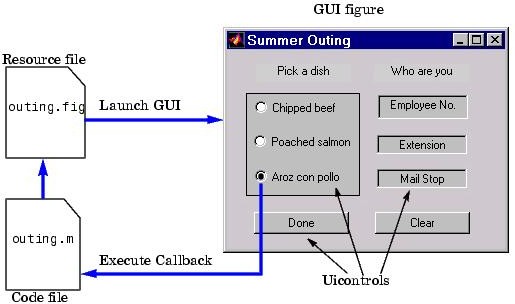


FIG 3.7.2 graphical user blocks

### Features of the GUIDE-Generated Application M-Fil

GUIDE simplifies the creation of GUI applications by automatically generating an M-file framework directly from your layout. You can then use this framework to code your application M-file. This approach provides a number of advantages:

The M-file contains code to implement a number of useful features (see Configuring Application Options for information on these features). The M-file adopts an effective approach to managing object handles and executing callback routines (see Creating and Storing the Object Handle Structure for more information). The M-files provides a way to manage global data (see Managing GUI Data for more information).

The automatically inserted subfunction prototypes for callbacks ensure compatibility with future releases. For more information, see Generating Callback Function Prototypes for information on syntax and arguments.

You can elect to have GUIDE generate only the FIG-file and write the application M-file yourself. Keep in mind that there are no uicontrol creation commands in the application M-file; the layout information is contained in the FIG-file generated by the Layout Editor.

### Beginning the Implementation Process

To begin implementing your GUI, proceed to the following sections:

### Getting Started with GUIDE - the basics of using GUIDE.

**Selecting GUIDE Application Options** - set both FIG-file and M-file options.

**Using the Layout Editor** - begin laying out the GUI.

**Understanding the Application M-File** - discussion of programming techniques used in the application M-file.

**Application Examples** - a collection of examples that illustrate techniques which are useful for implementing GUIs.

### Command-Line Accessibility

When MATLAB creates a graph, the figure and axes are included in the list of children of their respective parents and their handles are available through commands such as findobj, set, and get. If you issue another plotting command, the output is directed to the current figure and axes.

GUIs are also created in figure windows. Generally, you do not want GUI figures to be available as targets for graphics output, since issuing a plotting command could direct the output to the GUI figure, resulting in the graph appearing in the middle of the GUI.

In contrast, if you create a GUI that contains an axes and you want commands entered in the command window to display in this axes, you should enable command-line access.

### User Interface Control

The Layout Editor component palette contains the user interface controls that you can use in your GUI. These components are MATLAB uicontrol objects and are programmable via their Callback properties. This section provides information on these components.

* Push Buttons
* Sliders
* Toggle Buttons
* Frames
* Radio Buttons
* Listboxes
* Checkboxes
* Popup Menus
* Edit Text
* Axes
* Static Text
* Figures

### Push Buttons

Push buttons generate an action when pressed (e.g., an OK button may close a dialog box and apply settings). When you click down on a push button, it appears depressed; when you release the mouse, the button's appearance returns to its nondepressed state; and its callback executes on the button up event.

### Properties to Set

**String** - set this property to the character string you want displayed on the push button.

**Tag** - GUIDE uses the Tag property to name the callback subfunction in the application M-file. Set Tag to a descriptive name (e.g., close\_button) before activating the GUI.

### Programming the Callback

When the user clicks on the push button, its callback executes. Push buttons do not return a value or maintain a state.

### Toggle Buttons

Toggle buttons generate an action and indicate a binary state (e.g., on or off). When you click on a toggle button, it appears depressed and remains depressed when you release the mouse

button, at which point the callback executes. A subsequent mouse click returns the toggle button to the nondepressed state and again executes its callback.

### Programming the Callback

The callback routine needs to query the toggle button to determine what state it is in. MATLAB sets the Value property equal to the Max property when the toggle button is depressed (Max is 1 by default) and equal to the Min property when the toggle button is not depressed (Min is 0 by default).

### From the GUIDE Application M-File

The following code illustrates how to program the callback in the GUIDE application M-file. function varargout = togglebutton1\_Callback(h,eventdata,handles,varargin)

button\_state = get(h,'Value'); if button\_state == get(h,'Max')

% toggle button is pressed elseif button\_state == get(h,'Min')

% toggle button is not pressed end

### Adding an Image to a Push Button or Toggle Button

Assign the CData property an m-by-n-by-3 array of RGB values that define a truecolor image. For example, the array a defines 16-by-128 truecolor image using random values between 0 and 1 (generated by rand).

a(:,:,1) = rand(16,128);

a(:,:,2) = rand(16,128);

a(:,:,3) = rand(16,128);

set(h,'CData',a)

### Radio Buttons

Radio buttons are similar to checkboxes, but are intended to be mutually exclusive within a group of related radio buttons (i.e., only one button is in a selected state at any given time). To activate a radio button, click the mouse button on the object. The display indicates the state of the button.

### Implementing Mutually Exclusive Behavior

Radio buttons have two states - selected and not selected. You can query and set the state of a radio button through its Value property:

Value = Max, button is selected. Value = Min, button is not selected.

To make radio buttons mutually exclusive within a group, the callback for each radio button must set the Value property to 0 on all other radio buttons in the group. MATLAB sets the Value property to 1 on the radio button clicked by the user.

The following subfunction, when added to the application M-file, can be called by each radio button callback. The argument is an array containing the handles of all other radio buttons in the group that must be deselected.

function mutual\_exclude(off) set(off,'Value',0)

### Obtaining the Radio Button Handles.

The handles of the radio buttons are available from the handles structure, which contains the handles of all components in the GUI. This structure is an input argument to all radio button callbacks.

The following code shows the call to mutual\_exclude being made from the first radio button's callback in a group of four radio buttons.

function varargout = radiobutton1\_Callback(h,eventdata,handles,varargin) off = [handles.radiobutton2,handles.radiobutton3,handles.radiobutton4]; mutual\_exclude(off)

% Continue with callback

.

.

.

After setting the radio buttons to the appropriate state, the callback can continue with its implementation-specific tasks.

### Checkboxes

Check boxes generate an action when clicked and indicate their state as checked or not checked. Check boxes are useful when providing the user with a number of independent choices that set a mode (e.g., display a toolbar or generate callback function prototypes).

The Value property indicates the state of the check box by taking on the value of the Max or Min property (1 and 0 respectively by default):

Value = Max, box is checked. Value = Min, box is not checked.

You can determine the current state of a check box from within its callback by querying the state of its Value property, as illustrated in the following example:

function checkbox1\_Callback(h,eventdata,handles,varargin) if (get(h,'Value') == get(h,'Max'))

% then checkbox is checked-take approriate action else

% checkbox is not checked-take approriate action end

### Edit Text

Edit text controls are fields that enable users to enter or modify text strings. Use edit text when you want text as input. The String property contains the text entered by the user.

To obtain the string typed by the user, get the String property in the callback. function edittext1\_Callback(h,eventdata, handles,varargin)

user\_string = get(h,'string');

% proceed with callback...

### Obtaining Numeric Data from an Edit Test Component

MATLAB returns the value of the edit text String property as a character string. If you want users to enter numeric values, you must convert the characters to numbers. You can do this using the str2double command, which converts strings to doubles. If the user enters non-numeric characters, str2double returns NaN.

You can use the following code in the edit text callback. It gets the value of the String property and converts it to a double. It then checks if the converted value is NaN, indicating the user entered a non-numeric character (isnan) and displays an error dialog (errordlg).

function edittext1\_Callback(h,eventdata,handles,varargin) user\_entry = str2double(get(h,'string'));

if isnan(user\_entry)

errordlg('You must enter a numeric value','Bad Input','modal') end

% proceed with callback...

### Triggering Callback Execution

On UNIX systems, clicking on the menubar of the figure window causes the edit text callback to execute. However, on Microsoft Windows systems, if an editable text box has focus, clicking on the menubar does not cause the editable text callback routine to execute. This behavior is consistent with the respective platform conventions. Clicking on other components in the GUI execute the callback.

### Static Text

Static text controls displays lines of text. Static text is typically used to label other controls, provide directions to the user, or indicate values associated with a slider. Users cannot change static text interactively and there is no way to invoke the callback routine associated with it

### Frames

Frames are boxes that enclose regions of a figure window. Frames can make a user interface easier to understand by visually grouping related controls. Frames have no callback routines associated with them and only uicontrols can appear within frames (axes cannot).

### Placing Components on Top of Frames

Frames are opaque. If you add a frame after adding components that you want to be positioned within the frame, you need to bring forward those components. Use the Bring to Front and Send to Back operations in the Layout menu for this purpose.

### List Boxes

List boxes display a list of items and enable users to select one or more items.

The String property contains the list of strings displayed in the list box. The first item in the list has an index of 1.

The Value property contains the index into the list of strings that correspond to the selected item. If the user selects multiple items, then Value is a vector of indices. By default, the first item

in the list is highlighted when the list box is first displayed. If you do not want any item highlighted, then set the Value property to empty.

The ListboxTop property defines which string in the list displays as the top most item when the list box is not large enough to display all list entries. ListboxTop is an index into the array of strings defined by the String property and must have a value between 1 and the number of strings. Noninteger values are fixed to the next lowest integer

### Single or Multiple Selection

The values of the Min and Max properties determine whether users can make single or multiple selections:

If Max - Min > 1, then list boxes allow multiple item selection.

If Max - Min <= 1, then list boxes do not allow multiple item selection.

### Selection Type

Listboxes differentiate between single and double clicks on an item and set the figure SelectionType property to normal or open accordingly. See Triggering Callback Execution for information on how to program multiple selection.

### Triggering Callback Execution

MATLAB evaluates the list box's callback after the mouse button is released or a keypress event (including arrow keys) that changes the Value property (i.e., any time the user clicks on an item, but not when clicking on the list box scrollbar). This means the callback is executed after the first click of a double-click on a single item or when the user is making multiple selections. In these situations, you need to add another component, such as a Done button (push button) and program its callback routine to query the list box Value property (and possibly the figure SelectionType property) instead of creating a callback for the list box. If you are using the automatically generated application M-file option, you need to either:

Set the list box Callback property to the empty string ('') and remove the callback subfunction from the application M-file. Leave the callback subfunction stub in the application M-file so that no code executes when users click on list box items.

The first choice is best if you are sure you will not use the list box callback and you want to minimize the size and efficiency of the application M-file. However, if you think you may want to define a callback for the list box at some time, it is simpler to leave the callback stub in the M- file.

### Popup Menus

Popup menus open to display a list of choices when users press the arrow. The String property contains the list of string displayed in the popup menu. The Value property contains the index into the list of strings that correspond to the selected item. When not open, a popup menu displays the current choice, which is determined by the index contained in the Value property. The first item in the list has an index of 1.

Popup menus are useful when you want to provide users with a number of mutually exclusive choices, but do not want to take up the amount of space that a series of radio buttons requires.

### Programming the Popup Menu

You can program the popup menu callback to work by checking only the index of the item selected (contained in the Value property) or you can obtain the actual string contained in the selected item.

This callback checks the index of the selected item and uses a switch statement to take action based on the value. If the contents of the popup menu is fixed, then you can use this approach.

function varargout = popupmenu1\_Callback(h,eventdata,handles,varargin) val = get(h,'Value');

switch val

case 1

% The user selected the first item case 2

% The user selected the second item

% etc.

This callback obtains the actual string selected in the popup menu. It uses the value to index into the list of strings. This approach may be useful if your program dynamically loads the contents of the popup menu based on user action and you need to obtain the selected string. Note that it is necessary to convert the value returned by the String property from a cell array to a string.

function varargout = popupmenu1\_Callback(h,eventdata,handles,varargin) val = get(h,'Value');

string\_list = get(h,'String');

selected\_string = string\_list{val}; % convert from cell array to string

% etc.

### Enabling or Disabling Controls

You can control whether a control responds to mouse button clicks by setting the Enable property. Controls have three states:

on - The control is operational

off - The control is disabled and its label (set by the string property) is grayed out.

inactive - The control is disabled, but its label is not grayed out.

When a control is disabled, clicking on it with the left mouse button does not execute its callback routine. However, the left-click causes two other callback routines to execute: First the figure WindowButtonDownFcn callback executes. Then the control's ButtonDownFcn callback executes. A right mouse button click on a disabled control posts a context menu, if one is defined for that control. See the Enable property description for more details.

### Axes

Axes enable your GUI to display graphics (e.g., graphs and images). Like all graphics objects, axes have properties that you can set to control many aspects of its behavior and appearance. See Axes Properties for general information on axes objects.

### Axes Callbacks

Axes are not uicontrol objects, but can be programmed to execute a callback when users click a mouse button in the axes. Use the axes ButtonDownFcn property to define the callback.

### Plotting to Axes in GUIs

GUIs that contain axes should ensure the Command-line accessibility option in the Application Options dialog is set to Callback (the default). This enables you to issue plotting commands from callbacks without explicitly specifying the target axes.

### GUIs with Multiple Axes

If a GUI has multiple axes, you should explicitly specify which axes you want to target when you issue plotting commands. You can do this using the axes command and the handles structure. For example,

axes(handles.axes1)

makes the axes whose Tag property is axes1 the current axes, and therefore the target for plotting commands. You can switch the current axes whenever you want to target a different axes. See GUI with Multiple Axes for and example that uses two axes.

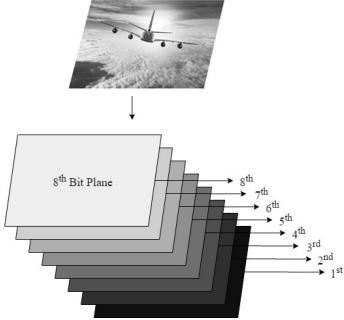
### Figure

Figures are the windows that contain the GUI you design with the Layout Editor. See the description of figure properties for information on what figure characteristics you can control.

In this section, we discuss the details of the flow of our proposed RDH scheme. Initially, we introduce the tuned prediction model, which comprises of the predictions of two regression models with different training data to predict the original pixel values of the image. In the next part, we discuss the concept and significance of the error map for complete reversibility. We then describe the process of embedding of secret data in the image by the sender. Lastly, the secret data is extracted, and the image recovery can be achieved by the receiver.

### β-Tuned prediction model

For recovery of the image, the receiver needs to predict the original value of pixels which contains the hidden data. This recovery would be possible due to the presence of regularities in images like spatially correlated neighbouring pixels. This property is used to predict the original value of the pixels. An example of a simple grayscale image of size H\*W is illustrated to show how the sender can hide secret data in an image. In the image, each pixel is made up of 8 bits either containing 0 or 1. An image is just a matrix of these pixels, and it is easy to imagine an image as a stack of eight single bit matrices or planes called bit planes, shown in Fig. 1. A single bit plane is just a matrix of 0 and 1. Sender then chooses a particular bit plane to hide the secret data, here hiding means simply overwriting the hidden data over the bit plane. There is a tradeoff between image distortion and pixel value prediction. Choosing a bit plane near Least significant bit (LSB) will provide the least distorted image, but the prediction would be difficult. Bit plane near Most significant bit (MSB) will provide accurate prediction, but its distortion is very high. For this reason, we have taken 3rd, 4th, 5th and 6th bit planes for validation of our results.



### Fig: Bit Planes of an image

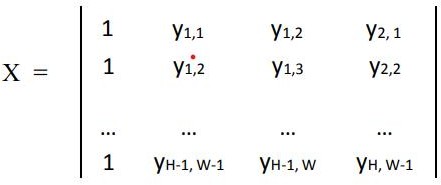
Sender trains a linear regression model over the image so that the value of the pixel can be predicted using neighbouring pixels, and this model can be called a Full image regression model. To predict the pixel, say yi, j value, model picks the three most neighbouring pixels one from top yi-1, j, one from left yi, j-1 and one from top left yi-1, j-1 and yi, j can be calculated as:

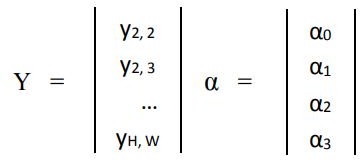


Where α0, α1, α2, and α3 are the training parameters for this regression model and these parameters can be obtained as:



Where X represents the matrix of input pixels, Y represents the vector of output labels, and α represents the vector of training parameters, as shown below:





This model does not include predicting the pixels present in the first row and the first column of the image as they do not have the neighbours specified above and therefore, these pixels are not considered for the data hiding. Naturally, an image is not regular entirely, there are many regions in the image where a drastic change in the pixel values can be seen, or the neighbouring pixels have quite different values, and these regions are named as complex regions. To improve this full image linear regression model’s predictability, the sender selects pixels only from regular regions, since pixels in these regions are easy to predict. The pixels from the complex regions have quite different values and only contribute to error. To filter out the pixels from complex regions, the model calculates the regularity value Ri, j of a pixel yi, j as follows:

Ri, j = |yi, j - yi-1, j-1| + |yi, j - yi-1, j| + |yi, j - yi, j-1|

Only those pixels are included in training data for which regularity value is less than a threshold R\_th (Ri, j ≤ R\_th). Through this process, all complex pixels can be removed, and this full image regression model can be trained over the regular regions, which will provide high accuracy. Along with the full image regression model, the sender trains a new Neighbour specific regression model. This new regression model is trained for every prediction over 8 neighbouring data points enclosed in 4\*4 block

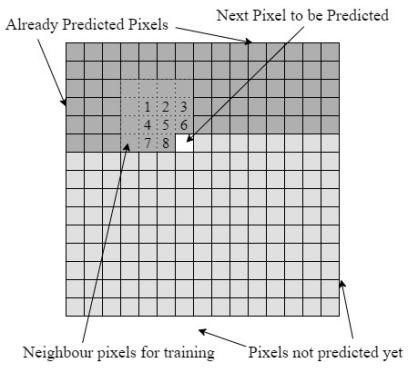


Fig: Process of prediction with neighbour specific model

Predicting the pixel value by using the neighbour specific regression model using surrounding pixels as its training data. The predicted value with this method can be called as Pf2. After getting two predicted values Pf1 and Pf2 for the pixel, the proposed scheme obtains final tuned prediction Pf from the full image as well as neighbour specific model using β (Beta) as shown in the equation:

Pf = β\*Pf1 + (1-β)\*Pf2

Here β is a hyperparameter such that 0 ≤ β ≤ 1. This parameter defines the proportions utilized of the predictions of the two different regression models in order to get optimized results as the final prediction of our combined model as:

1. If β = 1, then the prediction is only based on the full image regression model. No proportion of the prediction of the neighbour specific model will be present.
2. If β = 0, then the prediction is only based on a neighbour specific regression model.
3. If 0 < β < 1, proportions of both the predictions are present. The proposed scheme uses an optimal value of β as well as regularity threshold (R\_th) which is obtained using Particle Swarm Optimization (PSO) technique [9] such that the number of inaccurate predictions is as small as possible. Since evidently, the neighbour specific model does not include the predictions of first

three rows and first three columns, the predictions of the pixels present in these rows and columns is taken from the full image regression model.

### 2. SOURCE CODE

clc; % clear the command window clear all;

close all;% clear the workspace

% close all

disp (' ');

disp (' \*\*\*\*\* WELCOME TO IMAGE HIDER \*\*\*\*\*');

disp (' '); disp (' \*\*\*\*\*\*\* Enter ur choice \*\*\*\*\*\*\*\*\*\*');

task =input('\*\*\*\*---Encode :- 1 \n\*\*\*\*---Decode :- 2\n Enter your task:');

% select task

if isempty (task) task= 1;

end

if task == 1

% reads two image files [filename1,pathname]=uigetfile('\*.jpg','select cover the image'); x=imread(num2str(filename1)) [filename1,pathname]=uigetfile('\*.jpg','select hiding the image'); y=imread(num2str(filename1))

%checkcompatibility

|  |  |  |  |
| --- | --- | --- | --- |
| sx | = | size | (x); |
| sy | = | size | (y); |

x= imresize (x, [2\*sy(1),2\*sy(2)]);

% Applying shifting

x1 = bitand (x,uint8(252)); y1 = bitshift (y,-4);

y1\_= bitand (y1,12); y1\_= bitshift (y1\_,-2); y1 = bitand (y1,3);

y\_lsb1 = bitshift (bitand(y,12),-2); y\_lsb2 = bitand (y,3);

z= x1;

for j=1:sy(2) for i=1:sy(1) for k=1:3

z (i ,j ,k) = bitor (x1(i,j,k), y1\_(i,j,k));

z (i+sy(1) ,j+sy(2),k) = bitor (x1(i+sy(1) ,j+sy(2),k), y1(i,j,k));

z (i+sy(1) ,j,k) = bitor (x1(i+sy(1),j ,k), y\_lsb1(i,j,k));

z (i,j+sy(2) ,k) = bitor (x1(i,j+sy(2) ,k), y\_lsb2(i,j,k)); end

end end z=z

% the first image figure(1)

image (x);

xlabel(' Ist Image ');

% IInd image figure(2); image (y);

xlabel(' IInd Image ');

% encoded image figure(3);

image (z);

xlabel(' Encoded Image ');

% saving image file

sav= input('Do you want to save the file y/n [y] ','s'); if isempty(sav)

sav='y'; end

if sav == 'y'

name= input('Enter a name for the encoded image: ','s'); if isempty (sav)

name= 'encoded\_temp'; end

name=[name,'.bmp']; % concatination imwrite (z, name,'bmp');

end else

% Decoding encoded image clear;

[filename1,pathname]= uigetfile('\*.bmp','select cover the image'); z=imread (num2str(filename1));

sy = size(z)/2; % take the size of input file

% shifting

xo= bitand (z,uint8(252));

xo= imresize (xo,[sy(1),sy(2)]) % reduce the resolution to half so

%that it becoms the original image's resolution for j=1:sy(2) % y variation

for i=1:sy(1) % x variation for k=1:3

zout1(i,j,k) = bitshift (bitand(z(i,j,k),uint8(3)),2);

zout2(i,j,k) = bitand (z(i+sy(1),j+sy(2),k), uint8(3));

zout3(i,j,k) = bitshift (bitand(z(i+sy(1),j,k),uint8(3)),2);

zout4(i,j,k) = bitand (z(i,j+sy(2),k),uint8(3)); end

end end

zout = bitshift ((zout1+zout2),4)+zout3+zout4; yo = zout;

% display Ist & IInd image from encoded image figure(4);

image(xo);

xlabel ('Ist Decoded Image '); figure (5);

image (yo);

xlabel ('IInd Decoded Image');

% saving file

sav= input ('Do you want to save the file y/n [y] ','s');

if isempty ( sav)

sav='y'; end

if sav == 'y'

name1 = input('Enter a name for the first image: ','s'); name2 = input('Enter a name for the second image: ','s');

if isempty(name1) name1 = 'Ist\_temp'; end

if isempty(name2) name2 = 'IInd\_temp'; end

name1 = [name1, '.bmp'] name2 = [name2, '.bmp'] imwrite(xo,name1, 'bmp') imwrite(yo,name2, 'bmp')

end end