MAJOR PROJECT REPORT

ON

"Detection of Geometric Transformations in Copy-Move Forgery of Digital Images"

Submitted in partial fulfillment of the requirement for the award of the Degree of

BACHELOR OF TECHNOLOGY

Submitted to

Jawaharlal Nehru Technological University Hyderabad, Hyderabad

Submitted

By

S.ARCHANA **156B1A0460**

Under the Esteemed Guidance of

Mr.T.VENKATA RAMANA

M.Tech

Asst.Prof.

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

KAKATIYA INSTITUTE OF TECHNOLOGY AND SCIENCE FOR WOMEN

MANIKBHANDAR, NIZAMABAD, 503003

Approved by AICTE and Affiliated to JNTUH.

(2015-2019)

KAKATIYA INSTITUTE OF TECHNOLOGY AND SCIENCE FOR WOMEN
Approved by AICTE and affiliated to JNTUH-Hyderabad.
Manikbhandar, Nizamabad-503003.Ph:08462-281077

Date:

CERTIFICATE

This is to certify that the mini project entitled "Detection of Geometric Transformations in Copy-Move Forgery of Digital Images" is a record of bonafied work carried out by S.ARCHANA (156B1A0460) student of B.Tech, under my supervision and guidance in partial fulfillment for the award of Bachelor of Technology in Electronics and Communication Engineering during the academic year 2018-2019.

PROJECT GUIDE

Mr.T.VENKATA RAMANA
M.Tech
Asst.prof.

HEAD OF THE DEPARTMENT

Dr.M.MAHIPAL
M.Tech,Ph.D
Dept. of ECE

PRINCIPAL

Dr.S.SELVA KUMAR RAJA
M.E,Ph.D

EXAMINERS:

1.

2.

KAKATIYA INSTITUTE OF TECHNOLOGY AND SCIENCE FOR WOMEN Approved by AICTE and affiliated to JNTUH-Hyderabad.
Manikbhandar, Nizamabad-503003.Ph:08462-281077

Date:

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to our guide **T. VENKATRAMANA, M.Tech** of Electronics and Communication Engineering, KITSW who extended his unconditional support and his valuable time with his patience and valuable suggestions during the project. I Also extend my special thanks for the encouragement and constant supervision throughout the project.

I would like to give special vote of thanks to **HOD, Dr.M. MAHIPAL,M.tech.,Ph.D.,ECE** Dept. KITSW, for his unique way of inspiring students thought clarity of thought, enthusiasm and care. His constant encouragement and assistance were very helpful and made our effort a success.

I also greatful to **Dr.S.SELVA KUMAR RAJA, principal, KITSW** for providing us with the facilities and resources required for the successful completion of our project. I even thank him for his valuable suggestions at the time of seminars which encourage me to give our best in project.

I would also like to thank our faculty and supporting staff of **Electronics and Communication Engineering Department** and all other department for their kind cooperation directly or indirectly in making the project successful one.

Finally, I want to deeply acknowledge all our friends and family members to have encouraged us during the preparation of our project.

By

S.ARCHANA (156B1A0460)

KAKATIYA INSTITUTE OF TECHNOLOGY AND SCIENCE FOR WOMEN Approved by AICTE and affiliated to JNTUH-Hyderabad.
Manikbhandar, Nizamabad-503003.Ph:08462-281077

Date:

DECLARATION

I **S.ARCHANA (156B1A0460)** hereby declare this project has been carried out entirely under the esteemed guidance of **T.VENKATRAMANA, M.tech.,** for the partial fulfillment of the award of the degree of **B.Tech** in Electronics and Communication Engineering at **Kakatiya Institute of Technology and science for Women ,**Manikbhandar, Nizamabad,Affiliated to JNTUH and further it has not been submitted to any other University or Institutions for the award of any other Degree.

By

S.ARCHANA (156B1A0460)

Date:

ABSTRACT

Digital images and videos play the most important role in digital forensics. They are the prime evidences of any crime scene. Among all classes of image forgeries region-duplication is the simplest and most widely launched technique to forge an image. Region duplication is also known as copy-move forgery.

In this type of forgery one part of the image is copied and then pasted into another part of the same image. In this paper we particularly deal with copy-move form of digital image forgery. The existing techniques for identifying copy-move forgery are based on exact block matching approach.

However, the existing techniques detect an image forgery when the copied region(s) is/are directly moved to another location of the image, without undergoing any geometric transformation.

In this project we propose a technique to detect geometrically transformed copy-moved image regions. Specifically we aim to detect rotation,rescaling, as well as a combination of both. We use Scale Invariant Feature Transform (SIFT) for our purpose.

Date:

CONTENTS

Date:

LIST OF FIGURES

Date:

LIST OF TABLES

CHAPTER 1

INTRODUCTION TO PROJECT

Now-a-days digital images are easy to manipulate and edit. This is due to cheap availability of powerful image processing tools and editing software. Image forgery is an illegal manipulation done on the digital images to conceal some meaningful information of the image.

It is difficult to perceptually distinguish the edited image from the original image. Given the legal importance of credibility of digital images, not only in the court of law but also in several other areas, such as the media and broadcast industries, such illegal image manipulation or tampering are highly undesirable. Copy-move forgery is a very primitive as well as widely carried out form of digital image forgery.

In this type of forgery a region of the image is copied and moved into another portion of the same image. The intention of the attacker behind such kind of forgery is to add or hide some information to or from the image.

To identify such forgeries several researchers have proposed efficient block matching based forgery detection techniques over the past decade. The existing techniques are efficient enough to detect multiple copy-moved image regions too.

However, the existing forgery detection techniques are successful when the copied region is directly pasted onto another location of the image. If the attacker applies any form of geometric transformation (a spatial rearrangement of pixels) on the copied region, before pasting it onto the image, then it becomes difficult to detect the duplicated region, by following conventional copy-move forgery detection techniques. This is due to the fact that most conventional copy-move forgery detection techniques are based on exact block-matching approach .

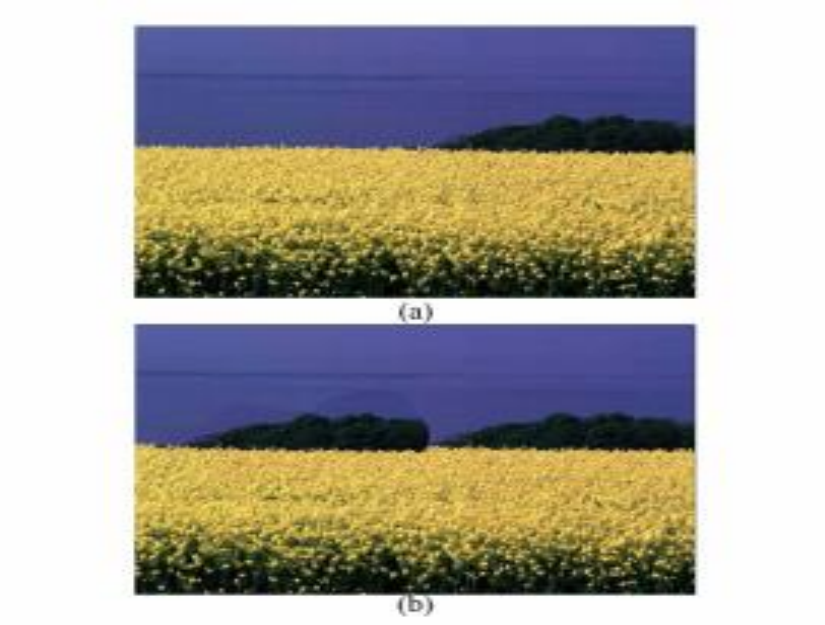


Fig. 1.1: (a) Original Image; (b) Forged Image: A portion of the foliage in the image background been duplicated.

Fig. 1.1 presents an example of copy-move forgery in digital images. The image at the top is the original image and the bottom image is the forged image. In the bottom image the background foliage is copy-pasted on to another location of the same image, to generate its forged version.

In addition, geometric transformations may also applied to the copied portion to obscure some sensitive information contained in the image, hence making detection all the more difficult. Hence we need some means to detect geometrically transformation copy-moved regions in an image. In this work we propose a method for detecting copy-move forgery with geometric transformations, specifically rotation and rescaling.

We use Scale Invariant Feature Transform (SIFT) to carry out detection of region duplication in our test images. The key- points extracted from and image using SIFT are invariant to scaling and rotation. Hence in this paper we propose a SIFT based technique to identify copy-move forgery involving the following forms of geometric transformations: rescaling, rotation as well as a combination of both. Rest of the paper is organized as follows. In next Section we have presented a survey of related work. The proposed method for identifying copy- move forgery with rescale and rotation transformations.

CHAPTER 2

INTRODUCTION TO DIGITAL IMAGE PROCESSING

2.1 IMAGE:

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

The image is 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. 2.1General 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.

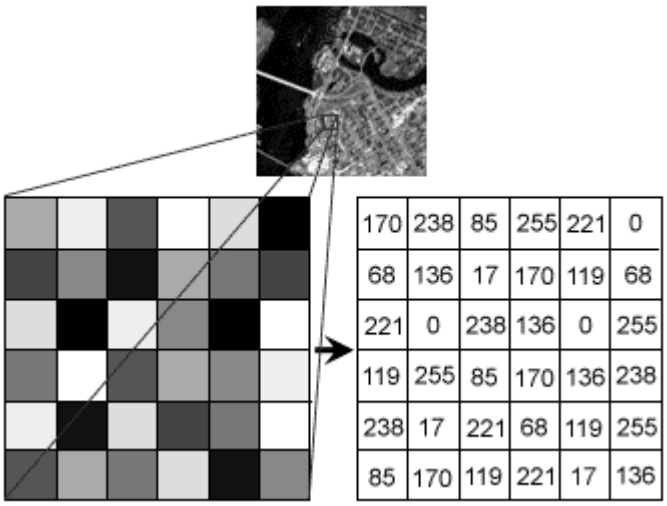


Fig. 2.2 Image pixel

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.

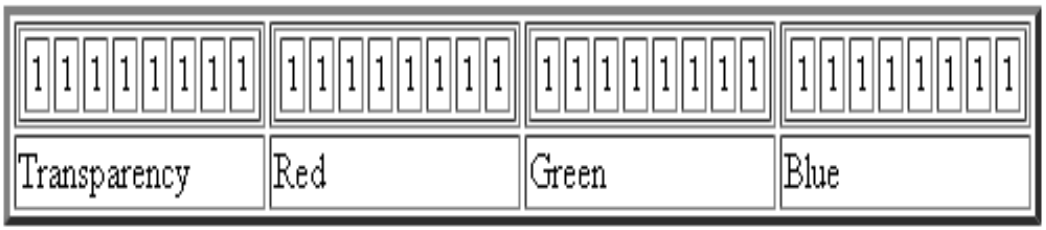


Fig. 2.3 Transparency image

2.2 IMAGE 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 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.

2.3 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.

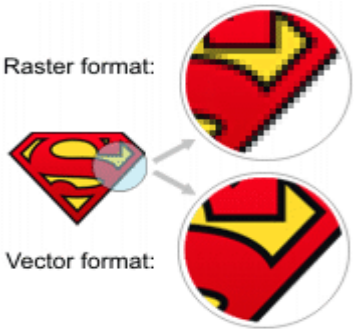


Fig. 2.4 Resolution image

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.

2.3.1 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 is recorded for individual images and include such things as camera settings, time and date, shutter speed, exposure, image size, compression, thename of thecamera, 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, gray scale, 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.

2.3.2 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.

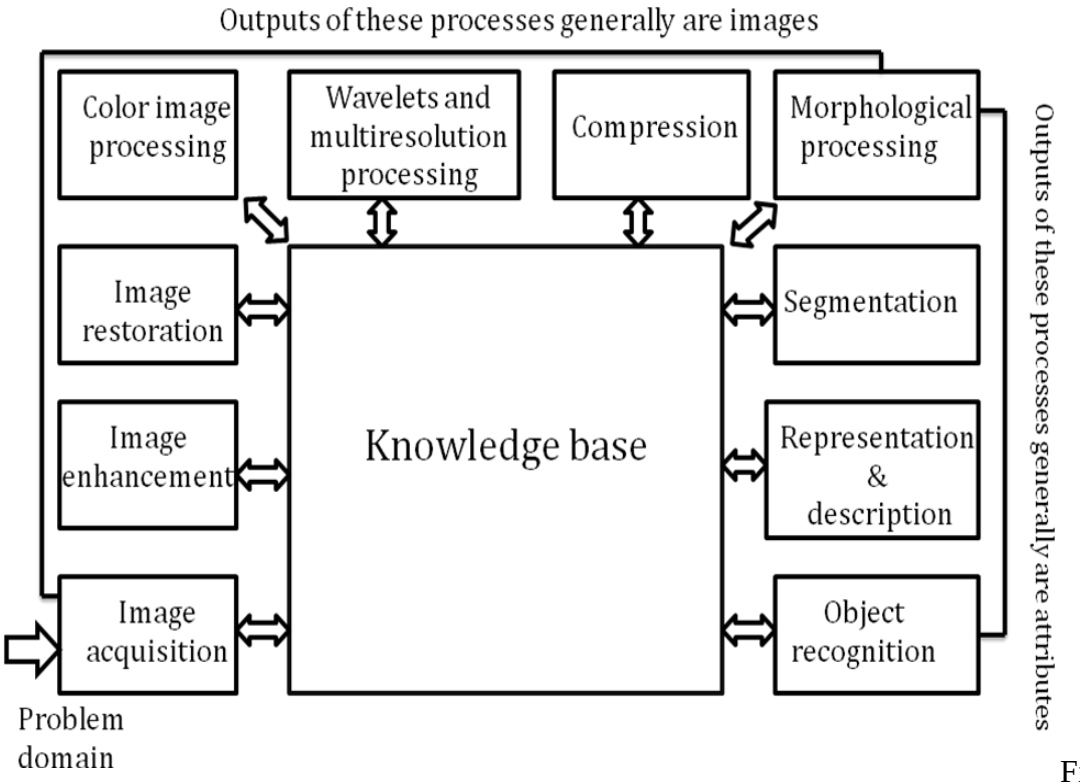
2.4 IMAGE PROCESSING:

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, miss-connect ions, misunderstandings and misinformation. It is thevast umbrella under which fall diverse aspect of optics, electronics, mathematics, photography graphics and computer technology. It is truly multidisciplinary endeavor plowed with imprecise jargon.

Several factorscombines 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.

2.5 FUNDAMENTAL STEPS IN DIGITAL IMAGE PROCESSING:



Fig

Fig. 2.5 Image fundamental

2.5.1 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 2.6 Digital camera image

The 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 2.7Digital camera cell

2.5.2 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.

Fig 2.8 Image enhancement

2.5.3 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 2.9 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, whereas removal of image blur by applying a deblurring function is considered a restoration technique.

2.5.4 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 2.10 Color & Gray scale image

2.5.5 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.

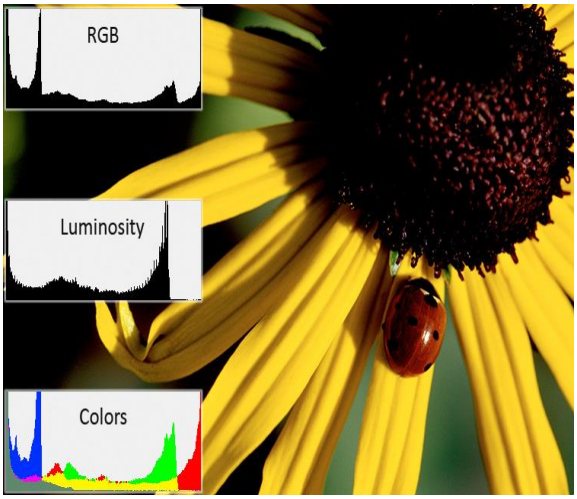


Fig 2.11 RGB histogram image

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.

2.5.6 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 is 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.

2.5.7 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 2.12 Blur to Deblur image

In binary images, the sets in question are members of the 2-D integer space Z^2, 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 Z^3. 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.

2.5.8 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 asuccessful solution of imaging problems that require objects to be identified individually.

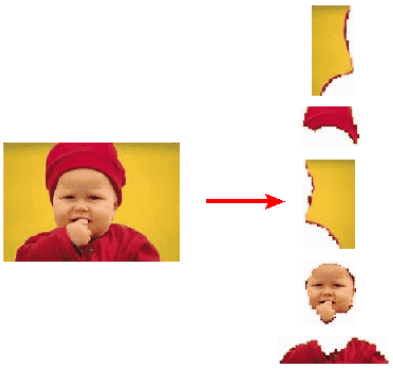


Fig 2.13Image segmentation

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.

2.5.9 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.

2.5.10 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.

2.5.11 Knowledgebase:

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 interrelated to list of all major possible defects in a materials inspection problem or an image database 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.

2.6 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.

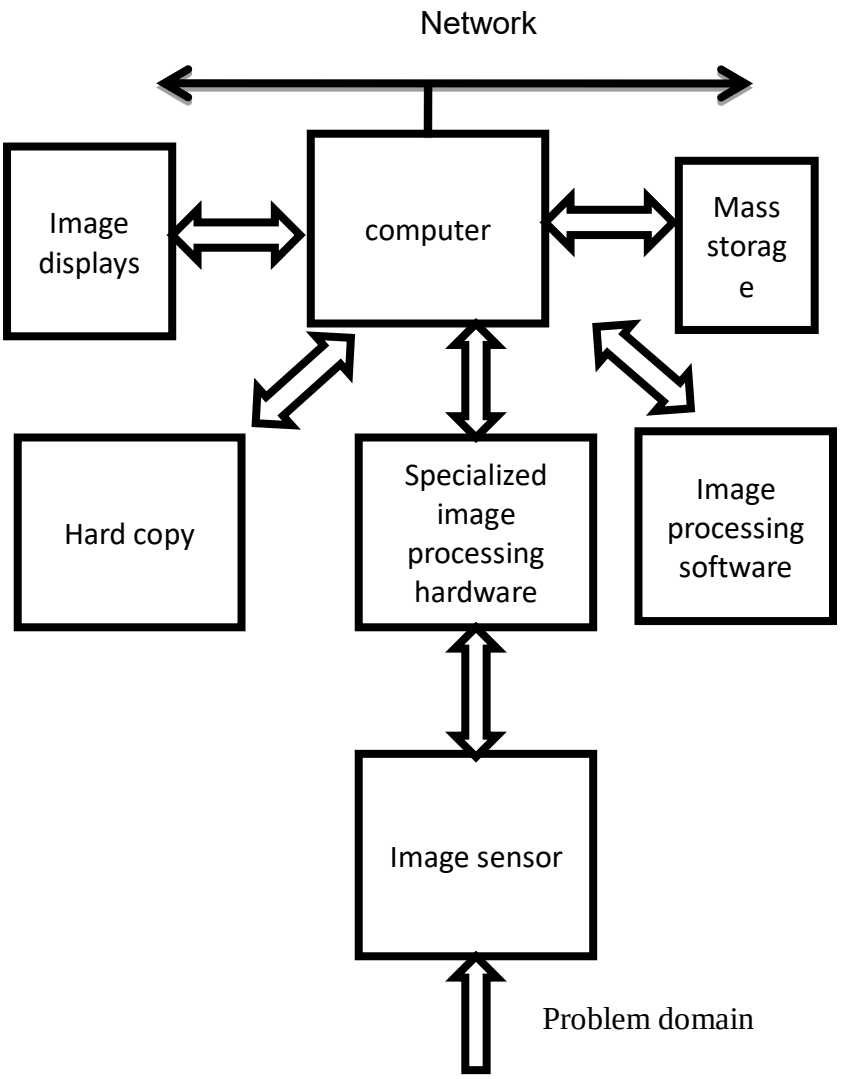


Fig 2.14 Component of image processing

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 sense, 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 for image processing applications falls into three principal categories: (1) short-term storage for use during processing, (2) online 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 gigs, or one billion, bytes), and Tbytes (meaning term, 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 ascroll(vertical shifts) and pan (horizontal shifts). Frame buffers usually are housed in the specialized image processing hardware unit shown. Online storage generally takes the form of magnetic disks or optical-media storage. The key factor characterizing online storage is frequent access to the stored data. Finally, archival storage is characterized by massive storage requirements but theinfrequent 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 their 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. The film provides the highest possible resolution, but thepaper is the obvious medium of choice for thewritten 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 a 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 theoptical fiber and other broadband technologies.

CCM:

In music

- Contemporary Christian music, a genre of popular music which is lyrically focused on matters concerned with the Christian faith
    - *CCM Magazine*, a magazine that covers Contemporary Christian music
- University of Cincinnati College-Conservatory of Music, the performing arts college of the University of Cincinnati
- Cincinnati Conservatory of Music, a conservatory formed in 1867 as part of a girls' finishing school which later became part of the University of Cincinnati College-Conservatory of Music.
- Contemporary Commercial Music
- In the context of MIDI: control change message.

In cryptography

- CCM mode, a mode of operation for cryptographic block ciphers
- Combined Cipher Machine, a common cipher machine system for securing Allied communications during World War II

In medicine

- Cerebral cavernous malformation, a vascular disorder of the central nervous system that may appear either sporadically or exhibit autosomal dominant inheritance
- Classical Chinese medicine, a medicine that developed from germ theory
- Comprehensive Care Management, a member of the Beth Abraham Family of Health Services

- *Critical Care Medicine*, a peer-reviewed medical journal in the field of critical care medicine

In politics

- Chama Cha Mapinduzi, the ruling political party of Tanzania
- Crown Council of Monaco, a seven-member administrative body which meets at least twice annually to advise the Prince of Monaco on various domestic and international affairs
- Convention on Cluster Munitions is an international treaty that prohibits the use of cluster bombs, a type of explosive weapon which scatters submunitions ("bomblets") over an area.
- Coalition for a Conservative Majority, a nonprofit, political advocacy group organized by Tom DeLay and Kenneth Blackwell

In religion

- Catholic Campus Ministry, a Catholic student organization on many college campuses
- Council of Churches of Malaysia, an ecumenical body in Malaysia comprising mainline Protestant churches and Oriental Orthodox Church
- Christ Church Manchester, otherwise known as CCM is a church in Manchester that meets in numerous locations
- Christian Compassion Ministries, a mission organization in the Philippines

In sports

- CCM (The Hockey Company), a manufacturing company of Canada
- CCM (cycle), a manufacturing company of Canada
- Central Coast Mariners FC, an Australian professional football (soccer) team based on the Central Coast of New South Wales, Australia

In technology

- Continuous Controls Monitoring describes techniques of continuously monitoring and auditing an IT system

- Continuous Current Mode, operational mode of DC-DC converters
- Cisco CallManager, a Cisco product
- Cloud Computing Manifesto
- Configuration & Change Management
- CORBA Component Model, a portion of the CORBA standard for software componentry
- A deprecated abbreviation for the cubic centimeter unit of volume measurement

In transportation

- CCM (cycle), a cycle manufacturer
- CCM Airlines, a regional airline based in Ajaccio, Corsica, France
- Clews Competition Motorcycles, a British motorcycle manufacturer based in Blackburn, England
- Cabin Crew Member, another name for flight attendant

In education

- County College of Morris, a two-year, public community college located off of Route 10 on Center Grove Road in Randolph Township, New Jersey
- City College Manchester, a Further Education college in the United Kingdom.
- City College of Manila, in the Philippines.

In military

- Center for Countermeasures, a United States military center based at White Sands Missile Range, New Mexico
- Command Chief Master Sergeant, a position in the United States Air Force

In other fields

- Corn cob mix, a kind of silage consisting of corn cobs and kernels.
- El Centro Cultural de Mexico, an alternative space in Santa Ana, Orange County, California
- Cerberus Capital Management, a large privately owned hedge fund

- Certified Consulting Meteorologist, a person designated by the American Meteorological Society to possess attributes as they pertain to the field of meteorology
- Crime Classification Manual, FBI produced text for a standardized system to investigate and classify violent crimes.
- Cervecería Cuauhtémoc Moctezuma, a major brewery in Mexico that produces brands such as Dos Equis and Tecate.

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 the global distribution 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 of 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 the ICICM feature in an in an image One of the main drawbacks of the histogram- based approaches are 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. Gray scale 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 has a particular gray scale combination. Let I am 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:

$$\mathrm{gcm}(i,j) = \Pr((\mathrm{gl}_{\mathrm{p}}, \mathrm{gl}_{N_{\mathrm{p}}}) = (i,j))$$

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 approach can effectively represent color distribution in an image. However, correlogram features do not capture intensity variation Many image databases often contain both colors 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 a color pair of the pixels p and Np are captured in the first two dimensions of the matrix and the spatial distance' 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 $\mathrm{hl} \in \{0, \ldots, \mathrm{HL} - 1\}$, then each component of CCM can be written as follows:

$$\mathrm{ccm}(i,j) = \Pr((\mathrm{hl}_{\mathrm{p}}, \mathrm{hl}_{N_{\mathrm{p}}}) = (i,j))$$

Four matrices representing neighbors at angles 0, 90, 180 and270 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.

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 VPN equals j, as a fraction of the total number of pixels in the image. Similarly, mcm(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.

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 image retrieval application from large image databases.

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.

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

$$H=\begin{cases}0 \ if \ h\in[316,20]\\ 1 \ if \ h\in[21,40]\\ 2 \ if \ h\in[41,75]\\ 3 \ if \ h\in[76,155]\\ 4 \ if \ h\in[156,190]\\ 5 \ if \ h\in[191,270]\\ 6 \ if \ h\in[271,295]\\ 7 \ if \ h\in[296,315]\end{cases} \qquad S=\begin{cases}0 \ if \ s\in[0,0.2)\\ 1 \ if \ s\in[0.2,0.7)\\ 2 \ if \ s\in[0.7,1)\end{cases} \qquad V=\begin{cases}0 \ if \ v\in[0,0.2)\\ 1 \ if \ v\in[0.2,0.7)\\ 2 \ if \ v\in[0.7,1)\end{cases}$$

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.

IMAGERETRIEVAL:

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.

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.

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:

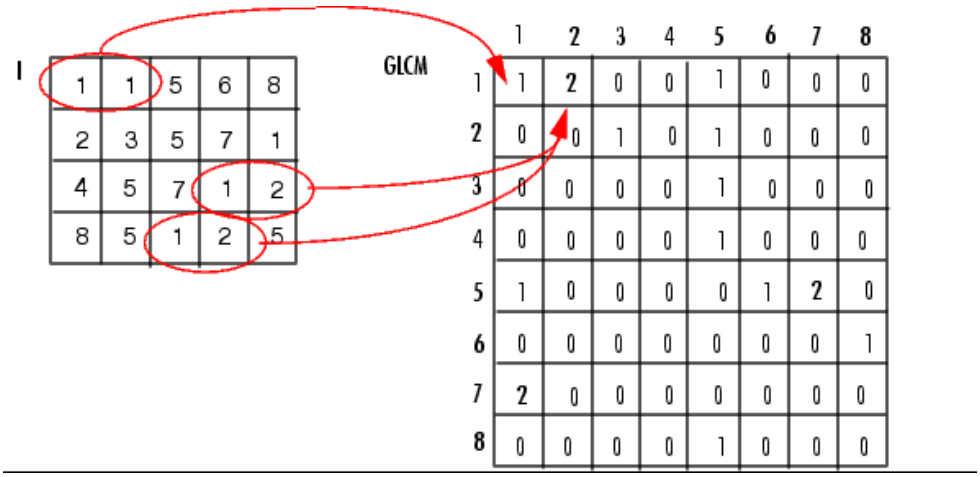


Fig2.15: Texture overview

CHAPTER 3

PROJECT DESCRIPTION

3.1 INTRODUCTION

One of the pioneer works in the direction of copy-move forgery detection was done by Fridrichet.a1. In this paper the authors have discussed different block-matching based techniques to identify copy-move forgery, such as exhaustive search, auto-correlation based block matching, exact block matching and robust block matching. In our work, we have also followed the exact block matching based approach.

The main idea of exact block matching is to divide the whole image into non-overlapping blocks and perform matching between them to locate the copied-moved blocks. In Cao et. a1. have proposed a region-duplication detection scheme which utilizes Discrete Cosine Transform, whereas Bayram et. a1. propose the use of Fourier-Mellin Transform and Counting.

Bloom filters for identification of forged image blocks. Pan and Lyu propose using SIFT to identify geometric transformations in duplicated regions, because SIFT is not a global vector but it gives local feature property of an image. The key points extracted by SIFT are invariant to different scales.

Each key point is associated with 128- dimension feature vector, which makes the key points distinctive. In matching and pruning of the SIFT key points are carried out to estimate duplicated image regions.

In also the authors have proposed image region duplication detection based on SIFT features extraction. For block matching in the authors have used kd-tree and Best Bin First (BBF) algorithm. The detection method proposed in identifies the forged regions involving some post operations like Gaussian blurring, lossy compression, rotation, scaling etc. Hashmia et.a1 use a combined approach of Dyadic Wavelet Transform (DyWT) and SIFT, for copy-move forgery detection.

The algorithm proposed in has considerably high block matching rate and is the most robust to image preprocessing operations. However it is not appropriate for downsampling of copied blocks in images. Kang et. a1 proposed a region-duplication detection method based on Singular Value Decomposition (SVD) and passive blind detection techniques. The proposed method obtains singular value block matrix, and correlation coefficients which helps to improve matching capabilities.

As evident from the above discussion, the existing literature contains a number of schemes which are capable of detecting copy-move forgery involving various forms of processing on the copied-moved blocks. In this paper we aim to devise a technique to detect specifically rotation and rescale operations, as well as a combination of the two, on copy-moved image regions.

3.2 EXISTING SYSTEM:

BLOCK PROCESSING

- Certain image processing operations involve processing an image in sections, rather than processing the entire image at once. A sliding neighborhood operation processes an image one pixel at a time, by applying an algorithm to each pixels neighborhood. In distinct block processing, an image is divided into equally-sized blocks without overlap, and the algorithm is applied to each distinct block.

- The neighborhoods and blocks are then reassembled to form the output image. Certain image processing operations involve processing an image in sections, rather than processing the entire image at once. A sliding neighborhood operation processes an image one pixel at a time, by applying an algorithm to each pixels neighborhood. In distinct block processing, an image is divided into equally-sized blocks without overlap, and the algorithm is applied to each distinct block. The neighborhoods and blocks are then reassembled to form the output image.

- Block processing to perform edge detection on a magnetic resonance image. When working with large images, normal image processing techniques can sometimes break down.

- The images can either be too large to load into memory, or else they can be loaded into memory but then be too large to process. blockprocess then divides the input image into blocks of the specified size, processes them using the function handle one block at a time, and then assembles the results into an output image. Blockprocess returns the output to memory or to a new file on disk.

- Edge detection is frequently the first step in recovering from images. An edge in an image is a significant local change in the image intensity. It is related with the first derivative of the image intensity. The gradient is a measure of change in a function.

- By analogy significant changes in the gray values in an image. The gradient is the two dimensional equivalent of the first derivative and is defined as the vector.

3.3 PROPOSED SYSTEM

3.3.1Detection of Image Region Duplication using SIFT

In this section we propose a copy-move forgery detection technique utilizing SIFT. SIFT is local feature aspect of image, and it stands for Scale Invariant Feature Transform. The key-points extracted using SIFT are insensitive to any geometric transformations, so that it makes the matching procedure easier. We first extract the SIFT key-points of the suspect image.

Once the key-points are extracted we divide the whole image into non-overlapping inspect blocks or segments. For each inspect block we are finding nearest match of that block into the image. For all the SIFT key-points present in the inspect block, we are calculating D-distance between each key-points which are present in inspect block. For finding D-distance we use Euclidean distance algorithm.

The Euclidean distance is calculated for 128 dimensional vectors of each key-points. And then we also find nearest neighbor of each key-points. With the help of matched keypoints we can find the moved region, which is geometrically transformed by applying scaling and rotation.

The overall detection method consists of the following steps:

1) SIFT Features Extraction
2) Key-points Matching and Pruning
3) Identifying Region Transformation
4) Display the original and forged region

In the next subsections we discuss the above mentioned steps in details. B.

3.3.2 Extract SIFT Features

In the following we discuss the procedure of extraction of SIFT key-points of an image . In the subsequent subsections we present the utilization of the extracted keypoints for copy-move forgery detection, as well as for addressing the problem of geometric transformations in copy-move forgery. Figure 2, presents a flow chart representing the different steps of the proposed technique. Next, we discuss all the steps all the steps shown in figure 2.

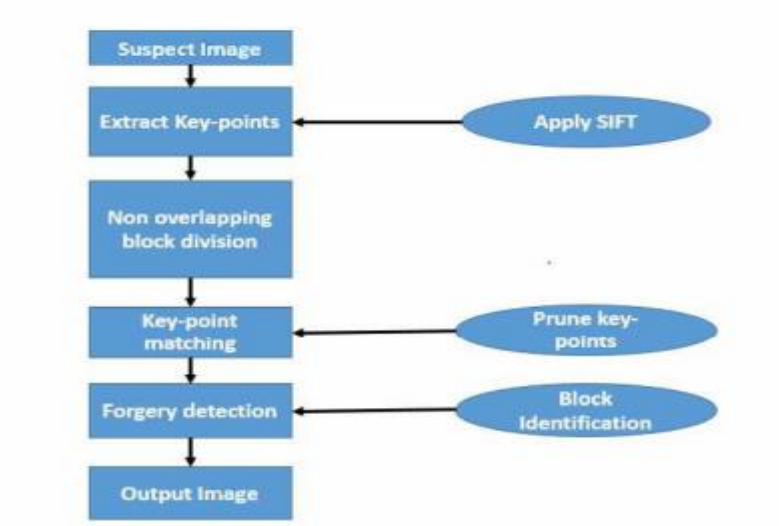


Figure 3.1: Detection Algorithm: Flow Chart

3.3.3 Scale Space Extrema Detection:

The first step identifies the location and the scale of the SIFT key-points. The points which are same for different scales or in different views of the suspected image are selected. Those points are invariant and more stable than the other key-points. The input image is convolve with the Gaussian function and gives scale space function (octaves). After that Difference of Gaussian is applied on scale spaces.

3.3.4 Detection of local extrema:

In order to find maximum and minimum points every point is a compared with its own 8 neighbors from 3x3 grid and 18 neighbors from upper and lower DOG's 3x3 grid. In this process some noisy point are also extracted as a selective point. So that elimination of noisy point and edges are required.

3.3.5 Orientation assignment:

In this step local orientation are assign to each key-point. It provides local features to key-points. The orientation assignment provides rotation invariance to SIFT. For each scale space function gradient orientation g(x,y) and magnitude m(x, y)are calculated.

3.3.6 Key-point descriptor:

The purpose behind key-point descriptor is to add more local information to the key-points. It adds 128 dimensional vector to each key-points. This 128 dimensional vector is created from all the orientation histogram and magnitude of the keypoints.

3.3.7 Key-points Matching and Pruning:

We are having two images original image and forged image. Let us take Kl and K2 such that:

- Kl = the key-points present in the inspect block in the original region and
- K2 = the key-points present in the moved block.

In this step for each SIFT feature points present in the original region, we are calculating the D-distance between the rest of the 128 dimensional vectors of the key-points presents in the inspect block. When we reach to the forged region by sliding the inspect block in non-overlapping manner, the same D-distance is calculated for that region.

The matching concept is that the key-points extracted in the original image must be extracted into the forged image. Only the duplicated region gives the difference in number of key-point which are extracted. The key-points of copied region and the moved region have some correlation, which helps to identify the forged portion. The key-points comes with SIFT are noisy so we have to prune them. The pruning removes all the false points so that false matches are removed and only correct matched will remains.

3.3.8 Identifying Region Transforms

Now we are finding the potential transformation between the original and forged blocks. In this work we are considering only copy-move forgery, re-scaling, rotation and both rotation and scaling forgery.

3.3.9 Copy-move forgery:

If the copied region is directly moved to any other location within the same image then key-points extracted in the original region and the key-points extracted in the moved region are must be same. The SIFT 128 feature vectors of each key-points are also same in both of the regions. And the distance D which we were calculated are also same for both original region as well as moved region. So that the forged region is easily be detected if there is no distortion has done.

3.3.10 Scaling:

If attacker re-scaled the duplicated region then D-distance plays most important role to identify the forgery. The number of key-points available on original portion are differ from number of the key-points available on the forged region. All the SIFT key-points which are present at original region are not extracted into the forged portion due

to scaling distortion. All the key points are not available so that the D-distance between pair of few key-points in the forged region must be multiple of their replicate presents in the original region.

Let us assume that we have two key-point pair present in both copied and moved part. These pairs are (a,b) E K1 and (a',b') E K2 and d1, dl' is D-distance between above pairs.

D(a,b) = d1, D(a',b') = d1'

Such that: (a,b) E K1 and (a',b') E K2

where D is the Euclidean distance between 128 feature vectors of the key-points. And

$$\frac{\|d1\|}{\|d1'\|} = constant$$

We plot histogram of the ratios of D-distances comes from pair of SIFT key-points in K1 and K2. The maximum frequency of the histogram gives the scaling factor or the forged region.

3.3.11 Rotation:

If the attacker rotates the copied portion and then pasted it into somewhere else in the image, then it is quite difficult to detect the forgery. The number of key-points detected in the inspect block of the original region are almost same with the number of key-points detected into the inspect block of forged region. The only difference between both regions is the key points occurs in rotated position. The D-distance and relative position of the SIFT key-points with respect to each other are constant. So we are taking D-distance between each key points presents in the inspect block. Let us take one key-point a in the original region, and suppose we have three other key-points a, b and c so the D-distance D(a,b), D(a,c), D(a,d) and the relative position of a with respect to the other key points is not going to change in the forged region.

D(a,b) = dl

D(a' ,b') = dl'

dl = dl'

3.3.12 Rotation and rescaling:

Till now identification of the forged block is not so difficult because we are having only one type of forgery applied on the copied region. But in this case both rotation and scaling are performed in the forged blocks. The order of the forgery does not make any difference.

The number of SIFT key-points extracted in the forged region only varies according to the scaling factor, rotation does not change the number of SIFT key-points extracted.

Now in both type of geometric distortion the scaling is the dominating one. Rotation just give angle to the region. So the SIFT key-points of the forged region are having a constant ratio of D-distances same as scaling forgery

$$\frac{\|d1\|}{\|d1'\|} = constant$$

Due to rotation the relative distances of the key-points and the relative position are constant.

D(a,b) = dl

D(a' ,b') = dl'

dl = dl'

Again the maximum frequency of the histogram gives the scaling factor which is drawn from the D-distance ratios.

3.3.13 Displaying the Original and Forged Regions:

In this step the matching results are displayed. All the SIFT key-points present in the original image are also present in the forged image except for the moved block, so that we are getting lots of matched SIFT key-points. But to display only the copied-moved block we have to remove some inliers for that purpose we are using RANSAC method. The RANSAC removes the inliers of the image.

CHAPTER 4

SCALE INVARIANT FEATURE TRANSFORM TECHNIQUE

4.1Introduction

Matching features across different images in a common problem in computer vision. When all images are similar in nature (same scale, orientation, etc) simple corner detectors can work. But when you have images of different scales and rotations, you need to use the Scale Invariant Feature Transform**.**

4.1.1Why care about SIFT

SIFT isn't just scale invariant. You can change the following, and still get good results:

- Scale (duh)
- Rotation
- Illumination
- Viewpoint

4.1.2The algorithm

SIFT is quite an involved algorithm. It has a lot going on and can become confusing, So I've split up the entire algorithm into multiple parts. Here's an outline of what happens in SIFT.

1. **Constructing a scale space** This is the initial preparation. You create internal representations of the original image to ensure scale invariance. This is done by generating a "scale space".
2. **LoG Approximation** The Laplacian of Gaussian is great for finding interesting points (or key points) in an image. But it's computationally expensive. So we cheat and approximate it using the representation created earlier.
3. **Finding keypoints** With the super fast approximation, we now try to find key points. These are maxima and minima in the Difference of Gaussian image we calculate in step 2

4. **Get rid of bad key points** Edges and low contrast regions are bad keypoints. Eliminating these makes the algorithm efficient and robust. A technique similar to the Harris Corner Detector is used here.
5. **Assigning an orientation to the keypoints** An orientation is calculated for each key point. Any further calculations are done relative to this orientation. This effectively cancels out the effect of orientation, making it rotation invariant.
6. **Generate SIFT features** Finally, with scale and rotation invariance in place, one more representation is generated. This helps uniquely identify features. Lets say you have 50,000 features. With this representation, you can easily identify the feature you're looking for (say, a particular eye, or a sign board). That was an overview of the entire algorithm. Over the next few days, I'll go through each step in detail. Finally, I'll show you how to **implement SIFT in OpenCV**!

CHAPTER 5

SOFTWARE INTRODUCTION

5.1 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.

5.2 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.

5.3 GRAPHICAL USER INTERFACE (GUI):

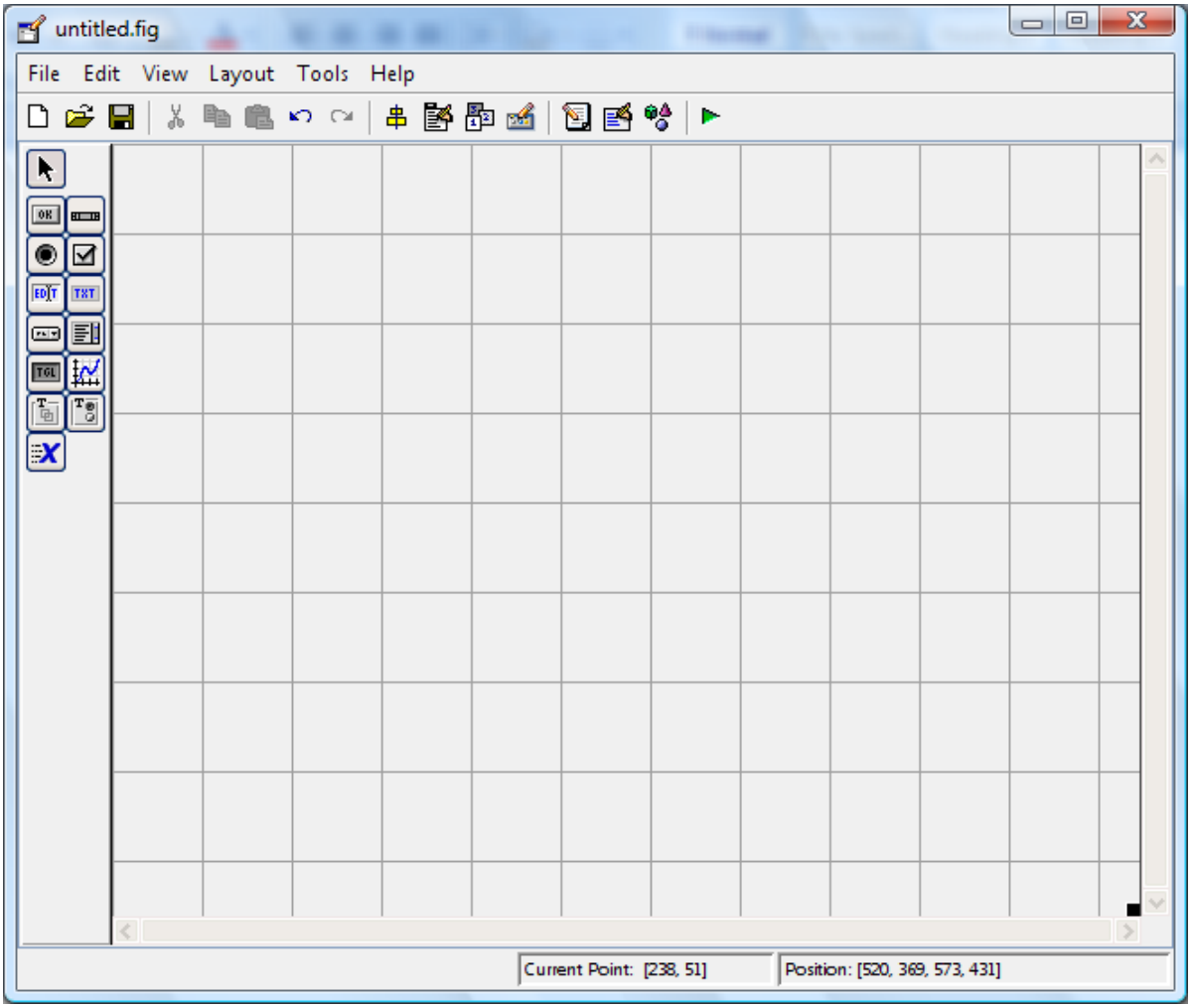
MATLAB's Graphical User Interface Development Environment (GUIDE) provides a rich set of tools for incorporating graphical user interfaces (GUIs) in M-functions. Using GUIDE, the processes of laying out a GUI (i.e., its buttons, pop-up menus, etc.)and programming the operation of the GUI are divided 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 filenameWhere 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.

5.4 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.

Introduction - describes the components of the MATLAB system.

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

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

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

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.

5.5 DEVELOPMENT ENVIRONMENT

5.5.1Introduction

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

5.5.2 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.

5.5.3 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.

5.5.4 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.

5.5.5 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 whose.

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 textediting, 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.

5.6 MANIPULATING MATRICES

5.6.1 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:

- Enter an explicit list of elements.
- Load matrices from external data files.
- Generate matrices using built-in functions.
- 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:

- Separate the elements of a row with blanks or commas.
- Use a semicolon, ; , to indicate the end of each row.

- 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.

5.6.2 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:

- Variables
- Numbers
- Operators
- 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.

5.6.3 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

Table 5.1: Operators

5.6.4 Functions

MATLAB provides a large number of standard elementary mathematical functions, including abs, sqrt, exp, and sin. Taking the square root or logarithm 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, typehelp specfunhelp 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.

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, $(2-\varepsilon)2^{1023}$
Inf	Infinity
NaN	Not-a-number

Table 5.2: Functions

5.7 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:

5.7.1 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.

5.7.2 GUI Development Environment

The process of implementing a GUI involves two basic task.

- Laying out the GUI components
- 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 sub-functions. 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.

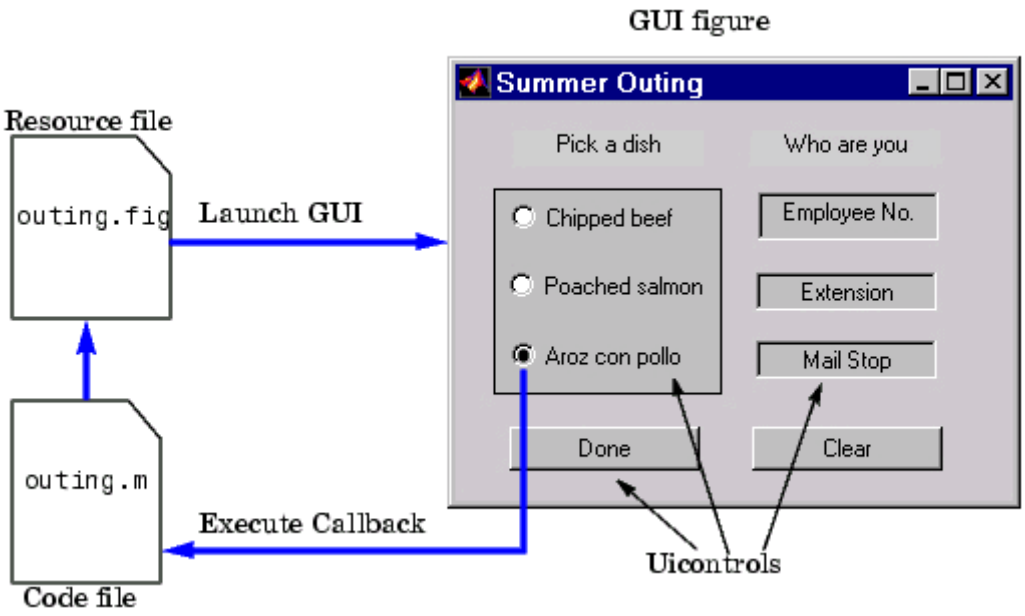


Fig 5.1 Graphical user blocks

5.7.3 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 sub-function 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.

5.7.4 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.

5.7.5 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 non-depressed 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.

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 non-depressed 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 true color image. For example, the array a defines 16-by-128 true color 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 sub-function, 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 appropriate action
else
   % checkbox is not checked-take appropriate 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 menu bar 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 menu bar 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. Listbox Top 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

List boxes differentiate between single and double clicks on an item and set the figure Selection Type 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 Selection Type 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 sub-function from the application M-file. Leave the callback sub-function 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 Window Button Down Fcn callback executes. Then the control's Button Down Fcn 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 Button Down Fcn property to define the callback.

5.7.6 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.

CHAPTER 6

RESULT

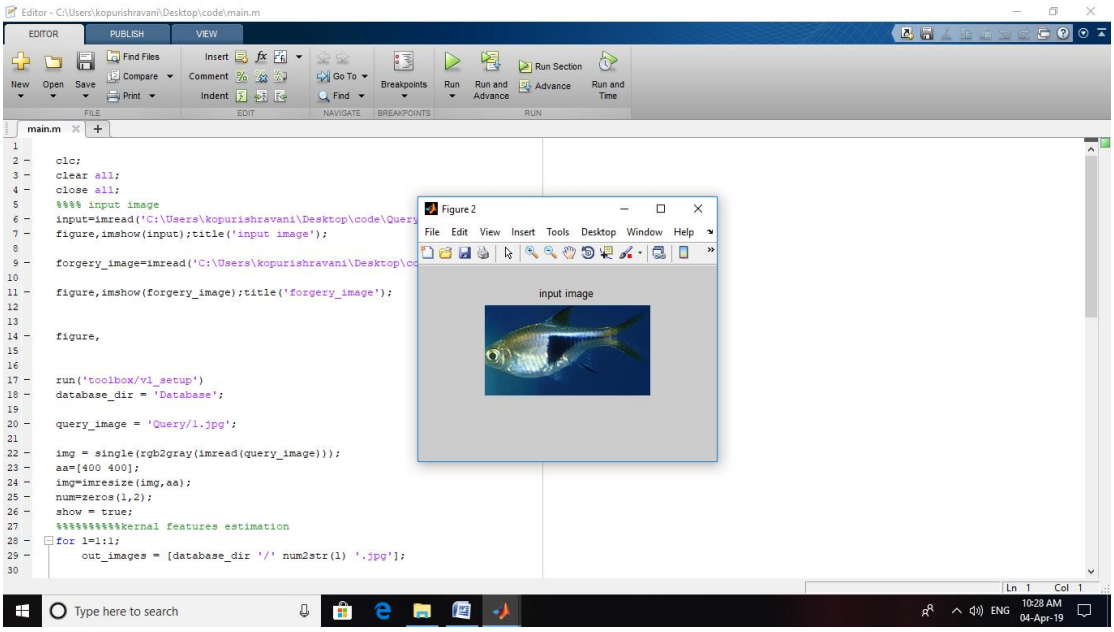


Fig 6.1Query image

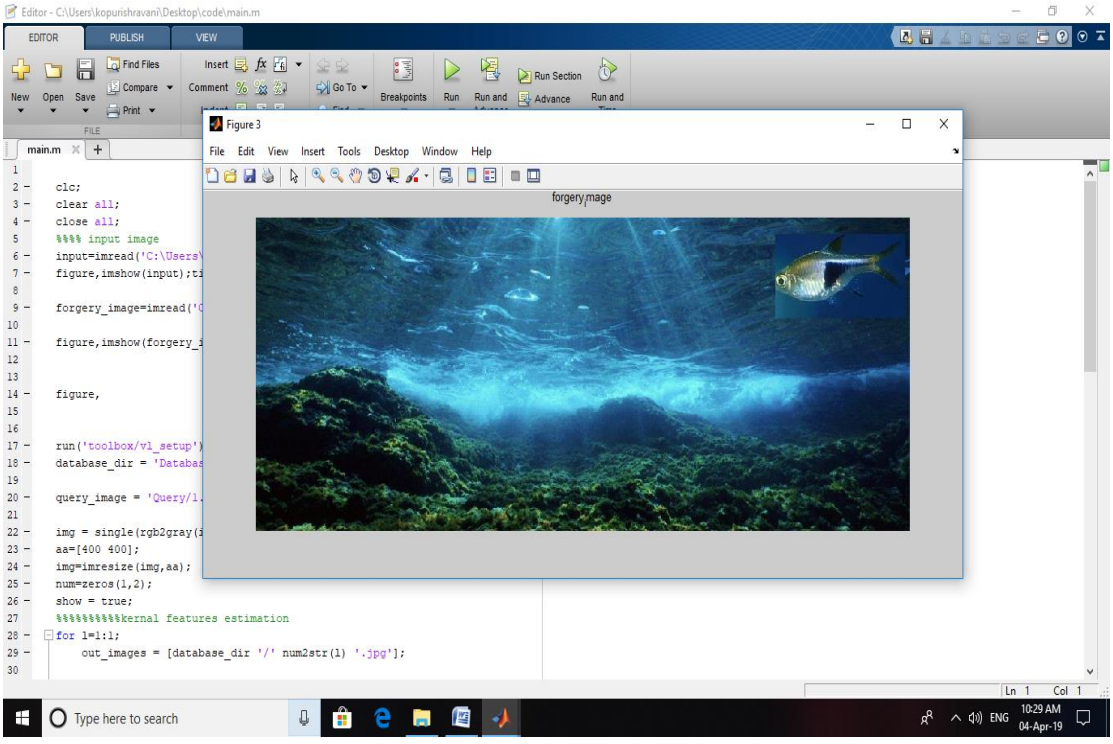


Fig 6.2 Database image

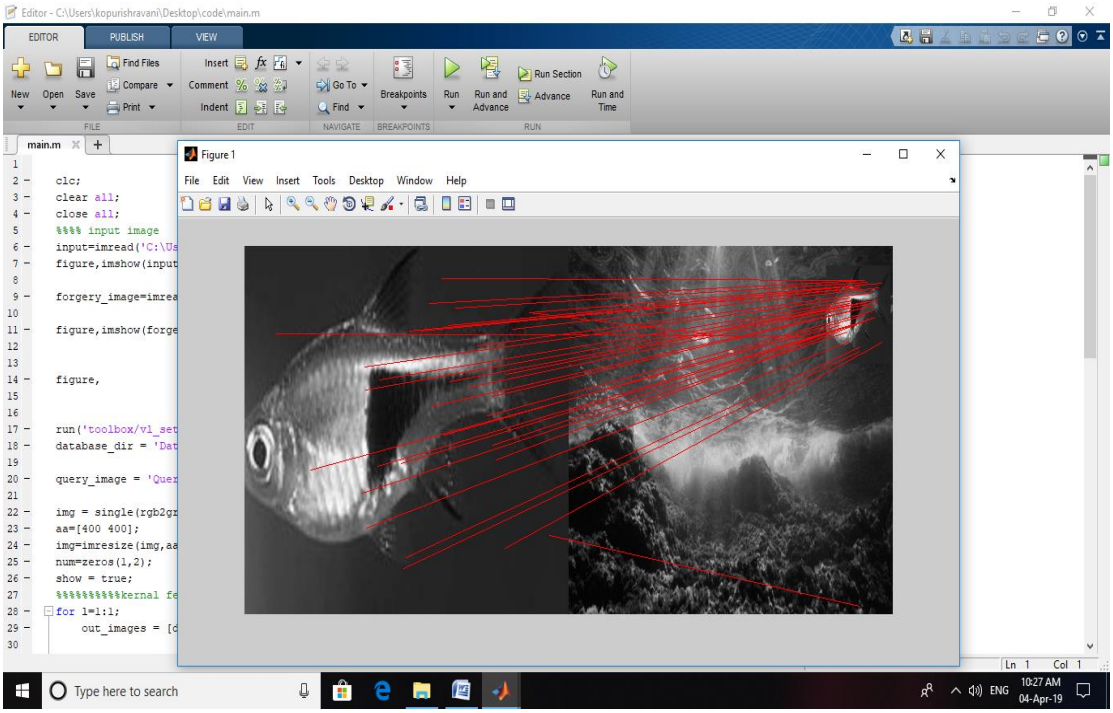


Fig 6.3 Output image

CHAPTER 7

APPLICATIONS, ADVANTAGES AND DISADVANTAGES

7.1 APPLICATIONS

- Copy- move forgery images are detected.
- Tampering images are identified.
- Used in all the forensic applications.
- Used in all the Security applications like thefting.

7.2 ADVANTAGES:

- Exact tampering and the forgery data analysis.
- Maximum features calculation.
- Accuracy and high efficieny.
- Low error rate compared to existing methods.
- Exact forgery part detection.

7.3 DISADVANTAGES:

- Maximum intensity values show SIFT features.
- Colour image and HD images are not applicable.
  Pre-processing should be done before execution if the image is inefficient

CONCLUSION

In this paper we have proposes an efficient method to detect copy-move forgery in digital images. The proposed method is also capable of detecting geometric transformations applied on the forged region, with the help of SIFT key-points. Our experimental results prove that the proposed method can detect a region duplication forgery, in which rescaling and rotation attacks are applied together or individually on the duplicated region. The proposed method is more effective and gives better performance than the exact block matching techniques, in terms of geometric attacks detection in region duplication.

FUTURE SCOPE

The future research direction includes identification of other forms of geometric attacks, relevant to copy-move forgery, such as reflection, as well as other image region transforms, such as grey level interpolation.

In future we are applying to colour images or real-time images to find exact forgery or tampering region.

BIBLIOGRAPHY:

[I] J. Fridrich, D. Soukal, and J. Lukas, "Detection of copy-move forgery in digital images," Proc. Digital Forensic Research Workshop, Cleveland, OH, Aug 2003.

[2] Y. Cao, T. Gao, L. Fan, and Q. Yang, "A robust detection algorithm for copy-move forgery in digital images", Forensic science international, pp.33-43, Jan 2012.

[3] S. Bayram,T. H.Sencar and N. Memon, "An efficient and robust method for detecting copy-move forgery", Acoustics, Speech and Signal Processing, ICASSP 2009. IEEE International Conference on, 2009.

[4] X. Pan and S. Lyu, "Detecting image region duplication using SIFT features", Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on. 2010.

[5] L. Jing, and C. Shao, "Image copy-move forgery detecting based on local invariant feature", Journal of Multimedia vol. 7, no. l,pp. 90-97, 2012.

[6] Md. F. Hashmi, V. Anand and A. G. Keskar, "Copy-move Image Forgery Detection Using an Efficient and Robust Method Combining Un-decimated Wavelet Transform and Scale Invariant Feature Transform", AASRI Procedia vol. 9, pp. 84-91, 2014.

[7] L. Kang and X. Cheng, "Copy-move forgery detection in digital image", Proc of the 3rd International Congress on Image and Signal Processing.[S. I.]: IEEE Computer Society, 2010.

[8] D. Lowe, "Distinctive image features from scale-invariant keypoints", International journal of computer vision vol. 60, no. 2 pp. 91-1 10, 2004.