# CHAPTER-1

# INTRODUCTION

IT IS well-known that the accuracy of face recognition systems deteriorates quite rapidly in unconstrained settings [1]. This can be attributed to degradations arising from blur, changes in illumination, pose, and expression, partial occlusions etc. Motion blur, in particular, deserves special attention owing to the ubiquity of mobile phones and hand-held imaging devices. Dealing with camera shake is a very relevant problem because, while tripods hinder mobility, reducing the exposure time affects image quality. Moreover, in-built sensors such as gyros and accelerometers have their own limitations in sensing the camera motion. In an uncontrolled environment, illumination and pose could also vary, further compounding the problem. The focus of this paper is on developing a system that can recognize faces across non-uniform (i.e., space-variant) blur, and varying illumination and pose.

Traditionally, blurring due to camera shake has been modeled as a convolution with a single blur kernel, and the blur is assumed to be uniform across the image [2], [3]. However, it is space-variant blur that is encountered frequently in hand-held cameras [4]. While techniques have been proposed that address the restoration of non-uniform blur by local space-invariance approximation [5]–[7], recent methods for image restoration have modeled the motion-blurred image as an average of projectively transformed images [8]–[12].

Face recognition systems that work with focused images have difficulty when presented with blurred data. Approaches to face recognition from blurred images can be broadly classified into four categories. (i) Deblurring-based [13], [14] in which the probe image is first deblurred and then used for recognition. However, deblurring artifacts are a major source of error especially for moderate to heavy blurs. (ii) Joint deblurring and recognition [15], the flip-side of which is computational complexity. (iii) Deriving blur-invariant features for recognition [16], [17]. But these are effective only for mild blurs. (iv) The direct recognition approach of [18] and [19] in which reblurred versions from the gallery are compared with the blurred probe image.

It is important to note that all of the above approaches assume a simplistic space-invariant blur model.

For handling illumination, there have mainly been two directions of pursuit based on (i) the 9D subspace model for face .In this paper, we propose a face recognition algorithm that is robust to non-uniform (i.e., space-varying) motion blur arising from relative motion between the camera and the subject. Following [19], we assume that only a single gallery image is available. The camera transformations can range from in-plane translations and rotations to out-of-plane translations, out-of plane rotations, and even general 6D motion.

Subsequently, we also show how the proposed method can be elegantly modified to account for variations in illumination and pose. We assume a planar structure for the face [14], [17], [19] and use the geometric framework proposed in [8]–[10]. The warped instances can be viewed as the intermediate images observed during the exposure time. Each warp is assigned a weight that denotes the fraction of the exposure duration for that transformation. The weights corresponding to the warps are referred to as the transformation spread function (TSF) in the literature. We develop our basic non-uniform motion blur (NU-MOB)-robust face recognition algorithm based on the TSF model. On each focused gallery image, we apply all the possible transformations that exist in the 6D space (3 dimensions for translations and 3 for rotations) and stack the resulting transformed images as columns of a matrix. We extend the convexity result proved for the simple convolution model in [19] to the TSF model and show that the set of all images obtained by blurring a particular gallery image is a convex set given by the convex hull of the columns of the corresponding matrix.

# CHAPTER - 2

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

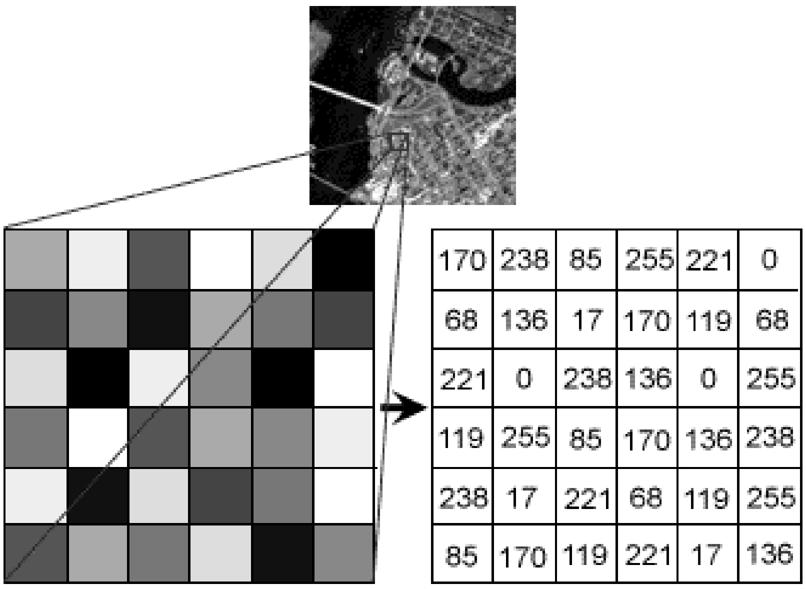
The word image is also used in the broader sense of any two-dimensional figure such as a map, a graph, a pie chart, or an abstract painting. In this wider sense, images can also be rendered manually, such as by drawing, painting, carving, rendered automatically by printing or computer graphics technology,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

.

Fig. 2.1. General image

An image is a rectangular grid of pixels. It has a definite height and a definite width counted in pixels. Each pixel is square and has a fixed size on a given display. However different computer monitors may use different sized pixels.

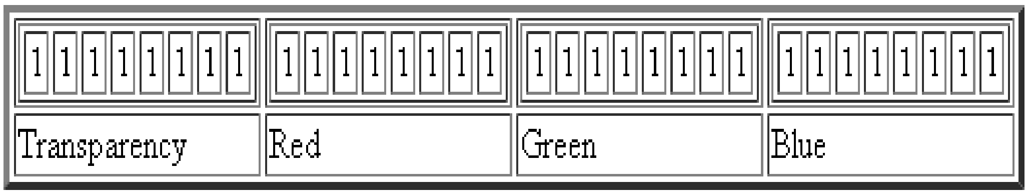
The pixels that constitute an image are ordered as a grid (columns and rows); each pixel consists of numbers representing magnitudes of brightness and color.



| | | | | | |
|---|---|---|---|---|---|
| 170 | 238 | 85 | 255 | 221 | 0 |
| 68 | 136 | 17 | 170 | 119 | 68 |
| 221 | 0 | 238 | 136 | 0 | 255 |
| 119 | 255 | 85 | 170 | 136 | 238 |
| 238 | 17 | 221 | 68 | 119 | 255 |
| 85 | 170 | 119 | 221 | 17 | 136 |

Fig. 2.2.Image pixel

Each pixel has a colour. The colour 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.



| | | | |
|---|---|---|---|
| 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 |
| Transparency | Red | Green | Blue |

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 colour 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 colour depth increases, an 8-bit pixel (1 byte) stores 256 colours, a 24-bit pixel (3 bytes) stores 16 million colours, the latter known as true colour.

Image compression uses algorithms to decrease the size of a file. High resolution cameras produce large image files, ranging from hundreds of kilobytes to megabytes, per the camera's resolution and the image-storage format capacity. High resolution digital cameras record 12 megapixel (1MP = 1,000,000 pixels / 1 million) images, or more, in true colour. For example, an image recorded by a 12 MP camera; since each pixel uses 3 bytes to record true colour, 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 data that are rasterized to pixels when displayed 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.

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.

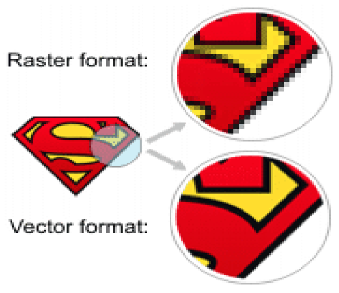


Fig. 2.4. Resolution image

### *2.3.1 RASTER FORMATS:*

These formats store images as bitmaps (also known as pix maps)

***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 colour (red, green, blue) for a 24-bit total, producing relatively small files. Photographic images may be better stored in a lossless non-JPEG format if they will be re-edited, or if small "artifacts" are unacceptable. The JPEG/JFIF format also is used as the image compression algorithm in many Adobe PDF files.

***EXIF:***

The EXIF (Exchangeable image file format) format is a file standard similar to the JFIF format with TIFF extensions. It is incorporated in the JPEG writing software used in most cameras. Its purpose is to record and to standardize the exchange of images with image metadata between digital cameras and editing and viewing software.

The metadata are recorded for individual images and include such things as camera settings, time and date, shutter speed, exposure, image size, compression, name of camera, colour 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 colour (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. Good lossless compression used 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 colour spaces, such as the CMYK defined by a particular set of printing press inks.

***GIF:***

GIF (Graphics Interchange Format) is limited to an 8-bit palette, or 256 colours. This makes the GIF format suitable for storing graphics with relatively few colours 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 colour, and ineffective for detailed images or dithered images. 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 colour spaces, such as the CMYK defined by a particular set of printing press inks. Till widely used to provide image animation effects. It also uses a lossless compression that is more effective when large areas have a single colour, and ineffective for detailed images or dithered images.

### *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 grip.

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

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

## 2.5 FUNDAMENTAL STEPS IN DIGITAL IMAGE PROCESSING:

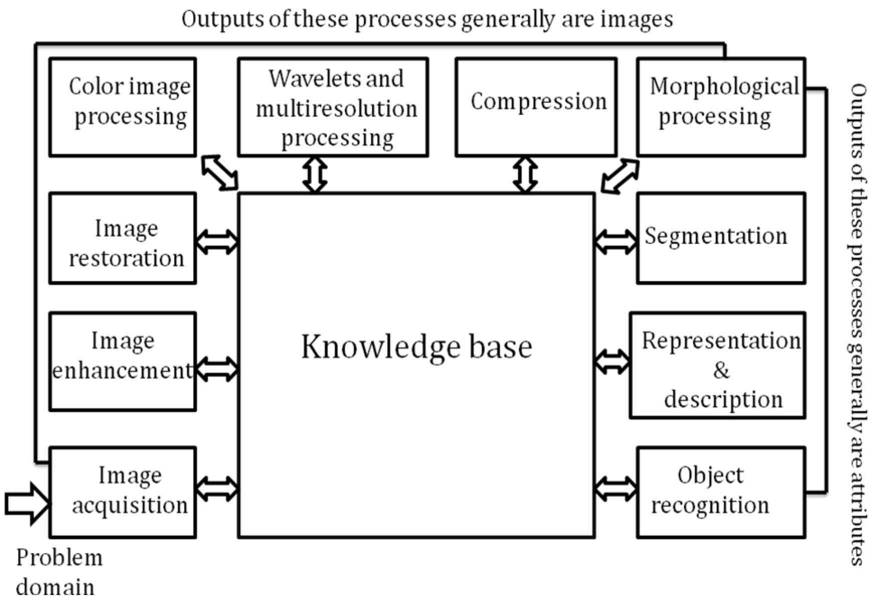


Fig.2.5. Image fundamental

### *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 colour 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

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 .2.7. digital camera cell

### *2.5.2 IMAGE ENHANCEMENT:*

Image enhancement 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.

Fig .2.8. Image enhancement

### *2.5.3 IMAGE RESTORATION:*

Image restoration 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, where as removal of image blur by applying a deblurring function is considered a restoration technique.

### *2.5.4 COLOUR IMAGE PROCESSING :*

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

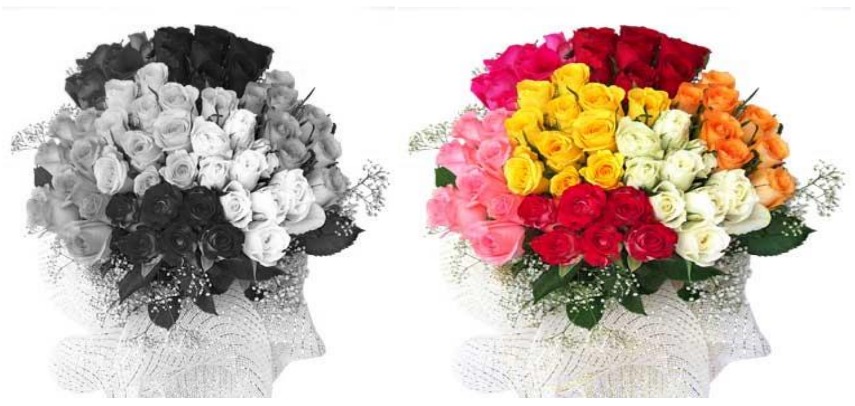


Fig. 2.10. Colour & Gray scale image

### *2.5.5 WAVELETS AND MULTI RESOLUTION 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.

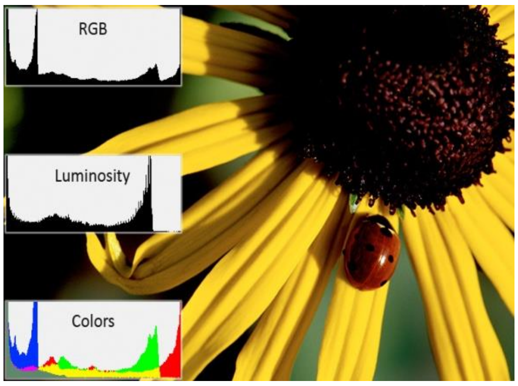


Fig.2.11. RGB histogram image

Wavelets were first shown to be the foundation of a powerful new approach to signal processing and analysis called Multi resolution theory. Multi resolution 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, 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. Blurs 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 successful solution of imaging problems that require objects to be identified individually. 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.

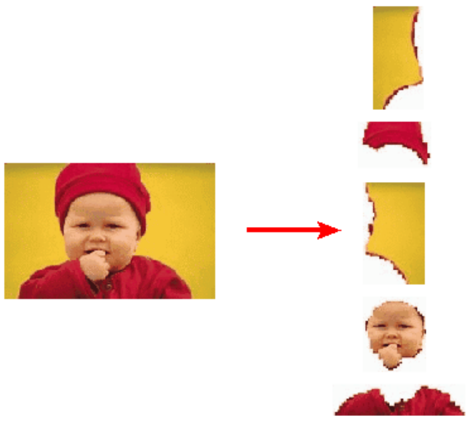


Figure. 2.13. Image segmentation

### *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.

### *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 KNOWLEDGE BASE :*

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

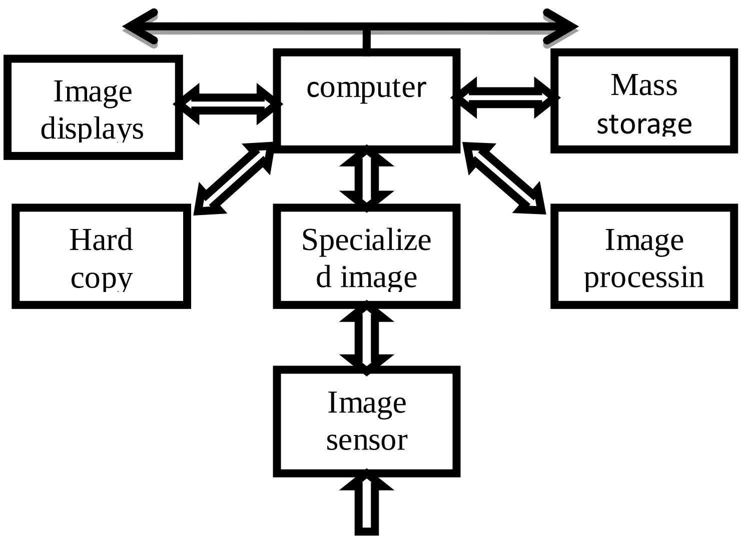


Fig. 2.14. Component of image processing

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 2.14 shows the basic components comprising a typical general-purpose system used for digital image processing.

# CHAPTER -3

# EXISTING SYSTEM

Existing methods for performing face recognition in the presence of blur are based on the convolution model and cannot handle non-uniform blurring situations that frequently arise from tilts and rotations in hand-held cameras.

Convolutional method for uniform motion blur due to camera shake. Face recognition from blurred images can be broadly classified into

1. De blurring method. the probe image is first de-blurred and then used for recognition.
2. Joint de-blurring and recognition: Computational Complexity
3. Deriving blur invariant features for recognition only for mild blurs
4. Direct recognition reblurred versions from the gallery are compared with the blurred probe image.

## 3.1 CONVOLUTION MODEL FOR SPACE-INVARIANT BLUR

The convolution model is sufficient for describing blur due to in-plane camera translations, a major limitation is that it cannot describe several other blurring effects (including out-of-plane motion and in-plane rotation) arising from general camera motion.

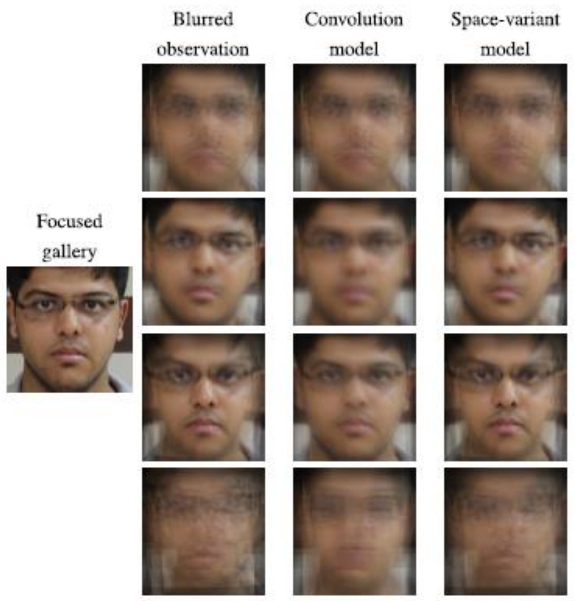


Fig 3.1Reconstructed Faces and RMS Errors

Fig. 3.1 - row 1: in-plane translations, row 2: in-plane translations and rotations, row 3: out-of-plane rotations, and row 4: full 6D blur. The reconstructed faces as well as the RMS errors are also shown in Fig. 3.1.

Note that there is no change in illumination or pose between the gallery and the probe, and only the blur has to be accounted for. Observe that except for in-plane translations (row 1), where, as expected, the RMS is the same for both the models, in all the other cases, the space-variant motion blur model gives significantly smaller RMS error than the convolution model. Note that the RMS value is smaller than one, except for 6D motion (row 4) for which it is marginally higher as our algorithm needs to search through a very large set of transformations.

Formally addresses the problem of recognizing faces from distant cameras across both blur and illumination wherein the observed blur can be well-approximated by the convolution model. However, they do not explicitly model illumination changes going from gallery to probe

Face recognition Dictionary based approach is a sparse minimization technique for recognizing faces across illumination and occlusion has been proposed

**DRAWBACKS:**

- De blurring artifacts are a major source of error especially for moderate to heavy blurs.
- The flipside of which is computational complexity in joint and recognition
- In deriving blur-invariant features is only effective for mild blurs.
- Although in subspace learning approach it is difficult to solve the problems like blur, pose, illumination etc..

# CHAPTER -4

# PROPOSED SYSTEM

## 4.1 INTRODUCTION

we propose a face recognition algorithm that is robust to non-uniform (i.e., space-varying) motion blur arising from relative motion between the camera and the subject. we assume that only a *single* gallery image is available. The camera transformations can range from in-plane translations and rotations to out-of-plane translations, out-of plane rotations, and even general 6D motion. We assume a planar structure for the face and use the geometric framework to model the blurred face as the weighted average of geometrically warped instances (homographies) of the focused gallery image. The warped instances can be viewed as the intermediate images observed during the exposure time. The weights of the wraps are referred as the transformation spread function (TSF).

The basic NU MOB robust face recognition algorithm based on the TSF. On each gallery image we apply 6D space (3 Dimensional 3 Rotations  3Translations) are stored as columns of the matrix. To recognize a blurred probe image, we minimize the distance between the probe and the convex combination of the columns of the transformation matrix corresponding to each gallery image. The gallery image whose distance to the probe is minimum is identified as a match.

We do not impose any constraints on the nature of the blur. Extensions to the basic framework to handle variations in illumination as well as pose. We approximate the face to a convex Lambertian surface, and use the 9D subspace model in [20] and the bi-convexity property of a face under blur and illumination variations in the context of the TSF model. Our motion blur and illumination (MOBIL)-robust face recognition algorithm uses an alternating minimization (AM) scheme wherein we solve for the TSF weights in the first step and use the estimated TSF to solve for the nine illumination coefficients in the second, and go on iterating till convergence. We finally transform (reblur and relight) each gallery image and compare it with the probe in the LBP space.we extend this formulation and propose an algorithm to handle motion blur, illumination and pose (MOBILAP) for non-frontal faces. The new synthesized gallery image is reblurred and relit as before, and compared with the probe using LBP (Local Bit Plane).

Our method allows for arbitrarily-shaped space-varying kernels across the image let $\mathbf{X} = [X\ Y\ Z]T$ denote the spatial coordinates of a point on the face. Let the corresponding image coordinates be $x = v\ X\ Z$ and $y = vY\ Z$ , where $v$ denotes the focal length of the camera. The projection of **X** on the image plane can be represented as $\mathbf{x} = \mathbf{KvX}$, where $\mathbf{Kv} = \mathrm{diag}(v,\ v,\ 1)$. To get the image coordinates $(x,\ y)$, the standard practice is to express **x** in homogeneous form i.e., scale **x** by its third element. At each instant of time $\tau$ during exposure, the coordinates of the 3D point **X** changes to $\mathbf{X}\boldsymbol{\tau} = \mathbf{R}\boldsymbol{\tau}\mathbf{X} + \mathbf{T}\boldsymbol{\tau}$ due to relative motion between the camera and the subject.

Here, $\mathbf{T}\boldsymbol{\tau} = [TX\tau\ TY\tau\ TZ\tau\ ]^T$ is the translation vector, and $\mathbf{R}\boldsymbol{\tau}$ represents the rotation matrix parameterized in terms of the angles of rotation $\theta X$, $\theta Y$ and $\theta Z$ about the three axes using the matrix, which seeks to explain the blur using a *single* PSF for the entire image. In fact, our scheme based on the TSF model subsumes for the special case of only in-plane translational motion, the TSF reduces to a PSF. The work proposed in this project advances the state-of-the art in many ways as discussed next.

• This is the first attempt to *systematically* address face recognition under (i) non-uniform motion blur and (ii) the combined effects of blur, illumination and pose.

• We prove that the set of all images obtained by non-uniformly blurring a given image forms a convex set. We also show that the set of all images obtained from a face image by non-uniform blurring and change of illumination forms a bi-convex set.

• We extend our method to non-frontal situations by transforming the gallery to a new pose.

• We propose a multi-scale implementation that is efficient both in terms of computation as well as memory usage.

• We demonstrate superior performance over contemporary methods on standard face databases (FERET, PIE, Extended Yale B) in the presence of blur, illumination and pose variations, as well as a real dataset which contains, in addition to these degradations, small occlusions and expression changes.

## 4.2 PROPOSED METHODOLOGY:

### 4.2.1 MOTION BLUR MODEL FOR FACES

The apparent motion of scene points in the image will vary at different locations when the camera motion is not restricted to in-plane translations. In such a scenario, the space-varying blur across the image cannot be explained using the convolution model and with a single blur kernel. In this section, we present the space-variant motion blur model [8]–[10], and illustrate how this model can explain geometric degradations of faces resulting from general camera motion. Later, we propose an optimization algorithm to recover the camera motion.

Let f : R2 → R denote the focused gallery face captured by a still camera. Assume the origin to be the camera center and let X = [X Y Z]T denote the spatial coordinates of a point on the face. Let the corresponding image coordinates be x = v X Z and y = vY Z , where v denotes the focal length of the camera. The projection of X on the image plane can be represented as x = KvX, where Kv = diag(v, v, 1).

To get the image coordinates (x, y), the standard practice is to express x in homogeneous form i.e., scale x by its third element. At each instant of time τ during exposure, the coordinates of the 3D point X changes to Xτ = RτX + Tτ due to relative motion between the camera and the subject.

Here, Tτ = [TXτ TYτ TZτ ]T is the translation vector, and Rτ represents the rotation matrix parameterized in terms of the angles of rotation θX, θY and θZ about the three axes using the matrix exponential

$$\mathbf{R}_\tau = e^{\boldsymbol{\Theta}_\tau} \text{ where } \boldsymbol{\Theta}_\tau = \begin{bmatrix} 0 & -\theta_{Z_\tau} & \theta_{Y_\tau} \\ \theta_{Z_\tau} & 0 & -\theta_{X_\tau} \\ -\theta_{Y_\tau} & \theta_{X_\tau} & 0 \end{bmatrix}$$

Intensity profile of Blurred image, where *Te* is the total exposure duration.

Let **g1** and **g2** be elements from the set ***B***. Then there exist TSFs **hT1** and **hT2** , both satisfying the conditions **hTi** ≥ 0 and ||**hTi** ||1 = 1, *i* = 1,2 such that **g1** = **AhT1** and **g2** = **AhT2** . To show that the set ***B*** is convex, we need to show that for any *γ* satisfying 0 ≤ *γ* ≤ 1, **g3** = *γ* **g1**+*(*1−*γ )***g2** is an element of ***B***. Now

$$\begin{aligned} \mathbf{g}_3 &= \gamma\,\mathbf{g}_1 + (1-\gamma)\mathbf{g}_2 \\ &= \gamma\,\mathbf{A}\mathbf{h}_{\mathbf{T}_1} + (1-\gamma)\mathbf{A}\mathbf{h}_{\mathbf{T}_2} \\ &= \mathbf{A}(\gamma\,\mathbf{h}_{\mathbf{T}_1} + (1-\gamma)\mathbf{h}_{\mathbf{T}_2}) \\ &= \mathbf{A}\mathbf{h}_{\mathbf{T}_3} \end{aligned}$$

**i. Multi scale Implementation:**

The discretization is performed in a manner that the difference in the displacements of a point light source due to two different transformations from the discrete set **T** is at least one pixel. It should be noted that since the TSF is defined over 6 dimensions, doubling their sampling resolution increases the total number of poses, *NT* , by a factor of 26. As the number of transformations in the space **T** increases, the optimization process becomes inefficient and time consuming, especially since only a few of these elements have non-zero values. Moreover, the resulting matrix **A** will have too many columns to handle. Following [9], we resort to a multiscale framework to solve this problem. We perform multiscaling in 6D (instead of 3D as in [9]). We select the search intervals along each dimension according to the extent of the blur we need to model, which is typically a few pixels for translation and a few degrees for rotation.

**ii. Face Recognition Across Blur**

Suppose we have M face classes with one focused gallery face fm for each class m, where m = 1, 2, . . . , M. Let us denote the blurred probe image which belongs to one of the M classes by g. Given fms and g, the task is to find the identity m∗ ∈ {1, 2, . . . , M} of g. Based on the discussions in Section III, the first step is to generate the matrix Am for each gallery face. Then, since g belongs to one of the M classes, it can be expressed as the convex combination of the columns of one of these matrices. Therefore, the identity of the probe image can be found by minimizing the projection error of g onto {Am}s. The reconstruction error dm can be obtained by solving *M* face classes with one focused gallery face **fm** for each class *m*, where *m* = 1, 2, . . . , *M*. first step is to generate the matrix **Am** for each gallery face. Error of **g** onto {**Am**}s. The reconstruction error *dm* can be obtained by solving

$$d_m = \min_{\mathbf{h_T}} ||\mathbf{g} - \mathbf{A_m h_T}||_2^2 + \beta ||\mathbf{h_T}||_1 \quad \text{subject to} \quad \mathbf{h_T} \geq 0.$$

by introducing a weighting matrix **W.**

$$d_m = \min_{\mathbf{h_T}} ||\mathbf{W}(\mathbf{g} - \mathbf{A_m h_T})||_2^2 + \beta ||\mathbf{h_T}||_1 \quad \text{subject to} \quad \mathbf{h_T} \geq 0$$

we first compute the optimal TSF **hTm** for each gallery image by solving

$$\mathbf{h}_{\mathbf{T}_\mathbf{m}} = \underset{\mathbf{h}_\mathbf{T}}{\operatorname{argmin}} ||\mathbf{W}(\mathbf{g} - \mathbf{A}_\mathbf{m}\mathbf{h}_\mathbf{T})||_2^2 + \beta||\mathbf{h}_\mathbf{T}||_1 \quad \text{subject to} \quad \mathbf{h}_\mathbf{T} \geq 0.$$

The Algorithm for Non-Uniform Blur-Robust Face Recognition is shown below:

```
Algorithm 1 NU-MOB: Non-Uniform Motion Blur-Robust Face Recognition
Input: Blurred probe image g and a set of gallery images
    f_m, $m = 1, 2, ..., M$.
Output: Identity of the probe image.
  1: For each gallery image f_m, find the optimal TSF $h_{T_m}$ by
     solving equation (13).
  2: Blur each gallery image f_m with its corresponding $h_{T_m}$
     and extract LBP features.
  3: Compare the LBP features of the probe image g with those
     of the transformed gallery images and find the closest
     match.
```

### 4.2.2 FACE RECOGNITION ACROSS BLUR, ILLUMINATION, AND POSE

Poor illumination is often an accompanying feature in blurred images because larger exposure times are needed to compensate for the lack of light which increases the chances of camera shake. Pose variation is another challenge for realizing the true potential of face recognition systems in practice. This section is devoted to handling the combined effects of blur, illumination and pose.

**A. Handling Illumination Variations**

To handle illumination variations, we modify our basic blur-robust algorithm (NU-MOB) by judiciously utilizing the following two results:

Using this "universal configuration" of lighting positions, an image f of a person under any illumination condition can be written as

$$f = \sum_{i=1}^{9} \alpha_i f_i$$

Image *f* of a person under any illumination condition where $\alpha i$ , $i$ = 1, 2, . . . , 9 are the corresponding linear coefficients. The *fi* s, which form a basis for this 9D subspace, can be generated using the Lambertian reflectance model as can be written as

$$f_i(r, c) = \rho(r, c)\ \max(\mathbf{n}(r, c)^T \mathbf{s}_\mathbf{i}, 0)$$

```
Algorithm 2 MOBIL: Motion Blur and Illumination-Robust Face Recognition
Input: Blurred and differently illuminated probe image g, and
    a set of gallery images $\mathbf{f_m}, m = 1, 2, ..., M$.
Output: Identity of the probe image.
1: For each gallery image $\mathbf{f_m}$, obtain the nine basis images
   $\mathbf{f_{m,i}}, i = 1, 2, ..., 9$.
2: For each gallery image $\mathbf{f_m}$, find the optimal TSF $\mathbf{h_{T_m}}$ and
   illumination coefficients $\alpha_{m,i}$ by solving equation (16).
3: Transform (blur and re-illuminate) the gallery images
   $\mathbf{f_m}$ using the computed $\mathbf{h_{T_m}}$ and $\alpha_{m,i}$ and extract LBP
   features.
4: Compare the LBP features of the probe image g with those
   of the transformed gallery images and find the closest
   match.
```

**B. Handling Pose Variations**

Most face recognition algorithms are robust to small variations in pose (~ 15°) [25], but the drop in performance is severe for greater yaw and pitch angles. In our xperiments, we found this to be true of our MOBIL algorithm also. The reason behind this drop in accuracy is that intra-subject variations caused by rotations are often larger than inter-subject differences.

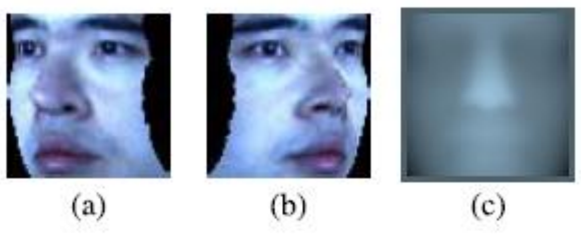
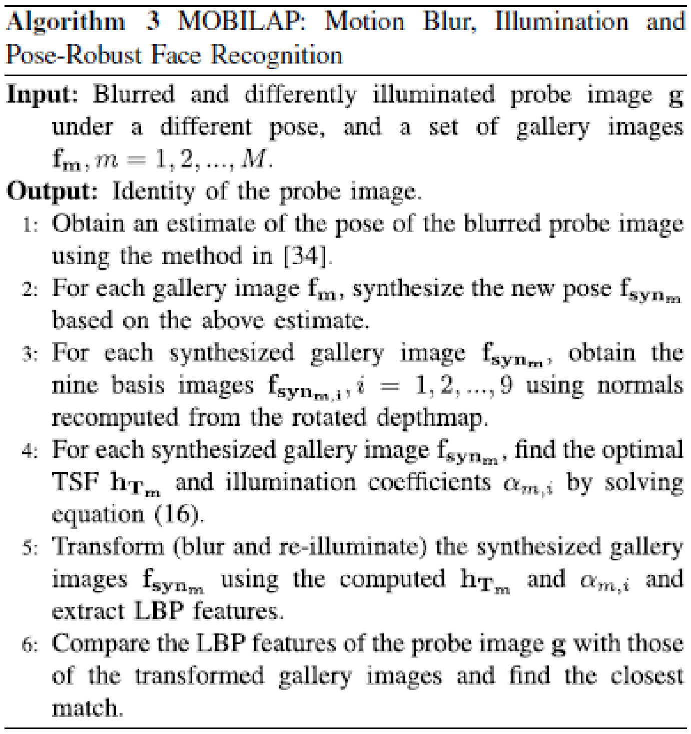


Fig. 4.2 Example images of a subject from the PIE database under new poses. The images in (a) and (b) are synthesized from the frontal gallery image using the average face depthmap shown in (c).

Clearly, there is no overstating the formidable nature of the problem at hand - recognizing faces across blur, illumination and pose. To this end, we next propose our MOBILAP algorithm which, using an estimate of the pose, matches the incoming probe with a synthesized non-frontal gallery image. To the best of the authors' knowledge, this is the first ever effort to even attempt this compounded scenario.



**Algorithm 3** MOBILAP: Motion Blur, Illumination and Pose-Robust Face Recognition

```
Input: Blurred and differently illuminated probe image g
    under a different pose, and a set of gallery images
    $\mathbf{f_m}, m = 1, 2, ..., M$.
Output: Identity of the probe image.
1: Obtain an estimate of the pose of the blurred probe image
   using the method in [34].
2: For each gallery image $\mathbf{f_m}$, synthesize the new pose $\mathbf{f_{syn_m}}$
   based on the above estimate.
3: For each synthesized gallery image $\mathbf{f_{syn_m}}$, obtain the
   nine basis images $\mathbf{f_{syn_{m,i}}}, i = 1, 2, ..., 9$ using normals
   recomputed from the rotated depthmap.
4: For each synthesized gallery image $\mathbf{f_{syn_m}}$, find the optimal
   TSF $\mathbf{h_{T_m}}$ and illumination coefficients $\alpha_{m,i}$ by solving
   equation (16).
5: Transform (blur and re-illuminate) the synthesized gallery
   images $\mathbf{f_{syn_m}}$ using the computed $\mathbf{h_{T_m}}$ and $\alpha_{m,i}$ and
   extract LBP features.
6: Compare the LBP features of the probe image g with those
   of the transformed gallery images and find the closest
   match.
```

**4.2.3 LBP Algorithm:**

To implement the face recognition in this work, we proposed the Local Binary patterns methodology. Local Binary Pattern works on local features that uses LBP operator which summarizes the local special structure of a face image. LBP is defined as an orders set of binary comparisons of pixels intensities between the center pixels and its eight surrounding pixels. Local Binary Pattern do this comparison by applying the following formula:

$$LBP(x_c , y_c) = \Sigma^{n}{}_{7} s(i_n - i_c)2^n$$

Where $i_c$ corresponds to the value of the center pixel ( , $y_c$), in to the value of eight surrounding pixels. It is used to determine the local features in the face and also works by using basic LBP operator. Feature extracted matrix originally of size 3 x 3, the values are compared by the value of the centre pixel, then binary pattern code is produced and also LBP code is obtained by converting the binary code into decimal one.

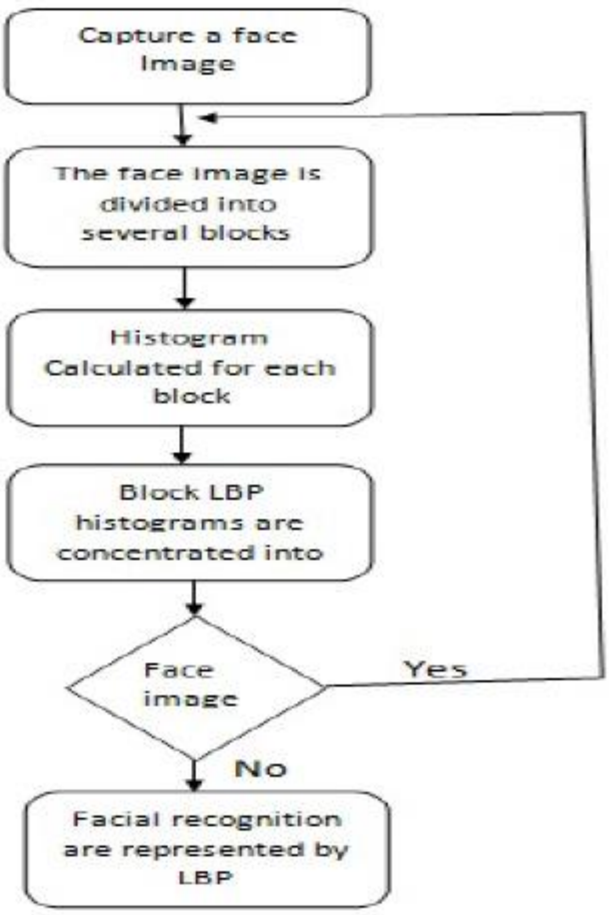
**The Face Recognition Algorithm**

Input: Training Image set.

Output: Feature extracted from face image and compared with centre pixel and recognition with unknown face image.

```
1. Initialize temp = 0
2. FOR each image I in the training image set
3. Initialize the pattern histogram, H = 0
4. FOR each center pixel $t_c$ € I
 5. Compute the pattern label of $t_c$, LBP(1)
6. Increase the corresponding bin by 1.
7. END FOR
8. Find the highest LBP feature for each face image and combined into single vector.
9. Compare with test face image.
10. If it match it most similar face in database then successfully recognized
```

**LBP Flowchart:**


Capture a face Image
The face image is divided into several blocks
Histogram Calculated for each block
Block LBP histograms are concentrated into
Face image
Yes
No
Facial recognition are represented by LBP

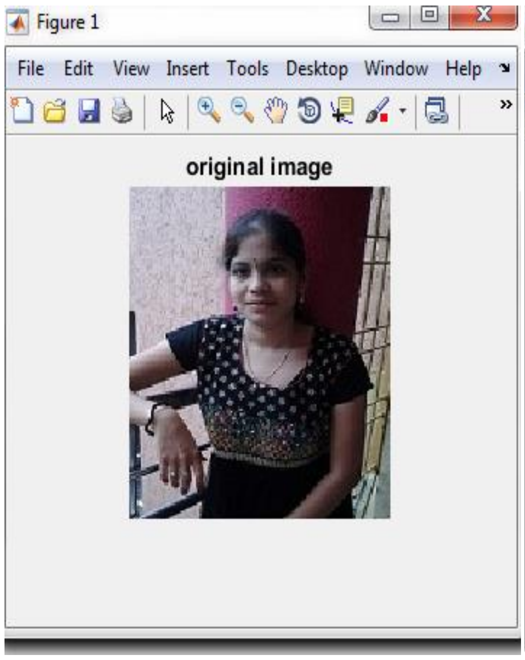
**ADVANATAGES OF PROPOSED SYSTEM:**

- It efficiently deals with blurred images.
- This is the first attempt to systematically address face recognition under (i) non-uniform motion blur and (ii) the combined effects of blur, illumination and pose.
- We prove that the set of all images obtained by non-uniformly blurring a given image forms a convex set. We also show that the set of all images obtained from a face image by non-uniform blurring and change of illumination forms a bi-convex set.
- We extend our method to non-frontal situations by transforming the gallery to a new pose.

# CHAPTER -5

# RESULTS

## Non-uniform motion blur:



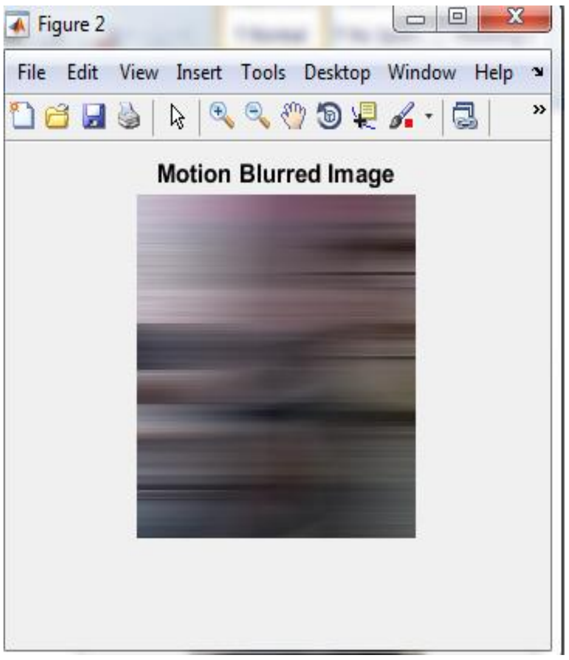


**FIG1: Test image taken as input** **FIG2: Motion blurred image**

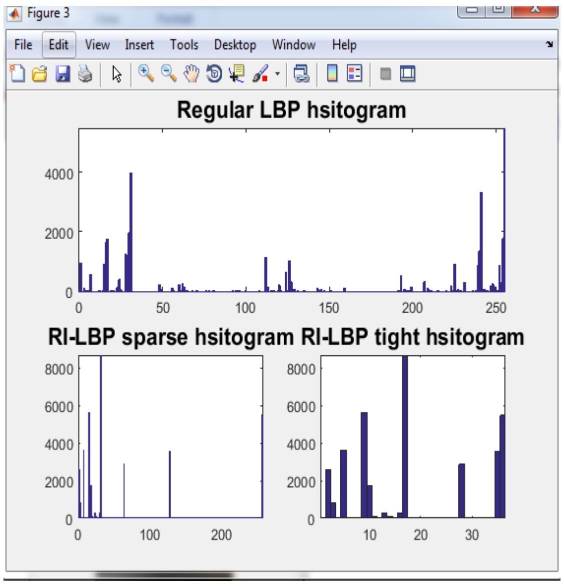


**FIG3:** **LBP histogram for the image**

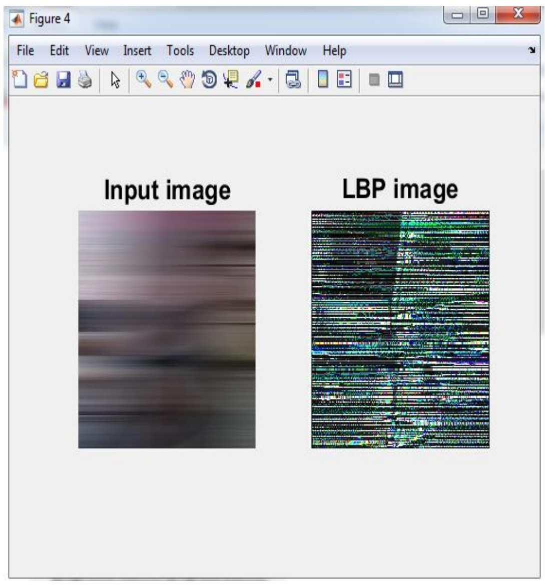


**FIG4:Compare input image with LBP image**

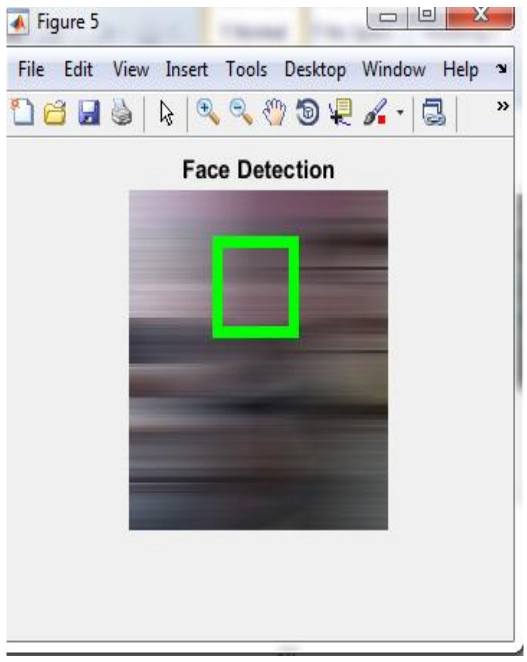


**FIG5:Face detection for the image**

Figure 6
File Edit View Insert Tools Desktop Window Help
Test Image from algorithm

**FIG6:Test image from algoritham**

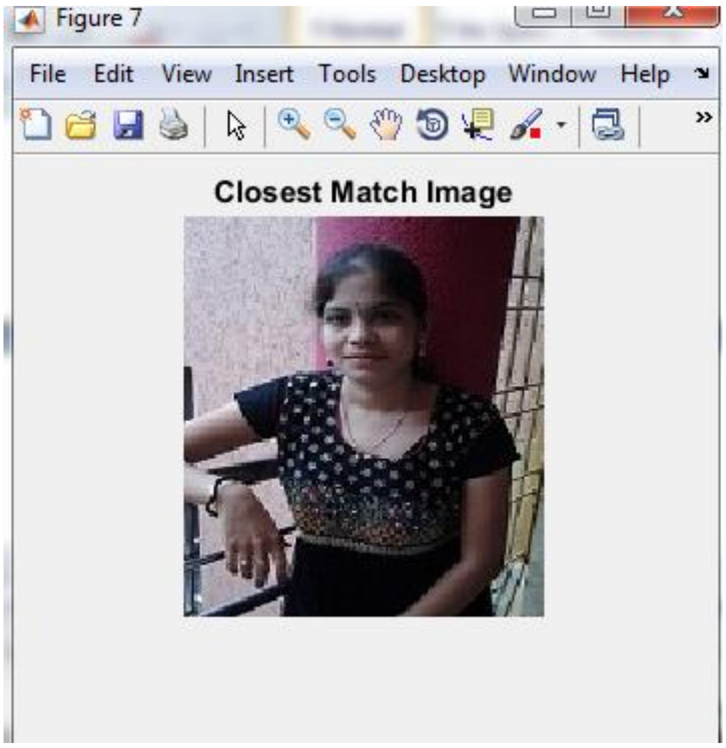
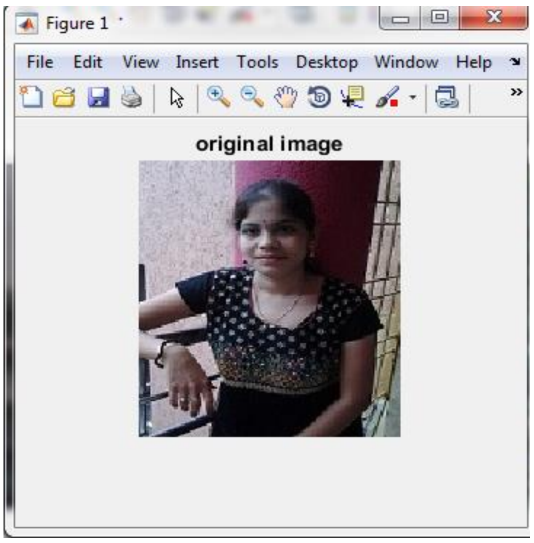


**FIG7:closest image for the input**

## Pose:



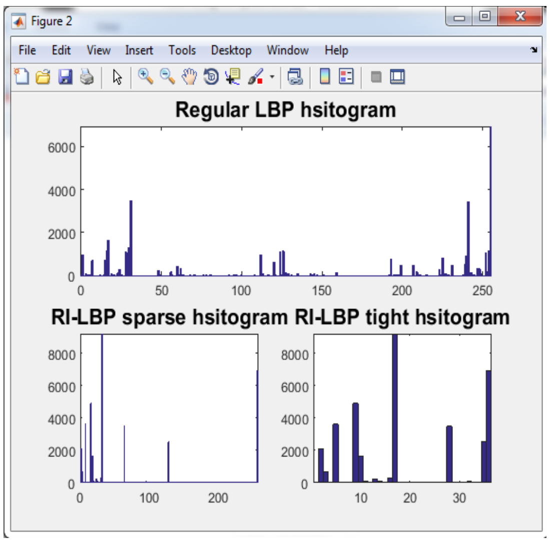


FIG1: Test image taken as input

Fig2: LBP histogram for the image

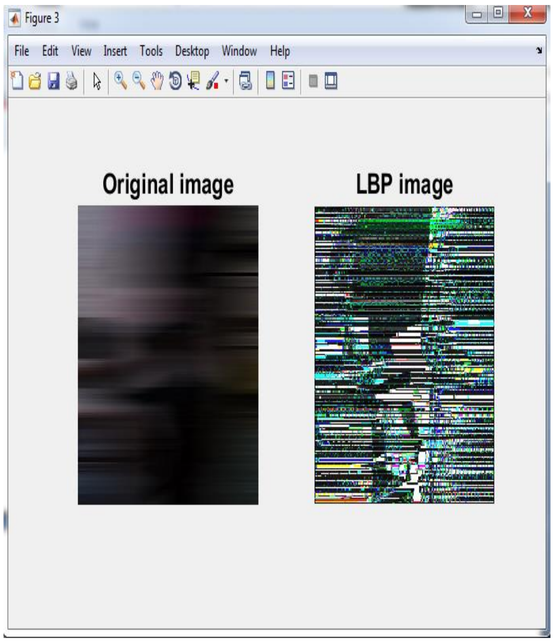


FIG3:compare test image with LBP image

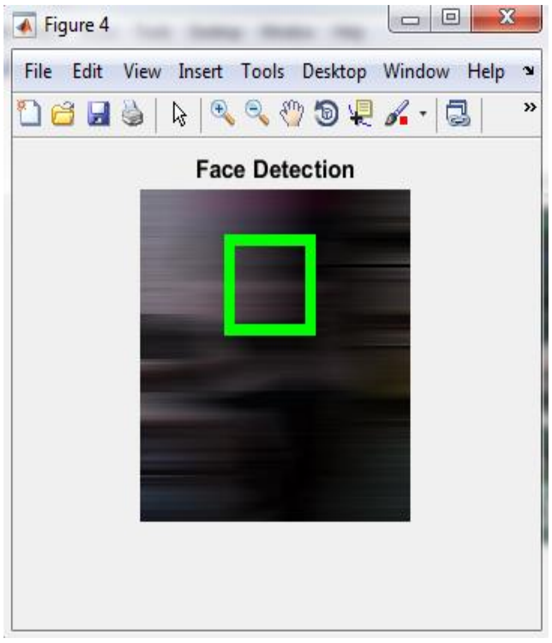


FIG4:Face detection from image

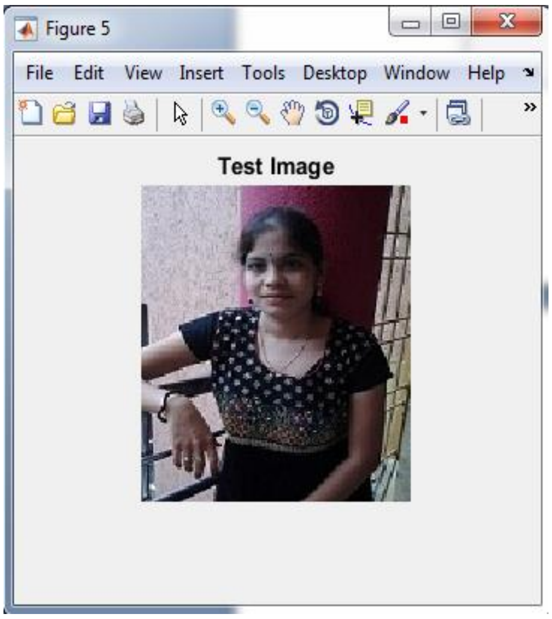


FIG5: Test image

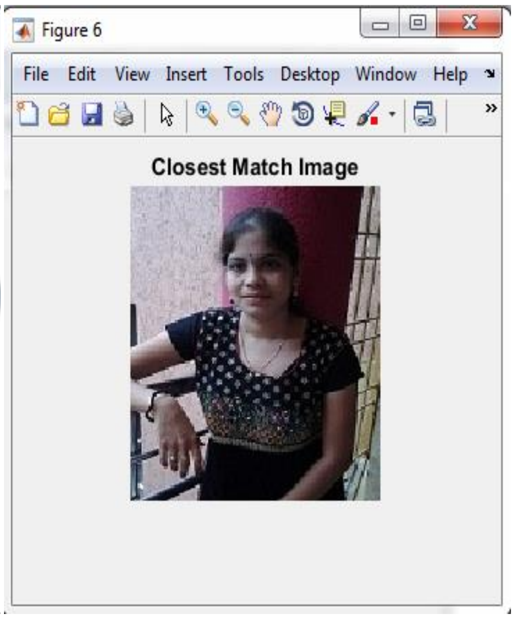
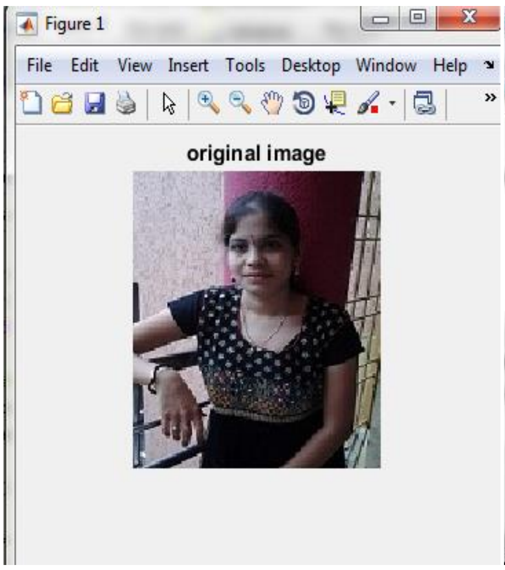


FIG6:closest match for the image

## Illumination:



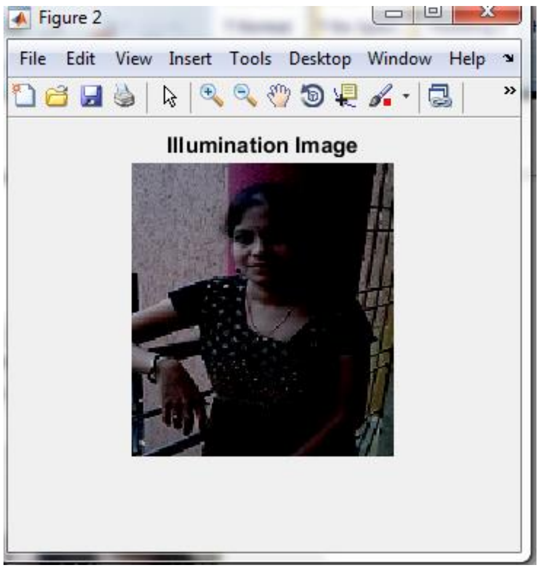
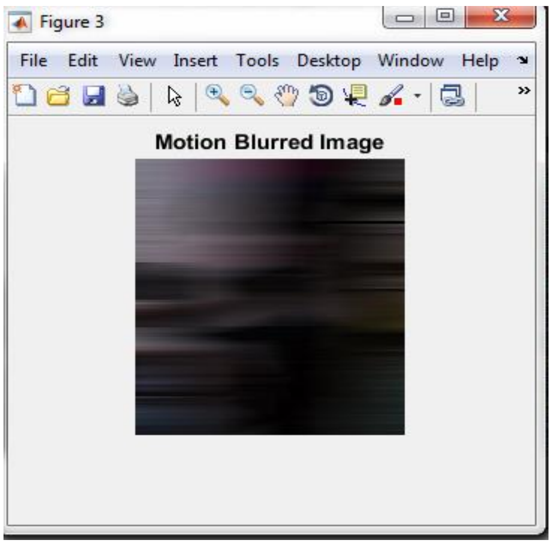


FIG1: Test image taken as input FIG2:Illuminated image



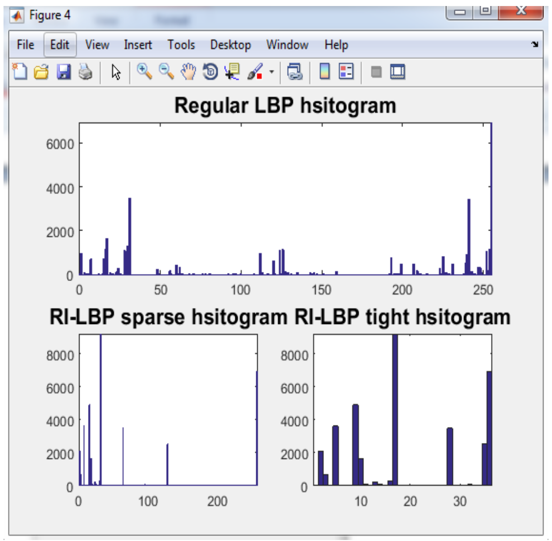


**FIG3:**motion blurred image **FiG4:**LBP histogram

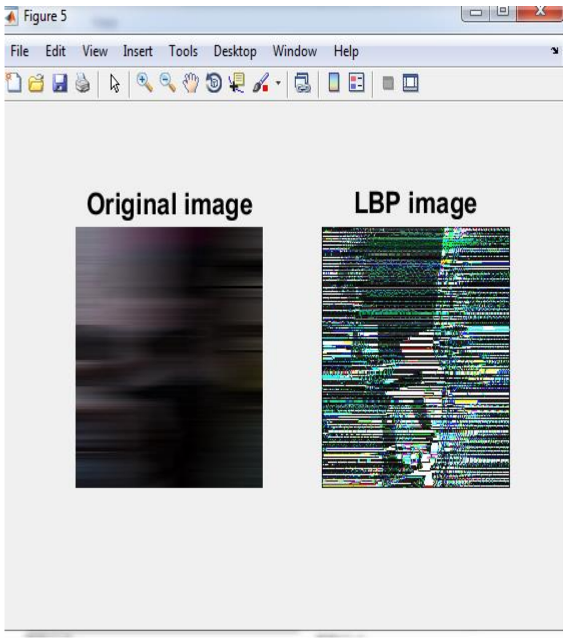


**FIG5:test image compared with LBP image**

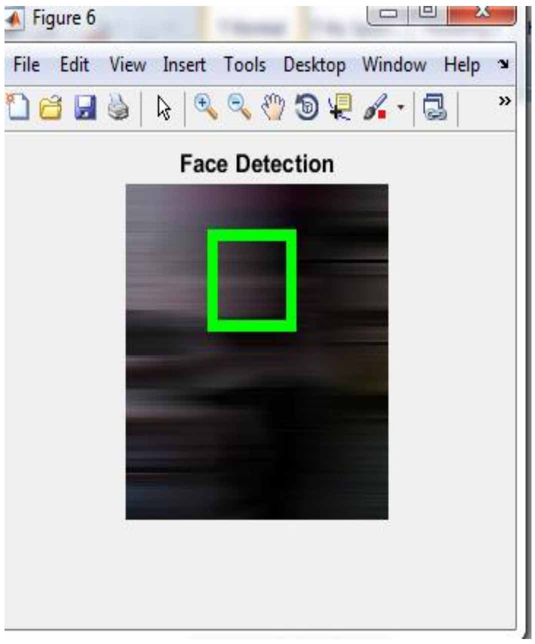
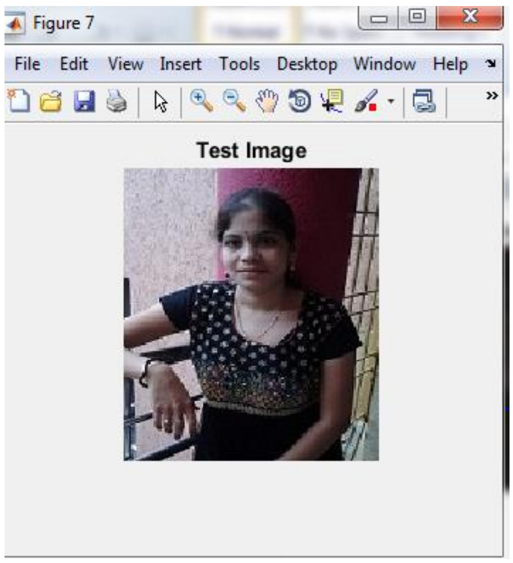


**FIG6:Face detection**



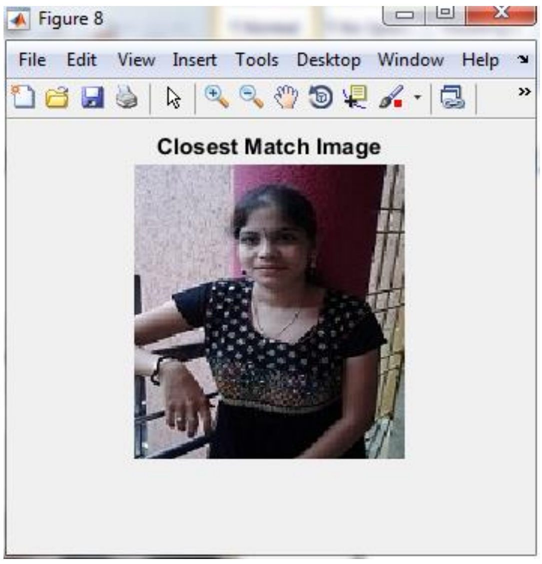


**FIG7: Test image** **FIG8:Closest match for the image**

# CHAPTER -6

# CONCLUSION

We proposed a methodology to perform face recognition under the combined effects of non-uniform blur, illumination, and pose. We showed that the set of all images obtained by non-uniformly blurring a given image using the TSF model is a convex set given by the convex hull of warped versions of the image. Capitalizing on this result, we initially proposed a non-uniform motion blur-robust face recognition algorithm NU-MOB. We then showed that the set of all images obtained from a given image by non-uniform blurring and changes in illumination forms a bi-convex set, and used this result to develop our non-uniform motion blur and illumination-robust Algorithm MOBIL. We then extended the capability of MOBIL to handle even non-frontal faces by transforming the gallery to a new pose. We established the superiority of this method called MOBILAP over contemporary techniques. Extensive experiments were given on synthetic as well as real face data. The limitation of our approach is that significant occlusions and large changes in facial expressions cannot be handled.

# CHAPTER -7

# BIBILOGRAPHY

[1] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: A literature survey," ACM Comput. Surv., vol. 35, no. 4, pp. 399–458, Dec. 2003.

[2] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman, "Removing camera shake from a single photograph," ACM Trans. Graph., vol. 25, no. 3, pp. 787–794, Jul. 2006.

[3] Q. Shan, J. Jia, and A. Agarwala, "High-quality motion deblurring from a single image," ACM Trans. Graph., vol. 27, no. 3, pp. 73:1–73:10, Aug. 2008.

[4] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman, "Understanding blind deconvolution algorithms," IEEE Trans. Pattern Anal. Mach. Intell., vol. 33, no. 12, pp. 2354–2367, Dec. 2011.

[5] M. Šorel and F. Šroubek, "Space-variant deblurring using one blurred and one underexposed image," in Proc. 16th IEEE Int. Conf. Image Process., Nov. 2009, pp. 157–160.

[6] H. Ji and K. Wang, "A two-stage approach to blind spatially-varying motion deblurring," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2012, pp. 73–80.

[7] S. Cho, Y. Matsushita, and S. Lee, “Removing non-uniform motion blur from images,” in Proc. Int. Conf. Comput. Vis., Oct. 2007, pp. 1–8.

[8] Y.-W. Tai, P. Tan, and M. S. Brown, “Richardson-Lucy deblurring for scenes under a projective motion path,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 33, no. 8, pp. 1603–1618, Aug. 2011.

[9] O.Whyte, J. Sivic, A. Zisserman, and J. Ponce, “Non-uniform deblurring for shaken images,” Int. J. Comput. Vis., vol. 98, no. 2, pp. 168–186, 2012.

[10] A. Gupta, N. Joshi, L. Zitnick, M. Cohen, and B. Curless, “Single image deblurring using motion density functions,” in Proc. Eur. Conf. Comput. Vis., 2010, pp. 171–184.

[11] Z. Hu and M.-H. Yang, “Fast non-uniform deblurring using constrained camera pose subspace,” in Proc. Brit. Mach. Vis. Conf., 2012, pp. 1–11.

[12] C. Paramanand and A. N. Rajagopalan, “Non-uniform motion deblurring for bilayer scenes,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2013, pp. 1115–1122.

[13] H. Hu and G. de Haan, “Adaptive image restoration based on local robust blur estimation,” in Proc. 9th Int. Conf. Adv. Concepts Intell. Vis. Syst., 2007, pp. 461–472.

[14] M. Nishiyama, A. Hadid, H. Takeshima, J. Shotton, T. Kozakaya, and O. Yamaguchi, “Facial deblur inference using subspace analysis for recognition of blurred faces,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 33, no. 4, pp. 838–845, Apr. 2011.

[15] H. Zhang, J. Yang, Y. Zhang, N. M. Nasrabadi, and T. S. Huang, “Close the loop: Joint blind image restoration and recognition with sparse representation prior,” in Proc. Int. Conf. Comput. Vis., Nov. 2011, pp. 770–777.

[16] T. Ahonen, E. Rahtu, V. Ojansivu, and J. Heikkila, “Recognition of blurred faces using local phase quantization,” in Proc. 19th Int. Conf. Pattern Recognit., Dec. 2008, pp. 1–4.

[17] R. Gopalan, S. Taheri, P. Turaga, and R. Chellappa, “A blur-robust descriptor with applications to face recognition,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 34, no. 6, pp. 1220–1226, Jun. 2012.

[18] I. Stainvas and N. Intrator, “Blurred face recognition via a hybrid network architecture,” in Proc. 15th Int. Conf. Pattern Recognit., vol. 2. Sep. 2000, pp. 805–808.

[19] P. Vageeswaran, K. Mitra, and R. Chellappa, “Blur and illumination robust face recognition via set-theoretic characterization,” IEEE Trans. Image Process., vol. 22, no. 4, pp. 1362–1372, Apr. 2013.

[20] K.-C. Lee, J. Ho, and D. Kriegman, “Acquiring linear subspaces for face recognition under variable lighting,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 27, no. 5, pp. 684–698, May 2005.

# APPENDIX

# SOFTWARE DESCRIPTION

## 1.1 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
- 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 makes it possible for you to solve many technical computing issues, particularly those with matrix and vector formulations, in a fraction of the time it will take to write a application in a scalar non interactive language similar to C or FORTRAN.

The title MATLAB stands for matrix laboratory. MATLAB was once in the beginning written to provide convenient entry to matrix software developed through the LINPACK and EISPACK projects. Today, MATLAB uses software developed through the LAPACK and ARPACK tasks, which collectively signify the brand new in application for matrix computation.

MATLAB has developed over a period of years with enter from many customers. In tuition environments, it's the usual instructional instrument for introductory and advanced courses in arithmetic, engineering, and science.

In enterprise, MATLAB is the tool of option for top-productivity study, development, and analysis.

MATLAB features a household of software-targeted options called toolboxes. Very primary to most customers of MATLAB, toolboxes enable you to learn and follow specialized technology. Toolboxes are complete collections of MATLAB functions (M-files) that lengthen the MATLAB atmosphere to clear up distinct courses of issues. Areas wherein toolboxes are to be had comprise sign processing, manage techniques, neural networks, fuzzy good judgment, wavelets, simulation, and lots of others.

## 1.2 THE MATLAB SYSTEM

The MATLAB system consists of five main parts:

***Development Environment:***

This is the set of tools and services that aid you employ MATLAB features and documents. Many of those instruments are graphical person interfaces. It includes the MATLAB computing device and Command Window, a command historical past, and browsers for viewing support, the workspace, files, and the search course.

***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:***

It is a high-degree matrix/array language with manage drift statements, features, knowledge structures, enter/output, and object-oriented programming points. It permits each "programming within the small" to rapidly create quick and dirty throw-away applications, and "programming within the giant" to create entire tremendous and tricky utility programs.

***Handle Graphics:***

This is the MATLAB pictures approach. It involves high-degree commands for 2-dimensional and third-dimensional data visualization, image processing, animation, and presentation snap shots. It also entails low-level commands that enable you to fully customize the appearance of photos as well as to build whole graphical user interfaces in your MATLAB functions.

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

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

## 1.3 DEVELOPMENT ENVIRONMENT

This presents a brief introduction to beginning and quitting MATLAB, and the tools and services that help you to work with MATLAB variables and files. For extra information about the topics blanketed right here, see the corresponding themes beneath development environment in the MATLAB documentation, which is to be had online as good as in print.

## 1.4 MATLAB DESKTOP

Whilst you begin MATLAB, the MATLAB laptop seems, containing tools (graphical consumer interfaces) for managing records, variables, and purposes associated with MATLAB.

The primary time MATLAB begins, the laptop appears as shown in the following illustration, even though your Launch Pad may contain specific entries.

That you may trade the way your desktop appears by means of opening, closing, moving, and resizing the tools in it. Which you can additionally move instruments external of the computing device or return them back inside the computer (docking).

The entire computer instruments furnish original elements corresponding to context menus and keyboard shortcuts.

That you can specify specified traits for the desktop tools via picking Preferences from the File menu. For example, that you would be able to specify the font characteristics for Command Window text. For more understanding, click the aid button within the Preferences dialog box.

### *1.4.1 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:

***Command Window***

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

***Command History***

Strains you enter within the Command Window are logged in the Command historical past window. Within the Command historical past, you could view beforehand used functions, and duplicate and execute selected traces. To avoid wasting the input and output from a MATLAB session to a file, use the diary function.

***Running External Programs***

That you may run external packages from the MATLAB Command Window. The exclamation point personality! Is a shell escape and indicates that the leisure of the enter line is a command to the operating approach. This is useful for invoking utilities or jogging different applications without quitting MATLAB. On Linux, for illustration, EmacsMagik.M invokes an editor called emacs for a file named Magik.M. While you give up the outside software, the operating system returns manipulate 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 on the aid button in the toolbar, or type aid browser within the Command Window. The support browser consists of two panes, the aid Navigator, which you utilize to find information, and the display pane, where you view the information.

***Editor/Debugger***

Use the Editor/Debugger to create and debug M-records, which can be applications you write to run MATLAB features. The Editor/Debugger provides a graphical person interface for basic text editing, as good as for M-file debugging.

You should use any textual content editor to create M-records, reminiscent of Emacs, and can use preferences (obtainable from the laptop File menu) to specify that editor as the default. If you happen to use an additional editor, that you can nonetheless use the MATLAB Editor/Debugger for debugging, or you should utilize debugging services, akin to db stop, which sets a breakpoint.

## 1.5 MANIPULATING MATRICES

### *ENTERING MATRICES:*

The best way for you to get started with MATLAB is to learn how to handle matrices. This exactly matches the numbers in the engraving. Once you have entered the matrix, it is automatically remembered in the MATLAB workspace. You can refer to it simply as A.

### *EXPRESSIONS:*

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

***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 means of any number of letters, digits, or underscores. MATLAB uses simplest the primary 31 characters of a variable title. MATLAB is case touchy; it distinguishes between uppercase and lowercase letters. A and a aren't the same variable. To view the matrix assigned to any variable, with ease enter the variable identify.

### *Numbers*

MATLAB makes use of traditional decimal notation, with an optional decimal factor and main plus or minus sign, for numbers. Scientific notation makes use of the letter e to specify a vigour-of-ten scale aspect. Imaginary numbers use either i or j as a suffix. Some examples of legal numbers are

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

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

***Operators***

Expressions use familiar arithmetic operators and precedence rules.

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

| | |
|---|---|
| ( ) | Specify evaluation order |

***Functions***

MATLAB provides a large number of standard elementary mathematical functions, including abs, sqrt, exp, and sin. Taking the square root or logarithm of a poor quantity just isn't an error; the suitable elaborate result is produced robotically. MATLAB also supplies many extra developed mathematical functions, including Bessel and gamma features. A lot of these capabilities be given complex arguments. For a record of the basic mathematical capabilities, kind support elfun. For a list of more advanced mathematical and matrix functions, type help spec fun help elmat. Some of the functions, like sqrt and sin, are built-in. . They're part of the MATLAB core so they are very efficient, but the computational small print aren't easily accessible. Different capabilities, like gamma and sinh, are implemented in M-records. You will see the code and even modify it if you wish to have. Several targeted capabilities furnish values of useful constants.

***GUI :***

A graphical user interface (GUI) is a user interface built with graphical objects, such as buttons, text fields, sliders, and menus. Probably, these objects already have meanings to most computer customers.

For illustration, whilst you move a slider, a value changes, while you press an good enough button, your settings are utilized and the dialog box is brushed aside of direction, to leverage this built-in familiarity, you have got to be constant in how you employ the quite a lot of GUI-building add-ons.

Applications that provide GUIs are almost always simpler to gain knowledge of and use on the grounds that the individual using the applying does now not have to be aware of what instructions are available or how they work. The motion that results from a targeted user action will also be made clear by using the design of the interface.

The sections that comply with describe create GUIs with MATLAB. This involves laying out the add-ons, programming them to do targeted things in accordance with consumer moves, and saving and launching the GUI; in other words, the mechanics of creating GUIs. This documentation does not attempt to cover the "art" of good user interface design, which is an entire field unto itself. Topics covered in this section include:

***Creating GUIs with GUIDE***

MATLAB implements GUIs as figure windows containing various styles of uicontrol objects. You must program each object to perform the intended action when activated by the user of the GUI. In addition, you must be able to save and launch your GUI. All of these tasks are simplified by GUIDE, MATLAB's graphical user interface development environment.

***GUI Development Environment***

The process of implementing a GUI involves two basic tasks:

- 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.

***Features of the GUIDE-Generated Application M-File***

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 call back 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).

***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 find obj, 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.

***User Interface Controls***

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 Call-back properties