



Jaypee University of Information Technology
Solan (H.P.)

LEARNING RESOURCE CENTER

Acc. Num.

Call Num:

General Guidelines:

- ◆ Library books should be used with great care.
- ◆ Tearing, folding, cutting of library books or making any marks on them is not permitted and shall lead to disciplinary action.
- ◆ Any defect noticed at the time of borrowing books must be brought to the library staff immediately. Otherwise the borrower may be required to replace the book by a new copy.
- ◆ The loss of LRC book(s) must be immediately brought to the notice of the Librarian in writing.

Learning Resource Centre-JUIT



SP03167

IMAGE COMPRESSION AND RETRIEVAL USING FOURIER SEREIS CO-EFFIECIENTS

By

ARUN KUMAR - 031045

HARIT JAMWAL - 031046

ANKUR MISHRA - 031108

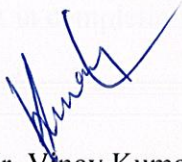


MAY-2007

DEPARTMENT OF ELECTRONICS AND
COMMUNICATIONS
JAYPEE UNIVERSITY OF INFORMATION
TECHNOLOGY-WAKNAGHAT

CERTIFICATE

This is to certify that the work entitled, "IMAGE COMPRESSION AND RETRIEVAL" submitted by Arun Kumar , Roll No. 031045, Harit Jmawal , Roll No. 031046 & Ankur Mishra , Roll No. 031108 in partial fulfillment for the award of degree of Bachelor of Technology in Electronics And Communication of Jaypee University of Information Technology has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.



Mr. Vinay Kumar
(Senior Lecturer,
Deptt. E.C.E)



Mr .S.V.Bhooshan
(H.O.D.,
Deptt. E.C.E.)

ACKNOWLEDGMENT

First of all, we would like to thank our project supervisor Mr. Vinay Kumar, Senior Lecturer (Department of Electronics And Communications) for his able guidance and support in the conception and development of this project. His suggestions in visualizing the project and sustained interest to attain the objective envisaged in the project are gratefully acknowledged.

A special thanks to Mr. S.V. Bhooshan, HOD (Department of Electronics And Communications) for his constant guiding vision and motivation that went a long way in materializing this project.

Along with this we would also like to thank Miss Shipra (Department of Computer Science) for helping and guiding us throughout the duration of the project.

We would also like to express thanks to our friend Sankalp Saurabh who helped us a lot in completion of the project.

Arun Kumar

(031045)

Harit Jamwal

(031046)

Ankur Mishra

(031108)

TABLE OF CONTENTS

1. INTRODUCTION

1.1 A Digital Image.....	2
1.2 Image Compression.....	2
1.3 The Need To Compress.....	4

2. VARIOUS STANDARDS AND THEIR PRAGMATISM

2.1 Picture Formats.....	5
--------------------------	---

3. INTRODUCTION TO BITMAP IMAGE

3.1 Definition Of Bitmap Image.....	7
3.2 Facts About Bitmap Image.....	8
3.3 Format Of Bitmap Image.....	9

4. COMPRESSION TECHNIQUES

4.1 Lossless Compression.....	11
4.2 Lossy Compression.....	11

5. METHODOLOGY EMPLOYED

5.1 Compression Methodology.....	12
5.2 Retrieval Methodology.....	17

6. SYSTEM DESIGN AND IMPLEMENTATION

6.1 Algorithms.....	20
6.2 Flowcharts for Compression and Retrieval.....	23

6.3 The Design	25
6.4 GUI'S.....	28
7. CONCLUSION.....	33
8. SCOPE OF PROJECT.....	34

LIST OF FIGURES

Chapter 3:- INTRODUCTION BITMAP IMAGE

- Fig 3.1(a) A zoomed Bitmap Image

Chapter 5: - METHODOLOGY EMPLOYED

- Fig:5.1(a) Original Image
- Fig:5.2(B) Image after Sub-parts

Chapter 6: - ALGORITHMS

- Fig:6.2(a) For compression
- Fig:6.2(b) For retrieval
- Fig:6.4(a) Main screen
- Fig:6.4(b) After loading the image
- Fig:6.4(c) After entering segmentation values and displaying subparts
- Fig:6.4(d) Compressed image
- Fig:6.4(e) The retrieved image

LIST OF ABBREVIATIONS

- **2-D** Two Dimensional
- **AFT** Applying Fourier Transform
- **BMP** Bitmap
- **DFT** Discrete Fourier Transform
- **FT** Fourier Transform
- **GIF** Graphics Interchange Format
- **IFT** Inverse Fourier Transform
- **JPEG** Joint Photographics Expert Group
- **PNG** Portable Networks Graphics
- **TIFF** Tagged Image File Format

ABSTRACT

Digital image compression is the process of reducing the image size by applying various techniques on the original input. Image compression is readily needed in today's paradigm of communication, especially when we have ever increasing demand of bandwidth. The image compression usually degrades the quality of the image. It is very high sometimes despite being an unwanted phenomenon. It can be said that it is primarily due to the Compression Techniques being used.

Thus, we can try to minimize the distortion in the image and produce an image of same eternal quality as the original one by applying techniques which provide us with the most sophisticated mathematical framework which describes a Digital Image most efficiently.

So, as new Compression techniques are being evolved as to save time and for better communications we have tried a new compression standard by applying Discrete Fourier Transform on the Digital Image.

This has been shown and performed in this project successfully.

CHAPTER 1 INTRODUCTION

Image compression is the application of data compression on digital images. In effect, the objective is to reduce redundancy of the image data in order to be able to store or transmit data in an efficient form.

Data Compression has become an important factor in relation to storage and transmission of large amounts of image data. Compression based on Fourier Transforms and Lempel Ziv Coding is a new and promising technique for image data compression. Digitalized images and sequence of images often have the property of being both data extensive and relatively redundant objects. This makes them an obvious target for data compression.

In a distributed environment large image files remain a major bottleneck within systems. Compression is an important component of the solutions available for creating file sizes of manageable and transmittable dimensions. Increasing the bandwidth is another method but the cost sometimes makes this a less attractive solution. Platform portability and performance are important in the selection of the compression/decompression technique to be employed. Compression solutions today are more portable due to the change from proprietary high-end solutions to accepted and implemented international standards.

So even with one of the densest storage capacity media commercially available the need for data compression of images is evident.

1.1 A Digital Image

An image may be defined as a two-dimensional function , $f(x,y)$ where x and y are spatial (Planar) co-ordinates , and the amplitude of f at any pair of co-ordinates (x,y) is called **intensity or graylevel** of the image of that point. When x,y and amplitude values of f are all discrete and finite quantities, we call the image as digital image.

A Digital Image is composed of a finite number of elements , each of which has a particular location and value. These elements are referred to as **Picture Elements , Image Elements , Pels or Pixels** . Pixel being the most well known and most widely used term . Hence, it is composed of discrete pixels of digitally quantized brightness and colour.

1.2 Image Compression

The goal of image compression is to remove redundancy in images and transmit only the minimum information necessary for reconstruction. The better this is achieved by signal processing , the more efficiently we can communicate with digital images. The difficulty lies in the task of coming up with a good model for an image ; that is , a mathematical framework that can be used describe every conceivable image that can be digitally captured .

Just to define some of the major types of redundancies that we face in a Digital Image are stated and explained below.

1.2.1 Coding Redundancy

Pixels have to be encoded in images. Like all processing in a digital computer. We have to assign some arbitrary set of bits for all different possible values. Sometimes , some

values are more probable than others. As such, they are more expected and thus have less information content. The key to reduce redundancy is to assign bits according to the information content. So, the thumb rule says that the more information, the more bits ; the less information , the fewer bits.

We can use variable-length coding for pixels or other information for images. One of the methods for doing so is Lempel Ziv coding which we have applied further in this project.

1.2.2 Interpixel Redundancy

Suppose if you were to scan across a row of binary image and read off the values, and if the values were “white”, “white”, “white”, what would you guess the next one to be ? While either “black” or “white” are possible, you’d probably guess that next one is white. This makes sense for images because they usually have regions of constant constant or similar values. This is called interpixel redundancy.

So, the coding process defined in earlier section, however would not alter the level of correlation between the pixels within the images. In other words the codes used to represent the gray levels of each image have nothing to do with correlation between the pixels.

1.2.3 Psychovisual Redundancy

The third way of reducing redundancy is to realize that our senses aren’t equally sensitive to all things. In this sense, information content relates to the ability of our visual systems to tell the difference.

Examples of this include :

Our eyes have greater sensitivity to changes in intensity than to changes in hue. Also , our eyes have greater sensitivity to changes in low-frequency changes than to high-frequency ones. By transforming our image into spaces that correspond to these properties, we can

better reduce the number of bits we devote to them. Such as transforming to frequency-based spaces (frequency domains of the Fourier Transforms).

1.3 The Need to Compress

Uncompressed image data requires a great deal of memory for storage or bandwidth for transmission. Minimizing the amount of data necessary to describe an image is an important consideration for any application that works with images or sequence of images. Thus the challenge for image concentrates to two constraints: high image quality and low data storage / transmission rates.

Therefore, in very case, where we needed to reduce the size of the image but we do not want the image to lose it's details, we must find out a solution which provide us with both of them.

2 Various Standards And Their Pragmatism

2.1 Picture Formats

2.1.1 JPEG

(Joint Photographic Experts Group) can have millions of colors and are often used for photographs and very complex images. JPEG files are also used for images that have minor color changes, depth, lighting effects, or other gradations of color or tone. JPEG images have the .jpg, .jpe, or .jpeg extension. JPEG files are compressed so data is actually removed from the graphic image to make the file size smaller.

2.1.2 GIF

(Graphic Interchange Format) images are limited to 256 colors, they are cross-platform, which means any computer can view them. GIF files are compressed which makes them small in file size but not in dimension. GIF files unlike JPEG files do not lose quality in compression. GIF files have the .gif extension. When to choose the GIF format GIF files are best used with large areas of solid colors, such as logos, and simple illustrations with flat colors.

2.1.3 BMP

(Bitmap) is the standard Windows image format on DOS and Windows compatible computers. The BMP format supports RGB (red, green, blue) indexed-colors, grayscale, and Bitmap color modes. BMP files have the .bmp extension.

2.1.4 TIFF

(Tagged-Image File Format) is used to exchange files between applications and computer platforms. TIFF format is supported by virtually all paint programs, image editing, and page layout applications. Most of the older desktop scanners produce TIFF images and

you should save images scanned as TIFF files unless you scan directly to PhotoShop. The TIFF format supports CMYK, RGB, and grayscale files. TIFF files have the .tif extension.

2.1.5 PNG

(Portable Network Graphics) Pronounced "ping" was developed as an alternative to GIF. PNG files support 24-bit images and produces background transparency without jagged edges. Some older versions of Web browsers may not support PNG images. Like GIF and JPEG files, PNG files are cross-platform and compressed. PNG files can have more colors than GIF files and also compress smaller. PNG files have the .png extension.

And there are some others which we would avoid at this stage.

CHAPTER 3 INTRODUCTION TO BITMAP IMAGE

3.1 DEFINITION OF BITMAP IMAGE

Bitmap image is made of pixels or dots that are arranged and colored differently to form an image or pattern such as photograph or logo. When you zoom in, you can see the individual squares that make up the total image.

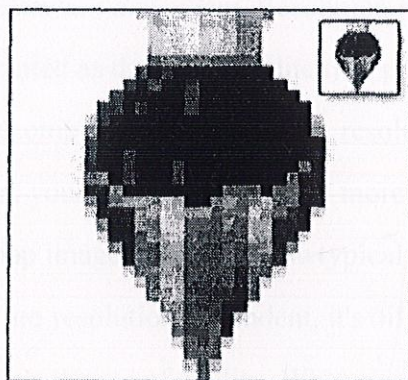


Fig 3.1(a) A zoomed Bitmap Image

A zoomed image is shown above with the original size in upper right corner. Notice the individual pixels that make up the image.

3.2 FACTS ABOUT BITMAP IMAGES

Bitmap images (also known as raster images) are made up of pixels in a grid. Pixels are picture elements; tiny dots of individual color that make up what you see on your screen. All these tiny dots of color come together to form the images you see. The typical computer monitor has 72 or 96 pixels per inch, depending on your monitor and screen.

Bitmap Images are *resolution dependent*. Resolution refers to the number of pixels in an image and is usually stated as dpi (dots per inch) or ppi (pixels per inch). Bitmap images are displayed on your computer screen at screen resolution: 72 or 96 ppi. However, when printing bitmaps, your printer needs much more image data than a monitor. In order to *render* a bitmap image accurately, the typical desktop printer needs 150-300 ppi. Because bitmaps are resolution dependent, it's difficult to increase or decrease their size without sacrificing a degree of quality. When you reduce the size of a bitmap image through your software's resample or resize command, you must throwaway pixels. When you increase the size of the bitmap Image through your software's resample or resize command, the software has to create new pixels. When creating pixels, the software must estimate the color values of the new pixels based on the surrounding pixels. This process is called *interpolation*.

3.3 FORMAT OF BITMAP IMAGE

A BMP computer image is the easiest to understand because it does not use compression, making pixel data retrieval much easier. The table below shows how the pixel is stored from the first byte to the last.

TABLE 2.1: BMP File Structure	
Byte # to fseek pointer	Information
0	Signature
2	File size
18	Width (number of columns)
22	Height (number of rows)
28	Bits / pixel
46	Number of colors used
54	Start of color table
$54 + 4 * (\text{number of colors})$	Start of raster data

The first 14 bytes are dedicated to the header information of the BMP. The next 40 bytes are dedicated towards the info header where one can retrieve such characteristics as

width, file size and number of colors used. Next, is the color table which is $4 * (\text{number of colors used})$ bytes long .So for an 8 bit gray scale image (number of colors is 256),the

color table would be 4×256 bytes long or 1024 bytes . And the last bit of data in a BMP file is the pixel data or raster data.

The raster data starts at byte 54 (header +info header)+ $4 * \text{number of colors}$ (color table). For an 8 bit gray scale image the raster data would start at byte $54 + 1024 = 1078$. The size of the raster data is $(\text{width} \times \text{height}) - 1$ bytes. Therefore a 100 row by 100 column 8-bit gray scale image would have $(100 \times 100) - 1 = 9,999$ bytes of raster data starting at byte 1078 and continuing to the end of the bmp .

Scaling an image does *not* affect the image permanently. In other words, it does not change the number of pixels in the image. However, if you scale a bitmap image to a larger size in your page layout software, you are going to see a definite jagged appearance. Even if you don't see it on your screen it will be apparent in your printed image. Scaling a bitmap image to a smaller size doesn't have any effect: in fact, when you do this you are effectively increasing the ppi of the image so that it will print clearer.

CHAPTER 4 COMPRESSION TECHNIQUES

Two categories of data compression algorithm can be distinguished: **Lossless and Lossy.**

4.1 LOSSLESS COMPRESSION:

Lossless coding guarantees that the decompressed image is absolutely identical to the image before the compression. This is an important requirement for some application domains, e.g. medical imaging, where not only high quality is in demand, but unaltered archiving is a legal requirement, text documents and program executables.

Lossless coding techniques :

- Run length Encoding
- Huffman Encoding
- Entropy Coding (Lempel/Ziv)
- Area Coding

4.2 LOSSY COMPRESSION:

Lossy techniques cause image quality degradation in each compression/decompression step. In general, lossy techniques provide far greater compression ratios than lossless techniques.

Lossy coding techniques

- Transform coding (DCT/Wavelets/Gabor)
- Vector quantisation
- Segmentation and approximation methods
- Spline approximation methods (Bilinear Interpolation / Regularization)

CHAPTER 5 METHODOLOGY EMPLOYED

5.1 COMPRESSION METHODOLOGY

5.1.1 EXTRACTING PIXEL VALUES

The very first step was to load a bitmap image. Then we went for finding out the values of each pixel of which the image comprised of. The values were read from the BMP header. The values were stored in a 2-D array to store the values of each pixel .The image could be sought as a map on X-Y plane. To see the image we have to read each pixel on X-Y plane.

5.1.2 IMAGE SEGMENTATION

There exists several different methods for image segmentation.. In computer vision, **segmentation** refers to the process of partitioning a digital image into multiple regions (sets of pixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. Here we have applied the same method . We have asked the user to input those set of value or the partitions which the user wants.

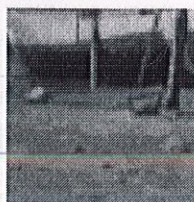
Example: Suppose there is an image whose original size is 100 X 100 pixels. The image is shown below.



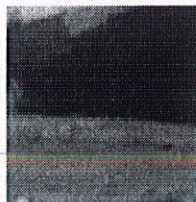
Fig:5.1(a) Original Image

Now we ask the user to input the set of values in which he wants the various sub-parts of the image. Suppose the user inputs 100 and 100 . Which means that the user wants to break the image into $((300 \times 300)/(100 \times 100))$ subparts. Which would come out to be 9 subparts.

The user wants to segment the image into 100 X 100 . He wants to image to be segmented into 9 subparts with each containing 100 rows and 100 columns of pixels as shown below.



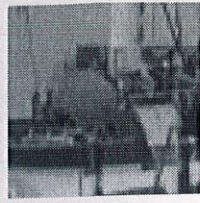
a.) Sub-part 1



b.) Sub-part 2



c.) Sub-part 3



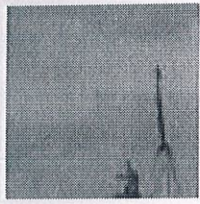
d.) Sub-part 4



e.) Sub-part 5



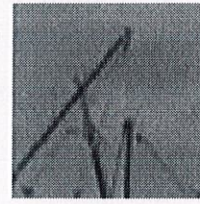
f.) Sub-part 6



g.) Sub-part 7



h.) Sub-part 8



i.) Sub-part 9

Fig:5.1(b) Image after Sub-parts

So this helps in transmitting the images in parts which could be more handy or useful instead of sending the whole image where image size may be a constraint.

5.1.3 APPLYING TWO DIMENSIONAL FOURIER TRANSFORMS TO THE PIXEL VALUES

Brief Description

The Fourier Transform is an important image processing tool which is used to decompose an image into its sine and cosine components. The output of the transformation represents

the image in the Fourier or frequency domain, while the input image is the spatial domain equivalent. In the Fourier domain image, each point represents a particular frequency contained in the spatial domain image.

The Fourier Transform is used in a wide range of applications, such as image analysis, image filtering, image reconstruction and image compression. Before we move onto further details we would like to bring the concept of two dimensional **Fourier Transform** and how it works.

Description

Returns the complex two-dimensional centered Fourier transform of z , where z is an integer, real, double-precision or complex matrix with an even number of rows and columns.

The Formula and working of **Fourier Transform** is shown below:

If M is the number of rows in z and N is the number of columns in z , the (m,n) -th element of the return value is equal to

$$\sum_{i=1}^M \sum_{j=1}^N z_{ij} \exp\{-2\pi i \sqrt{-1} [\frac{(i - M/2 - 1)(m - M/2 - 1)}{M} + \frac{(j - N/2 - 1)(n - N/2 - 1)}{N}]\}$$

If the only prime factors of M and N are 2, 3, 5, and 7, the transform is done in order $(M)(N)[\log(N) + \log(M)]$ operations; otherwise the transform is done in order $(M)(N)(N + M)$ operations.

An example to show the above operation

Example

In the following example M is 4, N is 2, and the $(4,1)$ -th and $(4,2)$ -th elements of z are one (the rest of the elements of z are zero). The (m,n) -th element of the transform is therefore equal to

$$\exp\{-2 \pi i \sqrt{-1} [(m - 3) / 4 - (n - 2) / 2]\} + \exp[-2 \pi i \sqrt{-1} (m - 3) / 4]$$

If you enter

```
z = [{0, 0, 0, 1}, {0, 0, 0, 1}]
fft2d(z)
```

O-Matrix replies

```
{
[ (0,0) , (-2,0) ]
[ (0,0) , (0,2) ]
[ (0,0) , (2,0) ]
[ (0,0) , (0,-2) ]
}
```

Due to numerical limitations, some of the zeros may be output as numbers that are nearly 0.

The Fourier Transform produces a complex number valued output image which can be displayed with two images, either with the *real* and *imaginary* part or with *magnitude* and *phase*. In image processing, often only the magnitude of the Fourier Transform is displayed.

5.1.4 FINDING THRESHOLD

In this case we have taken the average of all the values which we have obtained from the application of fourier transform on the pixel values. This threshold decides the values which should we get rid of to reduce redundancy and proceed for the further process .

5.1.5 REPLACEMENT OF VALUES LOWER THAN THRESHOLD

This is an important process because it checks out the redundancy. We replace all the values which lie below the threshold by 0. Thus unnecessary information is removed and only the useful information is left for the further process of compression.

5.1.6 LEMPER ZIV CODING

Brief Description

The Lempel-Ziv algorithm is a variable-to-fixed length code. Basically, there are two versions of the algorithm presented in the literature: the theoretical version and the practical version. Theoretically, both versions perform essentially the same. However, the

proof of the asymptotic optimality of the theoretical version is easier. In practice, the practical version is easier to implement and is slightly more efficient.

The basic idea is to parse the input sequence into non-overlapping blocks of different lengths while constructing a dictionary of blocks seen thus far.

The most common modification of LZ78 is LZW, first described by Terry Welch in 1984.

5.2 RETRIEVAL METHODOLOGY

This is process in which we would find out the efficiency of the designed code for the comparison to be made between the original image and the retrieved image so as to differentiate between the information lost during the transmission procedure.

As it is lossless procedure for compression but due to numerical limitations there might be a degradation of the quality of an image.

5.2.1 DECOMPRESSION

Once the end of the file is reached, all the blocks generated after reading the compressed file are transformed into a 2-D array. And finally, the pixel definitions can be read from the 2-D array generated as a result of decompression and can be written into a linear array. This linear array can then be decoded as a BMP header. And hence, the image is regenerated with acceptable quality

5.2.2 REPLACEMENT OF VALUES BY THRESHOLD

Those values which were replaced by zeroes in step 5.1.5 are again replaced by the threshold values found earlier. This is done for the next process to be followed.

5.2.3 APPLYING INVERSE FOURIER TRANSFORM

Brief description

This task computes the inverse Fourier transform of a 1- or 2-dimensional image. The input may consist of either the real and imaginary parts of an image, or either part independently.

The *forward* task does not normalize its output, but the *inverse* task normalizes by dividing by the number of pixels. Applying *forward* and then *inverse* therefore returns an image which is the same as the original, except for roundoff errors.

Description

Returns the complex two-dimensional centered inverse Fourier transform of z , where z is an integer, real, double-precision or complex matrix with an even number of rows and columns.

The formula and working of **Inverse Fourier Transform** is given below

If M is the number of rows in z and N is the number of columns in z , the (i,j) -th element of the return value is equal to

$$\sum_{m=1}^M \sum_{n=1}^N z_{m,n} \exp\left\{ 2 \pi i \sqrt{-1} \left[\begin{array}{l} (i - M/2 - 1) (m - M/2 - 1) / M \\ + \\ (j - N/2 - 1) (n - N/2 - 1) / N \end{array} \right] \right\}$$

The return value has the same type and dimension as z . If the only prime factors of M and N are 2, 3, 5, and 7, the transform is done in order $(M) (N) [\log(N) + \log(M)]$ operations; otherwise the transform is done in order $(M) (N) (N + M)$ operations.

Example

In the following example M is 4, N is 2, and the $(4,1)$ -th and $(4,2)$ -th elements of z are one (the rest of the elements of z are zero). The (i,j) -th element of the transform is therefore equal to

$$\exp\{2 \pi i \sqrt{-1} [(i - 3) / 4 - (j - 2) / 2]\} + \exp[2 \pi i \sqrt{-1} (i - 3) / 4]$$

If you enter

```
z = [{0, 0, 0, 1}, {0, 0, 0, 1}]
ifft2d(z)
```

O-Matrix replies

```
{
[ (0,0) , (-2,0) ]
[ (0,0) , (0,-2) ]
[ (0,0) , (2,0) ]
[ (0,0) , (0,2) ]
}
```

Due to numerical limitations, some of the zeros may be output as numbers that are nearly 0.

5.2.4 RECONSTRUCTION OF IMAGE

The values obtained from above steps are plotted to reconstruct the image so as to compare with the original image.

CHAPTER 6 SYSTEM DESIGN AND IMPLEMENTATION

6.1 ALGORITHMS

6.1.1 COMPRESSION

START

EXTRACTING PIXEL VALUES

STEP 1 : Read the image .

STEP 2 : Store it in the form of 2-D array.

PROCESS OF SEGMENTATION

STEP 1 : Ask the user to input those set of values to divide the image into sub-parts.

STEP 2 : From the above values make sub-part matrices by dividing the total no of rows and columns which the image comprises of with the no. of rows and columns which the user entered.

APPLYING FOURIER TRANSFORM

STEP 1 : Apply Fourier Transform to the values obtained in above step.

STEP 2 : Store these values in other matrix.

FINDING THRESHOLD AND REPLACING VALUES

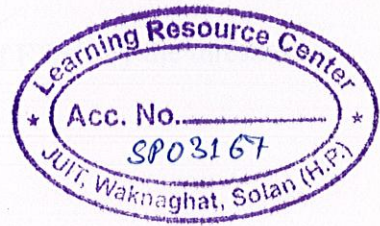
STEP 1 : Calculate the average values obtained in Step 2 of AFT.

STEP 2 : Replace the values by 0 which are lower than threshold found in Step 1.

LZW

STEP 1 : Store these values in a file .

STEP 2 : Apply winzip to the above file for compression.



6.1.2 RETRIEVAL

EXTRACTION

STEP 1 : Extract the above file .

STEP 2 : Read it in a 2-D array.

REPLACING THE VALUES

STEP 1 : The values which were replaced by 0's in Step 2 of FTAR by the threshold.

APPLYING IFT AND DISPLAYING THE IMAGE

STEP 1 : Apply IFT on the values obtained in Step 1 of RTV.

STEP 2 : Plot the image and compare with the original one.

END

6.2 FLOWCHARTS FOR COMPRESSION AND RETRIEVAL

6.2.1 For Compression

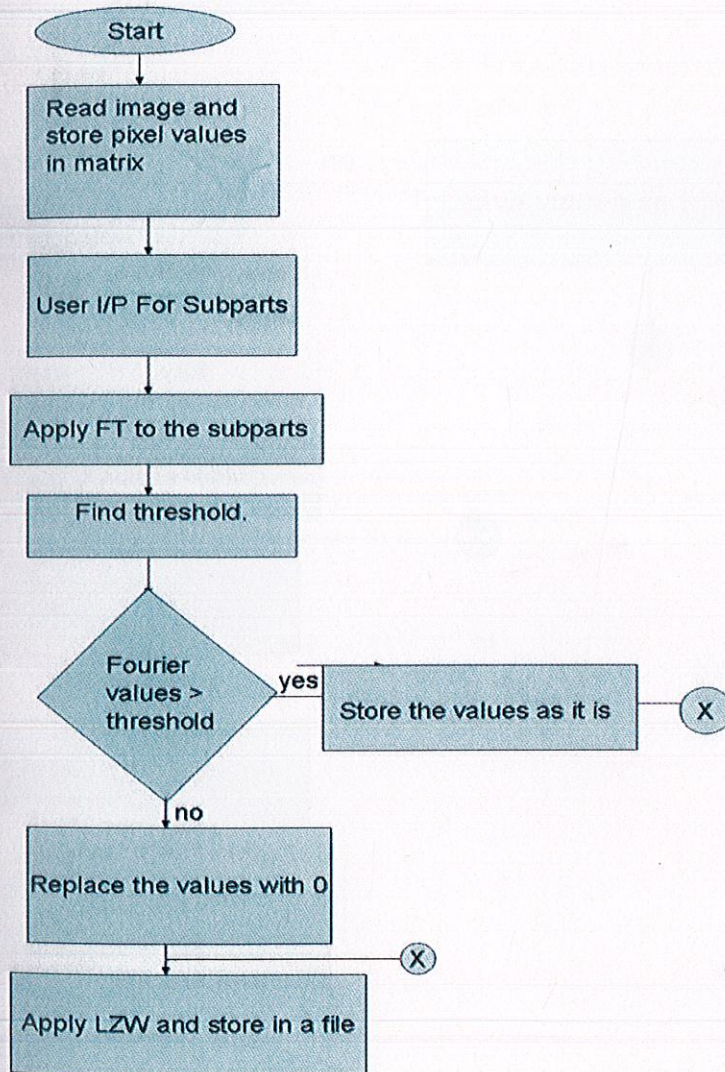


Fig:6.2(a)

6.2.2 For Retrieval

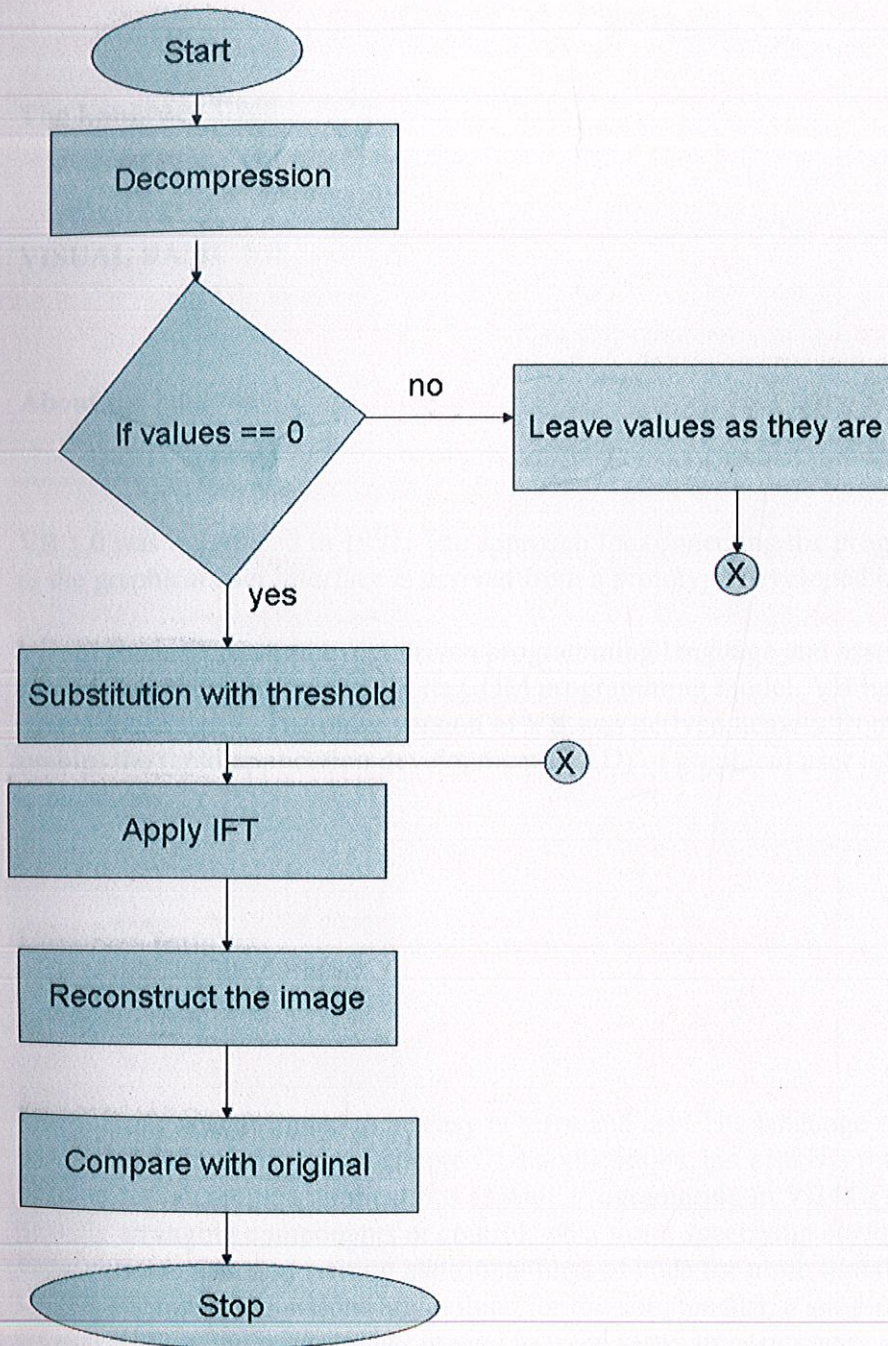


Fig:6.2(b)

6.3 THE DESIGN

The Language Used

VISUAL BASIC 6.0

About the language

VB 1.0 was introduced in 1991. The approach for connecting the programming language to the graphical user interface is derived from a prototype developed by Alan Cooper.

Visual Basic (VB) is an event driven programming language and associated development environment from Microsoft for its COM programming model. VB has been replaced by Visual Basic .NET. The older version of VB was derived heavily from BASIC and enables the rapid application development (RAD) of graphical user interface (GUI) applications.

Language features

Visual Basic was designed to be easy to learn and use. The language not only allows programmers to easily create simple GUI applications, but also has the flexibility to develop fairly complex applications as well. Programming in VB is a combination of visually arranging components or controls on a form, specifying attributes and actions of those components, and writing additional lines of code for more functionality. Since default attributes and actions are defined for the components, a simple program can be created without the programmer having to write many lines of code.

$1.05e+05 + (-0)i$ $-1.05e+05 + (0)i$ $1.05e+05 + (-0)i$ $-1.05e+05 + (-0)i$ $1.05e+05 + (0)i$

$-1.05e+05 + (0)i$ $1.05e+05 + (-0)i$ $-1.05e+05 + (0)i$ $1.05e+05 + (0)i$ $-1.05e+05 + (-0)i$

$1.05e+05 + (-0)i$ $-1.05e+05 + (0)i$ $1.05e+05 + (-0)i$ $-1.05e+05 + (0)i$ $1.05e+05 + (0)i$

6.4 GUI'S

Main Screen

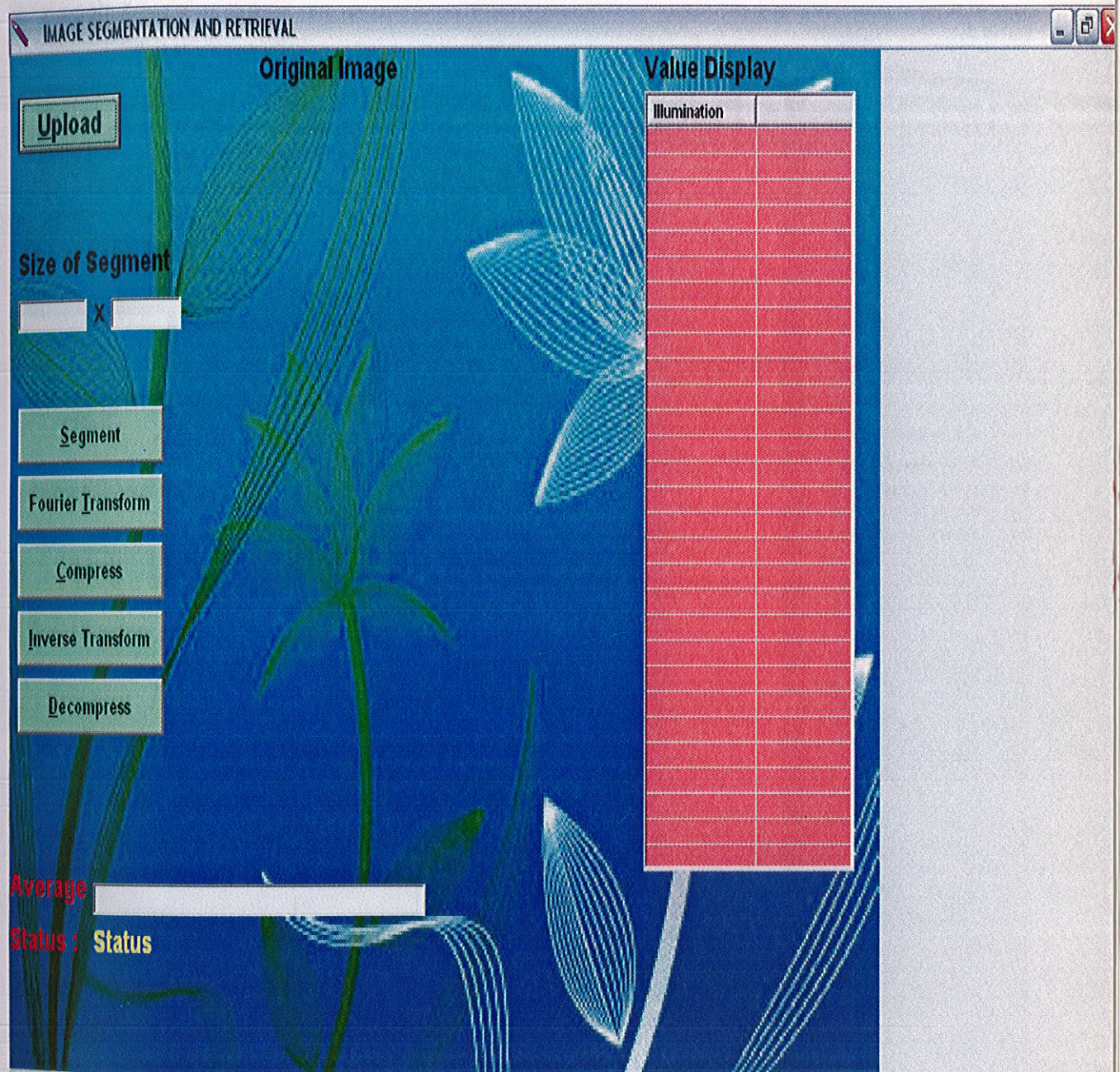


Fig:6.4(a)

After Loading The Image

The original image is uploaded and then illuminance is displayed.

Original Image

Value Display

Illumination	
15850269	
15850269	
15784476	
15784476	
15718683	
15718683	
15718683	
15784476	
15784476	
15784476	
15718683	
15718683	
15652890	
15652890	
15652890	
15652890	
15652890	
15652890	
15652890	
15652890	
15652890	
15652890	
15652890	
15652890	
15587097	
15587097	
15587097	
15587097	
15587097	
15587097	

Upload

100 x 100
Size of Segment

X

Segment

Fourier Transform

Compress

Inverse Transform

Decompress

Average

Status :

Fig:6.4(b)

After entering segmentation values and displaying illuminance of subparts.

Value Display

Part	1	2	3	4	5
Part 1	15850269	15850269	15784476	15784476	15784476
Part 1	15389210	15389210	15323417	15323417	15323417
Part 1	15850269	15784476	15784476	15718683	15718683
Part 1	15389210	15323417	15323417	15323417	15323417
Part 1	15784476	15784476	15718683	15718683	15718683
Part 1	15323417	15323417	15323417	15323417	15323417
Part 1	15784476	15718683	15718683	15718683	15718683
Part 1	15323417	15323417	15323417	15323417	15323417
Part 1	15718683	15718683	15718683	15784476	15784476
Part 1	15323417	15323417	15323417	15323417	15323417
Part 1	15718683	15718683	15784476	15784476	15784476
Part 1	15323417	15323417	15323417	15323417	15323417
Part 1	15784476	15784476	15784476	15718683	15718683
Part 1	15323417	15323417	15323417	15323417	15323417
Part 1	15784476	15784476	15718683	15718683	15652890
Part 1	15323417	15323417	15323417	15323417	15652890
Part 1	15718683	15718683	15652890	15652890	15652890
Part 1	15323417	15323417	15323417	15126554	15126554
Part 1	15718683	15652890	15652890	15652890	15652890
Part 1	15323417	15323417	15126554	15126554	15126554
Part 1	15652890	15652890	15652890	15652890	15652890
Part 1	15323417	15126554	15126554	15126554	15126554
Part 1	15652890	15652890	15652890	15652890	15652890
Part 1	15126554	15126554	15126554	15126554	15126554

Fig:6.4(c)

After Fourier Transform And Compression The original image is shown below the compressed image is of the size of user input of subparts.

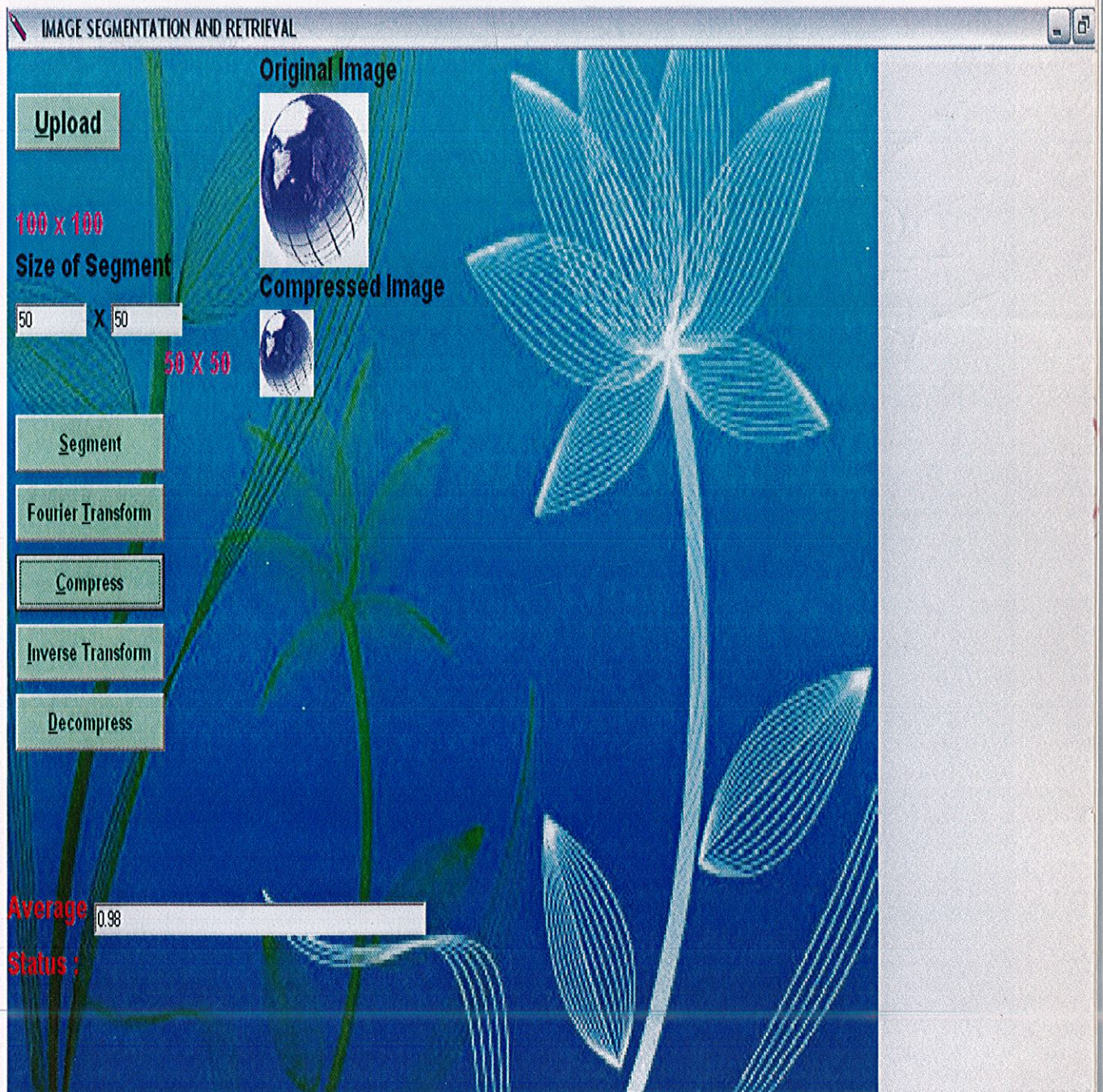


Fig:6.4(d)

The Retrieved Image After Inverse Fourier Transforms And Decompression

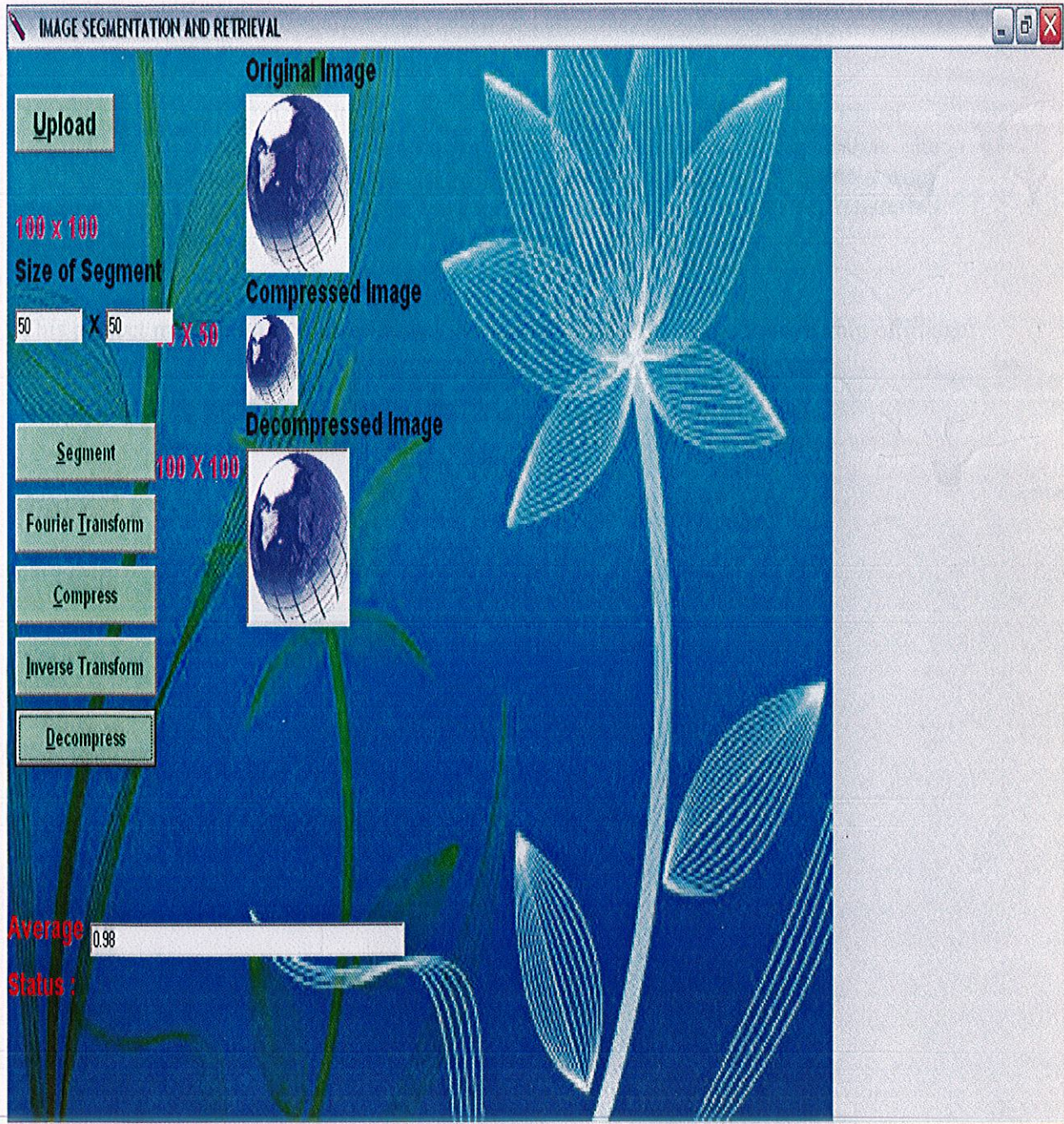


Fig6.4(e)

FUTURE SCOPE OF THE PROJECT

As far as we are concerned with the scope of the project and its effect on the life of normal human being we can say that one of the easiest means go for compression .The user will define the size of the image to be sent where devices may be prohibited from sending a larger image files just user would give the size of the image to be transferred.

This project may be implemented in cellular services for sending or transferring of image files.

CONCLUSION

Image segmentation is a relatively new and upcoming method for image compression and retrieval .

We have basically used the concept of matrices to offer something unique and innovative. The process which we have employed helps in increasing the effectiveness and precision of the image data.

Loss of information in the form of retrieval is very minimal. This method can be very helpful in handsets offering very high quality cameras with minimal space requirement.