



Jaypee University of Information Technology
Solam (H.P.)
LEARNING RESOURCE CENTER

Acc. Num. SP04136 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



SP04136

**JAYPEE UNIVERSITY OF INFORMATION
TECHNOLOGY-WAKNAGHAT
MAY-2008**

**Format Conversion and Image Compression &
Decompression**

**Submitted in partial fulfillment of the Degree of Bachelor of
Technology**

By

Digvijay Parihar – 041049

Akshey Mehta – 041058

Varun Jasrotia - 041059

Dr. S. V. Bhooshan
HOD of Electronics
And Communication

JUIT Waknaghat
Solani
H.P.
India



MAY-2008

**DEPARTMENT OF ELECTRONICS AND
COMMUNICATION
JAYPEE UNIVERSITY OF INFORMATION
TECHNOLOGY-WAKNAGHAT**

ACKNOWLEDGMENT

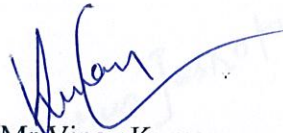
We would like to express our profound thanks to Mr. Vinay Kumar, without whose able guidance and support this work would not have been possible. We are also grateful for his constant support and help throughout the project.

CERTIFICATE

This is to certify that the work entitled, "Format Conversion & Image Compression/Decompression" submitted by Digvijay Parihar, Akshey Mehta and Varun Jasrotia in partial fulfillment for the award of degree of Bachelor of Technology in Electronics and Communication Department 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.



Dr. S.V. Bhooshan
HOD of Electronics
And Communication



Mr. Vinay Kumar
Department of Electronics
And Communication

JUIT Wagnaghat
Solan
H.P.
India

ACKNOWLEDGMENT

We would like to express our greatest gratitude towards Mr. Vinay Kumar, without whose able guidance the completion of this project would be anything but. We thank him for his constant support at the bleakest of times.

We would also like to thank the faculty of Electronics and Communication at Jaypee University of Information Technology, Waknaghat for their assistance.



DIGVIJAY PARIHAR



AKSHEY MEHTA



VARUN JASROTIA

TABLE OF CONTENTS

Abstract	6
Section 1: Image formats	
Chapter 1: Encapsulated Postscript Format.....	7
Chapter 2: Portable Network Graphics.....	10
Chapter 3: Scalable Vector Graphics.....	12
Chapter 4: Bitmap Format.....	14
Chapter 5: Tag Image File Format.....	17
Section 2: Project	
Chapter 1: Image Compression Using Block Truncation Coding.....	20
Chapter 2: Algorithm And Example.....	20
Chapter 3: Comparison Between Lossy and Lossless Compression.....	24
Chapter 4: Algorithm Of Image Formats Conversion.....	25
Chapter 5: Grayscale Conversion.....	27
Chapter 6: Inversion Of An Image.....	32
Conclusion	36
Bibliography	37

ABSTRACT:-

An in depth study of image formats was done by us. We were able to compress and decompress a grayscale image. We constructed a converter which converts a pixel based image into gif, jpeg, png, tiff format and also converts an image to grayscale. Using this converter we also inverted the image, the report of which is presented below.

CHAPTER 1 ENCAPSULATED POSTSCRIPT-

Encapsulated PostScript (EPS) is a standard format for importing and exporting PostScript language files in all environments.

PURPOSE:

The purpose of the EPS file is to be included as an illustration in other PostScript language page descriptions.

EPSI AND EPSF

EPSI is EPS with a device independent bitmap preview. EPSF is an all ASCII (no binary data or headers) version of EPS. EPSF provides for a hexadecimal encoded preview representation of the image that will be displayed or printed.

- The Macintosh has always favoured EPSF files with the data fork of the file containing the PostScript code, and the resource fork containing a PICT preview of the file.

HOW TO CONVERT PS TO EPS

- To convert a ps file to eps, you must edit the file using a text editor or word processor to add lines that will define the file as an eps.
- This all can be done by adding a header file and giving the dimension of bounding box.

DIFFERENCE BETWEEN PS AND EPS-

PostScript is a full-fledged programming language intended for producing printed matter on printers or any other ps driven output device.

SECTION 1:

IMAGE FORMATS:-

CHAPTER 1:

ENCAPSULATED POST SCRIPT:-

Encapsulated PostScript (EPS) is a standard format for importing and exporting PostScript language files in all environments.

PURPOSE:

The purpose of the EPS file is to be included as an illustration in other PostScript language page descriptions.

EPSI AND EPSF:-

- EPSI is EPS with a device independent bitmap preview. EPSI is an all ASCII (no binary data or headers) version of EPS. EPSI provides for a hexadecimal encoded preview representation of the image that will be displayed or printed.
- The Macintosh has always favoured EPSF files with the data fork of the file containing the PostScript code, and the resource fork containing a PICT preview of the file.

TO CONVERT PS TO EPS

- To convert a ps file to eps , you must edit the file using a text editor or word processor to add lines that will define the file as an eps.
- This all can be done by adding a header file and giving the dimension of bounding box.

DIFFERENCE B/W PS AND EPS:-

- PostScript is a full-fledged programming language intended for producing printed pages on printers or any other ps driven output device.

- Encapsulated PostScript is the same as PostScript, except that an EPS file contains a single image, never more, and it's not allowed to include certain commands that may appear in regular postscript.
- WHY EPS: PostScript was never intended by its inventors at Adobe as a way to exchange graphics, but they soon realized that with a few modifications, it could be made to do that, and do it very well indeed. That brings us to EPS.

USES OF EPS:-

- For high-resolution output EPS will provide the least hassles and the best quality. It is the standards for high-resolution printing.
- In desktop publishing

IMAGE FORMAT	DESIGNED FOR:-	TOP CHOICE FOR:-
BMP	Screen display under Windows	Windows Wallpaper
EPS	Printing to PostScript printers	High resolution printing of illustrations
GIF	Screen display, especially the Web	Online publishing of photographic images
JPEG	Screen display, especially the Web	Online publishing of images

CONVERSION OF EPS TO SVG:-

- Open EPS file in gimp and by some editing, it can be stored as png.
- That png file can be converted to svg by using inkscape

CHAPTER 2:-

PNG (Portable Network Graphics):-

- PNG file format is regarded, and was made as, the free and open-source successor to the GIF file format.
- It was designed to be an alternative to the GIF file format, but without the licensing issues that were involved in the GIF compression method at the time

TYPES OF PNG:-

- There are two types of PNG:
- *PNG-8* format, which holds 8 bits of color information (comparable to GIF)
- *PNG-24* format, which holds 24 bits of color (comparable to JPEG).

TECHNICAL DETAILS :-

File header:-

- A PNG file starts with an 8-byte signature. The hexadecimal byte values are
- 89 50 4E 47 0D 0A 1A 0A. Each of the header bytes is there for a specific reason
- Byte(s) 89
- Purpose Has the high bit set to detect transmission systems that do not support 8 bit data and to reduce the chance that a text file is mistakenly interpreted as a PNG, or vice versa
- Byte(s) 50 4E 47
- Purpose In ASCII, the letters "PNG", allowing a person to identify the format easily if it is viewed in a text editor.
- Byte(s) 0D 0AA
- Purpose DOS style line ending (CRLF) to detect DOS-UNIX line ending conversion of the data. 1AA byte that stops display of the file under DOS when the command type has been used – the end-of-file character
- Byte(s) 0AA
- Purpose UNIX style line ending (LF) to detect UNIX-DOS line ending conversion.

CHUNKS IN FILE:-

- After the header come a series of chunks, each of which conveys certain information about the image.
- Chunks declare themselves as *critical* or *ancillary*, and a program encountering an ancillary chunk that it does not understand can safely ignore it.

CHUNK STRUCTURE:-

- The chunks each have a header specifying their size and type. This is immediately followed by the actual data, and then the checksum of the data.
- Chunks are given a 4 letter case sensitive ascii name. The case of the different letters in the name (bit 5 of the numeric value of the character) provides the decoder with some information on the nature of chunks it does not recognize.

GIF	JPEG	PNG-8	PNG-24
Better for clipart and drawn graphics with few colours, or large blocks of colour	Better for photographs with lots of colours or fine colors detail	Better for clipart and drawn graphics with few colours, or large blocks of colour	Better for photographs with lots of colours or fine colour detail
Can only have up to 256 colours	Can have up to 16 million colours	Can only have up to 256 colours	Can have up to 16 million colours
Images are "lossless" - they contain the same amount of information as the original (but with only 256 colours)	Images are "lossy" - they contain less information than the original	Images are "lossless" - they contain the same amount of information as the original (but with only 256 colours)	Images are "lossless" - they contain the same amount of information as the original
Can be animated	Cannot be animated	Cannot be animated	Cannot be animated

COLOR DEPTH:-

- Color depth is a computer graphics term describing the number of bits used to represent the color of a single pixel in a bitmapped image. This concept is also known as bits per pixel (bpp), particularly when specified along with the number of bits used. Higher color depth gives a broader range of distinct colors.

CHAPTER 3: SCALABLE VECTOR GRAPHICS (SVG)

SVG is an XML specification and file format for describing two-dimensional vector graphics.

It is an open standard created by the World Wide Web Consortium's SVG Working Group.

SVG allows three types of graphic objects:

1. Vector graphics
2. Bitmap image
3. Text

CREATING SVG FILES:-

To create SVG files, you need a *vector* graphics package that can save SVG files. Fortunately, there are a wide range of packages for various operating systems. Inkscape, which is an open source package compatible with Windows, and various versions of Linux

INKSCAPE SVG /PLAIN SVG:-

Inkscape can save SVG files in two formats, Inkscape SVG, and Plain SVG. Inkscape SVG is a standards compliant SVG file that includes extra information, such as metadata about the file, and other information unique to Inkscape to aid future editing. However, some renderers can get confused by this extra information

WHEN TO USE SVG :-

1. A good choice for an SVG image. An image made of lines, shapes, and fills, that you may want to print small, or blow up to the size of a flag!
2. SVG is the preferred file format for line drawings, diagrams, graphs, maps, flow charts, and anything assembled out of lines and curves, as opposed to individual pixels as in a photograph. SVG should be used any time one can expect to edit an image after upload. Examples would be maps that may need to have borders adjusted, or text changed to be multilingual. SVG graphs showing data that might change with time can be edited to stay current.
3. Images such as photographs and animations should not be attempted in SVG format. Formats such as JPEG, PNG and GIF may be more appropriate.

SCRIPTING :-

- It is based on XML, it is entirely text-based which will open up for search engines to index SVG images, and users will thus be able to search for text within images (e.g. search for a button text, or a streetname on a map)

APPLICATION:-

- An example of how shapes, text and images could be combined into a nice SVG application could be an interactive map. For example bookstores could show the user an initial map, and then the user would be able to dive into the map and find the closest bookstore. The map could then have embedded images of how the bookstore looked.

BENEFITS OF SVG:-

- Scalable - the image can be as small as a thumbnail, or as large as a poster, without losing quality, or increasing file size.
- Open standard - anyone can create software to edit and generate SVG files. The standard is maintained by the W3C, a respected industry standards organization.
- Editable - images can be easily edited for updates, changing text for multilingual images, etc SVGs are built out of collections of lines, shapes, curves, and fills, not individual pixels. You can take a circle 10px wide, stretch it to 100px wide, and it stays a circle, instead of getting a 'jagged edge'. This is the power of a vector program

CHAPTER 4: BMP FORMAT

- Also stored as DIB
- DIB->Device independent bitmap
- It may use compression but by itself it is not capable of storing animation. animation can however be done by writing a code for it.

ADVANTAGES OF THE BMP FORMAT:-

- The simplicity of the bmp file format and its widespread familiarity in windows and also it is well documented makes it very commonly used formats.
- Free of patents

BMP DATA STRUCTURE:-

- BITMAP FILE HEADER: bmfh;
- BITMAP INFO HEADER: bmih;
- RGBQUAD: acolors[];
- BYTE: abitmapsbits[];

BITMAP FILE HEADER:-

- It contains some information about the bitmap file (about the file, not about the bitmap itself).
- It consists of bftype(2 bytes),bfsz(4 bytes),bfreserved(2 bytes),bfreserved2 bytes), and bffbits(4 bytes).

BITMAP INFO HEADER:-

- It contains information about the bitmap such as color, size, width etc.
- It consists of size, width, height,no of planes, no of bits per pixel, type of compression, image size, horizontal planes, vertical planes, no of colors used.
- All are size 4 bytes except for no of planes which is of 2 bytes.

start	Size	Name	Std value	purpose
15	4	biSize	40	specifies the size of the BITMAPINFOHEADER structure, in bytes.
19	4	biWidth	100	specifies the width of the image, in pixels.
23	4	biHeight	100	specifies the height of the image, in pixels.
27	2	biPlanes	1	specifies the number of planes of the target device, must be set to zero.
29	2	biBitCount	8	specifies the number of bits per pixel.
31	4	biCompression	0	Specifies the type of compression, usually set to zero (no compression)
35	4	biSizeImage	0	specifies the size of the image data, in bytes. If there is no compression, it is valid to set this member to zero.
39	4	biXPelsPerMeter	0	specifies the the horizontal pixels per meter on the designated target device, usually set to zero.
43	4	biYPelsPerMeter	0	specifies the the vertical pixels per meter on the designated target device, usually set to zero.
47	4	biClrUsed	0	specifies the number of colors used in the bitmap, if set to zero the number of colors is calculated using the biBitCount member.
51	4	biClrImportant	0	specifies the number of color that are 'important' for the bitmap, if set to zero, all colors are important.

BIBIT COUNT:-

- Actually specifies the color resolution
- 1=>black/white
- 4=>16 colors
- 8=>256 colors
- 24=>16.7 million colors

RGBQUAD:-

- This contains a color table.
- It consists of rgbblue,green, red,reserved.
- Rgb blue specifies the blue part of color.
- All are of the same size(1 byte).

FILE ORGANIZATION:-

FILE IMAGE HEADER:-

- This is the fixed part of tiff.
- The location of image file directory and data vary, making the tiff format complicated.
- It is the first 8 bytes of every tiff file.
- The data structure of tiff consists of identifier, version, tiff offset.
- The value of identifier indicate whether the tiff file is written in intel or the motorola format.
- Version number is always 42, regardless of tiff revision, so it may be regarded as a identification no rather as a version no.

FILE IMAGE DIRECTORY

- Data in a tiff file is found by using information found in the tiff file header and its associated tags are known as tiff metadata.
- Each tiff contains one or more data structures known as tags.
- Tags that are defined by the tiff specification are called public tags.
- User defined tags are called private tags.

CHAPTER 5:

TIFF FILE FORMAT :-

- Bit map type image
- Colors:1 to 24 bit
- Platforms: ms-dos, mac,unix,others
- Maximum size: $2^{32}-1$

USAGE:-

- Used for data storage and interchange.
- The tiff format is the most versatile and diverse format in existence
- Allows storage of multiple bitmap images through use of tags ,of any pixel depth, making it ideal for most image storage need
- Unlike standard jpeg, tiff files using lossless compression can be edited and reserved without suffering a compression loss.

FILE ORGANIZATION:-

FILE IMAGE HEADER:-

- This is the fixed part of tiff.
- The location of image file directory and data vary, making the tiff format complicated.
- It is the first 8 bytes of every tiff file.
- The data structure of ifh consists of identifier, version,ifd offset
- The value of identifier indicate whether the tiff file is written in intel or the motorola format.
- Version number is always 42,regardless of tiff revision, so it may be regarded as a identification no rather as a version no.

FILE IMAGE DIRECTORY

- Data in a tiff file is found by using information found in the ifd.
- Each ifd and its associated bitmap are known as tiff subfile.
- Each ifd contains one or more data structures known as tags.
- Tags: Each tag is a 12 byte record that contains specific piece of information about the bitmapped data.
- Tags that are defined by the tiff specification are called public tags.
- User defined tags are called private tags.

3 possible arrangements of data in TIFF:-

Header	Header	Header
Ifd 0	Ifd 0	Image 0
Ifd 1	Image 0	Image 1
Ifd 2	Ifd 1	Image 2
Image 0	Image 1	Ifd 0
Image 1	Ifd 2	Ifd 1
Image 2	Image 2	Ifd 2

IMAGE FILE DIRECTORY:-

- Ifd is a collection of information similar to a header and it is used to describe the bitmapped data to which it is attached.
- It contains information on the width, depth, height, no of color planes and type of compression used.
- Ifd is dynamic and may not only vary but also may be found anywhere within the tiff file.

FORMAT OF THE IMAGE FILE DIRECTORY:-

Tag entry count	2 bytes
Tag 0	12 bytes
Tag 1	12 bytes
Tag n	12 bytes
Next ifd offset	4 bytes

SECTION 2:
PROJECT:

CHAPTER 1:
Image Compression Using Block Truncation Coding

Block Truncation Coding, or **BTC**, is a type of lossy image compression technique for grayscale images.

It divides the original images into small sub-images and then using a quantizer, which adapts itself according to the image statistics, to reduce the number of gray levels in the image. It is an early predecessor of the popular hardware DXTC technique, although BTC compression method was first adapted to color long before DXTC using a very similar approach called Color Cell Compression. BTC has also been adapted to video compression

CHAPTER 2:
ALGORITHM FOR BTC:-

- 1. An image is divided into non-overlapping blocks. The size of a block could be 4 by 4 or 8 by 8 etc. Calculate the average gray level of the block 4 by 4.
- Avg. Gray Level $(M) = (1/16) \sum \sum x(i,j)$.
where $x(i,j)$ represents pixels in the block
- 2. Pixels in the image block is then classified into two kinds according to whether their gray level is greater than the block average gray level. The average gray levels of these two kinds of pixel are calculated
- $M(U)$ is the average grey level of the pixel having grey level greater than block average grey level.
- $M(L)$ is the average grey level of the pixel having grey level less than block average grey level.
- $M(U) = (1/k) \sum x(i,j)$ where $x(i,j) < m$
- $M(L) = (1/16-k) \sum x(i,j)$ where $x(i,j) \geq m$
- Where K is the number of pixels whose gray level is greater than m .

- 3. A binary block, denoted b , is also needed to classify the pixels. We can use "1" to represent a pixel whose gray level is greater than m and "0" to represent a pixel whose gray level is less than or equal to m .
- 4. The encoder writes $M(U), M(L)$ and B to a file. Assume that we use 8 bits to represent $M(U)$ and $M(L)$ respectively. Then the total number of bits required for a block is $8+8+16=32$ bits. Thus, the bit rate for the very basic BTC algorithm is 2 bits/pixel.
- 5. In the decoder, an image block is reconstructed by replacing the "1"s with $M(U)$ and the "0"s by $M(L)$.
- An example is shown in following tables:



The average gray level = 159

181	173	156	141
192	180	138	118
188	183	141	116
185	176	157	123

The original image block

1	1	0	0
1	1	0	0
1	1	0	0
1	1	0	0

The encoded information: the binary block (16 bits), $M(U)=182$ $M(L)=136$. Total = 32 bits/block.

The error (difference between the original and the reconstructed image block): Mean square error = 128.5.

182	182	136	136
122	182	136	136
182	182	136	136
182	182	136	136

The reconstructed image block by the decoder

-1	-9	20	5
10	-2	2	-18
6	1	5	-20
3	-6	21	-13

The error (difference between the original and the reconstructed image block): Mean square error = 128.5.

CHAPTER 3:
COMPARISON BETWEEN LOSSY AND LOSSLESS COMPRESSION:-

LOSSY COMPRESSION:-

With lossy compression, it is assumed that some loss of information is acceptable. The best example is a videoconference where there is an acceptable amount of frame loss in order to deliver the image in real time. People may appear jerky in their movements, but you still have a grasp for what is happening on the other end of the conference. In the case of graphics files, some resolution may be lost in order to create a smaller file. The loss may be in the form of color depth or graphic detail. For example, high-resolution details can be lost if a picture is going to be displayed on a low-resolution device. Loss is also acceptable in voice and audio compression, depending on the desired quality.

LOSSLESS COMPRESSION:-

With lossless compression, data is compressed without any loss of data. It assumes you want to get everything back that you put in. Critical financial data files are examples where lossless compression is required.

The removal of information in the lossy technique is acceptable for images, because the loss of information is usually imperceptible to the human eye. While this trick works on humans, you may not be able to use lossy images in some situations, such as when scanners are used to locate details in images.

- Lossy compression can provide compression ratios of 100:1 to 200:1, depending on the type of information being compressed.
- Lossless compression ratios usually only achieve a 2:1 compression ratio.

CHAPTER 4:
ALGORITHM FOR CONVERSION OF IMAGE FROM ONE FORMAT TO OTHER:-

To convert an image from one format to another following steps are required-

Step 1:-

First of all image is to be displayed in dialog box , for that a command "picthumbnail" is used.

e.g

```
picThumbnail.Image = Image.FromFile(openfiledialog.FileName);
```

Step 2:-

Only those images can be displayed whose format is given in load images title. This can be done as:

```
openfiledialog.Filter = "TIFF (*.tiff)|*.tiff" + "|PNG  
(*.png)|*.png" + "|GiF (*.gif)|*.gif" + "|BMP(*.bmp)|*.bmp" +  
"|JPEG(*.jpg;*.jpeg)|*.jpg;*.jpeg";
```

Step 3: -

In the "Form" insert a button to open a picture and other for conversion.

Step 4: -

Now put a "Combo box" in the Form in which different extensions of image are given. Select the desired format in which you want to convert the image using combo box.

Step 5: -

After selecting the format, using button "Convert" a dialog box is displayed in which we give the name of image and destination where we want to save the image.

Step 6: -

To save the image in other format three parameters are required

- (a) Original image.
- (b) Name of new image.
- (c) New extension.

Step 7:-

Now by using the function

`saveFile(img, newName, newExtension);` image can be saved to other format.

Step 8: -

First parameter "img" that we can get from "picThumbnail.Image", "new extension" from combo box and "new name" that we enter in dialog box.

Step 9: -

When we pass these three parameters in our save function, then by using "encoders and imagecodeinfo" our original image is converted to desired format and get saved at our given location.

CHAPTER 5: GRAYSCALE IMAGES:-

A grayscale (or graylevel) image is simply one in which the only colors are shades of gray. The reason for differentiating such images from any other sort of color image is that less information needs to be provided for each pixel. In fact a 'gray' color is one in which the red, green and blue components all have equal intensity in rgb space and so it is only necessary to specify a single intensity value for each pixel, as opposed to the three intensities needed to specify each pixel in a full color image..

Often, the grayscale intensity is stored as an 8-bit integer giving 256 possible different shades of gray from black to white. If the levels are evenly spaced then the difference between successive graylevels is significantly better than the graylevel resolving power of the human eye.

Grayscale images are very common, in part because much of today's display and image capture hardware can only support 8-bit images. In addition, grayscale images are entirely sufficient for many tasks and so there is no need to use more complicated and harder-to-process color images.

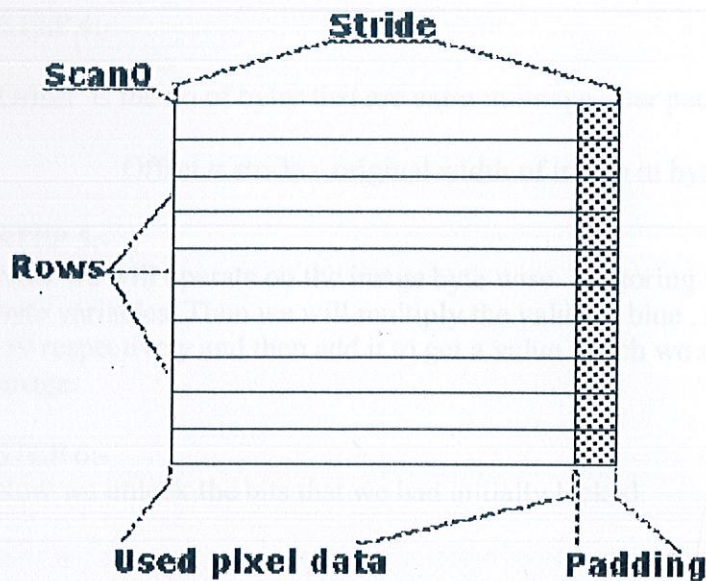
In computing, a grayscale or greyscale digital image is an image in which the value of each pixel is a single sample. Displayed images of this sort are typically composed of shades of gray, varying from black at the weakest intensity to white at the strongest, though in principle the samples could be displayed as shades of any color, or even coded with various colors for different intensities. Grayscale images are distinct from black and white images, which in the context of computer imaging are images with only two colors, black and white grayscale images have many shades of gray in between. In most contexts other than digital imaging, however, the term "black and white" is used in place of "grayscale"; for example, photography in shades of gray is typically called "black-and-white photography". The term monochromatic in some digital imaging contexts is synonymous with grayscale, and in some contexts synonymous with black-and-white. Grayscale images are often the result of measuring the intensity of light at each pixel in a single band of the electromagnetic spectrum.

Grayscale images intended for visual display are typically stored with 8 bits per sampled pixel, which allows 256 intensities (i.e., shades of gray) to be recorded, typically on a non linear scale. The accuracy provided by this format is barely sufficient to avoid visible banding artifacts, but very convenient for programming. Technical uses (e.g. in medical imaging or remote sensing applications) often require more levels, to make full use of the sensor accuracy (typically 10 or 12 bits per sample) and to guard against roundoff errors in computations. Sixteen bits per sample (65536 levels) appears to be a popular choice for such uses. The PNG image format supports 16 bit grayscale natively, although browsers and many imaging programs tend to ignore the low order 8 bits of each pixel.

ALGORITHM FOR CONVERTING AN IMAGE TO GRAYSCALE IMAGE:-

STEP1:-

We have to fix a portion of the bitmap pixel data array in memory. Then we can access it directly and finally replace the bits in the bitmap with the modified data.



By using Lockbits method we can store our image as shown in above diagram.

Now after storing image like this we will use Scan0. It is a pointer that will point to the first element of this rectangular box.

STEP 2:-

We will use the stride function to get the the width, in bytes, of a single row of pixel data. This width is a multiple, or possibly sub-multiple, of the pixel dimensions of the image and may be padded out to include a few more bytes.

The Stride property, as shown in the above figure, holds the width of one row in bytes. The size of a row however may not be an exact multiple of the pixel size because for efficiency, the system ensures that the data is packed into rows that begin on a four byte boundary and are padded out to a multiple of four bytes. This means for example that a 24 bit per pixel image 17 pixels wide would have a stride of 52. The used data in each

row would take up $3 * 17 = 51$ bytes and the padding of 1 byte would expand each row to 52 bytes or $13 * 4$ bytes. A 4BppIndexed image of 17 pixels wide would have a stride of 12. Nine of the bytes, or more properly eight and a half, would contain data and the row would be padded out with a further 3 bytes to a 4 byte boundary.

STEP 3:-

Unsafe mode is used here as c sharp doesn't allow the use of pointers but unsafe mode allows it.

STEP 4:-

Offset is the no of bytes that are extra in image after padding. It is calculated as follows:-

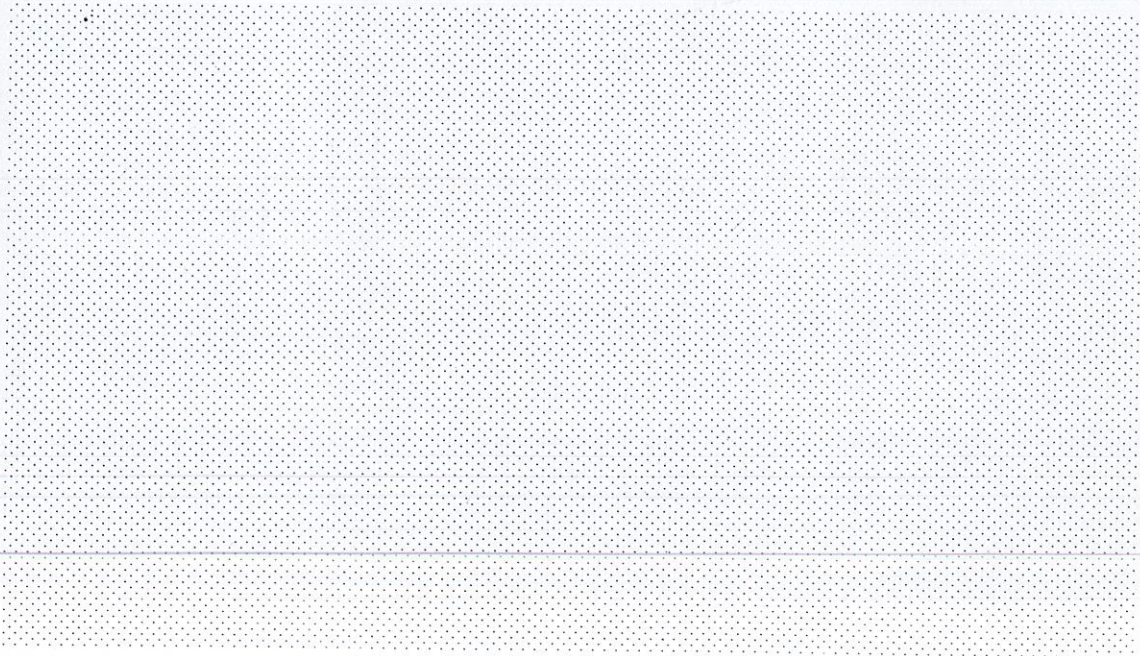
$$\text{Offset} = \text{stride} - \text{original width of image in bytes}$$

STEP 5:-

Now we will operate on the image byte wise by storing values of first three bytes in three byte variables. Then we will multiply the value of blue, red and green by .11, .30 and .59 respectively and then add it to get a value which we store in all of the three bytes of image.

STEP 6:-

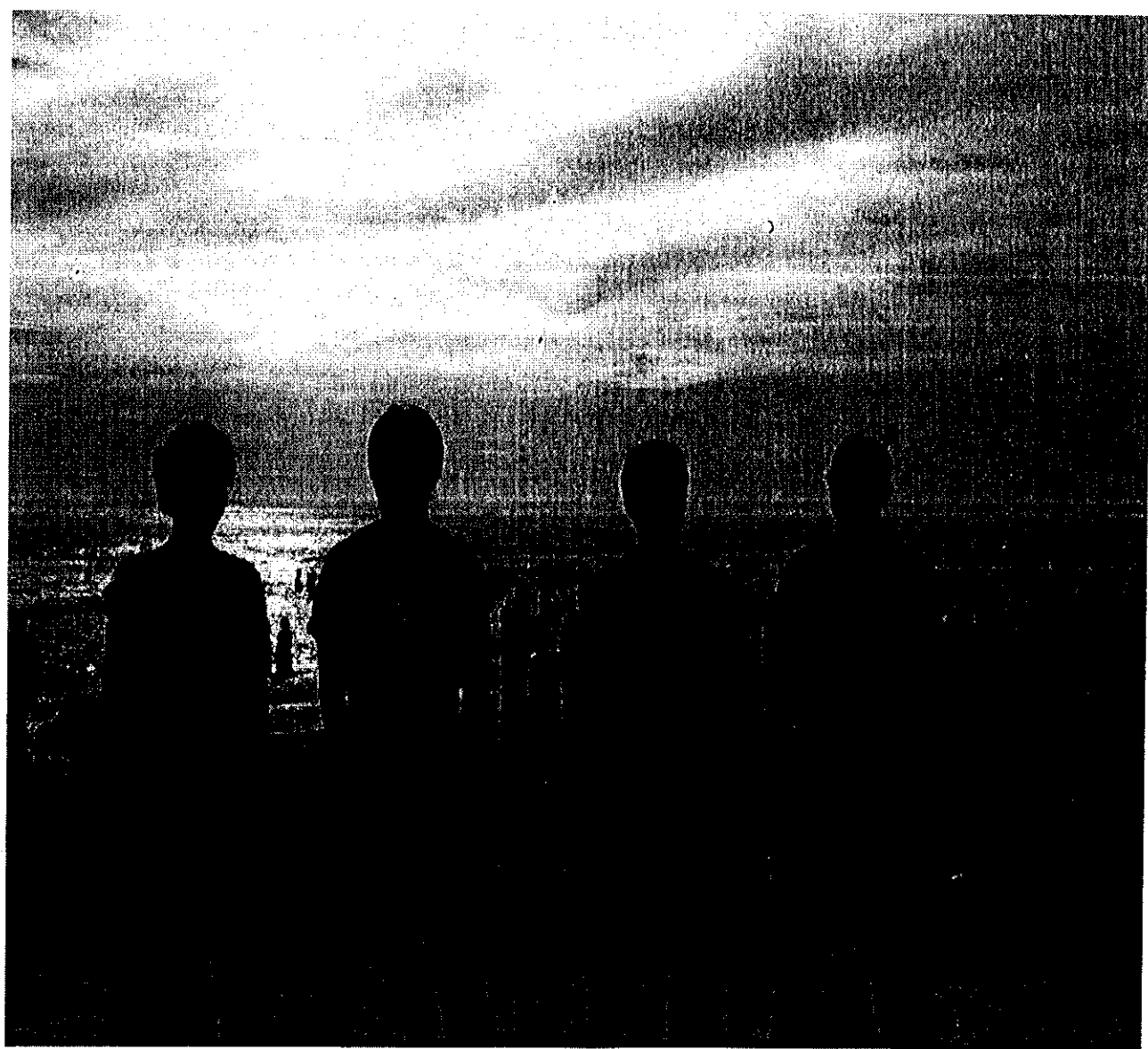
Now we unlock the bits that we had initially locked.



ORIGINAL IMAGE



GRAYSCALE IMAGE



CHAPTER 6:

INVERT:-

Image inversion is by far the easiest Image Manipulation there is. Outside the context of computers, an inverted image is referred to as a "negative". That is, every color is the exact opposite of the original color.

This same effect can be mirrored in the digital world. The human eye has three distinct color receptors, one for red, one for green, and one for blue. This is how color is represented in the .NET environment (**Red Green Blue**). The current range for each channel (it may be increased in the future) is 0 to 255 -- one Byte. The general, mathematical way to invert a number (where 0 is the minimum) is:

If you apply this formula to each color channel (RGB) of each pixel (a Point of Color in an Image), you will have an inverted image!

ALGORITHM FOR INVERTING AN IMAGE :-

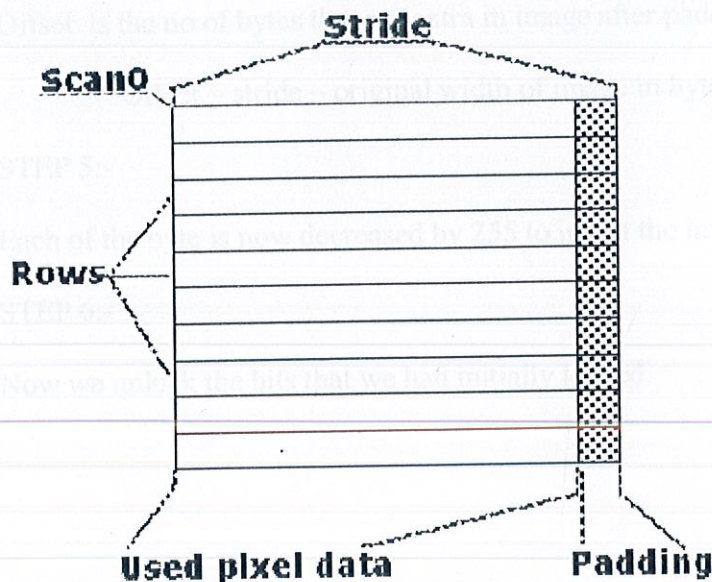
To invert an image the basic principle is :-

Operate each pixel individually is not enough .we have to operate byte wise in the image for e.g.

If we are dealing with 24 bpp image(24 bits per pixel image) that means each color is represented by one byte. So to invert that image we need to decrease values of each color by 255 (that is maximum value of each color in image) and then it will get inverted. To implement this following steps are required:-

STEP1:-

We have to fix a portion of the bitmap pixel data array in memory .Then we can access it directly and finally replace the bits in the bitmap with the modified data.



By using Lockbits method we can store our image as shown in above diagram. Now after storing image like this we will use Scan0. It is a pointer that will point to the first element of this rectangular box.

STEP 2:-

We will use the stride function to get the the width, in bytes, of a single row of pixel data. This width is a multiple, or possibly sub-multiple, of the pixel dimensions of the image and may be padded out to include a few more bytes.

The Stride property, as shown in the above figure , holds the width of one row in bytes. The size of a row however may not be an exact multiple of the pixel size because for efficiency, the system ensures that the data is packed into rows that begin on a four byte boundary and are padded out to a multiple of four bytes. This means for example that a 24 bit per pixel image 17 pixels wide would have a stride of 52. The used data in each row would take up $3 * 17 = 51$ bytes and the padding of 1 byte would expand each row to 52 bytes or $13 * 4$ bytes. A 4BppIndexed image of 17 pixels wide would have a stride of 12. Nine of the bytes, or more properly eight and a half, would contain data and the row would be padded out with a further 3 bytes to a 4 byte boundary.

STEP 3:-

Unsafe mode is used here as c sharp doesn't allow the use of pointers but unsafe mode allows it.

STEP 4:-

Offset is the no of bytes that are extra in image after padding.It is calculated as follows:-

$$\text{Offset} = \text{stride} - \text{original width of image in bytes}$$

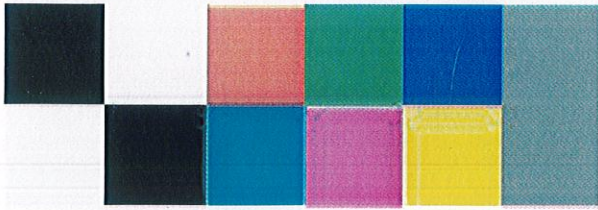
STEP 5:-

Each of the byte is now decreased by 255 to invert the image.

STEP 6:-

Now we unlock the bits that we had initially locked.

Here is a picture with five primary colors on the top row, and their inversions on the bottom row:



To help visualize the transformation, take a look at the before and after pictures

ORIGINAL IMAGE



INVERTED IMAGE



CONCLUSION:-

We compressed an image of size 270 kb to an image of size 53 kb and after decompression we got an image of size 203 kb.

We constructed a converter which has the following features:-

- a) Converts an image in pixel based format to other pixel based formats like png , tiff , jpeg, bmp, gif.
- b) Converts an image to grayscale image.
- c) Inverts an image.

FUTURE WORK:-

There is scope for conversion of an image in pixel based format to an image in vectored based format.

BIBLIOGRAPHY:-

- 1) A Guide To Matlab For Beginners
- 2) Wiley- C Sharp Bible
- 3) Msdn Programming With C Sharp
- 4) www.google.co.in
- 5) www.wikipedia.com