INDIAN INSTITUTE OF TECHNOLOGY
KHARAGPUR

DIGITAL IMAGE PROCESSING LABORATORY

A REPORT ON
Experiment-1

Group No: 10

B.Sai Manoj 17EC35007
Sanku Yogesh 17EC35020

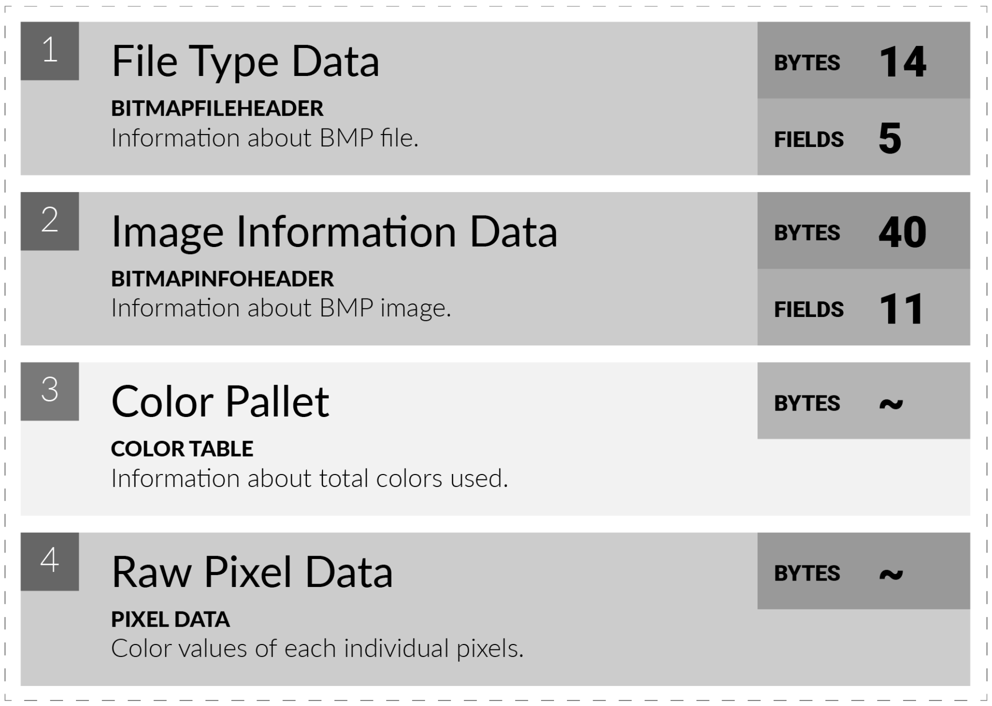
Dept of Electronics and Electrical Communication
Engineering
Visual Information and Embedded Systems

January 25, 2021

Contents

1 Introduction

Bitmap or BMP files are quite old file format used by "Windows" operating system. BMP images can range from 1 bit per pixel (thus a black and white image) to 24 bits per pixel (providing 1.67 million colours). In the experiment we used an 8 bits per pixel (Grayscale image) and 24 bits per pixel (RGB color image) formats for operating upon.



1. **BITMAP FILE HEADER:**This block of bytes is at the start of the file and is used to identify the file. A typical application reads this block first to ensure that the file is actually a BMP file and that it is not damaged. The first 2 bytes of the BMP file format are the character "B" then the character "M" in ASCII encoding. All of the integer values are stored in little-endian format (i.e. least-significant byte first).

2. **DIB HEADER :**This block of bytes tells the application detailed information about the image such as the height and width of the image, Bits per pixel, which will be used to display the image on the screen.

3. **COLOR TABLE :**The color table (palette) occurs in the BMP image file directly after the BMP file header. Therefore, its offset is the size of the BITMAPFILEHEADER plus the size of the DIB header. The color table is a block of bytes (a table) listing the colors used by the image and the color table is normally not used when the pixels are in the 16-bit per pixel (16bpp) format (and higher).

4. **PIXEL STORAGE :** The bits representing the bitmap pixels are packed in rows. The size of each row is rounded up to a multiple of 4 bytes by padding.The data starts from the address stored in the offset field of the BITMAPINFOHEADER

   For images with height above 1, multiple padded rows are stored consecutively, forming a Pixel Array.

   The total number of bytes necessary to store one row of pixels can be calculated as:

$$\mathrm{RowSize} = \left\lceil \frac{\mathrm{BitsPerPixel} \cdot \mathrm{ImageWidth}}{32} \right\rceil \cdot 4 = \left\lfloor \frac{\mathrm{BitsPerPixel} \cdot \mathrm{ImageWidth} + 31}{32} \right\rfloor \cdot 4,$$

Reading a BMP file : The function "readbmp" reads the BITMAP Header and DIB Header. According to details of the image from the DIB Header we have a function which takes the height and weight of the image and store the pixel values in an array.

Geometrical Functions : We have implemented separate functions for each and every geometrical transformation and these functions are called from a different function "geometrical".

Writing a BMP file : Creating a new BMP file, giving the height and width of the image. Now copying the headers in the new BMP file. Filling the colortable as per the definition if the image is a grayscale image. Now writing the pixel value accordingly.

2 Algorithms

2.1 Diagonally Flipped

1. Change the height and width for the transformed image.
2. Adjust the padding for the transformed image.
3. Allocate memory for the pixel array of the new flipped image.
4. Now copy the pixel values to the new array in transposed manner.

2.2 90 degrees Rotation

1. Change the height and width for the transformed image.
2. Adjust the padding for the transformed image.
3. Allocate memory for the pixel array of the new 90 degrees rotated image.
4. Now copy the pixel values of (y,height-x-1) to (x,y) to the allocated space.

2.3 45 degrees Rotation

1. Rotate the four corners of the image 45 degrees and calculate the height and width of the new image.
2. Adjust the padding for the transformed image.
3. Allocate memory for the pixel array of the new 45 degrees rotated image.
4. Copy the pixel values by checking whether it is inside the image or not.
5. Now interpolating the image.

2.4 Scaling

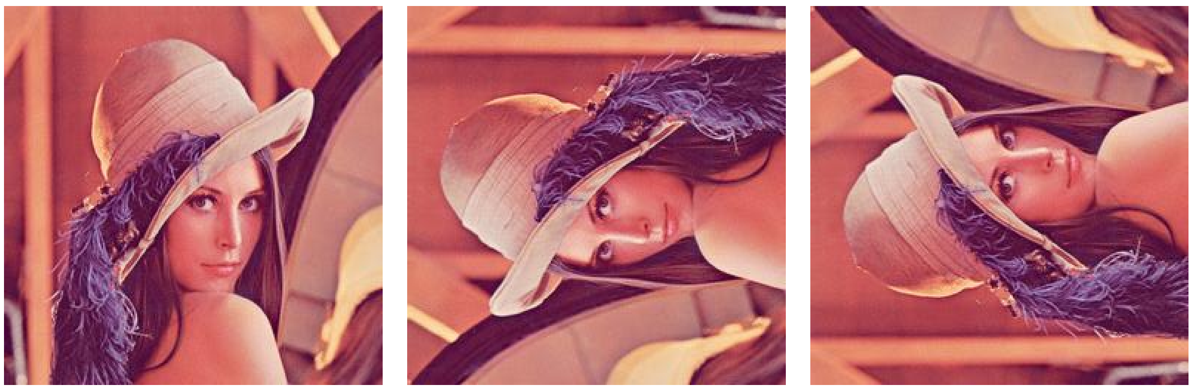
1. Adjust the height and width according to the scaling factor.
2. Adjust the padding for the transformed image.
3. Allocate memory for the pixel array of the new scaled image.
4. Now copy the pixel values to new pixel array in the ratio of scaling factor.
5. Now interpolating the image.

2.5 Gray Scale Converted

1. Copy the height and width of the original image.
2. Allocate memory for the new gray converted image.
3. Now determine the gray converted pixel value using the formula (0.3*red+ 0.6*green + 0.1*blue).
4. copy the calculated pixel values to newly created array.

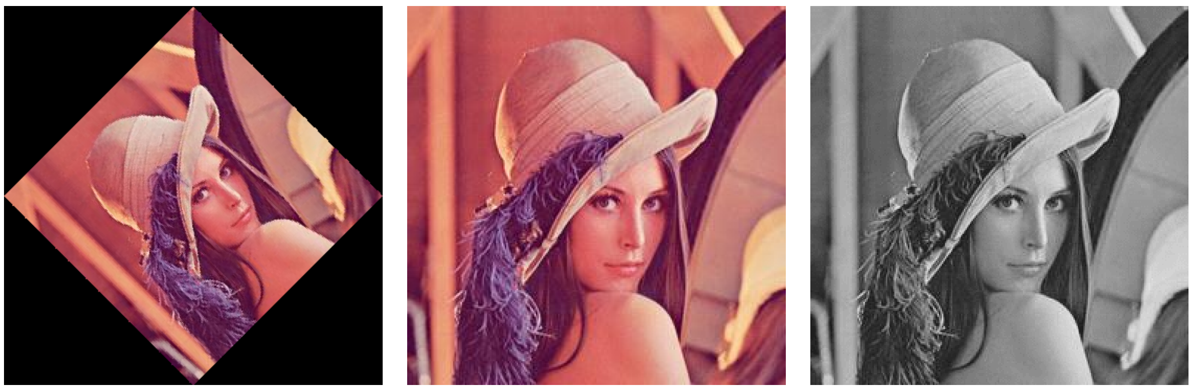
3 Results

3.1 Lena



(a) (b) (c)

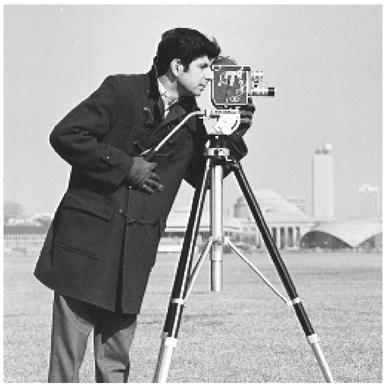
Figure 1: (a)actual image (b)flipped image (c)90 degrees rotated image

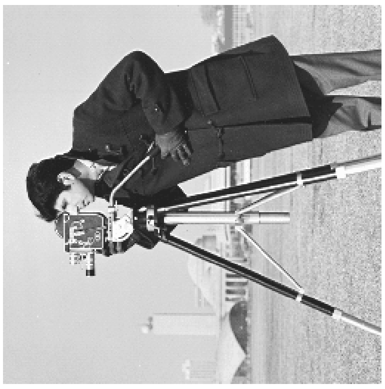


(a) (b) (c)

Figure 2: (a)45 degrees rotated image (b)scaled image (c)gray converted image

3.2 Cameraman


(a)


(b)

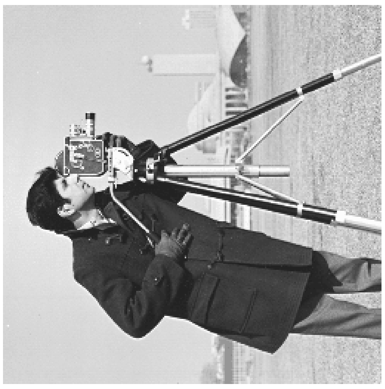
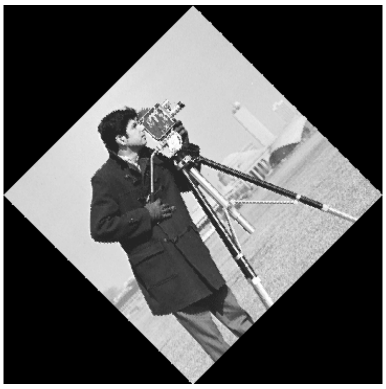

(c)

Figure 3: ((a)actual image (b)flipped image (c)90 degrees rotated image


(a)

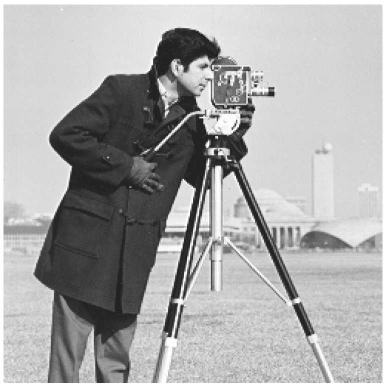

(b)

Figure 4: (a)45 degrees rotated image (b)scaled image

4 Analysis

1. BMP file format is well defined, the file format consists of file header, information header, and the pixel array. BMP file header gives important information such as height, width, size, offset and number of bits per pixel.
2. The location of the pixel array is different in case of Red-Blue-Green(RGB) images and Gray-scale images.
3. For converting the RGB image to gray scale image we have changed some fields in the header like bits per pixel , image offset, image size.
4. We have populated the contents of the color table with 0,1,2,3,.....,255 for gray scale images to inform the application about the actual color.
5. To get the image we have to read the pixel array coordinates from the bottom left not from top left.
6. The header field used to identify the BMP and DIB file is 0x42 0x4D in hexadecimal, same as BM in ASCII. The possible entries are BM, BA, CI, CPIC, PT.
7. For rotating the image by 45 degrees we have firstly calculated the possible height and width and then filled the pixel array. While interpolating the unoccupied pixels inside the new image we have assumed that at least one of the 8 neighbouring pixels will be occupied before interpolating.