NEGATIVE IMAGE

A Project Report
SUBMITTED BY

Name & Roll number

Yeshwant Balaji	CB.EN.U4AIE22102
Nithish Ariyha	CB.EN.U4AIE22140
Nishanth S	CB.EN.U4AIE22149
Shruthi R	CB.EN.U4AIE22154

Submission date: 7th February 2023

Batch-B Team1

Semester 1

As a part of the subject

PROBLEM SOLVING AND C PROGRAMMING

Centre for Computational Engineering and Networking

AMRITA SCHOOL OF ENGINEERING

AMRITA VISHWA VIDYAPEETHAM

COIMBATORE - 641 112 (INDIA)

ACKNOWLEDGEMENT

This project has been possible due to the sincere and dedicated efforts of many. We thank each individual who has specifically or by implication helped and helped us in growing our psyches and investigating the learning which lies in profundities of room. We might initially want to express gratitude towards Department of Computational Engineering and Networking for

This project has given us part of learning about Image processing using C Language has really made us consider something we could just envision.

We might want to thank our Mentor Ms. Aswathy P, for giving every one of us the vital data about the project

Our folks have given their most extreme help by helping us fiscally and giving us moral impact. We healthily express gratitude toward them for their assistance. We likewise thank our establishment Amrita Vishwa Vidyapeetham, to give us this chance to take an interest in this challenge.

Table of Contents

Table of Figures

Abstract

One of the most common ways to convey data or information is by making use of images. Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which the input is an image and the output may be an image or a characteristic or feature associated with that image. One of the alterations that can be done to an image is to generate a negative image

Negative images are representations of the same object or scene but with inverted colors, i.e., it is a total inversion, in which the light areas in the image appear dark and vice-versa.

The method will use basic image processing techniques such as color channel manipulation, which basically RGB manipulation, to generate the negative image. This has the potential to have applications in the fields of image processing and computer vision.
This project aims to develop a C programming-based method for generating negative images from positive ones. Here, we have selected a BMP type of image, and generated the negative image. The output of this project will be a C program that can take a positive image as input and generate its corresponding negative image.

Chapter-1

Introduction

Negative image

A positive image is a normal image. A negative image is a complete inversion, where, the bright areas appear dark and vice versa. A color-negative image is further color-inverted so that red areas become cyan, green areas become magenta, blue areas become yellow, or vice versa.

Negatives are usually used to make positive prints by reversing the image a second time to restore the light and dark areas to their original tones. Like digital files, a single negative image can be used to generate many final prints with widely differing appearances. A print can often be replaced, as long as its negative is available, but the negative itself is irreplaceable.
Film negatives usually have lower contrast than final printed positive images, but have greater dynamic range. Printing on photographic paper usually increases the contrast.
Bringing negative film images into the digital realm allows contrast adjustments during scanning, or more commonly during subsequent post-processing.

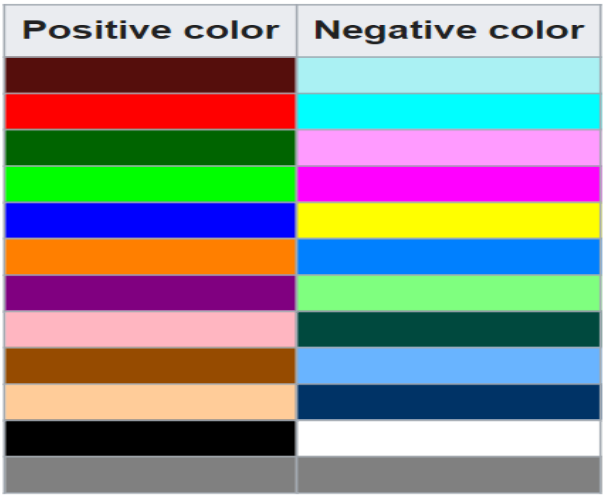


Figure 1 : Positive vs Negative color

BMP- Bitmap Image File

The BMP file format, or bitmap, is a raster graphics image file format used by Microsoft Windows and OS/2 operating systems in particular to store digital bitmap images regardless of the display device (graphics adapter, etc.).

File extensions having .bmp represent Bitmap Image files that are used to store bitmap digital images. These images are independent of graphics adapter and they are also called Device Independent Bitmap (DIB) file format.

The BMP file format can store monochrome and color two-dimensional digital images of various color depths, with optional data compression, alpha channel, and color profile. The Windows Metafile (WMF) specification covers the BMP file format. The color can consist of 16-bit entries that constitute indexes to the currently referenced pallet instead of explicit RGB color definitions

The bit depth refers to the numbers of colors that can be stored in that pixels. The higher the bit depth of an image, the more the colors it can store. For example, a simple 1-bit image can store only two colors, white and black. This is because the 1-bit can only store one of two values, 0 (white) and 1 (black).

The BMP file is of 6 types depending upon the bit depth of the pixels.

Bits per pixel	Number of colors that can be assigned to a pixel
1	2^1 = 2
2	2^2 = 4
4	2^4 = 16
8	2^8 = 256
16	2^16 = 65,536
24	2^24 = 16,777,216

Table 1 : Types of BMP files based on Bit depth

File Structure

A bitmap image file consists of a fixed-size structure (header) and variable-size structures that appear in a predetermined order. Since this file format has a long history, several different versions of these structures can appear in the file.

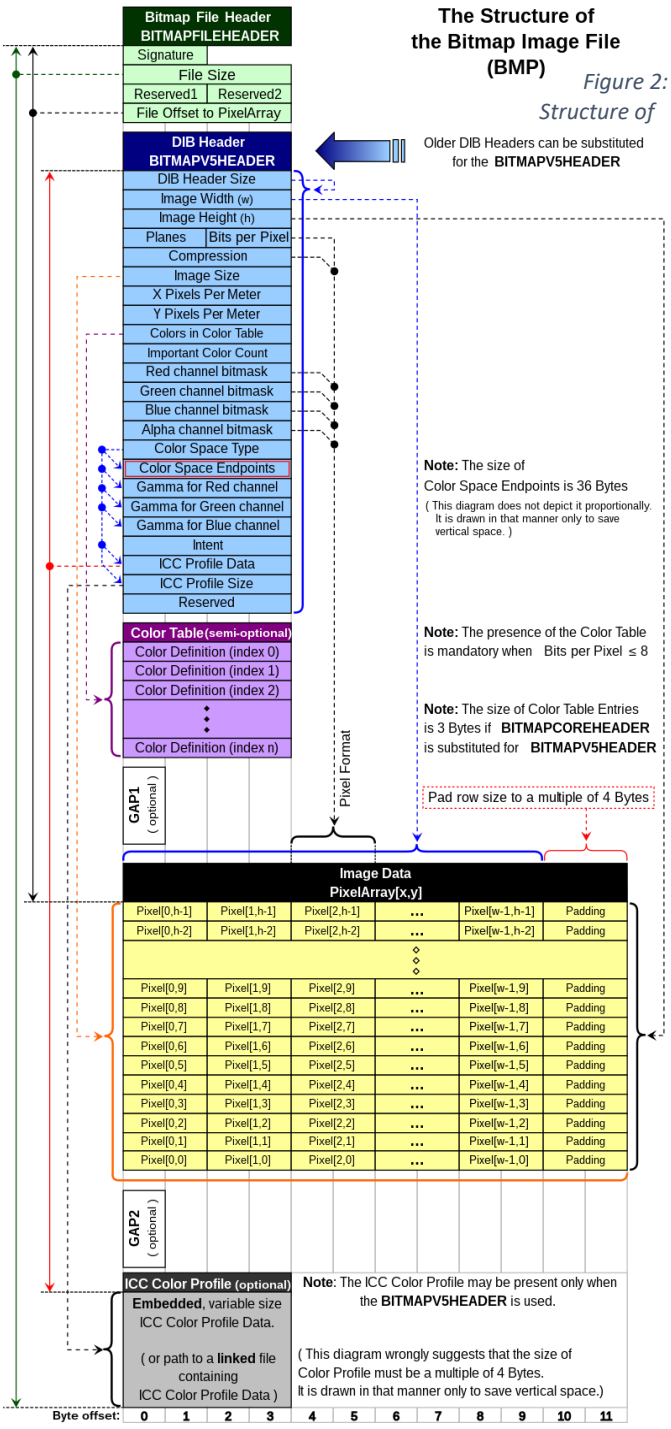


Figure 2: Structure of BMP file/Image [1]

Header structure

BMP image has a 54-byte image header, 1024-byte colorTable if present, and the rest is the image data. The header field used to identify a BMP file is 0x420x4D in hexadecimal. The header values must be same for both the original file and the negative image file as only the pixel values differ between them. This is applicable only in cases like producing negative images

offset	size	description
0	2	signature, must be 4D42 hex
2	4	size of BMP file in bytes (unreliable)
6	2	reserved, must be zero
8	2	reserved, must be zero
10	4	offset to start of image data in bytes
14	4	size of BITMAPINFOHEADER structure, must be 40
18	4	image width in pixels
22	4	image height in pixels
26	2	number of planes in the image, must be 1
28	2	number of bits per pixel (1, 4, ,8 or 24) (bitDepth)
30	4	compression type (0=none, 1=RLE-8, 2=RLE-4)
34	4	size of image data in bytes (including padding)
38	4	horizontal resolution in pixels per meter (unreliable)
42	4	vertical resolution in pixels per meter (unreliable)
46	4	number of colors in image, or zero
50	4	number of important colors, or zero

Table 2: Bitmap Header [2]

The header values we process in this topic are height, width and bitDepth.

- The 19th bit in the header gives us the height of the image.
- The 23rd bit in the header gives us the width of the image
- The 29th bit in the header gives us the bitDepth of the image.

Color table:

A bitmap file stores an image as a grip of pixels, where each pixel is a single color. The file stores a list of the color for each pixel, stored as a binary code indicating color. Pixel is short for Picture Element. Each pixel is a square of a single color. The color table occurs in the BMP file directly after the BMP file header, the DIB header.

The color table is a block of bytes listing the colors used by the image. Each pixel in an indexed color image is described by a number of bits (1, 4, or 8) which is an index of a single color. The purpose of the color palette in indexed color bitmaps is to inform the application about the actual color that each of these index values corresponds to.

The color table is normally not used when pixels are in 16-bit per pixel format (and higher).

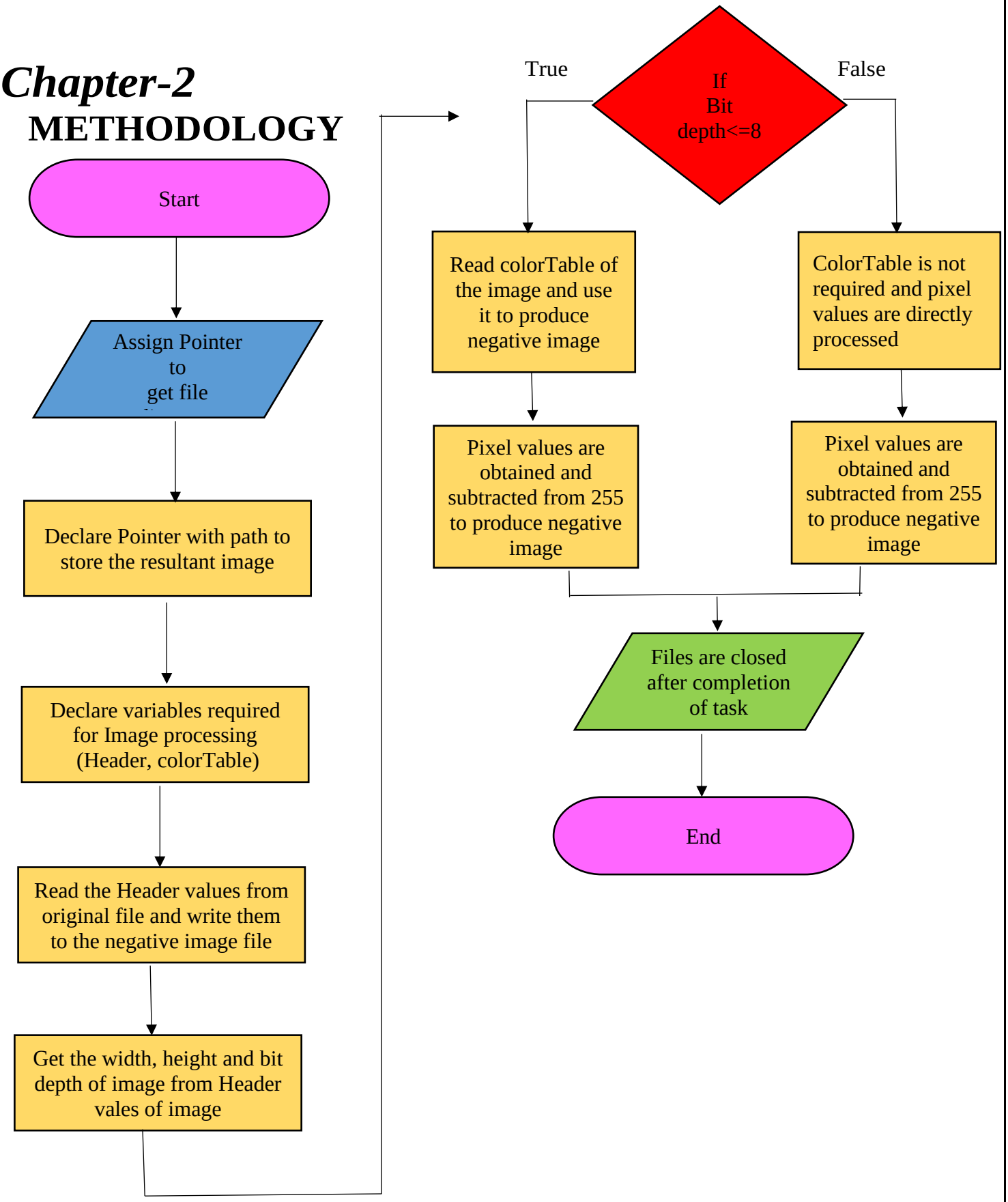
Creation of Negative image

To create a negative image, we need to compliment the value of the color in each pixel obtained from the original file. We achieve this complement by subtracting the read image value from 255. We use BMP image here as it is less complex to understand and decode and its relative ease of processing using C language.

Here the images used are in the form of 24-Bit BMP files as it's the default file obtained by converting images of other formats in MS Paint application.

Chapter-2

METHODOLOGY



Chapter-3

SOURCE CODE

```
#include <stdio.h>

int main() {
    int i, j;
    FILE *fp =
fopen("D:/c/projects/images/testmax5x.bmp",
"rb");
    FILE *fp2 =
fopen("D:/c/projects/images/testmax5x_neg.b
mp", "wb");
    unsigned char header[54];
    unsigned char colorTable[1024];
    unsigned char r, g, b;
    fread(header, sizeof(unsigned char),
54, fp);
    fwrite(header, sizeof(unsigned char),
54, fp2);
    int width = *(int*)&header[18];
    int height = *(int*)&header[22];
    int bitDepth = *(int*)&header[28];
    printf("%d,%d,%d",width,height,bitDepth
);

    if (bitDepth <= 8)
    {
        fread(colorTable, sizeof(unsigned
char), 1024, fp);
        fwrite(colorTable, sizeof(unsigned
char), 1024, fp2);
        for (i = 0; i < width; i++)
        {
            for (j = 0; j < height; j++)
            {
                r=fgetc(fp);
                fputc((255-r),fp2);
            }}}

    else
    {
        for (i = 0; i < height; i++)
        {
            for (j = 0; j < width ; j += 1)
            {
                r = fgetc(fp);
                g = fgetc(fp);
                b = fgetc(fp);
                fputc((255-r), fp2);
                fputc((255-g), fp2);
                fputc((255-b), fp2);
            }
        }
    }

    fclose(fp);
    fclose(fp2);

    return 0;
}
```

SOURCE CODE EXPLANATION

```
FILE *fp = fopen(" ", "rb");
```

Image file is obtained in the form of pointer in binary read mode.

```
FILE *fp2 = fopen(" ", "wb");
```

Output Image file is declared in the form of pointer in binary write mode.

```
unsigned char header[54];
unsigned char colorTable[1024];
unsigned char r, g, b;
```

Variables are declared which are required for image processing. These variables provide us information about the image.

```
fread(header, sizeof(unsigned char), 54,
fp);
fwrite(header, sizeof(unsigned char), 54,
fp2);
```

Header values of the image is obtained from original file and is written onto the output image file. Here, the unsigned char represents the positive values from 0 to 255.

```
int width = *(int*)&header[18];
int height = *(int*)&header[22];
int bitDepth = *(int*)&header[28];
```

Required Variables which is the Width, Height and the bitDepth are declared and obtained from Header array.

```
if (bitDepth <= 8)
    {
        fread(colorTable, sizeof(unsigned char), 1024, fp);
        fwrite(colorTable, sizeof(unsigned char), 1024, fp2);
        for (i = 0; i < width; i++)
        {
            for (j = 0; j < height; j++)
            {
                r=fgetc(fp);
                fputc((255-r),fp2);
            }
        }
    }
```

First the Bit depth of the image is checked and if it is less than 8, colorTable is obtained from the original image file and is written in output file Then by traversing and obtaining each pixel values and subtracting it from 255, we obtain pixel values of the negative image, which are then written on the output file.

```
else
    {
      for (i = 0; i < height; i++)
        {
            for (j = 0; j < width ; j += 1)
            {
                r = fgetc(fp);
                g = fgetc(fp);
                b = fgetc(fp);
                fputc((255-r), fp2);
                fputc((255-g), fp2);
                fputc((255-b), fp2);
            }
        }
    }
```

If the Bit depth of the image is more than 8, colorTable is not required and is not obtained from the original image. The pixel values are directly changed to negative pixel values.

Then by traversing and obtaining each pixel values and subtracting it from 255, we obtain pixel values of the negative image, which are then written on the output file

```
    fclose(fp);
    fclose(fp2);

    return 0;
}
```

After completion of loop, files are closed and the program ends.

Chapter-4

RESULTS

Figure 3: landscape

Figure 4: landscape negative

Figure 5: landscape 2

Figure 6: landscape 2 negative

Here in Figure 3 and 5, images are in their original colors and in Figure 4 and 6, images are in their negative form.

By traversing through each image using for loops pixel is obtained and then subtracted from 255 to obtain negative pixels value and then each pixel is appended to output file.

We can note that all the lighter regions in the original file are converted to a darker shade and vice-versa. The red color in the original image is converted to cyan color. The areas of green are converted to magenta and the areas which are in blue color becomes yellow color.

Conclusion

The scope of image processing is vast and has many applications out of which we have covered a small section of negative image transformation using the platform of C-Programming. This is useful in increasing our knowledge in coding as well as get an idea of the real-life applications of C as well as of image transformations. Throughout this course of C-Programming we have learnt to input texts, process it according to the desired instructions and get an output. Through this project, we have learnt how to input an image, process and modify it, as well as receive the required output.

References

[1] "http://www.fastgraph.com/help/bmp_header_format.html," [Online].

[2] "https://en.wikipedia.org/wiki/BMP_file_format," [Online].

[3] "https://docs.fileformat.com/image/bmp/," [Online].

[4] "https://abhijitnathwani.github.io/blog/2017/12/19/Introduction-to-Image-Processing-using-C," [Online].

[5] "https://abhijitnathwani.github.io/blog/2017/12/21/Negative-Image-using-C," [Online].