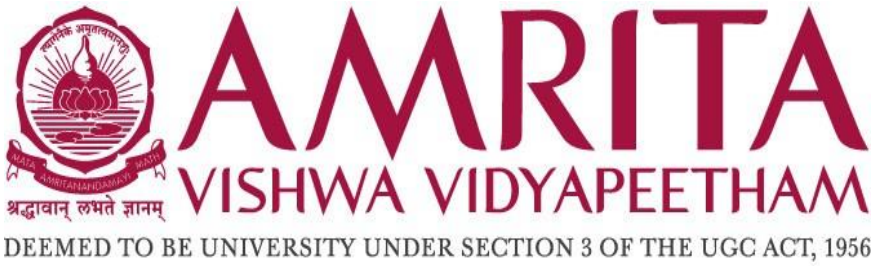

Centre for Computational Engineering and Networking

AMRITA SCHOOL OF ENGINEERING

AMRITA VISHWA VIDYAPEETHAM

COIMBATORE - 641 112
(INDIA)

February - 2023

Project Report

Submitted by

GROUP – 12

GROUP MEMBERS:

S.NO	NAME	ROLL NO
1	M.NILAVARASAM	CB.EN.U4AIE22139
2	P.GUNA VARDHAN	CB.EN.U4AIE22145
3	P.SAI SANTHAN	CB.EN.U4AIE22146
4	KJ.MADHAVAN NARAYANAN	CB.EN.U4AIE22135

As a part of the subject

PROBLEM SOLVING AND C PROGRAMMING

END-SEM PROJECT

DECLARATION

We declare that the Submitted Report is our original work and no values and context of it has been copy pasted from anywhere else. We take full responsibility, that if in future, the report is found invalid or copied, the last decision will be of the faculty concerned. Any form of plagiarism will lead to disqualification of the report.

Place: Ettimadai

Date: 07-2-2023

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our teacher (MS. Aswathy madam), who gave us the golden opportunity to do this wonderful project on the topic Image Thresholding, which also helped us in doing a lot of Research and we came to know about so many new things. We are thankful for the opportunity given. We would also like to thank our group members, as without their cooperation, we would not have been able to complete the project within the prescribed time.

Table of Contents

Table of figures

Chapter-1 Thresholding image in C

1.1.Abstract

Image thresholding is a technique in image processing used to convert a grayscale or color (RGB) image into a binary image, where pixels are either set to 0 or 1 or 0 and 1 based on a defined threshold value by the user. In this project we aim to implement or make an c-program such that it take's greyscale or color image as input from user and do image thresholding. Our program will take the user defined image and convert the image pixels(the specified pixels which satisfies the condition of the threshold of an pixel)to either black or white or an black and white image depending on the condition (sets the value of the pixel's value to '0' or '1' based on the condition).The threshold value is user defined or an inbuilt value is also assigned ,and finally this image is stored in the path defined by the user. This project is built only with the help of the stdlib library. This program is an efficient way to convert an image to binary image (black and white image).

1.2.Introduction

Image thresholding is a technique in image processing used to convert a grayscale or color (RGB) image into a binary image(generally), where pixels are either set to 0 or 1 or 0 and 1 based on a defined threshold value by the user. We will show in the upcoming section's how we have implemented a c-program code for the above topic, with the help of only stdlib and stdio libraries.in the below section we explain what are the meaning of the terms that will come in the project.

1.2.1.Grey-Scale Image

We know that each pixel's in an image has an certain value,in an grey-scale image all the pixels will have some value and the value ranges from 0-255,and 0-black,255-white.the pixel's value in an image can be anything in the above mentioned range(0-255),for instance if the pixel value is 50 it is grey in colour,but an darkish grey.due to this range is why we can see an image in grey shades.The greyscale image has only one channel because the pixel's will determine only the brightness of the image (if near '255' brighter and if near '0' then darker),unlike an RGB image discussed in the below section.

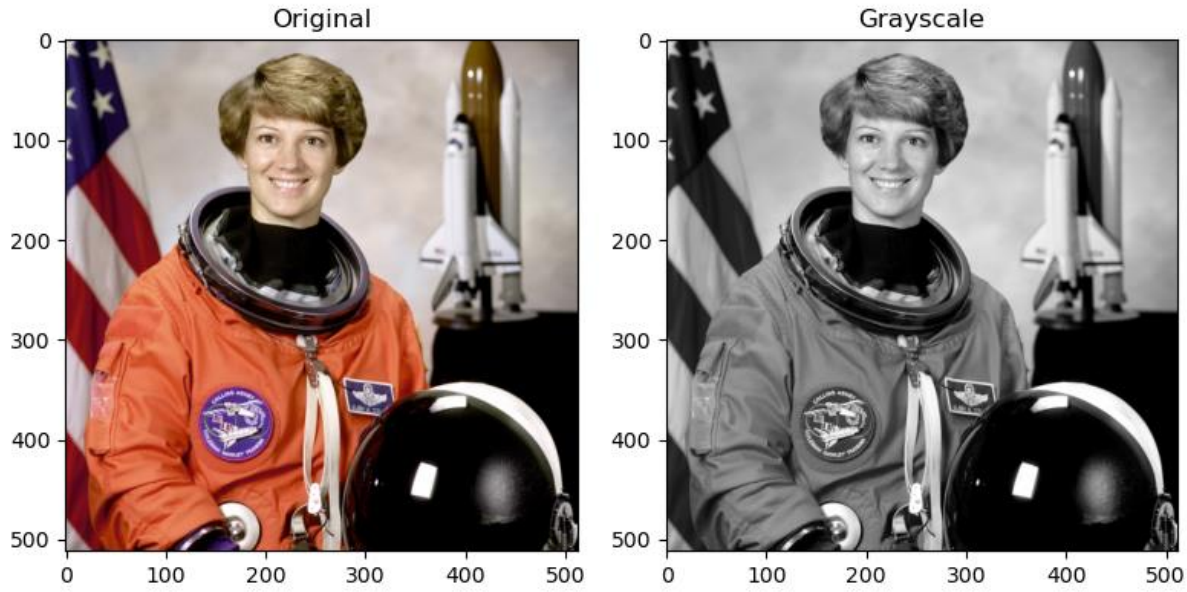

***Figure 1**-Grey-scale image.*

In mathematical point of view,or to say how an image is stored genrally is, grey-scale image is stored as an two-dimensional array of each element in the array is an integer values ranging from 0-255 and this values is non other than the pixels,due to this form of storing image's(in the form of matrix) is why we can access the pixel's of image in c and the other language's with ease.Now an image is seen as matrix, and operation's can be done accordingly on the pixels.

1.2.2.RGB image

As we discussed above each pixel has certain value.RGB-red green blue in mathematical point of view it is an three-dimensional matrix.suppose x and y are the coordinates of any element in an two-dimensional matrix,

then the coordinates of red layer refers to (x,y,0) , the green layer coordinates refers to the (x,y,1) and the blue layer coordinates refers to (x,y,2).blue layer refer to the two dimesional array,therefore the blue, green and red layers each seperately refers to an two-dimesional matrix and when combined is an three-dimensional matrix.note any color

image,which Is mixture of any color is non other then but an combination of the three colors.

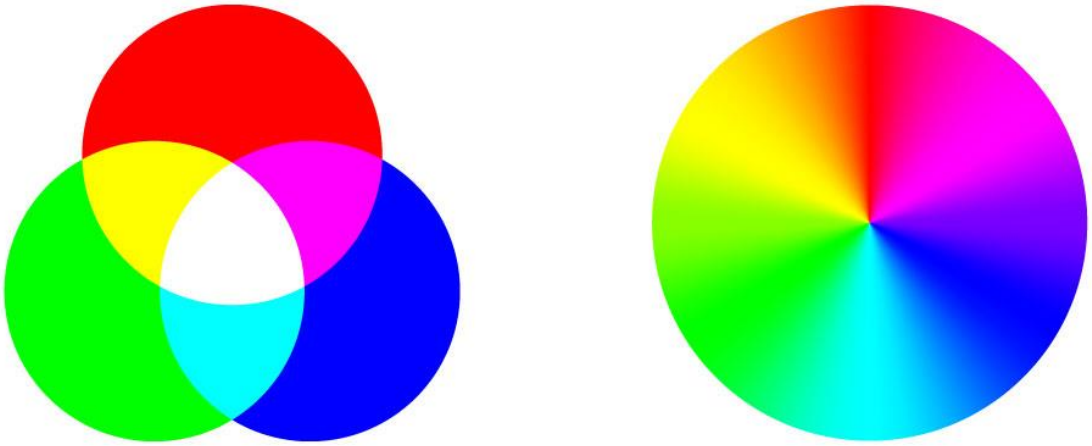

***Figure 2**-RGB-image.*

The intensity values of the red, green, and blue channels can range from 0 to 255, where 0 represents the minimum intensity (no color) and 255 represents the maximum intensity (full color). For example, a pixel with RGB values of (255, 0, 0) would be fully red, a pixel with RGB values of (0, 255, 0) would be fully green, and a pixel with RGB values of (0, 0, 255) would be fully blue. A pixel with RGB values of (255, 255, 255) would be white, and a pixel with RGB values of (0, 0, 0) would be black. RGb images has three channels(red,green,blue) together represents any color image.

1.2.3.Black And White Image.

Black an white image is an special case of greyscale image,where each pixel's in an image has value eithier '0'(black) or '255'(white) only, therfore the resulting image will have only two shades of color black and white,in other way to say black and white image is an binary image.

***Figure 3**-Black and white image.*

1.2.4.Digital image.

Digital image is an array of pixels in 2-dimesional space,each pixel's in an image correspond's to some value a numeric value,eg (in a greycale image,each pixel will have value only in the range of 255-0),this digital representaion of pixels in an image in form of array is known as digital image.In terms of c-programming an digital image is viewed in a different way , an digital image can be considered as 2-dimesional array of pixels.c-is an basic language unlike java or python it is not velversed,in c we can do image processing with the help of digital image since , each pixel in an image can be considered as some integer value.the pixels of an image can be stored as two-dimesional array of integers where each integer represents the intesity of a pixel,and this matrix which stores the numeric value of the integers can be accesed by doing the basic matrix operation and file handling etc.hence this digital images are very usefull in do image processing in c-language.

1.2.4.1.pseudo code for digital image

```
#define WIDTH 300
#define HEIGHT 200
int image[HEIGHT][WIDTH];
for (int i = 0; i < HEIGHT; i++)
 {
 for (int j = 0; j < WIDTH; j++)
{
    image[i][j] = ...; // assign the intensity value for each pixel
}
}
```

This an pseudo code for acessing all the integer values from an binary image.

1.2.5.Tresholding of image

Generally Image tresholding is a process of converting an greyscale image to an binary image(binary image where each pixel's in an binary image has value eithier '0' or '255'),an black and white image is considered as an binary image.which pixel's in an image should be assigned the value '0' or '255' is decided by the treshold value,we know that in an greyscale image each pixels value will be in the range of 0-255.The treshold value are user defined,if an pixel value is greater than the treshold value assigned by the user then we will assign it's value as '0'(black) and the other pixels which don't satisfy the above condition will be assigned value as '255'(white) and vice-versa(the values '0' and '255' can be assigned

in the opposite way also),this is also an process of converting greyscale image to black&white image.

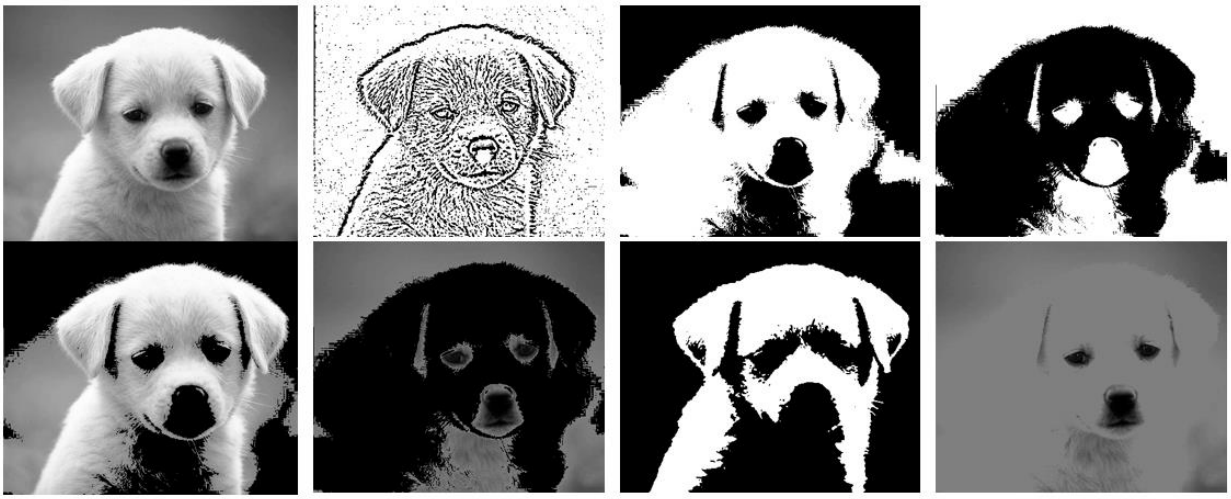

***Figure 4**-All type's of tresholding of an image*

The above discussed definiation for treshold is also known as global tresholding of an image ,which is also said as otsu's tresholding of image.But,there are many types of tresholding of an image some of the following types are mentioned below:

1. In this process each pixel value is checked through if-condition and if the value is greater then the threshold value it is assigned as white '255' and the pixel value's lesser than the treshold value is not disturbed and left as such,(to enhace the brightness)
2. This is just an opposite of the above discussed case .In this process each pixel value is checked and if the value is greater then the treshold value it is not disturbed and the pixel value's lesser than the treshold value is assigned as '255', ,(to enhace the brightness).
3. In this case as usual each pixel value is checked and if the value is greater then the threshold value it is assigned as black('0') and the pixel value's lesser than the treshold value is not disturbed. ,(to enhace the darkness).
4. This case is just the opposite of the above case . so in this process the each pixel value is checked and if the value is greater then the

treshold value it left as such and the pixel value's lesser than the treshold value is assigned '0'. ,(to enhace the darkness).

5. If an pixel value is greater than the treshold value assigned by the user then we will assign it's value as '0'(black) and the other pixels which don't satisfy the above condition will be assigned value as '255'(white) and vice-versa(the values '0' and '255' can be assigned in the opposite way also),to make black&white image

The above disccused are some of the cases for image tresholding,and we will carry the above discussed six cases in our project.

1.2.5.Bmp images

Bmp stand for Bit-Map image.As we discussed above about digital image's,bmp image's usually store the the value of pixels in integer form in simple words to say,it is an digital image.The simplicity of the BMP format makes it easy to use and understand. It stores digital images in a straightforward way, using a pixel-based format that is easy for computers to read and process,note it stores image as two dimensional-matrix(2-D).

1.2.6.PPM images

PPM (Portable Pixmap Format) is a basic image which stores image data as ASCII text or binary data, and includes information about the image dimensions and color depth,it stores in three-dimesional matrix(3-D).

1.2.7.Image processing in BMP images.

Here are some of the reasons we figgured out why we used bmp image:

- The BMP format is straightforward to work with, as it stores images in a pixel-based format that is easy for computers to understand.(that each pixel in the image is represented by a specific color value

in the file. This makes it easy to manipulate and edit the image by directly accessing and changing the pixel values, but also results in larger file sizes compared to other image formats that use compression).

- Finally, the BMP format supports lossless compression, which is important in image processing, where it is important to preserve the quality of the image. BMP files can be compressed without losing any quality, making it an ideal format for images that will be processed or modified in C.
- Therefore the important reason why we used bmp image is that it stores in pixel-format,each pixel has some digital values,and this digital values are some integers.basically bmp image is,stored as array of integers.
- And the image is not in compressed form,because it is stored as values, where it is important to preserve the quality of the image.

1.2.8.Image processing in PPM image's.

PPM (Portable Pixmap Format) is very portable image format, we figgured it is an best option for image processing in C language. Here are some of the reasons we figured out why we used ppm image's.

- PPM is a plain and easy-to-read and write format that makes reading and writing picture data in C programmes simple.
- PPM is text-based, the picture data may be easily transferred between many platforms and computer systems.
- In c programms PPM image is widely used since there are many tools and libraries to read and write images,which makes it very compact.
- PPM is a lossless format, thus when a picture is saved and subsequently read back into memory, no image data is lost. For the

purpose of processing images,so there is no loss in ascii-values, which make's it to easily acces values.

1.2.9.Image processing in JPEG and PNG image's.

We know that c is an low level language,the jpeg and png type of image format is usually in compressed form due to which some amount of data of the pixels are compressed and are lossed,which is why it is not possible to acces this type of image format's in c,with the help of regular libraries.But this type of image format's(jpeg an png) can be manipulated with the help of special libraries like libjpeg,libpeg and std image.h .But the image type like bmp,ppm the image is not stored in compressed form ,so the values are not lost which is why bmp is preffered over bmp images.The images also stores digital value's of the image.

- Bmp,ppm, no loss of data,preffered.
- Jpeg,png loss of data,some amount of data is lossed,less preffered.

1.2.10.RGB-to greyscale.

The RGB (Red, Green, Blue) image can be converted to a greyscale image by using a mathematical formula that determines that gives the value in the range of 0-255 ,which is in the range of an greyscale image of each pixel. The most common formula used is:

$Y = 0.2989 * R + 0.5870 * G + 0.1140 * B$

where Y is the pixel value we get after substituing the value of the red green and blue componeets.To convert an RGB image to greyscale in C, you can loop through each pixel in the image, extract its R, G, and B components, and use the formula above to calculate the corresponding Y

value. Then, you can set all three RGB components of the pixel to the same Y value, effectively converting it to greyscale.

Below is an pseudo for this conversion in c,

```
int i, j;
    for (i = 0; i < height; i++)
    {
        for (j = 0; j < width; j++)
        {
            int index = (i * width + j) * 3;
            unsigned char r = image[index];
            unsigned char g = image[index + 1];
            unsigned char b = image[index + 2];
            unsigned char y = (unsigned char)(0.2989 * r + 0.5870 * g + 0.1140 *
b);
            image[index] = image[index + 1] = image[index + 2] = y;
        }
    }
}
```

This an pseudo code for image processing in c.

1.3.Methadology

From the above section's it is clear that why we preffered bmp and ppm type of image's over the other images jpeg and png and the other type of format's.our aim is to design an c-program code such that it take's input from user as image and to do tresholding.the image can be greyscale or BMP image.note that our aim is to do design a c-code which doesn't uses external libraie's so,the type of image's which support our code's are BMP and PPM etc.

1.3.1.Line wise explanation (BMP image)

#include<stdio.h> - This line includes the standard input/output library.

#include<stdlib.h> - This line includes the standard library.

int main() - This line declares the main function which returns an integer value.

{ - This line marks the beginning of the main function.

int width, height, max_color; - This line declares three integer variables, width, height, and max_color.

unsigned char *image; - This line declares an unsigned char pointer variable, image.

int new_width, new_height; - This line declares two integer variables, new_width and new_height.

unsigned char *new_image; - This line declares an unsigned char pointer variable, new_image.

double x_ratio, y_ratio; - This line declares two double variables, x_ratio and y_ratio.

int floor_x, floor_y; - This line declares two integer variables, floor_x and floor_y.

char str[100]; - This line declares a character array, str, of size 100.

printf("Enter the name of the file (with .ppm) :"); - This line prints the string "Enter the name of the file (with .ppm) :" to the console.

gets(str); - This line gets the input from the user and stores it in the variable str.

FILE *in_file = fopen(str, "rb"); - This line opens the file with the name provided by the user in read binary mode and assigns the file pointer to the variable in_file.

fscanf(in_file, "P6\n%d %d\n%d\n", &width, &height, &max_color); - This line reads the width, height, and max_color from the file provided by the user.

image = (unsigned char *) malloc(width * height * 3); - This line allocates memory for the image pointer.

fread(image, 1, width * height * 3, in_file); - This line reads the image from the file provided by the user.

fclose(in_file); - This line closes the file.

for (int i = 0; i < width * height * 3; i += 3) { - This line is a for loop that iterates from 0 to the width * height * 3, incrementing i by 3 each time.

int avg = (0.299*image[i] + 0.587*image[i + 1] + 0.114*image[i + 2]); - This line calculates the average of the r, g, and b values of the pixel.

image[i] = avg; - This line assigns the average to the r value of the pixel.

image[i + 1] = avg; - This line assigns the average to the g value of the pixel.

image[i + 2] = avg; - This line assigns the average to the b value of the pixel.

}

printf("enter the treshold value that u want or inbuild value is ""150"" \n"); - This line prints the string "enter the treshold value that u want or inbuild value is ""150"" " to the console.

printf("1.)inbuilt treshold value is ""150"" \n"); - This line prints the string "1.)inbuilt treshold value is ""150"" " to the console.

printf("2.)write the treshold value that i want \n"); - This line prints the string "2.)write the treshold value that i want " to the console.

int n,treshold; - This line declares two integer variables, n and treshold.

scanf("%d",&n); - This line gets input from the user and stores it in the variable n.

printf("u have entered (%d) option\n",n); - This line prints the string "u have entered (%d) option\n" and the value of n to the console.

while((n!=1)&&(n!=2)) - This line is a while loop that executes as long as n is not equal to 1 and 2.

{ - This line marks the beginning of the while loop.

printf("%d is not the correct option\n",n); - This line prints the string "%d is not the correct option\n" and the value of n to the console.

printf("enter the correct option\n"); - This line prints the string "enter the correct option\n" to the console.

scanf("%d",&n); - This line gets input from the user and stores it in the variable n.

} - This line marks the end of the while loop.

printf("u have entered (%d) option\n",n); - This line prints the string "u have entered (%d) option\n" and the value of n to the console.

switch(n) - This line is the beginning of a switch statement.

{ - This line marks the beginning of the switch statement.

case 1: - This line is the beginning of the first case.

treshold=150; - This line assigns the value 150 to the variable treshold.

break; - This line exits the switch statement.

case 2: - This line is the beginning of the second case.

printf("enter the treshold value:\n"); - This line prints the string "enter the treshold value:\n" to the console.

scanf("%d",&treshold); - This line gets input from the user and stores it in the variable treshold.

break; - This line exits the switch statement.

default: - This line is the default case.

printf("%d is not the correct constraint\n",treshold); - This line prints the string "%d is not the correct constraint\n" and the value of treshold to the console.

} - This line marks the end of the switch statement.

printf("therfore the treshold value that u want is %d\n",treshold); - This line prints the string "therfore the treshold value that u want is %d\n" and the value of treshold to the console.

printf("enter the type of operation that u want to do\n"); - This line prints the string "enter the type of operation that u want to do\n" to the console.

printf("1.)to make the values greater than treshold brigther(brigther image)\n"); - This line prints the string "1.)to make the values greater than treshold brigther(brigther image)\n" to the console.

printf("2.)to make the values lesser than treshold brigther(brigther image)\n"); - This line prints the string "2.)to make the values lesser than treshold brigther(brigther image)\n" to the console.

printf("3.)to make the values greater than treshold darker(darker image)\n"); - This line prints the string "3.)to make the values greater than treshold darker(darker image)\n" to the console.

printf("4.)to make the values lesser than treshold darker(darker image)\n"); - This line prints the string "4.)to make the values lesser than treshold darker(darker image)\n" to the console.

printf("5.)to make the values greater than treshold value brigther and lesser darker(black&white image)\n"); - This line prints the string "5.)to

make the values greater than treshold value brigther and lesser darker(black&white image)\n" to the console.

printf("6.)to make the values greater than treshold value darker and lesser brighter(black&white image)\n"); - This line prints the string "6.)to make the values greater than treshold value darker and lesser brighter(black&white image)\n" to the console.

printf("7.)to make rgb to greyscale\n"); - This line prints the string "7.)to make rgb to greyscale\n" to the console.

int new;

- This line declares a variable of type integer called "new".

scanf("%d",&new);

- This line reads in an integer value from the user and stores it in the variable "new".

printf("%d is the option that u have selected\n",new);

- This line prints out the value of the "new" variable and a message.

while((new!=1)&&(new!=2)&&(new!=3)&&(new!=4)&&(new!=5)&&(new!=6)&&(new!=7))

- This line checks if the value of the "new" variable is not equal to 1, 2, 3, 4, 5, 6, or 7. If the condition is true, the code inside the while loop will execute.

printf("%d is not the correct option\n",new);

- This line prints out the value of the "new" variable and a message.

END-SEM PROJECT

printf("enter the correct option::");

- This line prints out a message.

scanf("%d",&new);

- This line reads in an integer value from the user and stores it in the variable "new".

printf("%d is the correct option that u have selected\n",new);

- This line prints out the value of the "new" variable and a message.

switch(new)

- This line begins a switch statement using the value of the "new" variable.

case 1:

- This line is the first case of the switch statement. If the value of the "new" variable is equal to 1, the code inside the case statement will execute.

for (int i = 0; i < width * height*3; i++) {

- This line starts a for loop which will loop through the image array.

if (image[i] < treshold)

```
{
 image[i] = 255;
}
else
image[i] = image[i];
```

- This line checks if the value of the current index in the image array is less than the treshold. If it is, it sets the value to 255, otherwise, it sets the value to itself.

break;

- This line breaks out of the switch statement.

default:

- This line is the default case of the switch statement. If none of the other cases are executed, this code will be executed.

printf("%d",width*height);

- This line prints out the result of the multiplication of width and height.

FILE *out_file = fopen("outwithB.ppm", "wb");

- This line opens the file "outwithB.ppm" in write binary mode and stores a pointer to the file in the variable "out_file".

fprintf(out_file, "P6\n%d %d\n%d\n", width, height, max_color);

END-SEM PROJECT

- This line writes a string to the file pointed to by "out_file".

fwrite(image, 1, width * height * 3, out_file);

- This line writes the contents of the image array to the file pointed to by "out_file".

fclose(out_file);

- This line closes the file pointed to by "out_file".

free(image);

- This line frees the memory allocated to the image array.

return 0;

- This line returns 0 as the exit code of the program.

1.3.2.Line wise explanation (PPM image)

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int width, height, color;
    unsigned char *image;
    char str[100];
    printf("Enter the name of the file (with .ppm) :");
    gets(str);
    FILE *in_file = fopen(str, "rb");
    fscanf(in_file, "P6\n%d %d\n%d\n", &width, &height, &color);
    image = (unsigned char *) malloc(width * height * 3);
    fread(image, 1, width * height * 3, in_file);
    fclose(in_file);
    for (int i = 0; i < width * height * 3; i += 3) {
        int avg = (0.299*image[i] + 0.587*image[i + 1] + 0.114*image[i + 2]);
```

```
        image[i] = avg;
        image[i + 1] = avg;
        image[i + 2] = avg;
```

```
    FILE *out_file = fopen("outwithhhB1111.ppm", "wb");
    fprintf(out_file, "P6\n%d %d\n%d\n", width, height, color);
    fwrite(image, 1, width * height * 3, out_file);
    fclose(out_file);
    free(image);
    return 0;
```

Except these 2 sets of lines of codes, everything for PPM image is similar to that of BMP images. Therefore, both follows same logic and explanations.

Explanation for these 2 sets of codes is as follows :

#include <stdio.h>: This line includes the standard input/output header file which provides functions such as printf() and scanf() to take input and display output.

#include <stdlib.h>: This line includes the standard library header file which provides functions such as malloc() to allocate memory dynamically.

int main(): This line declares the main function which is the entry point of the program.

int width, height, color: These three lines declare three integer variables.

unsigned char *image: This line declares an unsigned character pointer variable.

char str[100]: This line declares a character array of size 100.

printf("Enter the name of the file (with .ppm) :");: This line prints out a message asking the user to enter the name of the file.

gets(str);: This line reads the name of the file entered by the user and stores it in the character array str.

END-SEM PROJECT

FILE *in_file = fopen(str, "rb");: This line opens the file entered by the user in read binary mode and stores the pointer to the file in in_file.

fscanf(in_file, "P6\n%d %d\n%d\n", &width, &height, &color);: This line reads the first three values from the file and stores them in width, height, and color variables.

image = (unsigned char *) malloc(width * height * 3);: This line allocates memory for the image array using malloc() and stores the pointer to the memory allocated in the image variable.

fread(image, 1, width * height * 3, in_file);: This line reads the binary data from the file and stores it in the image array.

fclose(in_file);: This line closes the file.

```
for (int i = 0; i < width * height * 3; i += 3) {
    int avg = (0.299*image[i] + 0.587*image[i + 1] + 0.114*image[i + 2]);
    image[i] = avg;
    image[i + 1] = avg;
    image[i + 2] = avg;
```

}: This loop calculates the average of the three color components of each pixel and stores it in all three of the components.

FILE *out_file = fopen("outwithhhB1111.ppm", "wb");: This line opens a file called outwithhhB1111.ppm in write binary mode and stores the pointer to the file in out_file.

fprintf(out_file, "P6\n%d %d\n%d\n", width, height, color);: This line writes the first three values to the file.

fwrite(image, 1, width * height * 3, out_file);: This line writes the binary data to the file.

fclose(out_file);: This line closes the file.

END-SEM PROJECT

free(image);: This line deallocates the memory allocated to the image array using malloc().

return 0;: This line returns 0 to the calling function.

1.4.Source code:

We have done the image tresholding for two types of image format BMP and PPM type of image format.below are the source code for BMP image and PPM image.

1.4.1 source code for BMP image.

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
void case1(unsigned char *imageData,unsigned char *newimageData,int imgDataSize);
void case2(unsigned char *imageData,unsigned char *newimageData,int imgDataSize);
void case3(unsigned char *imageData,unsigned char *newimageData,int imgDataSize);
void case4(unsigned char *imageData,unsigned char *newimageData,int imgDataSize);
void case5(unsigned char *imageData,unsigned char *newimageData,int imgDataSize);
void case6(unsigned char *imageData,unsigned char *newimageData,int imgDataSize);
int treshoold(int n);
void printtreshold();
void printoption();
int optioncheck(int new);
void mainswitch(int new,unsigned char *imageData,unsigned char *newimageData,int
imgDataSize);
void extraswitch(int n,unsigned char *imageData,unsigned char *newimageData,int
imgDataSize);
void check(char *name);
int main()
{
    char name[50];
    check(name);
    FILE* fp = fopen(name, "rb");
    if(fp==NULL)
```

```
    {
        printf("no such file exist,enter the valid file");
        return 0;
    }
    printf("1.)to continue the loop\n2.)to break\nclick any one option
mentioned\n");
    unsigned char *imageData;
    unsigned char *newimageData;
    unsigned char imageHeader[54];
    unsigned char colorTable[1024];
    fread(imageHeader, sizeof(unsigned char), 54, fp);
    int i,j;
    int width = *(int*)&imageHeader[18];
    int height = *(int*)&imageHeader[22];
    int bitDepth = *(int*)&imageHeader[28];
    int imgDataSize = width * height;
    imageData = (unsigned char*)malloc (imgDataSize * sizeof(unsigned char));
    newimageData = (unsigned char*)malloc (imgDataSize * sizeof(unsigned char));
    if(bitDepth <= 8)
    {
    fread(colorTable, sizeof(unsigned char), 1024, fp);
    }
    fread( imageData, sizeof(unsigned char), imgDataSize, fp);
    imgDataSize=width*height;
    int n;
    scanf("%d",&n);
    extraswitch(n,imageData,newimageData,imgDataSize);
    FILE *fo = fopen("nilaaaa.bmp", "wb");
    fwrite(imageHeader, sizeof(unsigned char), 54, fo);
    if(bitDepth <= 8)
    {
        fwrite(colorTable, sizeof(unsigned char), 1024, fo);
    }
        fwrite( newimageData, sizeof(unsigned char), imgDataSize, fo);
         printf("the image is proccesed go look your directory\n");
    fclose(fo);
    fclose(fp);
    ;
    return 0;
}



void case1(unsigned char *imageData,unsigned char *newimageData,int imgDataSize)
{
```

```
    int i;
     for(i = 0; i <= imgDataSize; i++) {
    if(imageData[i]>150)   {
      newimageData[i] =255;
    } else {
      newimageData[i]=imageData[i];
    }
  }
}
void case2(unsigned char *imageData,unsigned char *newimageData,int imgDataSize)
{
    int i;
    for(i = 0; i <= imgDataSize; i++)
    {
    if(imageData[i]>150)   {
      newimageData[i]=imageData[i];
    } else {
      newimageData[i]=255;
    }
  }
}
void case3(unsigned char *imageData,unsigned char *newimageData,int imgDataSize)
{
    int i;
    for(i = 0; i <= imgDataSize; i++)
    {
    if(imageData[i]>150)   {
      newimageData[i]=0;
    } else {
      newimageData[i]=imageData[i];
    }
  }
}
void case4(unsigned char *imageData,unsigned char *newimageData,int imgDataSize)
{
    int i;
    for(i = 0; i <= imgDataSize; i++)
    {
    if(imageData[i]>150)   {
      newimageData[i]=imageData[i];
    } else {
      newimageData[i]=0;
    }
  }
}
```

```
void case5(unsigned char *imageData,unsigned char *newimageData,int imgDataSize)
{
    int i;
    for(i = 0; i <= imgDataSize; i++)
    {
    if(imageData[i]>150)   {
      newimageData[i]=255;
    } else {
      newimageData[i]=0;
    }
  }
}
void case6(unsigned char *imageData,unsigned char *newimageData,int imgDataSize)
{
    int i;
    for(i = 0; i <= imgDataSize; i++)
    {
    if(imageData[i]>150)   {
      newimageData[i]=0;
    } else {
      newimageData[i]=255;
    }
  }
}
int treshoold(int n)
{
    int treshold;
while((n!=1)&&(n!=2))
    {
        printf("%d is not the correct option\n",n);
        printf("enter the correct option\n");
        scanf("%d",&n);
    }
    printf("u have entered (%d) option\n",n);
    switch(n)
    {
        case 1:
        treshold=150;
        break;
        case 2:
        printf("enter the treshold value:\n");
        scanf("%d",&treshold);
        break;
        default:
        printf("%d is not the correct constraint\n",treshold);
```

```
    }
    return treshold;
}
void printtreshold()
{
    printf("enter the treshold value that u want or inbuild value is ""150""
\n");
    printf("1.)inbuilt treshold value is ""150"" \n");
    printf("2.)write the treshold value that i want \n");
}
void printoption()
{
    printf("enter the type of operation that u want to do\n");
    printf("1.)to make the values greater than treshold brigther(brigther
image)\n");
    printf("2.)to make the values lesser than treshold brigther(brigther
image)\n");
    printf("3.)to make the values greater than treshold darker(darker image)\n");
    printf("4.)to make the values lesser than treshold darker(darker image)\n");
    printf("5.)to make the values greater than treshold value brigther and lesser
darker(black&white image)\n");
    printf("6.)to make the values greater than treshold value darker and lesser
brighter(black&white image)\n");
}
int optioncheck(int new)
{
    while((new!=1)&&(new!=2)&&(new!=3)&&(new!=4)&&(new!=5)&&(new!=6)&&(new!=7))
    {
    printf("%d is not the correct option\n",new);
    printf("enter the correct option::");
    scanf("%d",&new);
    }
}
void mainswitch(int new,unsigned char *imageData,unsigned char *newimageData,int
imgDataSize)
{
    switch(new)
    {
        case 1:
        case1(imageData,newimageData,imgDataSize);
        break;
        case 2:
        case2(imageData,newimageData,imgDataSize);
        break;
        case 3:
```

```
        case3(imageData,newimageData,imgDataSize);
        printf("hi......");
        break;
        case 4:
        case4(imageData,newimageData,imgDataSize);
        break;
        case 5:
        case5(imageData,newimageData,imgDataSize);
        break;
        case 6:
        case6(imageData,newimageData,imgDataSize);
        break;
        default:
        printf("some problem.....\n");
        }
}
void extraswitch(int n,unsigned char *imageData,unsigned char *newimageData,int
imgDataSize)
{

     switch(n)
    {
    case 1:
    printtreshold();
    int n,treshold;
    scanf("%d",&n);
    printf("u have entered (%d) option\n",n);
    treshold=treshoold(n);
    printoption();
    int new;
    scanf("%d",&new);
    int hi=optioncheck(new);
    printf("%d is the correct option that u have selected\n",hi);
    mainswitch(new,imageData,newimageData,imgDataSize);
    break;
    case 2:
    printf("hi......");
    break;
    }
}
void check(char *name)
{
    printf("enter the file name with .bmp\n");
    gets(name);
    printf("the name u entered is %s\n",name);
```

```
}
```

The above is the source code we have created for BMP image,note it supports only for .BMP image format,and it used only for greyscale image.The code work's well and good for greyscale image,and the code is done with help of function-reffrence.

1.4.2. source code for PPM image.

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int width, height, color;
    unsigned char *image;
    char str[100];
    printf("Enter the name of the file (with .ppm) :");
    gets(str);
    FILE *in_file = fopen(str, "rb");
    fscanf(in_file, "P6\n%d %d\n%d\n", &width, &height, &color);
    image = (unsigned char *) malloc(width * height * 3);
    fread(image, 1, width * height * 3, in_file);
    fclose(in_file);
    for (int i = 0; i < width * height * 3; i += 3) {
        int avg = (0.299*image[i] + 0.587*image[i + 1] + 0.114*image[i + 2]);
        image[i] = avg;
        image[i + 1] = avg;
        image[i + 2] = avg;
    }
    printf("enter the treshold value that u want or inbuild value is ""150""
\n");
    printf("1.)inbuilt treshold value is ""150"" \n");
    printf("2.)write the treshold value that i want \n");
    int n,treshold;
    scanf("%d",&n);
    printf("u have entered (%d) option\n",n);
    while((n!=1)&&(n!=2))
    {
        printf("%d is not the correct option\n",n);
        printf("enter the correct option\n");
        scanf("%d",&n);
```

```
    }
    printf("u have entered (%d) option\n",n);
    switch(n)
    {
        case 1:
        treshold=150;
        break;
        case 2:
        printf("enter the treshold value:\n");
        scanf("%d",&treshold);
        break;
        default:
        printf("%d is not the correct constraint\n",treshold);
    }
    printf("therfore the treshold value that u want is %d\n",treshold);
    printf("enter the type of operation that u want to do\n");
    printf("1.)to make the values greater than treshold brigther(brigther
image)\n");
    printf("2.)to make the values lesser than treshold brigther(brigther
image)\n");
    printf("3.)to make the values greater than treshold darker(darker image)\n");
    printf("4.)to make the values lesser than treshold darker(darker image)\n");
    printf("5.)to make the values greater than treshold value brigther and lesser
darker(black&white image)\n");
    printf("6.)to make the values greater than treshold value darker and lesser
brighter(black&white image)\n");
    printf("7.)to make rgb to greyscale\n");
    int new;
    scanf("%d",&new);
    printf("%d is the option that u have selected\n",new);
    while((new!=1)&&(new!=2)&&(new!=3)&&(new!=4)&&(new!=5)&&(new!=6)&&(new!=7))
    {
    printf("%d is not the correct option\n",new);
    printf("enter the correct option::");
    scanf("%d",&new);
    }
    printf("%d is the correct option that u have selected\n",new);
    switch(new)
    {
        case 1:
        for (int i = 0; i < width * height*3; i++) {
      if (image[i] < treshold)
       {
        image[i] = 255;
      }
```

END-SEM PROJECT

```
      else
      image[i] = image[i];

    }
    break;
        case 2:
       for (int i = 0; i < width * height*3; i++) {
      if (image[i] < treshold) {
        image[i] = image[i];
      }
      else
      image[i] = 255;

    }
    break;
        case 3:
        for (int i = 0; i < width * height*3; i++) {
      if (image[i] < treshold) {
        image[i] = image[i];
      }
      else
      image[i] =0;

    }
    break;
        case 4:

       for (int i = 0; i < width * height*3; i++) {
      if (image[i] < treshold) {
        image[i] = 0;
      }
      else
      image[i] = image[i];

    }
break;
        case 5:
      for (int i = 0; i < width * height*3; i++) {
      if (image[i] < treshold) {
        image[i] = 0;
      }
      else
      image[i] = 255;

    }
```

```
    break;
        case 6:
       for (int i = 0; i < width * height*3; i++) {
      if (image[i] < treshold) {
        image[i] = 255;
      }
      else
      image[i] = 0;

    }
    break;
        default:
        printf("choose the right option completed\n");
        }
        printf("%d",width*height);
    FILE *out_file = fopen("outwithhhB1111.ppm", "wb");
    fprintf(out_file, "P6\n%d %d\n%d\n", width, height, color);
    fwrite(image, 1, width * height * 3, out_file);
    fclose(out_file);
    free(image);
    return 0;
}
```

The above section is the source code for PPM type of image format,the above section convert's both rgb and greyscale image to the specific type of treshold.Above is done with the help of for loop inside the main function,since we have aldready used the reffernce-function call in the above section.

It is important to note that the above code is for p6,and for the p3 type of image just

```
    FILE *in_file = fopen(str, "rb");
```

```
FILE *out_file = fopen("outwithhhB1111.ppm", "wb");
```

Replace the 'rb' and 'wb' as 'r' and 'w'.

1.5.Results

Figure 5*-input lena image(BMP).*

Figure 6*-Output of all the six types of input images discused below.*

Figure 7-*Input and the output images for RGB image(1) using PPM format.*

***Figure 8**-Input and the output images for RGB image(1) using PPM format.*

1.6.Conclusion

Image thresholding is an important technique used in image processing and can be used to reduce noise and enhance the visibility of an image. The C language is a powerful computing language that is well suited to image processing applications. In this case study, we have explored how to implement image thresholding in C language. We have discussed the various algorithms used for image thresholding, including the global thresholding and the local thresholding algorithms. We have also discussed how to implement the algorithms in C language, with detailed explanations of the code.

In conclusion, image thresholding is an important image processing technique that can be used to reduce noise and enhance the visibility of an image. The C language is an ideal language for implementing these algorithms, as it is a powerful and flexible language with extensive libraries and capabilities. We have explored how to implement image thresholding in C language, with detailed explanations of the code and the various algorithms. We hope that this case study has been helpful in understanding how image thresholding can be implemented in C language.

1.7.Referrences

1. Chat GPT (For Definitions)

2. Open AI Playground (For line wise code explanations)

3.DALL-E(For BMP and PPM images).

END-SEM PROJECT