Tecnológico de Monterrey

Act 2.1

Periodo Semestral Agosto-Diciembre 2022
Fecha de entrega |*Septiembre 29, 2022*

Multiprocesadores TE3061.1

Profesor:

Emmanuel Torres Rios

Abril Jimenez Alatriste | *A01732412*

Instituto Tecnológico de Estudios Superiores de Monterrey

Intruccion

Desarrollar un programa en c que a través de threads permita cambiar una imagen a color en escala de grises además de a)invertir con respecto a la horizontal y b) vertical de la imagen. Como evidencia de la actividad deberá entregar un reporte de los detalles de su experimento para una imagen .bmp normal (700x700) y gran formato (más de 2000 pixeles en ancho o alto). Use malloc para mejorar el tiempo de ejecución de su programa

Introduccion

Dentro del procesamiento de imagenes, existe algo a lo cual se le conoce como conversión de espacio de color, un ejemplo es el pasar una imagen de RGB o bien una imagen a “colores” a una imagen Grayscale o bien una imagen en escala de grises, existen varios tipos de metodos para esto entre los cuales se encuentran los siguientes, tomando el valor promedio de R, G y B como el valor de la escala de grises, La razón es que los globos oculares humanos reaccionan de manera diferente a RGB. Los ojos son más sensibles a la luz verde, menos sensibles a la luz roja y menos sensibles a la luz azul. Por lo tanto, los tres colores deben tener diferentes pesos en la distribución. Eso nos lleva al método ponderado o bien El método ponderado, también llamado método de luminosidad, pesa el rojo, el verde y el azul según sus longitudes de onda. La fórmula mejorada es la siguiente.

$$Grayscale = 0.299R + 0.587G + 0.114B$$

Quedando asi algo similar a esto en el lenguaje C:

```
FILE* fp = fopen("/path/to/your/image.bmp", "rb");

fread(imageHeader, sizeof(unsigned char), 54, fp);

int width = *(int*)&imageHeader[18];

int height = *(int*)&imageHeader[22];

int bitDepth = *(int*)&imageHeader[28];

if(bitDepth <= 8)

read(colorTable, sizeof(unsigned char), 1024, fp);

FILE *fo = fopen("/path/to/output/image.bmp", "wb");
```

```
fwrite(imageHeader, sizeof(unsigned char), 54, fo);

int imgDataSize = width * height;

unsigned char buffer[imgDataSize][3];

    buffer[i][2] = getc(fp);

    buffer[i][1] = getc(fp);

    buffer[i][0] = getc(fp);

if(bitDepth <= 8)

        fwrite(colorTable, sizeof(unsigned char), 1024,

y   =   (buffer[i][0]*0.3)   +   (buffer[i][1]*0.59)   +   (buffer[i][2]*0.11);
//conversion formula of rgb to gray

    putc(y,fo);

    putc(y,fo);

    putc(y,fo);}

fclose(fo);

fclose(fp);
```

Una vez esto entendido, debemos preguntarnos, ¿Qué es invertir en el procesamiento de imágenes? Esta se definiria como el color de una imagen o video, quedando similar a esto en codigo c.

```
void mirror(int height, int width, RGBTRIPLE image[height][width]) {
 for( int j = 0; j < height; j++) {
j for(int i = 0; i < width/2; i++) {
RGBTRIPLE temp = image[j][width-1-i];
 temp = right side pixel image[j][width-1-i] = image[j][i];
 right side pixel = left side pixel image[j][i] = temp;
```

```
}
}
}
```

Por ultimo debemos entender que es es cambiar el vertical de una imagen, se definiria como "voltear" o "reflejar" una imagen en la dirección vertical (arriba-abajo) y se obtendria un codigo similar a el siguiente el lenguaje c.

```
int flip (PIXEL *original, PIXEL **new, int rows, int cols)

{

int row, col;

if ((rows <= 0) || (cols <= 0)) return -1;

*new = (PIXEL*)malloc(rows*cols*sizeof(PIXEL));

for (row=0; row < rows; row++)

for (col=0; col < cols; col++) {

PIXEL* o = original + row*cols + col;

PIXEL* n = (*new) + (rows-1-row)*cols + col;

*n = *o;

}

return 0;

}
```

Marco Teórico.

Caso 1

Iniciando por intentar correr una imagen con dimensiones equivocadas o bien muy “pequeñas” nos deberia mandar el siguiente mensaje junto con sus dimensiones o bien que sean dos diferentes imagenes las que se intenten abrir.

```
(base) PS C:\Users\Yukin\Desktop\multi\GS2H2V2.1> .\a.exe
largo img 16777215
ancho img 16777215
Error
```

Caso 2

Al usar imagenes que correctamente, cumplean con las especificaciones dadas por las siguientes lineas de codigo.

```
ancho = (long)cdr[20] * 65536 + (long)cdr[19] * 256 + (long)cdr[18];

alto = (long)cdr[24] * 65536 + (long)cdr[23] * 256 + (long)cdr[22];
```

donde podremos ver las dimensiones generales dadas, esto de igual manera es algo de lo que nos basamos del codigo original dado por el profesor.

Finalmente se obtendra lo siguiente:

```
(base) PS C:\Users\Yukin\Desktop\multi\GS2H2V2.1> gcc -fopenmp prueba.c
(base) PS C:\Users\Yukin\Desktop\multi\GS2H2V2.1> .\a.exe
largo img 16776136
ancho img 1080
```

Esto nos indicara que la conversion fue correcta pues no nos arrojara un “ERROR”.

Ahora bien de salida deberemos obtener 3 diferentes images bmp con el nombre predefinido en el programa en lenguaje c, como se muestra a continuacion.

```
void main()

{

FILE *image, *outputImage_GrayScale, *outputImage_Flip_vertical,
```

```
*outputImage_Flip_horizontal;

image = fopen("2.bmp", "rb");

outputImage_GrayScale = fopen("2Gris.bmp", "wb");

outputImage_Flip_vertical = fopen("2.2v.bmp", "wb");

outputImage_Flip_horizontal = fopen("2.3hbmp", "wb");
```

Siendo el resultado el siguiente.

De la imagen original “2.bmp”

Imagen 1.1 “2.bmp”

La primera imagen de salida sera la la imagen en escala de grises, seguida por las imagenes en vertical y horizontal.

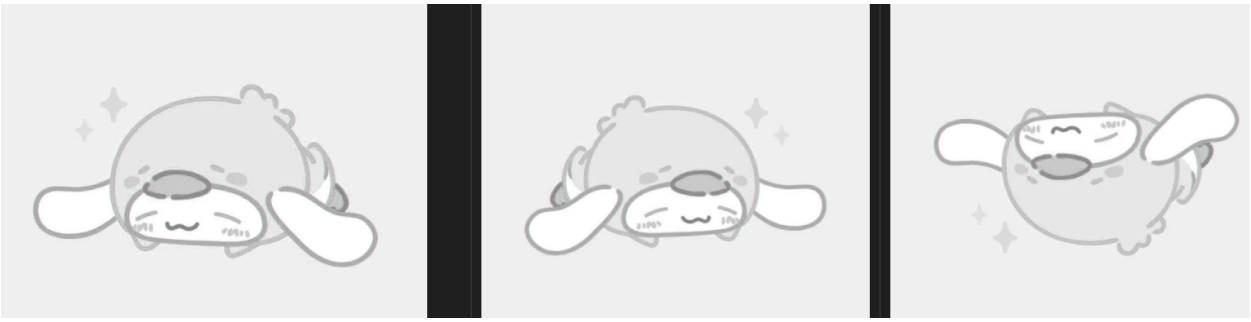

Imagen 1.2 “imagenes resultados”

Conclusión

Finalmente de esta actividad se logra concluir varias cosas, primeramente que existian varias maneras de solucion, y que estas, podian variar desde tomar de base el programa dado por el profesor o bien optar por estructuras, ambas muy buenas opciones, y muy validas, consiguientemente se logra concluir que para lo que luciria como una simple actividad, tal cual voltear imagenes, se pueden presentar muchas dificultades como las siguientes imagenes adjuntas que fueron pruebas antes de lograr el programa final, esto me llevo a darme cuenta que entre mas threads usara menor seria la calidad, asi como ademas si usaba una imagen de dimensiones exageradamente minusculas esta trataria de “llenar dichas dimenciones”.

Apendice.

Imagen 1.3 imagen de prueba “vango.bmp”

Imagen 1.4 imagen de prueba “vangoGris.bmp”

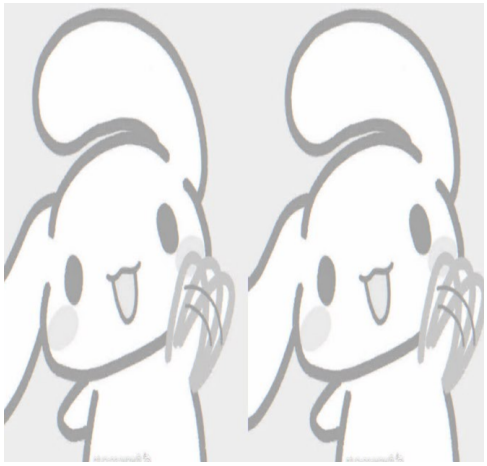

Imagen 1.5 imagen de prueba “vangoR1.bmp”

Imagen 1.5 imagen de prueba “vangoR2.bmp”