1.- Introducción

Un área donde es importante lograr buen desempeño del software es en el área de procesamiento de imágenes. En este trabajo práctico se implementarán algoritmos básicos de manipulación de imágenes utilizando assembler embebido en código C.

El objetivo consiste en tomar una imagen a color, pasarla a tonos de gris y manipularla geométricamente aplicando las operaciones de rotación desplazamiento y estiramiento/compresión.

Para la realización del trabajo tanto la lectura de la imagen como la proyección en pantalla de las imágenes se implementarán en lenguaje C utilizando la librería OpenCV.. Los cálculos necesarios para implementar las manipulaciones se harán en lenguaje assembler x86_64. Todo el código fuente (C + Assembler) debe compilar y generar un archivo ejecutable que recibe como parámetro una imagen color en formato JPG y muestra en pantalla la imagen original y la imagen modificada.

1.1.- Conversión color a tonos de gris

Una forma de hacer esta conversión es tomar los componentes RGB de la imagen y mapearlos a un tono de gris equivalente, con lo cual la imagen pasa de tener 3 canales de color RGB a tener un solo canal de escala de gris (que va de negro a blanco). En este trabajo se pide mantener los 3 canales de color RGB, por lo que se debe lograr una conversión de los 3 canales RGB de la imagen a color a 3 canales RGB de la imagen en tonos de gris (sigue siendo una imagen a color, pero los colores forman tonos de gris).

Ejemplo

	
Original	Tono de Gris

La conversión de canales RGB color a canales RGB tono de gris comprende 3 etapas: debido a que el ojo humano percibe la intensidad de color en forma nolineal, los canales RGB están comprimidos en forma no lineal y se debe obtener los valores lineales mediante una transformación no lineal.

Entones la primera etapa es obtener las tres componentes lineales (R_{linear}, G_{linear}, B_{linear}) a partir de los valores comprimidos sRGB de la imagen según la siguiente transformación:

$$C_{\text{linear}} = \begin{cases} \frac{C_{\text{srgb}}}{12.92}, & \text{if } C_{\text{srgb}} \leq 0.04045 \\ \left(\frac{C_{\text{srgb}}+0.055}{1.055}\right)^{2.4}, & \text{otherwise} \end{cases}$$

Donde C_{srgb} es una componente (R, G, o B) obtenida del archivo de imagen y C_{linear} es la componente lineal equivalente. Las componentes RGB están mapeadas en un rango [0…1], mientras que en un archivo de imagen en general están mapeadas en el rango [0…255], por lo que es necesario dividir por 255 antes de aplicar la transformación.

La segunda etapa es obtener la luminancia lineal Y_{linear} equivalente a la combinación de los 3 canales RGB lineales. Este color equivalente en tonos de gris está relacionado con la luminancia, de manera que se puede obtener la luminancia a partir de las componentes RGB lineales mediante la siguiente ecuación

$$Y_{\text{linear}} = 0.2126 R_{\text{linear}} + 0.7152 G_{\text{linear}} + 0.0722 B_{\text{linear}}.$$

Esta intensidad o luminancia debe ser comprimida (debido a la percepción no lineal del ojo humano) mediante la transformación inversa a la usada para obtener los componentes lineales. La tercera etapa consiste en obtener la luminancia comprimida Y_{srgb} a partir de la luminancia lineal Y_{linear} de acuerdo a la transformación inversa:

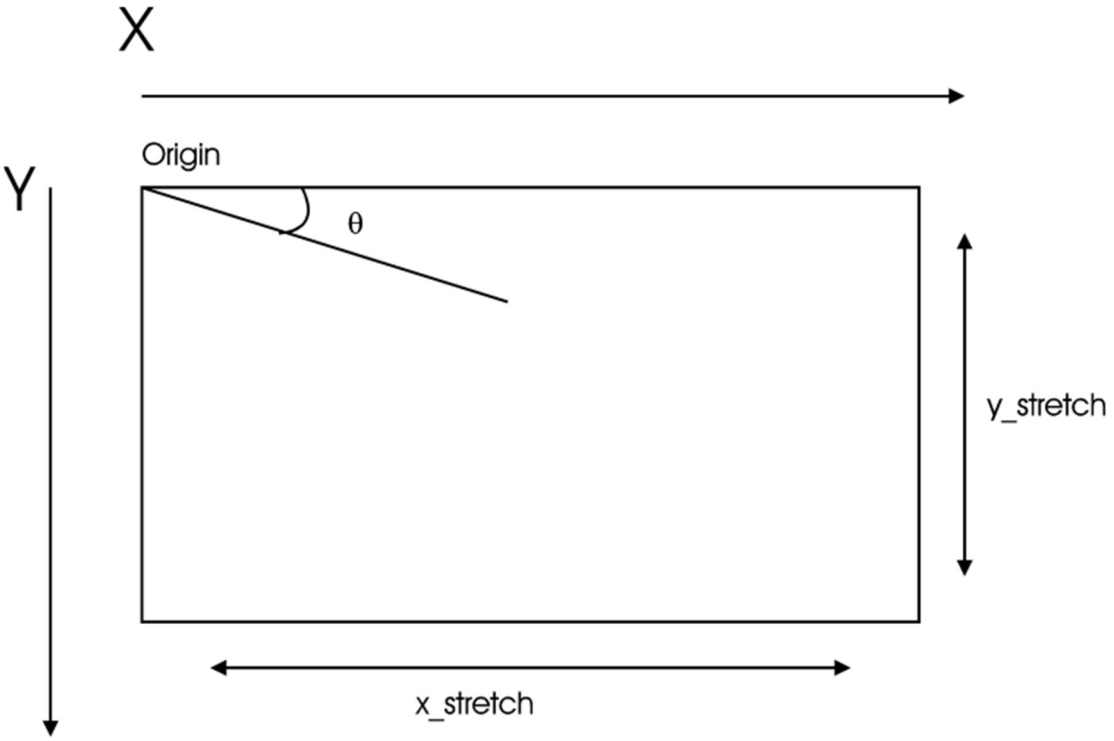
$$Y_{\text{srgb}} = \begin{cases} 12.92\, Y_{\text{linear}}, & \text{if } Y_{\text{linear}} \leq 0.0031308 \\ 1.055\, Y_{\text{linear}}^{1/2.4} - 0.055, & \text{otherwise} \end{cases}$$

Esta luminancia comprimida pasa a ser el nuevo valor de los canales RGB (el mismo valor de Y_{srgb} para cada componente R, G y B), obteniéndose una imagen en tonos de gris

(Fuente:
https://en.wikipedia.org/wiki/Grayscale#Converting_color_to_grayscale)

1.2.- Rotación de imagenes

Las tres operaciones geométricas que se pueden aplicar a una imagen son desplazamiento, rotación y compresión/expansión.



La rotación se indica con un ángulo, el desplazamiento es un valor positivo o negativo en los ejes X e Y, y la compresión/expansión es una constante que alargo o comprime la imagen a lo largo de los ejes X e Y.

Estas tres operaciones mapean la nueva ubicación de un pixel en posición x-y a una nueva posición X-Y de acuerdo con las siguientes ecuaciones:

$$X = x \cdot cos(\theta) + y \cdot sin(\theta) + x_{displace} + x \cdot x_{stretch}$$

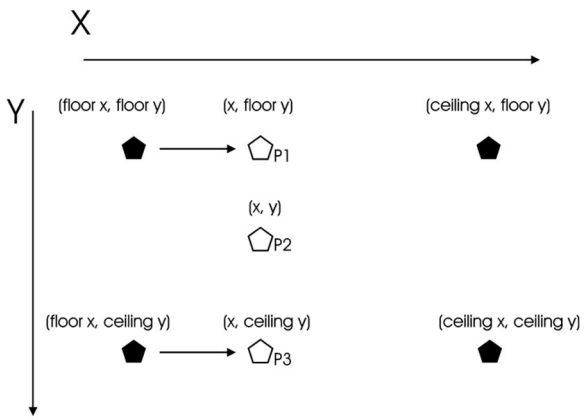
$$Y = y \cdot cos(\theta) - x \cdot sin(\theta) + y_{displace} + y \cdot y_{stretch}$$

En estas ecuaciones Θ es el ángulo de rotación, $xy_{displace}$ es un desplazamiento de la imagen (positivo o negativo) y $xy_{stretch}$ es un factor de compresión/expansión (valores menores a 1 comprimen y valores mayores a 1 expanden la imagen).

1.3.- Interpolación bilineal

Al aplicar las operaciones geométricas surgen espacios vacíos entre las nuevas ubicaciones de los pixels. Para calcular los valores RGB de estos espacios vacíos se utiliza la interpolación bilineal. El nombre bilineal surge porque se calcula una aproximación lineal entre dos valores de pixel conocidos tanto en el eje X como en el eje Y.
La interpolación bilineal calcula un valor promedio de la siguiente manera: supongamos un punto P2 ubicado en algún lugar intermedio entre 4 pixels de

valor conocido. La idea es calcular un valor de pixel promedio sobre el eje X (P1), un valor de pixel promedio sobre el eje Y (P3), y con esos valores calcular un valor de pixel promedio en P2



Las ecuaciones para realizar estos cálculos en una imagen en escala de grises son:

$$gray(P1) = (1-x) \cdot gray(floor(x), floor(y)) + x \cdot gray(ceiling(x), floor(y)) \quad (13.5)$$

$$gray(P2) = (1-x) \cdot gray(floor(x), ceiling(y)) + x \cdot gray(ceiling(x), ceiling(y)) \quad (13.6)$$

$$gray(P3) = (1-y) \cdot gray(P1) + y \cdot gray(P2) \quad (13.7)$$

Para el caso de imágenes a color, se debe hacer una interpolación bilineal sobre cada componente (RGB)

(Fuente: Image processing in C, 2nd Edition. Phillips)

2.- Conformación de Grupos:

Los Grupos con **1 o 2 integrantes** deben Implementar la conversión de una imagen color a tonos de gris de acuerdo con lo descripto en el enunciado, es decir, deben implementar el ítem **1.1.-**

Los equipos de **3 integrantes** deben realizar la misma consigna que los equipos de 1 o 2 integrantes con el agregado de la implementación de un algoritmo de rotación de la imagen en un ángulo arbitrario que se recibe como parámetro de entrada junto con el archivo de imagen; sin realizar interpolación de los espacios vacíos, por lo que deben implementar los ítems **1.1.- y 1.2.-**

Los equipos de **4 integrantes** deben realizar la misma consigna que los equipos de 3 integrantes con el agregado de la implementación de una interpolación bilineal para los espacios vacíos de la imagen rotada, implementando los ítems **1.1.-; 1.2.- y 1.3.-**

3.- Observaciones

Los cálculos para obtener

- Las tres componentes lineales (R_{linear}, G_{linear}, B_{linear}),
- La luminancia lineal Y_{linear}
- La luminancia comprimida Y_{srgb}
- La nueva ubicación de un pixel
- El valor interpolado de los pixeles de los espacios vacíos

Deben implementarse como funciones externas en assembler x86_64 invocadas desde el código C. Dados los valores numéricos de las transformaciones, se debe usar precisión doble para todos los cálculos.

Las funciones implementadas deben ser código original generado por los Grupos (no obtención desde C, no librerías externas, no código generado por IA, etc). Es aceptable utilizar funciones de C declaradas en <math.h> si se las invoca desde el código assembler.

4.- Implementación de Referencia

Como referencia se adjunta una implementación completamente en C de la consigna para Grupos de 1 o 2 alumnos. En el trabajo práctico se debe implementar en assembler x86_64 las funciones *procesarImagen()*; *valorRGBlineal()* y *valorYcomprimido()*.

Para compilar la implementación de referencia se debe crear un subdirectorio con los archivos:

- CMakeLists.txt
- Casa.jpg
- DisplayImage.cpp

Y ejecutar la siguiente secuencia de comandos:

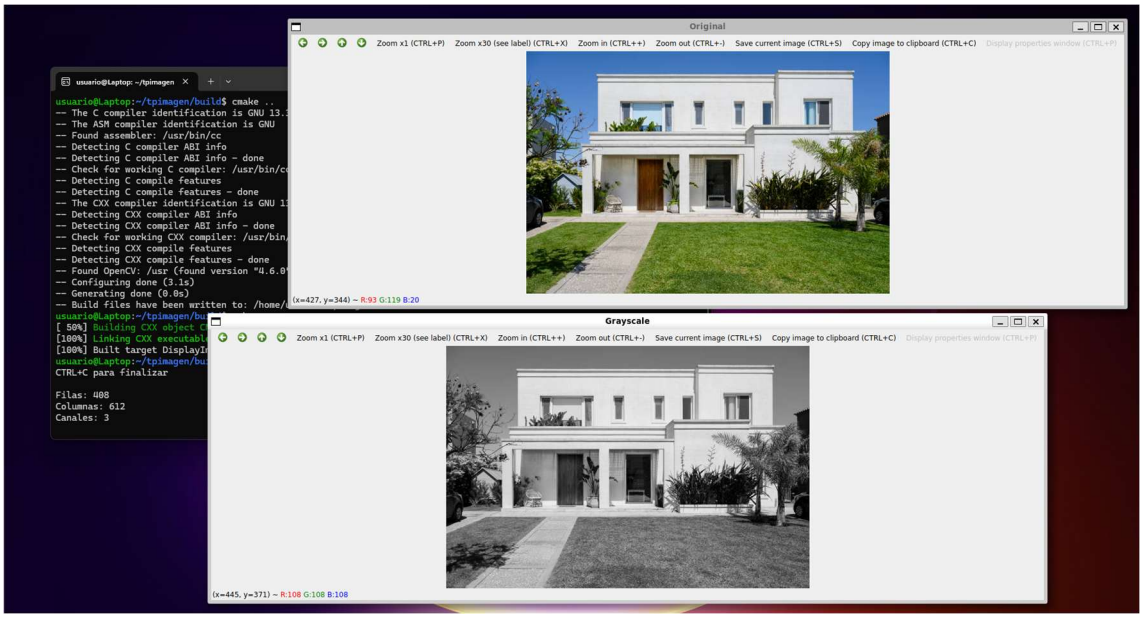
```
mkdir -p build
cd build
cmake ..
make
```

Esta secuencia de instrucciones crea un subdirectorio “build” que va a tener todos los productos de la compilación, incluido el archivo ejecutable. De esta manera se separan los archivos fuente de los productos de la compilación y se puede borrar estos últimos con el comando “rm -r ./build” (ejecutado desde el directorio con el código fuente) antes de hacer una nueva compilación.

Una vez compilado el proyecto, verificar su funcionamiento ejecutando la aplicación:

```
usuario@Laptop:~/tpimagen/build$ ./DisplayImage ../Casa.jpg
```

Deberán verse dos imágenes en pantalla: la original y la versión en escala de grises



De esta manera se verifica la correcta compilación de la implementación de referencia.