Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Tarea 0

IIC2133 - Estructuras de datos y algoritmos

Primer semestre, 2017

Entrega: Martes 21 de marzo

Objetivos

- Familiarizar al alumno con C
- Implementar y analizar un algoritmo simple

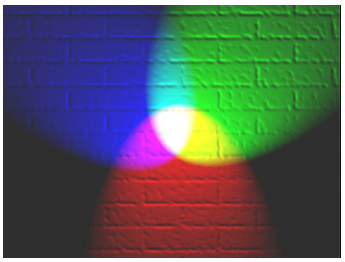
Introducción

La compañía de fotográfica *Instakilogram* ha solicitado tu ayuda para la implementación de su nuevo filtro fotográfico `A E S T H E T I C`. Los usuarios de *Instakilogram* cada día están más creativos y quieren estar menos limitados por las opciones de filtro que la aplicación tiene para ofrecerles. Por ende, la compañía decidió hacer un filtro que sea modificable por el usuario. Pero primero, te entregarán la base de conocimientos que necesitarás para entender el problema.

Colores e Imágenes

Las imágenes son básicamente una matriz de colores, y estos a su vez pueden ser representados de múltiples maneras. En el caso de esta tarea se usará la forma estándar de representarlos: RGB.

RGB es un modelo aditivo de color en el cual los tres colores primarios de la luz, Rojo (R), Verde (G) y Azul (B) se combinan en diferentes intensidades para formar los distintos colores que podemos ver.



Las luces de distintos colores se combinan para formar blanco.

Esta representación es la estándar ya que es la que deben seguir los monitores para emitir la luz de distintos colores. Siguiendo estas reglas, cada color en la imagen es básicamente un vector de 3 componentes: R, G y B, llamados "canales". Dado que el valor de cada uno representa la intensidad, este va de 0 a 1.

A E S T H E T I C

El filtro consiste en lo siguiente: para cada píxel de la imagen original, el valor del píxel correspondiente en la imagen resultante corresponde a una ponderación entre el píxel y sus vecinos. Esta se puede especificar fácilmente con una matriz. Por ejemplo, para el píxel x en cualquier posición de la imagen, y su píxel correspondiente x' de la imagen resultante:

$$\begin{bmatrix} \ddots & \vdots & \vdots & \vdots & \iddots \\ \dots & a & b & c & \dots \\ \dots & d & x & e & \dots \\ \dots & f & g & h & \dots \\ \iddots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} * \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \implies x' = -b - d + 5x - e - g$$

Es decir, centrar la matriz en el píxel determinado, multiplicar cada vecino y el píxel por el valor correspondiente en la matriz y sumarlos. Dado que cada píxel es un vector de 3 componentes, se pueden aplicar las operaciones normales de suma de vectores. Esta operación se conoce como *convolución* de matrices, y no es lo mismo que la multiplicación tradicional de matrices. Bajo este contexto, a la matriz ponderadora se le conoce como *kernel*. Por ejemplo, aplicando el kernel anterior sobre una imagen real obtenemos lo siguiente:

Imagen original

Imagen luego de aplicar el filtro

Pero, ¿Qué hacer cuando el píxel a calcular queda cercano al borde de la imagen? Después de todo, no hay vecinos hacia allá afuera para ser ponderados. Para no perder información, lo que se hace es extender el valor de los bordes de la imagen hacia afuera de esta. Visto desde otro ángulo, si se necesita un píxel que queda fuera de la imagen, se usa en su lugar el píxel más cercano que forma parte de la imagen. De forma visual:

Este filtro no se limita a matrices de 3×3, en realidad se puede realizar con matrices de $m \times n$, siempre y cuando m y n sean números impares (ya que debe haber un píxel central).

Según los coeficientes de la matriz es posible que algún canal en algún píxel quede fuera del rango [0,1] definido para los colores. Esto es una consecuencia normal del algoritmo y **no debes ajustar estos datos al intervalo**: ya serán normalizados a la hora de guardar la imagen.

Tu objetivo es implementar este filtro en C tomando como parámetros una imagen y un kernel.

Librería

Para todo el manejo de imágenes tus ayudantes han preparado una librería. Puedes revisar lo que hacen sus funciones en `Programa/src/imagelib/imagelib.h` en tu repositorio. Esta librería solo es capaz de manejar imágenes en formato PNG.

Análisis

Deberás entregar un informe donde analices la complejidad del algoritmo, tanto en tiempo de ejecución como en uso de memoria.

Input

Tu programa deberá recibir los siguientes parámetros:

1. La ruta de la imagen que se desea filtrar
2. La ruta del archivo que describe el kernel
3. La ruta de la imagen resultante

de la siguiente manera:

```
./filter lena.png kernel.txt lenafiltrada.png
```

El archivo del kernel sigue el siguiente formato:

Las primeras dos líneas contienen la cantidad de filas r y columnas c de la matriz. Las siguientes r líneas contienen c números decimales cada una. Por ejemplo:

```
3
5
0.125 0.250 0.500 0.250 0.125
0.500 0.750 1.000 0.750 0.500
0.125 0.250 0.500 0.250 0.125
```

Output

El output de tu programa es la imagen resultante, la cual deberás guardar en la ruta que se te ha otorgado.

Evaluación

La nota de tu tarea está descompuesta en distintas partes:

- 50 % a que las imágenes generadas sean correctas
- 20 % a la calidad de tu código. Específicamente,
  - 10 % a que *valgrind* indique que no tienes **errores**
  - 10 % a que *valgrind* indique que no tienes **memory leaks**
- 30 % a tu análisis de complejidad. Específicamente,
  - 20 % a que el análisis en sí sea correcto
  - 10 % a que la complejidad sea la esperada

Si bien para esta tarea no interesa la velocidad de tu programa, tampoco podemos permitir que te demores mucho. Tu programa será cortado luego de 5 segundos. El puntaje por uso de memoria solo aplica si el output de tu programa es correcto.

Entrega

Deberás entregar tu tarea en el repositorio que se te será asignado; asegúrate de seguir la estructura inicial de éste.

Se espera que tu código compile con `make` dentro de la carpeta **Programa** y genere un ejecutable de nombre `filter` en esa misma carpeta.

Se espera que dentro de la carpeta **Informe** entregues tu informe en formato *PDF*, con el nombre *Informe.pdf*

Cada regla tiene asociado un puntaje, asegúrate de cumplirlas para no perder puntos.

Se recogerá el estado de la rama `master` de tu repositorio, 1 minuto pasadas las 23:59 horas del día de entrega. Recuerda dejar ahí la versión final de tu tarea. No se permitirá entregar tareas atrasadas.

Bonus

Como regla general, un bonus a tu nota se aplica solo si la nota correspondiente es $\geq 3{,}95$. A continuación, las distintas formas de aumentar tu nota.

Buen uso del espacio y del formato (+5 % a la nota de *Informe*)

La nota de tu informe aumentará en un 5 % si tu informe, a criterio del corrector, está bien presentado y usa el espacio y formato a favor de entregar la información.