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

Tarea 2

IIC2133 - Estructuras de datos y algoritmos

Segundo semestre, 2016

Entrega: Martes 4 de Octubre

Objetivos

- Investigar sobre una estructura de datos y sus usos
- Modificar algoritmos vistos en clases para un propósito específico
- Familiarizar al alumno con conceptos de procesamiento de imágenes

Introducción

La compañía de videojuegos *Mintendo* acaba de anunciar su nueva consola, la *Mintendo MX*. Después de todas las críticas respecto al rendimiento de su consola anterior, para esta consola han decidido aumentar el rendimiento al reducir la capacidad gráfica de la consola: ahora solo se permitirá una cantidad limitada de colores en pantalla, trabajando con paletas de colores como se hacía en los 90.

Usted forma parte de una compañía independiente que estaba haciendo un juego para esta última generación, y se viene a enterar ahora que todas las gráficas HD 1080p que tenía preparadas para su juego ahora son inviables, por lo que debe convertir todo a una paleta limitada de n colores según se le solicite.

*CIE L*a*b** color space

El espacio de colores RGB usado típicamente tiene la conveniencia de que modela directamente el comportamiento de la luz, pero puede resultar impráctico para otros propósitos, en especial cuando se quiere hacer algo con rangos de colores.

Cuando se trata de procesamiento de imágenes, se suele necesitar poder encontrar la "distancia" entre dos colores. Podemos considerarlos como vectores de 3 dimensiones y simplemente calcular la distancia euclidiana entre ambos. El problema que surge de esto es que esta distancia no se condice con la percepción humana. Por ejemplo, en RGB el rojo está a la misma distancia del verde que el verde del azul, lo cual perceptualmente no es cierto.

Calcular la distancia entre dos colores analíticamente es un proceso complicado, pero existe un espacio de color que aproxima bastante bien esta distancia al usar simplemente la distancia euclidiana, llamado *CIE L*a*b** o simplemente *Lab*. Usando este espacio de color se pueden obtener mejores resultados, manteniendo la simplicidad algorítmica de calcular la distancia.

Esto significa que cada color ahora esta representado por una 3-tupla:

- L: luminosidad, de 0 a 100
- a: cromaticidad verde-magenta, de -128 a 128
- b: cromaticidad azul-amarillo, de -128 a 128

Este espacio modela la percepción humana, pero contiene muchos más colores que los que podemos percibir: no todas las combinaciones de L,a,b son colores RGB válidos.

Reducción de colores

Tu objetivo es para dado un numero n y una imagen I de ancho w alto h buscar una imagen R con tal de minimizar el error E

$$E = \sum_{i=0}^{w-1} \sum_{j=0}^{h-1} d(I_{ij}, R_{ij})$$

donde X_{ij} es el píxel (i, j) de la imagen X, y $d(u, v)$ es la distancia euclidiana entre los colores u y v en espacio Lab. R debe tener como máximo n colores distintos entre ellos.

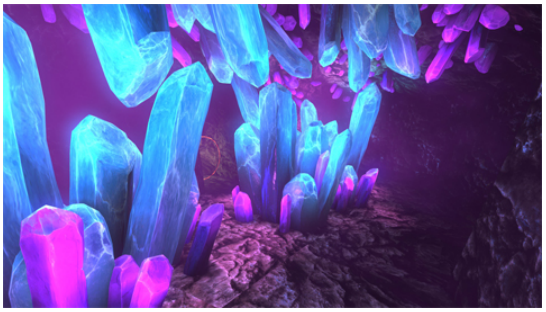

Imagen original

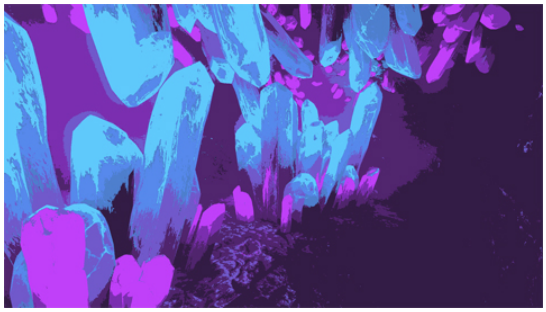

Imagen con sólo 8 colores

Paleta de colores

Para hacer esto lo primero que debe hacerse es encontrar la *paleta óptima de colores*, es decir el conjunto P de n o menos colores tal que al reemplazar cada color en la imagen por su color más cercano en P, E sea mínimo.

Así, podemos expresar cada color en la imagen por el índice del color que le corresponde en P, usando de manera efectiva menos memoria para representar la imagen. Esto significa que si P tiene n elementos, cada color en la imagen ahora necesita $log_2(n)$ bits para ser almacenado. Dado que sólo se pueden usar cantidades enteras de bits, solo tiene sentido usar $n = 2^k, k \in \mathbb{N}$, por lo que se considerará k como el parámetro del problema.

Encontrar la paleta óptima no es trivial, pero existen algoritmos capaces de aproximarla muy bien. En este caso, se espera que uses el algoritmo de *clustering* llamado *Median Cut*, el cual agrupa los colores similares dividiendo sucesivemente según la mediana en alguno de los ejes L,a,b.

Búsqueda de colores

Una vez obtenido P, debes recorrer la imagen entera, y para cada píxel en ella, reemplazarlo por el color más cercano en P. Si nuestra imagen tiene $m = h \times w$ píxeles esta operación es $\mathcal{O}(m \times n)$ dado que por cada pixel debes recorrer P entero viendo cual es el más cercano.

Para mejorar esto, debería construirse una estructura de datos sobre P de manera de llevar a cabo el proceso de búsqueda del color en $\mathcal{O}(log(n))$ para cada píxel de la imagen.

Se espera que la estructura que construyas sea un *KD-Tree* de puntos en 3 dimensiones para encontrar los vecinos más cercanos.

Fusión de algoritmos

El algoritmo de búsqueda de la paleta y el de construcción del árbol son muy similares, por lo que sería útil si trataras de condensarlos en un solo proceso. Para encontrar la mediana se recomienda usar *Quickselect* como se vió en clases.

Librería y código base

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. Actualmente la librería solo es capaz de manejar imágenes en formato PNG.

Análisis

Deberás entregar un informe donde analices teóricamente el problema. Se espera lo siguiente:

- Analizar la complejidad total del proceso usando *median cut* y *kd-tree*

Además deberás incluir un análisis empírico de los resultados de tu programa. En particular se espera lo siguiente:

- Analizar la variación del tiempo de tu proceso según el k y ver como se relaciona con la complejidad teórica calculada anteriormente.
- Analizar la variación del tiempo de tu proceso según las dimensiones de la imagen, y ver como se relaciona con la complejidad teórica calculada anteriormente.

Input

Tu misión es completar la función `void reduce(Image_Lab* image, int k)` en la carpeta `Programa/src/alumno`, la cual es llamada por el main ubicado en `Programa/src/alumno` el cual no deberás modificar, ya que no se corregirá con ese.

El main se compila a un ejecutable llamado `reduce` que recibe los siguientes parámetros

1. La ruta de la imagen que se desea reducir
2. La ruta de la imagen resultante
3. k

de la siguiente manera:

```
./reduce crystals.png crystals_8.png 3
```

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:

- 60 % a que el error de tu imagen esté suficientemente cerca del esperado
- 20 % a tu análisis de complejidad teórico.
- 20 % a tu análisis de complejidad empírico. Específicamente:
  - 10 % al análisis según k
  - 10 % al análisis según dimensiones de la imagen

Si la imagen que entregas tiene más colores que los especificados tendrás automáticamente 0 puntos para ese test. Si tu algoritmo demora más de 10 segundos en un test, será cortado y tendrás 0 puntos en ese test.

Entrega

Deberás entregar tu tarea en el repositorio que se te será asignado; asegurate 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 `reduce` en esa misma carpeta

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

Por cada regla que no cumplas se te aplicará un descuento porcentual a tu nota final.

Se recogerá el estado de la rama `master` de tu repositorio, 1 minuto pasadas las 24 horas del día de entrega. Recuerda dejar ahí la versión final de tu tarea. No se permitirá entregar tareas atrasadas.

Bonus

A continuación, formas de aumentar la nota obtenida en tu tarea. Estos aplicarán solo si la nota involucrada es mayor o igual a 4.

Manejo de memoria perfecto (+5 % a la nota de *Código*)

Se aplicará este bonus si valgrind reporta en tu código 0 leaks y 0 errores de memoria, considerando que tu programa haga lo que tiene que hacer.

Buen uso del lenguaje y la estructura (+5 % a la nota de *Código*)

Se aplicará este bonus si tu código es ordenado, bien estructurado y utilizas el estilo de programación de C. A criterio del corrector.

Ortografía y gramática perfecta (+5 % a la nota de *Informe*)

Se aplicará este bonus solo si tu ortografía y gramática en el informe son impecables, considerando que tengas una cantidad razonable de texto.

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 está bien presentado y usa el espacio y formato a favor de entregar la información. A juicio del corrector.

Anexo: Más ejemplos de Reducción

Imagen original

Imagen con sólo 16 colores

Imagen original

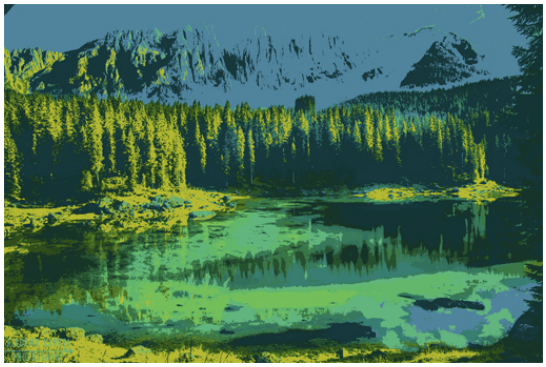

Imagen con sólo 16 colores

Imagen original

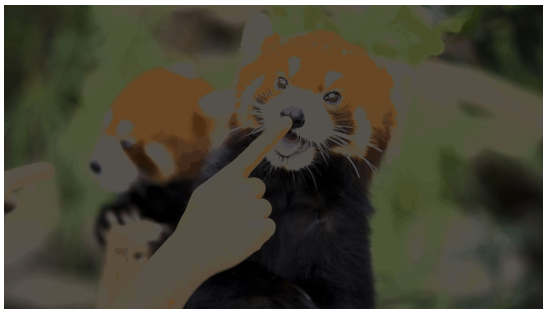

Imagen con sólo 4096 colores