UNIVERSIDAD NACIONAL AUTONOMA DE MÉXICO

Tarea 02 - Proces digital de imágenes

Gerardo Daniel Martínez Trujillo
311314348

7 de septiembre de 2019

1. DESCRIPCIÓN DEL PROBLEMA

Se quiere programar una aplicación que aplique filtros a imágenes. La aplicación proveerá una interfaz gráfica desde la cual poder aplicar los filtros.
El formato de las imágenes proveídas son de formato jpeg.

Los filtros que se necesitan son: filtro rojo, filtro azul, filtro verde y filtro de mosaico.

Para el filtro de mosaico se debe de poder escoger el largo de la región a la cual se le aplicará el filtro desde una ventana secundaria.

2. ANÁLISIS DEL PROBLEMA

Para poder hacer los filtros necesitamos alguna manera de acceder al valorde los pixeles de la imagen. Para la mayoría de los lenguajes de programación existe alguna biblioteca con la capacidad de trabajar con pixeles. Por lo tanto la elección de este estará en función de la eficiencia y facilidad con la que realiza este trabajo.

Por la definición del problema parece indicado utilizar orientación a objetos para modelar una clase filtro donde la imagen sea un atributo y los metodos realizen transformaciones sobre ésta. También, para evitar la perdida de información de la imagen de original la clase filtro devería tener alguna especie de caché donde guardar los datos iniciales.

De nuevo, la mayoría de los lenguajes cuenta con una forma de trabajar con interfaz gráfica y este probleme se reduce a escoger aquel donde sea más fácil crearla.

3. Selección de la mejor alternativa

Para solucionar el problema usaremos el lenguaje de programación go, debido que se biblioteca standar proveé la posibilidad de trabajar con pixeles. En la mayoría de los lenguages que se revisaron se necesitaba alguna biblioteca adicional.

Para la interfaz gráfica usaremos un binding de gtk para go que se llama go-gtk. Por desgracia esta biblioteca no está terminada y no está muy bien documentada, pero nos termite usar gtk de manera fácil.

Go no tiene como tal una abstracción de clase o objeto. Pero es posible utilizar ciertos conceptos de orientación a objetos. Es posible asociar un tipo de dato (o un conjunto de datos) a una serie de funciones y hacer subtipos de estos. Por lo tanto, go ofrece características suficientes para los propósitos de este proyecto.

4. Pseudocódigo

Algoritmo .1: Filtro de color

Data: Matriz(M) de tamaño $n \times m$ con entradas de la forma (r,g,b), donde cada r,g,b es un entero entre 0 y 255

Data: flotantes x,y,z entre 0 y 1

Result: Matriz de tamaño $n \times m$ con entradas de la forma (r,g,b), donde cada r,g,b es un enteros entre 0 y 255

1 **for** *x=0 hasta n* **do**
2 **for** *y=0 hasta m* **do**
3 $M_{xy}.r \leftarrow M_{xy}.r \cdot x$;
4 $M_{xy}.g \leftarrow M_{xy}.g \cdot y$;
5 $M_{xy}.b \leftarrow M_{xy}.b \cdot z$;
6 Regresa M;

Aclacaciones

- datos de entrada x,y,z

  Este es un algoritmo generalizado para hacer filtros de colores. Para el filtro rojo x,y,z serían iguales a 1,0,0 respectivamente; para el verde, 0,1,0 y para el azul, 0,0,1

Algoritmo .2: Filtro de mosaico

Data: Matriz(M) de tamaño $n \times m$ con entradas de la forma (r,g,b), donde cada r,g,b, es un enteros entre 0 y 255

Data: entero s donde, s.p.g. si $n \leq m$, entonces $0 \leq s \leq m$, además $s|n$ y $s|m$

Result: Matriz de tamaño $n \times m$ con entradas de la forma (r,g,b) donde cada r,g,b es un entero entre 0 y 255

```
1  var sumR,sumG, sumB;
2  for x=0 hasta n con saltos de s do
3      for y=0 hasta m con saltos de s do
4          sumR=0, sumG=0, sumB=0;
5          for a=0 hasta s do
6              for b=0 hasta s do
7                  sumR ← sumR + M_xy.r;
8                  sumG ← sumG + M_xy.g;
9                  sumB ← sumB + M_xy.b;
10         for a=0 hasta s do
11             for b=0 hasta s do
12                 M_xy.r ← ⌊sumR/s^2⌋;
13                 M_xy.g ← ⌊sumG/s^2⌋;
14                 M_xy.b ← ⌊sumB/s^2⌋;
15 Regresa M;
```

Aclacaciones

- dato de entrada s

  Por simplicidad para este algoritmo se pide que s divida a m y a n. Si se quiere que un s cualquiera se debe considerar un caso especial para lidiar con las regiones a la derecha y abajo de la imagen. Éste se implementa en el proyecto.

5. A FUTURO

Para el mantenimiento creo que lo más inmediato sería mejorar la interfaz gráfica porque considero que tiene varias limitaciones como tener que cargar una imagen por default para que funcione el programa y que las imágenes se muestre siempre del mismo tamaño.

También sería posible añadir nuevos filtros, pero lo primero que agregaría sería una opción donde el usuario pueda escoger los parametros que se le pasan al filtro de color para hacer filtros de colores más variados.

Como ya hay muchas aplicaciones gratuitas que hacen lo mismo que esta aplicación y más, no cobraría más de 500.

6. Documentación

- Go

  Documentación de paquete image
  Orientación a objetos con Go
  Documentación de Go
  Documentación de paquete testing
  Tutorial para hacer pruebas unitarias en Go

- GTK

  Página de presentación de go-gtk
  Documentación de GTK3
  Repositorio de go-gtk. Se revisó el código para saber, primero si estaban implimentadas algunas opciones de GTK, y después para ver los datos de entrada y salida.

- Proceso Digital de Imágenes

  Tutorial para hacer un filtro gris en go
  Distintos algoritmos para el filtro gris.