

Visor de Imágenes

Enunciado

Se quiere construir una aplicación que permita visualizar imágenes en formato BMP y realizar operaciones de transformación sobre estas. El formato BMP consiste en guardar la información del color de cada píxel que conforma a la imagen. El color de un píxel está representado en el sistema RGB (Red-Green-Blue), es decir un componente rojo, uno verde y uno azul. Estos componentes son representados por un número que va de 0 a 255.

La aplicación debe permitir:

1. Visualizar una imagen BMP
2. Cargar una imagen BMP
3. Transformar imagen a negativo
4. Transformar imagen a escala de grises
5. Reflejar imagen
6. Binarizar imagen
7. Pixelar imagen
8. Transformar una imagen a través de una convolución

Transformaciones

Transformación 1: Negativo

Para calcular el negativo de una imagen calcularemos el color negativo de cada píxel. Para ello debemos restar a cada componente 255 (máximo valor para un componente) y tomar el valor de absoluto de dicha resta. Con los nuevos valores de los componentes formamos el nuevo color.



Transformación 2: Flip Vertical

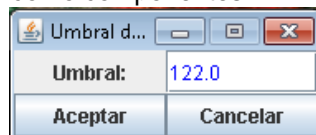
Sirve para invertir verticalmente una imagen. Para ello se intercambian las columnas de píxeles de la imagen: La primera con la última, la segunda con penúltima, etc.



Transformación 3: Binarización

Consiste en llevar los colores de la imagen a dos colores: Negro y Blanco. Para ello se establece un umbral y los píxeles con colores que están por encima o igual al umbral se cambian a blanco y por debajo se cambian a negro.

En este caso se sugiere el umbral como el color promedio de la imagen. Para calcular el color promedio, de un grupo de píxeles se promedian cada uno de sus componentes y se forma un nuevo color con dichos promedios como componentes.



Transformación 4: Pixelamiento

Para esta transformación se divide la imagen en pequeñas regiones de más de un píxel, y se busca el color promedio para esa región (ver transformación3 para la descripción del color promedio), para después cambiar el color de toda la región al color promedio. Las regiones deben ser de un alto y un ancho que sean divisores del ancho y alto de la imagen total respectivamente.

En este caso buscaremos las dimensiones de la región como:

Ancho región: menor divisor mayor a uno del ancho de la imagen.

Alto región: menor divisor mayor a uno del alto de la imagen.



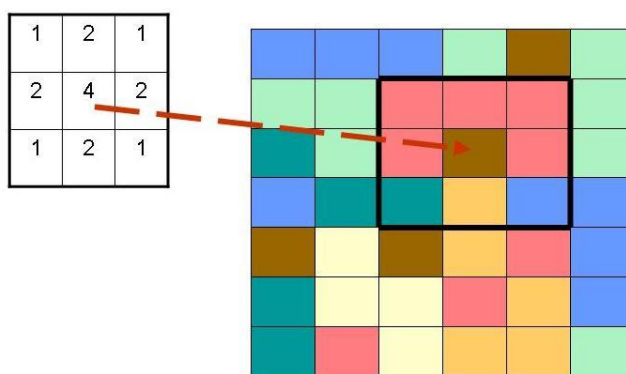
Transformación 5: Escala de Grises

Para convertir la imagen a grises, se promedian los componentes de cada píxel y se crea un nuevo color donde cada componente (RGB) tiene el valor de dicho promedio.



Transformación 6: Convolución

La convolución consiste en operar para cada píxel, su región de píxeles vecinos (incluyéndolo), con unos factores que se ingresan en una matriz cuadrada de dimensión impar (para este ejemplo utilizaremos una matriz de 3x3). La dimensión de la matriz de convolución define la región de píxeles vecinos.



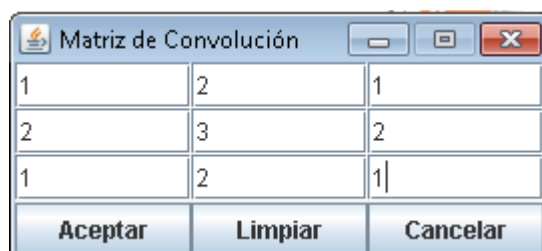
La forma en que se opera la región de píxeles con esta matriz es:

El centro de la matriz coincide con el píxel que se está procesando. Cada píxel (en realidad cada componente del píxel) de la región se multiplica con su factor correspondiente.

Se suman los resultados de estas multiplicaciones.

La sumatoria anterior se divide por la **suma** de los factores que fueron operados: Si el píxel está cerca del borde de la imagen (tomando en cuenta la dimensión de la matriz de convolución) no aplican todos los factores de la matriz (porque no todos tienen un píxel correspondiente), sólo los que efectivamente se operaron.

El color que se crea con el resultado de las operaciones anteriores (que se hace para cada componente) reemplaza al color del píxel original (el del centro). Sin embargo, puede darse el caso de que la suma de los factores sea 0: en ese caso no se realiza la división. Esto puede dar origen a interesantes efectos como la detección de bordes. La operación de la imagen con la matriz de convolución debe hacerse sobre el mapa de píxeles original y no acumular las transformaciones a píxeles hechas por operaciones anteriores con la matriz.





Interfaz

