



Práctica 1

Carga, visualización y modificación de una imagen

**Procesamiento Digital de Imágenes
Grupo 4BV1**

Alumno: Padilla García Andrea Miranda

Objetivo: Realizar un programa en Java que le permita al usuario cargar una imagen jpeg y procesarla para mostrar canales RGB, escala de grises y modificar contraste y brillo.

Introducción

El procesamiento de imágenes es una disciplina encargada de analizar, modificar y mejorar imágenes digitales por medio de algoritmos y técnicas computacionales, su principal objetivo es transformar imágenes para interpretarlas con facilidad, extrayendo información útil o prepararlas para otros procesos más avanzados.

En esta práctica nos enfocaremos en cargar y visualizar una imagen y a esa imagen se le harán los procesos de: ver los canales RGB, ver la escala de grises y modificar tanto el brillo como el contraste de las imágenes a color y en grises.

El modelo RGB (Red, Green, Blue), cada píxel se puede representar como la combinación de estos tres colores, lo que pretendemos es manipular individualmente cada canal y mostrarlos como imágenes por separado, este proceso permite analizar las contribuciones de cada color. Para realizarlo solo hay que mandar a cero dos de los colores del píxel y conservar con valor al color que se desea visualizar.

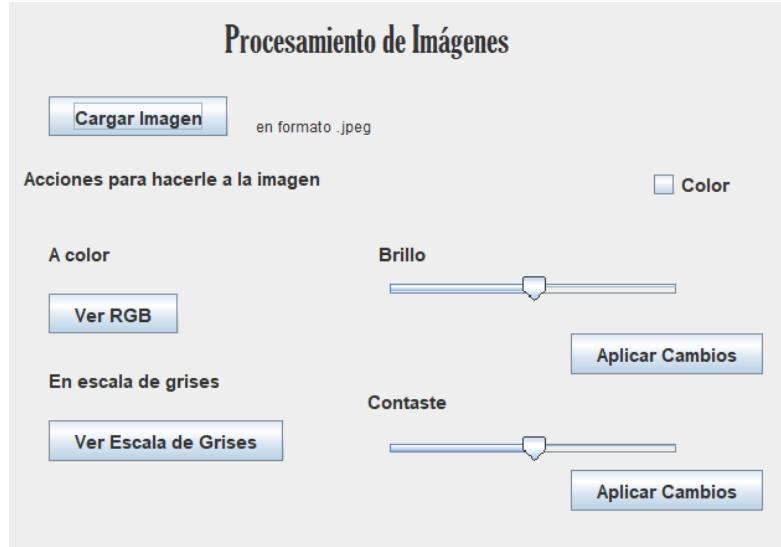
La escala de grises de una imagen consiste en eliminar la información cromática y solo conservar la intensidad luminosa. Para obtenerla, la forma más fácil es obteniendo la media del RGB de los pixeles $\frac{(R+G+B)}{3}$ lo que nos da una imagen monocromática con diferentes tonalidades de gris, dichas tonalidades están en el rango de 0 a 255 niveles de gris.

El brillo aumenta la luminosidad total de la imagen, es decir, se aclaran los colores oscuros y se blanquean los claros; el contraste ajusta la diferencia entre los colores más claros y los más oscuros. El brillo corresponde al desplazamiento uniforme de los valores de intensidad de todos los pixeles, ya sea aclarándola u oscureciéndola. El contraste está relacionado con la diferencia entre las intensidades más oscuras y claras, lo que implica que al codificar el contraste se ve afectada la nitidez y la percepción de los detalles.

Para realizar esta práctica, se realizará una interfaz gráfica para que el usuario pueda cargar una imagen desde la ubicación que prefiera, se le mostrará la imagen que cargó y podrá ver los diferentes canales de color (RGB) y gris y también podrá modificar por medio de un *slider* el brillo y el contraste de la imagen tanto a color como en grises, siendo la escala de color por defecto la escala de grises y por medio de una casilla podrá cambiar al modo de color.

Desarrollo

Lo primero que realicé para la práctica fue la interfaz para el usuario, después de un par de ajustes se optó por esta distribución:



Con este acomodo se le va a permitir al usuario cargar una imagen con el formato especificado, una vez cargado la imagen se podrán visualizar los canales RGB de la imagen y la escala de grises, adicional a esto se permite modificar el brillo y el contraste de la imagen, siendo por default la escala de grises la imagen a modificar, si el usuario desea modificar el brillo y/o contraste del a imagen a color deberá seleccionar la casilla “Color”.

Dentro de la interfaz gráfica de usuario, NetBeans genera el código necesario para implementar la ventana. Para cargar la imagen, se emplea el siguiente código:

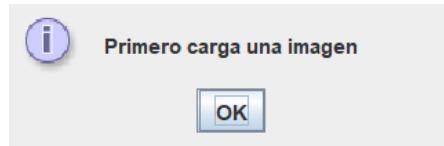
```
private void btnCargarActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser selector = new JFileChooser();
    int res = selector.showOpenDialog(this);
    if (res == JFileChooser.APPROVE_OPTION) {
        try {
            File archivo = selector.getSelectedFile();
            imagenOriginal = ImageIO.read(archivo);
            // Crear canales y escala de grises
            rojo = logica.extraerCanalRojo(imagenOriginal);
            verde = logica.extraerCanalVerde(imagenOriginal);
            azul = logica.extraerCanalAzul(imagenOriginal);
            gris = logica.crearImagenGris(imagenOriginal);
            // Mostrar imagen original al subir la imagen
            logica.mostrarImagen	escalarImagen(imagenOriginal, 800,
                600), "Imagen Original");
        } catch (Exception e) {
            JOptionPane.showMessageDialog(this, "Error al cargar
                imagen: " + e.getMessage());
        }
    }
}
```

Para los botones para mostrar los canales RGB y gris se utilizan los canales rojo, verde, azul, gris definido anteriormente y con parte del código que está en la lógica del programa, se muestra la imagen.

```
private void verGrisActionPerformed(java.awt.event.ActionEvent evt) {
    if (gris != null) {
        logica.mostrarImagen(escalarImagen(gris, 800, 600), "Imagen
        en Escala de Grises");
    } else {
        JOptionPane.showMessageDialog(this, "Primero carga una
        imagen");
    }
}

private void verColorActionPerformed(java.awt.event.ActionEvent evt) {
    if (imagenOriginal != null) {
        logica.mostrarImagen(escalarImagen(rojo, 800, 600), "Canal
        Rojo");
        logica.mostrarImagen(escalarImagen(verde, 800, 600), "Canal
        Verde");
        logica.mostrarImagen(escalarImagen(azul, 800, 600), "Canal
        Azul");
    } else {
        JOptionPane.showMessageDialog(this, "Primero carga una
        imagen");
    }
}
```

Si el usuario presiona estos botones sin haber cargado una imagen con anterioridad, se le muestra una advertencia.



Para el par de JSlider's para el brillo y contraste, se queda establecido con un rango de valores de ±100, para visualizar la imagen modificada se debe presionar el botón de “Aplicar Cambios” y sale la imagen modificada. Este método tiene una variable de tipo booleano que indica el estado de la casilla del color y dependiendo de su valor, es la imagen que arrojará.

```
private void aplicaContrasteActionPerformed(java.awt.event.ActionEvent evt) {
    if (imagenOriginal != null) {
        int valor = contrasteSlider.getValue();
        boolean enColor = colorSelect.isSelected();
        BufferedImage modificada =
            logica.ajustarContraste(imagenOriginal, valor, enColor);
        logica.mostrarImagen(escalarImagen(modificada, 800, 600),
            "Imagen con Contraste");
    } else {
        JOptionPane.showMessageDialog(this, "Primero carga una
            imagen");
    }
}

private void aplicaBrilloActionPerformed(java.awt.event.ActionEvent evt)
{
    if (imagenOriginal != null) {
        int valor = brilloSlider.getValue();
        boolean enColor = colorSelect.isSelected();
        BufferedImage modificada =
            logica.ajustarBrillo(imagenOriginal, valor, enColor);
        logica.mostrarImagen(escalarImagen(modificada, 800, 600),
            "Imagen con Brillo");
    } else {
        JOptionPane.showMessageDialog(this, "Primero carga una
            imagen");
    }
}
```

Al igual que con los canales RGB y el gris, sale el mensaje de advertencia si no se sube inicialmente una imagen.

Por último, está el método que sirve para darle a la imagen una dimensión (tamaño) específico.

```
private BufferedImage escalarImagen(BufferedImage imagen, int maxAncho,
int maxAlto) {
    int ancho = imagen.getWidth();
    int alto = imagen.getHeight();
    double factor = Math.min((double) maxAncho / ancho, (double)
maxAlto / alto);
    int nuevoAncho = (int) (ancho * factor);
    int nuevoAlto = (int) (alto * factor);

    Image temp = imagen.getScaledInstance(nuevoAncho, nuevoAlto,
Image.SCALE_SMOOTH);
    BufferedImage escalada = new BufferedImage(nuevoAncho,
nuevoAlto, BufferedImage.TYPE_INT_RGB);
    Graphics2D g2d = escalada.createGraphics();
    g2d.drawImage(temp, 0, 0, null);
    g2d.dispose();
    return escalada;
}
```

En la zona de la lógica están los métodos para extraer los canales RGB, obtener los grises, modificar brillo y contraste, mostrar la imagen en pantalla.

```
public BufferedImage extraerCanalRojo(BufferedImage imagen) {
    int ancho = imagen.getWidth();
    int alto = imagen.getHeight();
    BufferedImage rojo = new BufferedImage(ancho, alto,
BufferedImage.TYPE_INT_RGB);

    for (int y = 0; y < alto; y++) {
        for (int x = 0; x < ancho; x++) {
            int rgb = imagen.getRGB(x, y);
            int r = (rgb >> 16) & 0xFF;
            int valor = (r << 16);
            rojo.setRGB(x, y, valor);
        }
    }
    return rojo;
}

public BufferedImage extraerCanalVerde(BufferedImage imagen) {
    int ancho = imagen.getWidth();
    int alto = imagen.getHeight();
    BufferedImage verde = new BufferedImage(ancho, alto,
BufferedImage.TYPE_INT_RGB);

    for (int y = 0; y < alto; y++) {
        for (int x = 0; x < ancho; x++) {
            int rgb = imagen.getRGB(x, y);
            int g = (rgb >> 8) & 0xFF;
            int valor = (g << 8);
            verde.setRGB(x, y, valor);
        }
    }
    return verde;
}

public BufferedImage extraerCanalAzul(BufferedImage imagen) {
    int ancho = imagen.getWidth();
    int alto = imagen.getHeight();
    BufferedImage azul = new BufferedImage(ancho, alto,
BufferedImage.TYPE_INT_RGB);

    for (int y = 0; y < alto; y++) {
        for (int x = 0; x < ancho; x++) {
            int rgb = imagen.getRGB(x, y);
            int b = rgb & 0xFF;
            azul.setRGB(x, y, b);
        }
    }
    return azul;
}

public BufferedImage crearImagenGris(BufferedImage imagen) {
    int ancho = imagen.getWidth();
    int alto = imagen.getHeight();
    BufferedImage gris = new BufferedImage(ancho, alto,
BufferedImage.TYPE_INT_RGB);

    for (int y = 0; y < alto; y++) {
        for (int x = 0; x < ancho; x++) {
            int rgb = imagen.getRGB(x, y);
            int r = (rgb >> 16) & 0xFF;
            int g = (rgb >> 8) & 0xFF;
            int b = rgb & 0xFF;
            int promedio = (r + g + b) / 3;
            int valor = (promedio << 16) | (promedio << 8) |
promedio;
            gris.setRGB(x, y, valor);
        }
    }
    return gris;
}
```

```

public BufferedImage ajustarBrillo(BufferedImage original, int valor,
boolean enColor) {
    BufferedImage salida = new BufferedImage(original.getWidth(),
    original.getHeight(), BufferedImage.TYPE_INT_RGB);

    for (int y = 0; y < original.getHeight(); y++) {
        for (int x = 0; x < original.getWidth(); x++) {
            Color c = new Color(original.getRGB(x, y));

            int r = c.getRed();
            int g = c.getGreen();
            int b = c.getBlue();

            if (!enColor) {
                // Escala de grises
                int gris = (r + g + b) / 3;
                gris = limitar(gris + valor);
                r = g = b = gris;
            } else {
                r = limitar(r + valor);
                g = limitar(g + valor);
                b = limitar(b + valor);
            }

            salida.setRGB(x, y, new Color(r, g, b).getRGB());
        }
    }
    return salida;
}

public BufferedImage ajustarContraste(BufferedImage original, int valor,
boolean enColor) {
    BufferedImage salida = new BufferedImage(original.getWidth(),
    original.getHeight(), BufferedImage.TYPE_INT_RGB);
    // Fórmula de contraste
    double factor = (259 * (valor + 255.0)) / (255 * (259 - valor));

    for (int y = 0; y < original.getHeight(); y++) {
        for (int x = 0; x < original.getWidth(); x++) {
            Color c = new Color(original.getRGB(x, y));
            int r, g, b;
            if (!enColor) {
                int gris = (c.getRed() + c.getGreen() + c.getBlue()) / 3;
                gris = limitar((int) (factor * (gris - 128) + 128));
                r = g = b = gris;
            } else {
                r = limitar((int) (factor * (c.getRed() - 128) + 128));
                g = limitar((int) (factor * (c.getGreen() - 128) + 128));
                b = limitar((int) (factor * (c.getBlue() - 128) + 128));
            }
            salida.setRGB(x, y, new Color(r, g, b).getRGB());
        }
    }
    return salida;
}

```

```
public static void mostrarImagen(BufferedImage imagen, String
titulo) {
    JFrame ventana = new JFrame(titulo);
    JLabel label = new JLabel(new ImageIcon(imagen));
    JScrollPane scroll = new JScrollPane(label);
    ventana.add(scroll);
    ventana.pack();
    ventana.setLocationRelativeTo(null);
    ventana.setVisible(true);
}
```

También esta función que limita los valores que se dan al momento de cambiar el contraste y el brillo, para evitar que se salga del rango establecido:

```
private int limitar(int val) {
    return Math.min(255, Math.max(0, val));
}
```

Conclusiones

Esta práctica fue el primer acercamiento al procesamiento digital de imágenes, fue muy interesante para mí el emplear de manera práctica los conceptos explicados en el salón para desarrollar un programa en Java capaz de cargar una imagen y realizar varios procesos que hoy en día son comunes en aplicaciones móviles como cambiar el brillo y contraste o hacer la conversión a escala de grises. Además, descubrí el proceso de extracción de los canales RGB.

En general, considero que esta práctica me ayudó a sentar las bases para próximos trabajos de esta y otras materias.

Referencias

Corel PHOTO-PAINT X6 help. (s. f.). https://product.corel.com/help/PHOTO-PAINT/540240626/Main/ES/Doc/wwhelp/wwimpl/common/html/wwhelp.htm?context=Corel_PHOTO_PAINT_Help&file=CorelDRAW_Working_with_color_channels.html

Función de brillo y contraste—ArcMap | Documentación. (s. f.).
<https://desktop.arcgis.com/es/arcmap/latest/manage-data/raster-and-images/contrast-and-brightness-function.htm>

Jimenez, E. C. (2025, 2 julio). Procesamiento digital de imágenes y su importancia. MasScience.
<https://masscience.com/procesamiento-digital-de-imagenes-y-su-importancia/>