Instituto Politécnico Nacional
Escuela Superior de Cómputo

Práctica 4

Binarización de una imagen

Procesamiento Digital de Imágenes
Grupo 4BV1

Alumno: Padilla García Andrea Miranda

Objetivo: Obtener la binarización de una imagen en escala de grises empleando de 1 a 4 umbrales, además de obtener la inverza la de imagen binarizada

Introducción

La binarización de imágenes, también conocida como *umbralización* es un proceso que convierte las imágenes originales capturadas ya sea en color o en escala de grises en imágenes binarias formadas por pequeños píxeles en blanco y negro (B/N). Esta técnica permite separar el primer plano del fondo de la imagen, esto se realiza con el objetivo de extraer información útil de la imagen o segmentar una región de interés.

La binarización se hace en imágenes en escala de grises, si se sube una imagen a color, se convierte a escala de grises (en este caso tomando Y del modelo YIQ), cada valor de píxel en niveles de grises se compara con un valor umbral, si el valor del píxel es menor que el umbral, se manda a cero; de lo contrario, se manda al máximo valor posible (255). El umbral se puede expresar de la siguiente manera:

```
Si P(x, y) < Umbral:
    P(x, y) = 0
  de lo contrario:
    P(x, y) = 255

donde, P(x, y) = Valor de píxel
```

Pero cuando un umbral no es suficiente para capturar bien distintas intensidades (por ejemplo, si hay varios grupos tonales), se puede usar binarización con múltiples umbrales (también llamada "umbralización múltiple"), lo que implica dividir el rango de intensidad en más de dos regiones.

En pocas palabras:

- 1 umbral divide la escala en dos partes: todo lo que esté por debajo del Umbral va a clase "0", lo que esté por encima va a clase "1".
- 2 umbrales permiten dividir en tres regiones (por debajo de $Umbral_1$, entre $Umbral_1$ y $Umbral_2$, por encima de $Umbral_2$).
- 3 umbrales permiten cuatro regiones, etc.

Es un tipo de segmentación más sofisticada que puede capturar distintos niveles de intensidad, útil cuando la imagen no es simplemente "fondo / objeto".

La práctica requiere la transformación de RGB al modelo YIQ para obtener la escala de grises (utilizando Y) y también, se requiere la transformación de YIQ a RGB (utilizando Y_b para reconstruir la imagen con los valores I y Q originales). Para ello se utiliza la siguiente matriz:

$$\begin{pmatrix} Y \\ I \\ Q \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ 0.5959 & -0.2746 & -0.3213 \\ 0.2115 & -0.5227 & 0.3112 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

RGB → YIQ

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1.0 & 0.956 & 0.619 \\ 1.0 & -0.272 & -0.647 \\ 1.0 & -1.106 & 1.703 \end{pmatrix} \begin{pmatrix} Y \\ I \\ Q \end{pmatrix}$$

YIQ → RGB

El proceso de **reconstrucción** tendría estos efectos en la imagen resultante, comparada con la original:

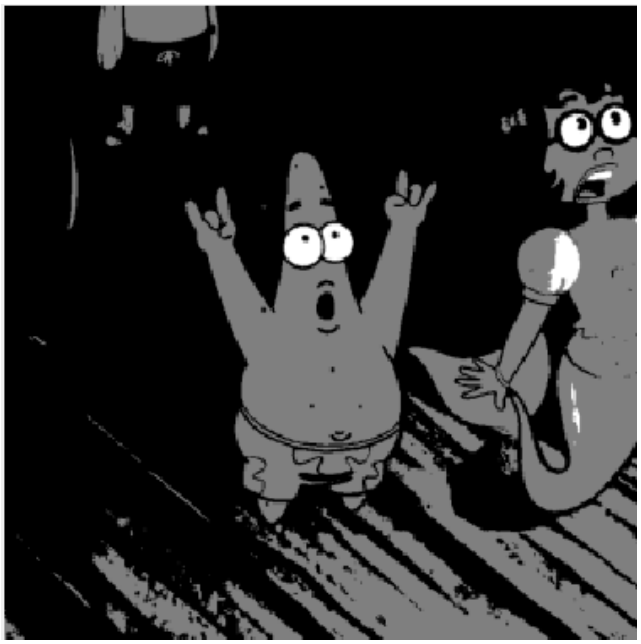
- Colores "lavados" o saturados: Porque Y_b es extremo (0, 64, 127, etc.), multiplicado por I/Q puede causar *overflows/underflows*, pero el limitador (Math.max/min) hace que quede en un rango de valores permitidos, aunque podría hacer partes muy brillantes u oscuras.
- Pérdida de detalles finos: La binarización de Y elimina gradientes suaves, haciendo la imagen más “estilo caricatura” en cuestión del brillo.

Por ejemplo:

Imagen Original

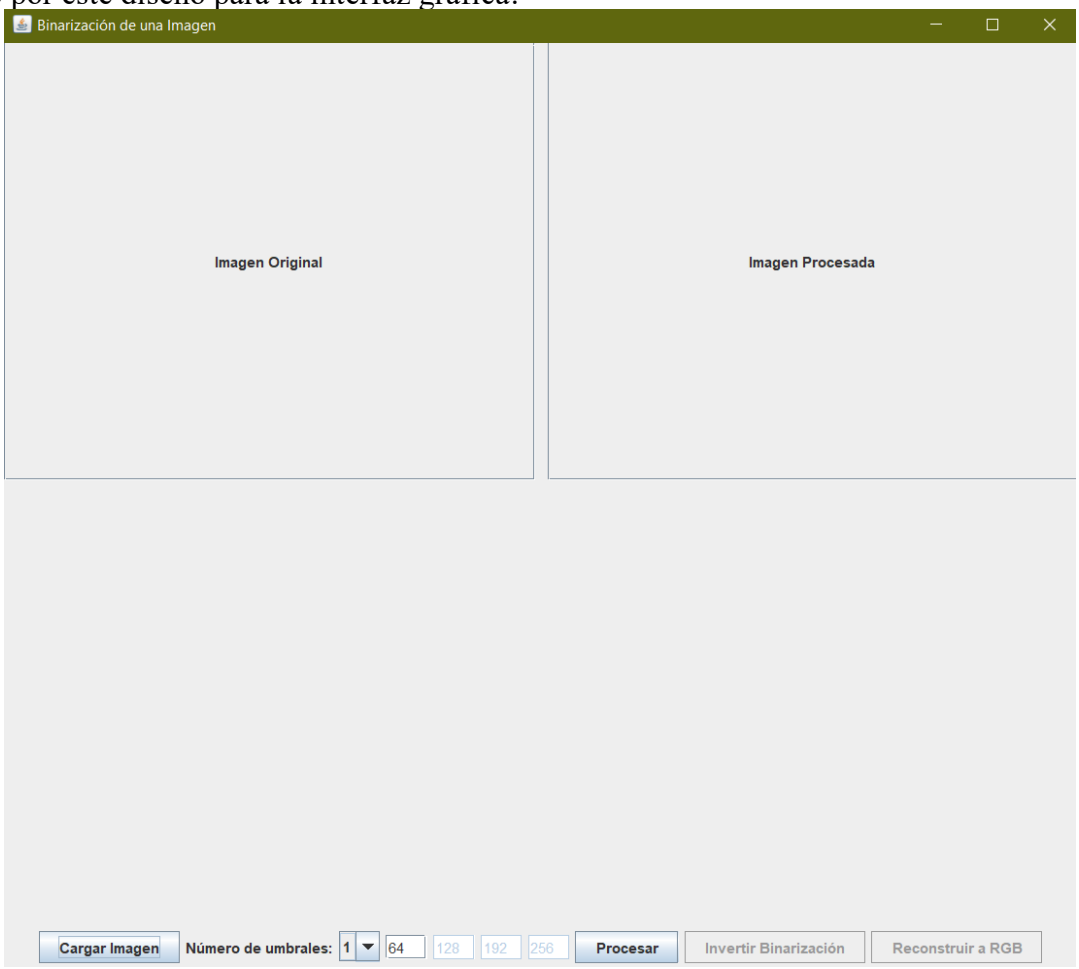
Imagen reconstruida a partir de una binarización con 2 umbrales

Imagen binaria con 2 umbrales



Desarrollo

Se optó por este diseño para la interfaz gráfica:



Una vez cargada la imagen, se muestra en pantalla en escala de grises y debajo de ella se ve su histograma. Con los siguientes métodos auxiliares obtenemos los valores de YIQ:

```
public static double rgbAY(int r, int g, int b) {  //componente Y
    return 0.299 * r + 0.587 * g + 0.114 * b;
}

public static double rgbAI(int r, int g, int b) {  //componente I
    return 0.596 * r - 0.275 * g - 0.321 * b;
}

public static double rgbAQ(int r, int g, int b) {  //componente Q
    return 0.212 * r - 0.523 * g + 0.311 * b;
}
```

Al presionar el botón de procesar, se emplea lo siguiente:

```
        private void procesarImagen() {
                if (imagenOriginal == null) {
                    JOptionPane.showMessageDialog(this, "Carga primero una imagen.");
                    return;
                }

                int numUmbrales = comboUmbrales.getSelectedIndex() + 1;
                double[] umbrales = new double[numUmbrales];
                try {
                    for (int i = 0; i < numUmbrales; i++) {
                        umbrales[i] = Double.parseDouble(camposUmbrales[i].getText());
                    }
                    java.util.Arrays.sort(umbrales);
                } catch (NumberFormatException e) {
                    JOptionPane.showMessageDialog(this, "Introduce umbrales válidos
                   (números).");
                    return;
                }

                int altura = imagenOriginal.getHeight();
                int ancho = imagenOriginal.getWidth();
                int[][] binarizada = new int[altura][ancho];
                //recorrer y aplicar binarización según los umbrales
                for (int fila = 0; fila < altura; fila++) {
                    for (int col = 0; col < ancho; col++) {
                        double valorY = matrizY[fila][col];
                        switch (numUmbrales) {
                            case 1:
                                binarizada[fila][col] = binarizarUnoUmbral(valorY, umbrales[0]);
                                break;
                            case 2:
                                binarizada[fila][col] = binarizarDosUmbrales(valorY,
                                umbrales[0], umbrales[1]);
                                break;
                            case 3:
                                binarizada[fila][col] = binarizarTresUmbrales(valorY,
                                umbrales[0], umbrales[1], umbrales[2]);
                                break;
                            case 4:
                                binarizada[fila][col] = binarizarCuatroUmbrales(valorY,
                                umbrales[0], umbrales[1], umbrales[2], umbrales[3]);
                                break;
                        }
                    }
                }

                ultimaBinarizada = binarizada; //guardar para futuras operaciones
                imagenProcesada = convertirYBinariaAImagenGris(binarizada); //mostrar en grises
                int[] histProcesada = calcularHistogramaDeYb(binarizada);
                histogramaProcesada.setData(histProcesada);
                etiquetaProcesada.setIcon(new ImageIcon(escalarImagen(imagenProcesada,
                etiquetaProcesada.getWidth(), 300)));
                etiquetaProcesada.setText("Imagen Procesada");
                invertida = false;
                botonInvertir.setEnabled(true);
                botonReconstruir.setEnabled(true); //reconstrucción RGB
            }
```

Lo anterior se encarga, primero, de que se haya subido una imagen y se ingresaran valores válidos para los umbrales, después se obtiene la dimensión de la imagen y se recorre por píxel aplicando la binarización pertinente dependiendo de los umbrales que se hayan seleccionado. Fuera del ciclo se almacenan los valores de la última binarización realizada y se muestra la imagen ya procesada junto con su histograma. Igual se habilitan los botones para invertir la binarización y reconstruir a RGB.

Para realizar lo anterior, se implementan los métodos siguientes para realizar la umbralización:

```
//con un umbral
    public static int binarizarUnoUmbral(double y, double umbral) {
        return y >= umbral ? 255 : 0;
    }

    //con dos umbrales
    public static int binarizarDosUmbrales(double y, double t1, double t2) {
        if (y < t1) return 0;
        else if (y < t2) return 127;
        else return 255;
    }

    //con tres umbrales
    public static int binarizarTresUmbrales(double y, double t1, double t2, double t3) {
        if (y < t1) return 0;
        else if (y < t2) return 85;
        else if (y < t3) return 170;
        else return 255;
    }

    //con cuatro umbrales
    public static int binarizarCuatroUmbrales(double y, double t1, double t2, double t3,
    double t4) {
        if (y < t1) return 0;
        else if (y < t2) return 64;
        else if (y < t3) return 128;
        else if (y < t4) return 192;
        else return 255;
    }
```

Para hacer la inversión de la binarización se hace:

```
private void invertirImagen() {
        if (ultimaBinarizada == null) return;

        int altura = ultimaBinarizada.length;
        int ancho = ultimaBinarizada[0].length;

        // Invertir cada valor en la matriz binarizada
        for (int fila = 0; fila < altura; fila++) {
            for (int col = 0; col < ancho; col++) {
                ultimaBinarizada[fila][col] = 255 - ultimaBinarizada[fila][col];
            }
        }
        //recrear la imagen procesada en grises con la inversión
        imagenProcesada = convertirYBinariaAImagenGris(ultimaBinarizada);
        int[] histProcesada = calcularHistogramaDeYb(ultimaBinarizada);
        histogramaProcesada.setData(histProcesada);
        etiquetaProcesada.setIcon(new ImageIcon(escalarImagen(imagenProcesada,
        etiquetaProcesada.getWidth(), 300)));
        invertida = !invertida;
        botonReconstruir.setEnabled(true);
    }
```

Para reconstruir la imagen utilizamos:

```
private BufferedImage convertirYIQaRGB(int[][] Yb, double[][] I, double[][] Q) {
        int altura = Yb.length;
        int ancho = Yb[0].length;
        BufferedImage img = new BufferedImage(ancho, altura,BufferedImage.TYPE_INT_RGB);

        for (int fila = 0; fila < altura; fila++) {
            for (int col = 0; col < ancho; col++) {
                double yVal = Yb[fila][col]; // Y binarizado
                double iVal = I[fila][col];  // I original
                double qVal = Q[fila][col];  // Q original

                //fórmulas de conversión YIQ a RGB
                double r = yVal + 0.956 * iVal + 0.621 * qVal;
                double g = yVal - 0.272 * iVal - 0.647 * qVal;
                double b = yVal - 1.105 * iVal + 1.702 * qVal;

                //limitar a rango 0-255
                int rInt = limitarValor((int) Math.round(r));
                int gInt = limitarValor((int) Math.round(g));
                int bInt = limitarValor((int) Math.round(b));

                int rgb = (rInt << 16) | (gInt << 8) | bInt;
                img.setRGB(col, fila, rgb);
            }
        }
        return img;
    }

    //limitar valores entre 0 y 255
    private static int limitarValor(int val) {
        return Math.min(255, Math.max(0, val));
    }
```

Donde se implementa el limitador antes mencionado para conservar el rango permitido.

Conclusión

Concluyo diciendo que esta práctica me permitió entender la técnica de binarización empleando de 1 a 4 umbrales siendo la cantidad de umbrales seleccionada la segmentación que se obtiene de la imagen logrando distinguir más allá del fondo u objeto dando varios niveles de brillo en la imagen procesada y, por ende, más o menos detalles de la imagen. Y finalmente, la reconstrucción de la imagen a partir de los niveles de gris generados en la binarización fue un proceso que si bien al inicio no entendí del todo cómo hacerlo, logré llevarlo a cabo y me pareció un proceso bastante interesante ya que el factor Y se modificó lo que arroja una imagen distinta a la original.

Referencias

¿Qué es la binarización de imágenes? (s. f.). Bettersize Instruments Ltd. https://www.bettersizeinstruments.com/es/learn/bettersize-wiki/what-is-image-binarization/

Piyadasa, T. D. (2022, 6 junio). *Image binarization in a nutshell*. Medium. https://medium.com/@tharindad7/image-binarization-in-a-nutshell-b40b63c0228e

Image thresholding. (s. f.). MathWorks. https://la.mathworks.com/discovery/image-thresholding.html

Wikipedia contributors. (2025, 26 agosto). *YIQ*. Wikipedia. https://en.wikipedia.org/wiki/YIQ