Instituto Politécnico Nacional
Escuela Superior de Cómputo

Práctica 5

Operaciones entre imágenes

Procesamiento Digital de Imágenes
Grupo 4BV1

Alumno: Padilla García Andrea Miranda

Objetivo: Aplicar transformaciones geométricas y operaciones aritméticas, lógicas y relacionales

Introducción

Para esta práctica requerimos conocer las operaciones que se pueden realizar sobre las imágenes.

Transformaciones geométricas

Las transformaciones geométricas modifican la relación espacial entre píxeles

- **Traslación**: la traslación o desplazamiento consiste en trasladar el píxel *(i, j)* en la imagen original hasta que alcance la posición $(i+t_x, j+t_y)$ y sustituir el valor de la intensidad $(i+t_x, j+t_y)$ por el correspondiente a la posición *(i, j)* de la imagen original. Para realizar la traslación se emplean las ecuaciones: $x = v + t_x$ y $y = w + t_y$
- **Rotación**: esta transformación difiere de la traslación ya que es un giro. Para realizarla se emplean las ecuaciones: $x = v\cos\theta - wsen\theta$ y $y = v\cos\theta + wsen\theta$
- **Interpolación/escalamiento**: la interpolación es la modificación, al alza o a la baja, del número de píxeles de una imagen digital multiplicando sus coordenadas por factores de escala, se emplean las fórmulas: $x = c_x v$ y $y = c_y w$

Operaciones aritméticas

Estas operaciones se hacen por píxel, cuando se tienen dos imágenes, se llevan a cabo entre el píxel de una con el píxel de otra en la misma posición/coordenada de ambas imágenes.

Las operaciones que se pueden realizar son:

- Suma: la suma de dos imágenes del mismo tamaño da como resultado una nueva imagen del mismo tamaño, donde los píxeles son la suma de los píxeles de las imágenes originales, se calcula con: $f'(x,y) = \frac{f(x,y)+g(x,y)}{2}$
- Resta: la imagen resultante se ve más oscura, se calcula con:

$$f'(x,y) = round(|(double)f(x,y) - (double)g(x,y)|)$$

- Multiplicación: intensifica regiones, puede oscurecer mucho si los valores son pequeños se calcula como:

$$maximo = max(f'(x,y))$$
$$minimo = min(f'(x,y))$$
$$f''(x,y) = (f'(x,y) - minimo) * (255.0/(maximo - minimo))$$

- División: suele dar efectos de realce o corrección, se calcula como:

$$double\ b = g(x,y)$$
$$si\ b == 0\ \ entonces\ b = 0.01$$
$$f'(x,y) = (double)f(x,y)/b$$

Operaciones lógicas/booleanas

- AND: ayuda a seleccionar áreas de interés (la luz representa 1 y oscuro 0) o para crear una tercera imagen a partir de dos imágenes (intersección), generalmente da una imagen más oscura que resalta las zonas oscuras. Se calcula con:

$$f_{ANDg}(x,y) = f_{g1}(x,y)\&f_{g2}(x,y)$$

$$f_{ANDb}(x,y) = f_{b1}(x,y)\&f_{b2}(x,y)$$

- OR: al igual que la AND, se utiliza para enmascarar y se crea a partir de ella una tercera imagen, generalmente da una imagen más clara. Se calcula con:

$$f_{ORg}(x,y) = f_{g1}(x,y) \mid f_{g2}(x,y)$$

$$f_{ORb}(x,y) = f_{b1}(x,y) \mid f_{b2}(x,y)$$

- XOR: muestra las diferencias entre las imágenes, donde son iguales, queda negro; donde son diferentes, se resalta. Se calcula con:

$$f_{XORg}(x,y) = f_{g1}(x,y) \wedge f_{g2}(x,y)$$

$$f_{XORb}(x,y) = f_{b1}(x,y) \wedge f_{b2}(x,y)$$

- NOT: muestra el negativo de la imagen, intercambiando 1 y 0. Se calcula:

$$f_{NOTg}(x,y) = \sim f_g(x,y)$$

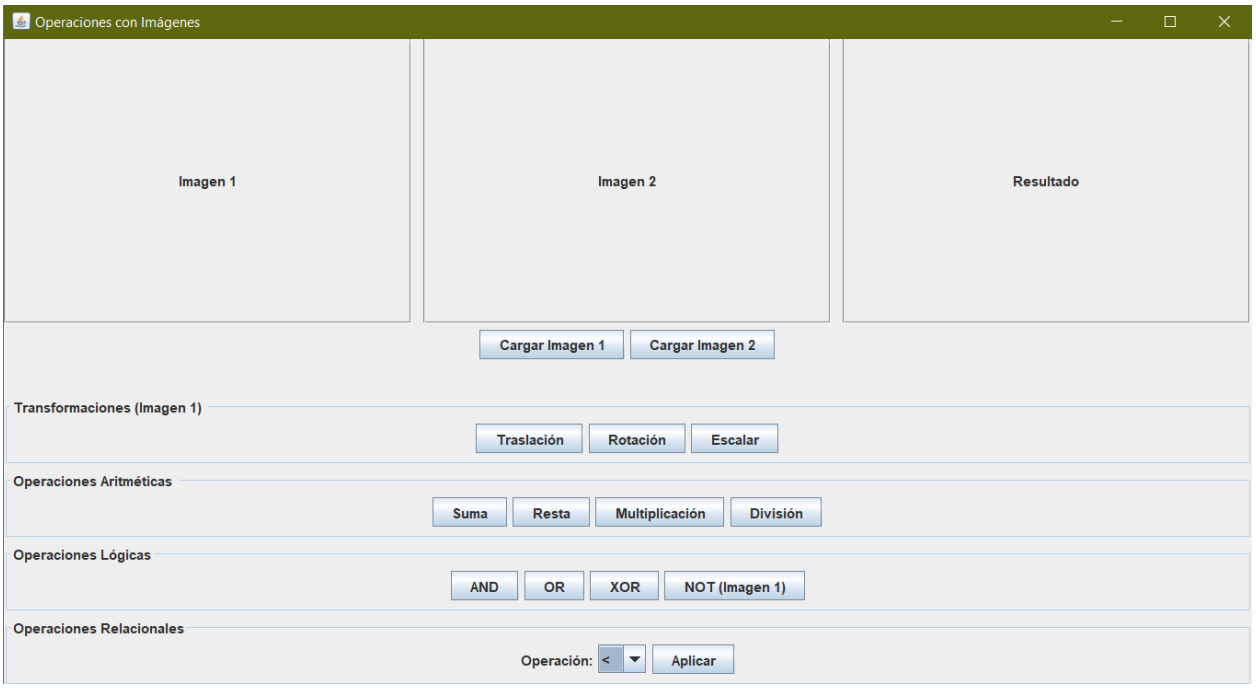
$$f_{NOTb}(x,y) = \sim f_b(x,y)$$

Operaciones relacionales

Operador	Operación	
<	Verifica si, píxel a píxel, una imagen es menor a otra	Pone en blanco los píxeles de la primera imagen que sean menores (o iguales) a los de la segunda. Resalta zonas oscuras.
<=	Verifica si una imagen es menor o igual a otra	
>	Verifica si una imagen es mayor a otra	Pone en blanco los píxeles de la primera imagen que sean mayores (o iguales) a los de la segunda. Resalta zonas brillantes.
>=	Verifica si una imagen es mayor o igual a otra	
==	Compara píxel a píxel si dos imágenes son iguales	Queda blanco donde ambas imágenes tienen exactamente el mismo valor de píxel.
!=	Compara los píxeles de dos imágenes para ver si son diferentes	Marca en blanco los píxeles diferentes entre ambas imágenes.

Desarrollo

Para realizar la interfaz gráfica, se optó por el siguiente diseño:



Al cargar la imagen 1, se habilitan los botones de la zona de transformaciones y la operación lógica NOT, con la imagen 2 se habilitan las demás operaciones.

Para las transformaciones y lógica se implementa de este modo en código:

```java
//aritméticas
private void arithmeticOperation(String operation) {
    if (img1 == null || img2 == null) {
        JOptionPane.showMessageDialog(this, "Carga ambas imágenes primero.");
        return;
    }
    BufferedImage im1 = toGrayscale(img1);
    BufferedImage im2 = toGrayscale(img2);
    int w = Math.min(im1.getWidth(), im2.getWidth());
    int h = Math.min(im1.getHeight(), im2.getHeight());
    resultImg = new BufferedImage(w, h, BufferedImage.TYPE_INT_RGB);

    for (int y = 0; y < h; y++) {
        for (int x = 0; x < w; x++) {
            int g1 = im1.getRGB(x, y) & 0xff;
            int g2 = im2.getRGB(x, y) & 0xff;
            int val = 0;
            switch (operation) {
                case "suma": val = Math.min(255, g1 + g2); break;
                case "resta": val = Math.max(0, g1 - g2); break;
                case "multiplicación": val = Math.min(255, g1 * g2 / 255); break;
                case "division": val = (g2 == 0) ? 255 : Math.min(255, (g1 * 255) / g2);
                break;
            }
            int rgb = (val << 16) | (val << 8) | val;
            resultImg.setRGB(x, y, rgb);
        }
    }
    labelResult.setIcon(new ImageIcon(scaleImage(resultImg, labelResult.getWidth(),
     labelResult.getHeight())));
    labelResult.setText("Resultado " + operation);
}
```

```java
//lógicas
private void logicalOperation(String operation) {
    if (operation.equals("not")) {
       if (img1 == null) {
           JOptionPane.showMessageDialog(this, "Carga la Imagen 1 primero.");
           return;
       }
       BufferedImage im1 = toGrayscale(img1);
       int w = im1.getWidth();
       int h = im1.getHeight();
       resultImg = new BufferedImage(w, h, BufferedImage.TYPE_INT_RGB);
       for (int y = 0; y < h; y++) {
           for (int x = 0; x < w; x++) {
               int g1 = im1.getRGB(x, y) & 0xff;
               int val = ~g1 & 0xff;
               int rgb = (val << 16) | (val << 8) | val;
               resultImg.setRGB(x, y, rgb);
           }
       }
    labelResult.setIcon(new ImageIcon(scaleImage(resultImg, labelResult.getWidth(),
     labelResult.getHeight())));
    labelResult.setText("NOT Imagen 1");
    return;
    }
    if (img1 == null || img2 == null) {
        JOptionPane.showMessageDialog(this, "Carga ambas imágenes primero.");
        return;
    }
    BufferedImage im1 = toGrayscale(img1);
    BufferedImage im2 = toGrayscale(img2);
    int w = Math.min(im1.getWidth(), im2.getWidth());
    int h = Math.min(im1.getHeight(), im2.getHeight());
    resultImg = new BufferedImage(w, h, BufferedImage.TYPE_INT_RGB);
   for (int y = 0; y < h; y++) {
       for (int x = 0; x < w; x++) {
           int g1 = im1.getRGB(x, y) & 0xff;
           int g2 = im2.getRGB(x, y) & 0xff;
           int val = 0;
           switch (operation) {
              case "and": val = g1 & g2; break;
              case "or": val = g1 | g2; break;
              case "xor": val = g1 ^ g2; break;
           }
           int rgb = (val << 16) | (val << 8) | val;
           resultImg.setRGB(x, y, rgb);
       }
   }
    labelResult.setIcon(new ImageIcon(scaleImage(resultImg, labelResult.getWidth(),
      labelResult.getHeight())));
    labelResult.setText("Resultado " + operation.toUpperCase());
}
```

```
private void rotateImage() {
    if (img1 == null) {
        JOptionPane.showMessageDialog(this, "Carga la Imagen 1 primero.");
        return;
    }
    String angleStr=JOptionPane.showInputDialog(this,"Ángulo de rotación (grados):", "0")
    try {
        double angle = Math.toRadians(Double.parseDouble(angleStr));
        resultImg = rotate(img1, angle);
        labelResult.setIcon(new ImageIcon(scaleImage(resultImg, labelResult.getWidth(),
        labelResult.getHeight())));
        labelResult.setText("Imagen Rotada");
    } catch (Exception e) {
        JOptionPane.showMessageDialog(this, "Valor inválido.");
    }
}
         JOptionPane.showMessageDialog(this, "Valores inválidos.");
    }
 }
//relacionales
private void relationalOperation(String operation) {
    if (img1 == null || img2 == null) {
        JOptionPane.showMessageDialog(this, "Carga ambas imágenes primero.");
        return;
    }
    BufferedImage im1 = toGrayscale(img1);
    BufferedImage im2 = toGrayscale(img2);
    int w = Math.min(im1.getWidth(), im2.getWidth());
    int h = Math.min(im1.getHeight(), im2.getHeight());
    resultImg = new BufferedImage(w, h, BufferedImage.TYPE_INT_RGB);
    for (int y = 0; y < h; y++) {
        for (int x = 0; x < w; x++) {
            int g1 = im1.getRGB(x, y) & 0xff;
            int g2 = im2.getRGB(x, y) & 0xff;
            boolean cond = false;
            switch (operation) {
                 case "<": cond = g1 < g2; break;
                 case "<=": cond = g1 <= g2; break;
                 case ">": cond = g1 > g2; break;
                 case ">=": cond = g1 >= g2; break;
                 case "==": cond = g1 == g2; break;
                 case "!=": cond = g1 != g2; break;
            }
            int val = cond ? 255 : 0;
            int rgb = (val << 16) | (val << 8) | val;
            resultImg.setRGB(x, y, rgb);
       }
    }
    labelResult.setIcon(new ImageIcon(scaleImage(resultImg, labelResult.getWidth(),
      labelResult.getHeight())));
    labelResult.setText("Resultado Relacional " + operation);
}
```

```
private void scaleImage() {
    if (img1 == null) {
        JOptionPane.showMessageDialog(this, "Carga la Imagen 1 primero.");
        return;
    }
    String scaleStr = JOptionPane.showInputDialog(this, "Factor de escala (ej. 0.5 para
    reducir, 2 para agrandar):", "1");
    try {
        double scale = Double.parseDouble(scaleStr);
        if (scale <= 0) throw new Exception("El factor de escala debe ser positivo");
        int maxSize = 5000;
        resultImg = scale(img1, scale, maxSize);
        labelResult.setIcon(new ImageIcon(scaleImage(resultImg, labelResult.getWidth(),
        labelResult.getHeight())));
        labelResult.setText("Imagen Escalada (factor: " + scale + ")");
    } catch (Exception e) {
        JOptionPane.showMessageDialog(this, "Valor inválido.");
    }
}

private BufferedImage translate(BufferedImage img, int dx, int dy) {
    int w = img.getWidth();
    int h = img.getHeight();
    BufferedImage res = new BufferedImage(w, h, BufferedImage.TYPE_INT_RGB);
    Graphics2D g = res.createGraphics();
    g.setColor(Color.BLACK);
    g.fillRect(0, 0, w, h);
    g.drawImage(img, dx, dy, null);
    g.dispose();
    return res;
}
private BufferedImage rotate(BufferedImage img, double angleRad) {
    int w = img.getWidth();
    int h = img.getHeight();
    double sin = Math.abs(Math.sin(angleRad));
    double cos = Math.abs(Math.cos(angleRad));
    int newW = (int) Math.floor(w * cos + h * sin);
    int newH = (int) Math.floor(h * cos + w * sin);

    BufferedImage res = new BufferedImage(newW, newH, BufferedImage.TYPE_INT_RGB);
    int cx = w / 2;
    int cy = h / 2;
    int ncx = newW / 2;
    int ncy = newH / 2;

    for (int y = 0; y < newH; y++) {
        for (int x = 0; x < newW; x++) {
            int dx = x - ncx;
            int dy = y - ncy;
            double ox = Math.cos(angleRad) * dx + Math.sin(angleRad) * dy + cx;
            double oy = -Math.sin(angleRad) * dx + Math.cos(angleRad) * dy + cy;
            int rgb = bilinearInterpolation(img, ox, oy);
            res.setRGB(x, y, rgb);
        }
    }
    return res;
}
```

```java
private BufferedImage scale(BufferedImage img, double scale, int maxSize) {
    int w = img.getWidth();
    int h = img.getHeight();
    int newW = (int) (w * scale);
    int newH = (int) (h * scale);
    if (newW <= 0) newW = 1;
    if (newH <= 0) newH = 1;

    if (newW > maxSize || newH > maxSize) {
        double adjustScale = Math.min((double) maxSize / newW, (double) maxSize / newH);
        newW = (int) (w * adjustScale);
        newH = (int) (h * adjustScale);
    }

    BufferedImage res = new BufferedImage(newW, newH, BufferedImage.TYPE_INT_RGB);

    for (int y = 0; y < newH; y++) {
        for (int x = 0; x < newW; x++) {
            double ox = x / scale;
            double oy = y / scale;
            int rgb = bilinearInterpolation(img, ox, oy);
            res.setRGB(x, y, rgb);
        }
    }
    return res;
}
```

Conclusión

Puedo decir que con esta práctica entendí cómo llevar a cabo las transformaciones y operaciones entre imágenes. Me resulta interesante especialmente las operaciones que permiten mostrar las diferencias entre un par de imágenes, lo que me lleva a pensar en todas las aplicaciones que tienen estas operaciones con operaciones relativamente simple.

Referencias

Arithmetic operations — Basics of Image Processing. (s. f.). https://vincmazet.github.io/bip/digital-images/operations.html

Barreto Melo, S. (s. f.). *Transformaciones geométricas*. https://alammi.info/2congreso/memorias/Documentos/martes/TRANSFORMGEOMETRICAS.pdf

GeeksforGeeks. (2025, 23 julio). *Geometric Transformation in Image Processing*. GeeksforGeeks. https://geeksforgeeks.org/electronics-engineering/geometric-transformation-in-image-processing-1/

Herrera, P., & Guijarro, M. (2016). *Operaciones de transformación*. https://researchgate.net/publication/323200190_Operaciones_de_Transformacion_de_Imagenes

Martínez, J. (2011). *Operadores puntuales*. https://arantxa.ii.uam.es/~jms/tdi/TDI_Tema3_2011.pdf