2.2. Transformaciones lógicas de imágenes digitales

Las operaciones lógicas que involucran imágenes se realizan píxel por píxel entre dos o más imágenes (esto excluye la operación lógica NOT, que se realiza en una sola imagen). Dependiente en el hardware y/o software que se utiliza, la mecánica real de implementación. Las operaciones lógicas se pueden realizar secuencialmente, un píxel a la vez en el tiempo, o en paralelo, donde todas las operaciones se realizan simultáneamente. Las operaciones lógicas operan procesando píxel por píxel. Solo necesitamos preocuparnos por la capacidad de implementar los operadores lógicos AND, OR y NOT porque estos tres operadores son funcionalmente completos. En otras palabras, cualquier otro operador lógico puede implementarse usando solo estas tres funciones básicas. Cuando se trata de operaciones lógicas en imágenes en escala de grises, los valores del pixel se procesan como cadenas de números binarios.

2.2.1. AND (&)

La operación AND se utiliza para enmascarar; es decir, para seleccionar subimágenes en una imagen, como se ilustra en la figura 2.3, también se puede hacer la operación entre dos imágenes diferentes para crear una tercera (figura 2.4). En la máscara de imagen AND, la luz representa un binario 1 y oscuro representa un 0 binario. El enmascaramiento a veces se denomina región de procesamiento de intereses (ROI). En términos de mejora, el enmascaramiento se utiliza principalmente para aislar un área para el procesamiento. Esto se hace para resaltar esa área y diferenciarla del resto de la imagen. Las operaciones lógicas también se utilizan con frecuencia.
en conjunción con operaciones morfológicas.

La operación AND toma dos imágenes binarias o en niveles de grises como entrada, se leen de arriba hacia abajo los pixeles de cada imagen en la posición (*x*, *y*), con los dos valores con el operador lógico AND (&) de algún lenguaje de programación de realiza dicha operación bit a bit de los valores para producir una tercera imagen que cumple la siguiente tabla

Pixel A	Pixel B	AND(bit pixel nuevo)
0	0	0
0	1	0
1	0	0
1	1	1

Las ecuaciones de esta operación para imágenes en niveles de gris y binaria se muestran a continuación.

$$f_{ANDg}(x,y) = f_{g1}(x,y)\&f_{g2}(x,y) \tag{2.21}$$

$$f_{ANDb}(x,y) = f_{b1}(x,y)\&f_{b2}(x,y) \tag{2.22}$$

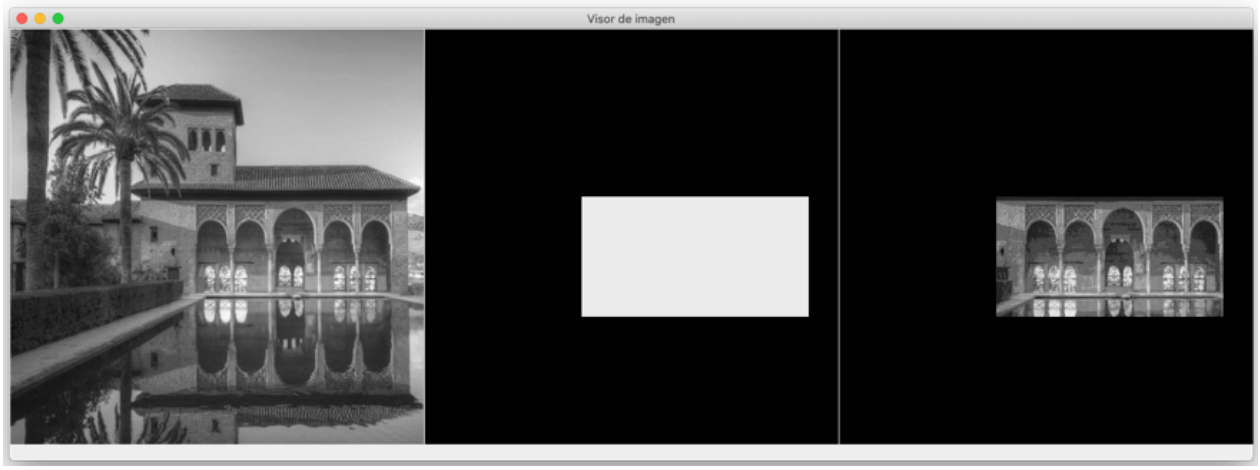

Figura 2.3 Enmascaramiento AND para obtener una porción de interés.

Figura 2.4 Composición de dos imágenes mediante operación lógica AND.

2.2.2. OR (|)

La operación OR se utiliza para enmascarar; es decir, para seleccionar subimágenes en una imagen, como se ilustra en la figura 2.5, también se puede hacer la operación entre dos imágenes diferentes para crear una tercera (figura 2.6). En la máscara de imagen OR, la luz representa un binario 1 y oscuro representa un 0 binario. El enmascaramiento a veces se denomina región de procesamiento de intereses (ROI). En términos de mejora, el enmascaramiento se utiliza principalmente para aislar un área para el procesamiento. Esto se hace para resaltar esa área y diferenciarla del resto de la imagen. Las operaciones lógicas también se utilizan con frecuencia.
en conjunción con operaciones morfológicas.

La operación OR toma dos imágenes binarias o en niveles de grises como entrada, se leen de arriba hacia abajo los pixeles de cada imagen en la posición (*x*, *y*), con los dos valores con el operador lógico OR (|) de algún lenguaje de programación de realiza dicha operación bit a bit de los valores para producir una tercera imagen que cumple la siguiente tabla

Pixel A	Pixel B	OR(bit pixel nuevo)
0	0	0

0	1	1
1	0	1
1	1	1

Las ecuaciones de esta operación para imágenes en niveles de gris y binaria se muestran a continuación.

$$f_{ORg}(x,y) = f_{g1}(x,y) \mid f_{g2}(x,y) \tag{2.23}$$

$$f_{ORb}(x,y) = f_{b1}(x,y) \mid f_{b2}(x,y) \tag{2.24}$$

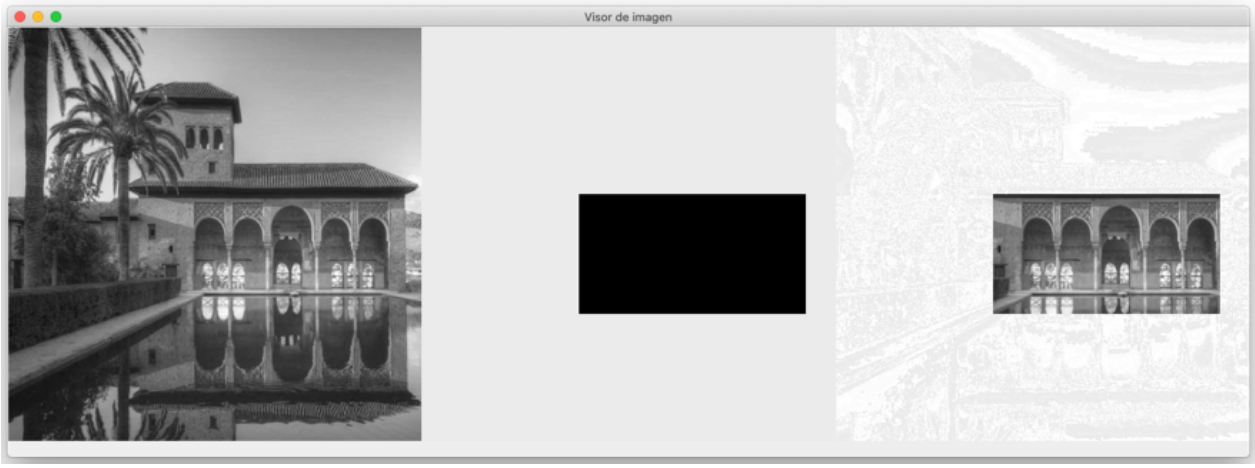

Figura 2.5 Enmascaramiento con operación OR.

Figura 2.6 Composición entre dos imágenes con la operación OR.

2.2.3. XOR (^)

Las figuras 2.7 y 2.8 muestran el enmascaramiento de la operación lógica XOR, también se puede hacer la operación entre dos imágenes diferentes para crear una tercera (figura 2.9). La operación XOR toma dos imágenes binarias o en niveles de grises como entrada, se leen de arriba hacia abajo los pixeles de cada imagen en la posición (*x*, *y*), con los dos valores con el operador lógico XOR (^) de algún lenguaje de programación de realiza dicha

operación bit a bit de los valores para producir una tercera imagen que cumple la siguiente tabla

Pixel A	Pixel B	XOR(bit pixel nuevo)
0	0	0
0	1	1
1	0	1
1	1	0

Las ecuaciones de esta operación para imágenes en niveles de gris y binaria se muestran a continuación.

$$f_{XORg}(x,y) = f_{g1}(x,y) \text{ ^ } f_{g2}(x,y) \quad (2.25)$$

$$f_{XORb}(x,y) = f_{b1}(x,y) \text{ ^ } f_{b2}(x,y) \quad (2.26)$$

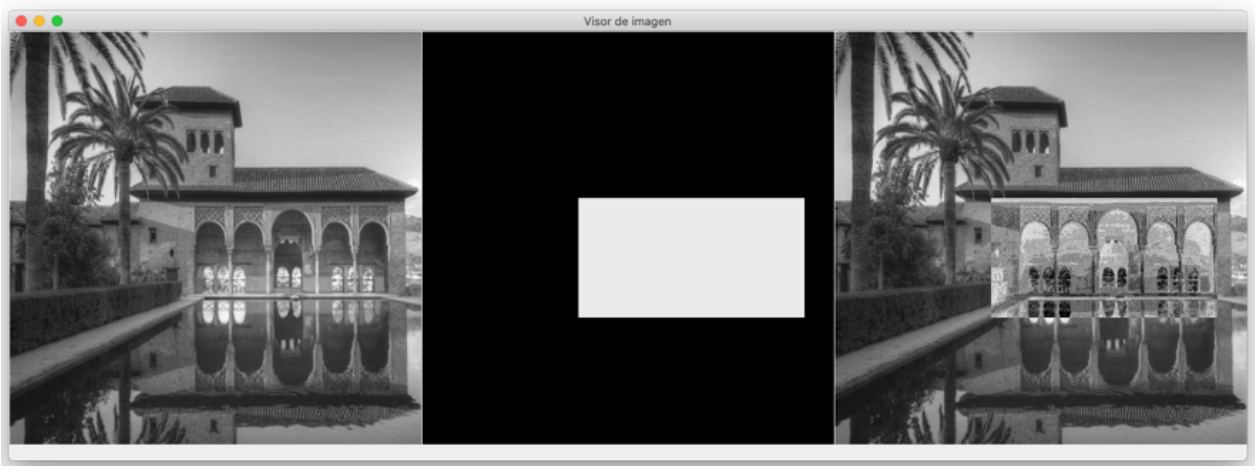

Figura 2.7 Enmascaramiento con operación XOR.

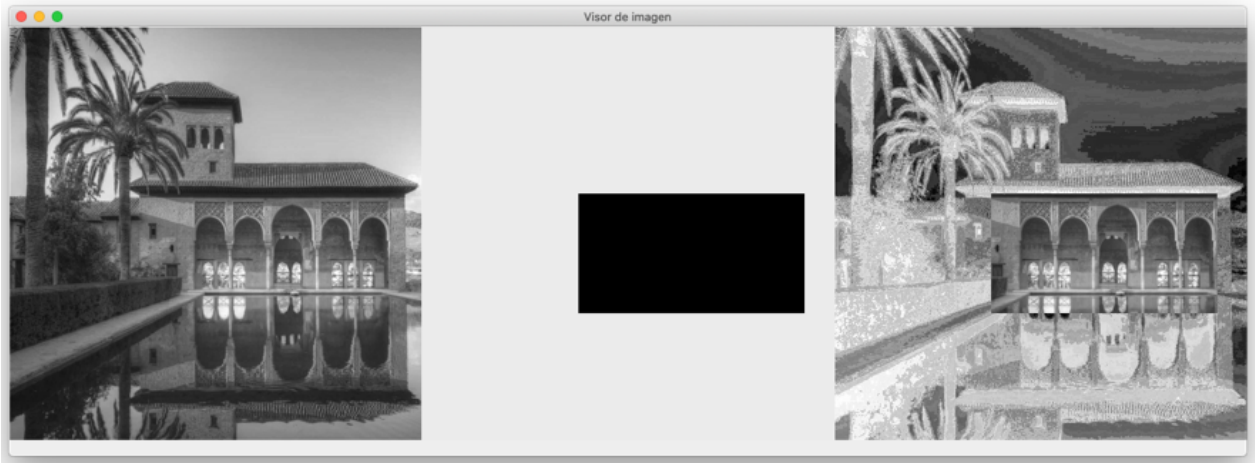

Figura 2.8 Enmascaramiento con operación XOR.

Figura 2.9 Composición entre dos imágenes con la operación XOR.

2.2.4. Negación NOT (~)

Por ejemplo, realizar la operación NOT en un píxel negro de 8 bits (una cadena de ocho ceros 0) y produce un píxel blanco (una cadena de ocho unos 1) vea la figura 2.10. Los valores intermedios se procesan de la misma manera, cambiando todo de 1 a 0 y viceversa. Por lo tanto, el operador lógico NOT realiza el mismo funcionamiento como la transformación negativa de la ecuación. La operación NOT toma una imagen binaria o en niveles de grises como entrada, se leen de arriba hacia abajo los pixeles de la imagen en la posición (*x*, *y*), con el valor del pixel con el operador lógico NOT (~) de algún lenguaje de programación de realiza dicha operación negando bit a bit los valores para producir una tercera imagen que cumple la siguiente tabla

Pixel	NOT(bit pixel nuevo)
0	1
1	0

Las ecuaciones de esta operación para imágenes en niveles de gris y binaria se muestran a continuación.

$$f_{NOTg}(x,y) = \sim f_g(x,y) \tag{2.27}$$

$$f_{NOTb}(x,y) = \sim f_b(x,y) \tag{2.28}$$

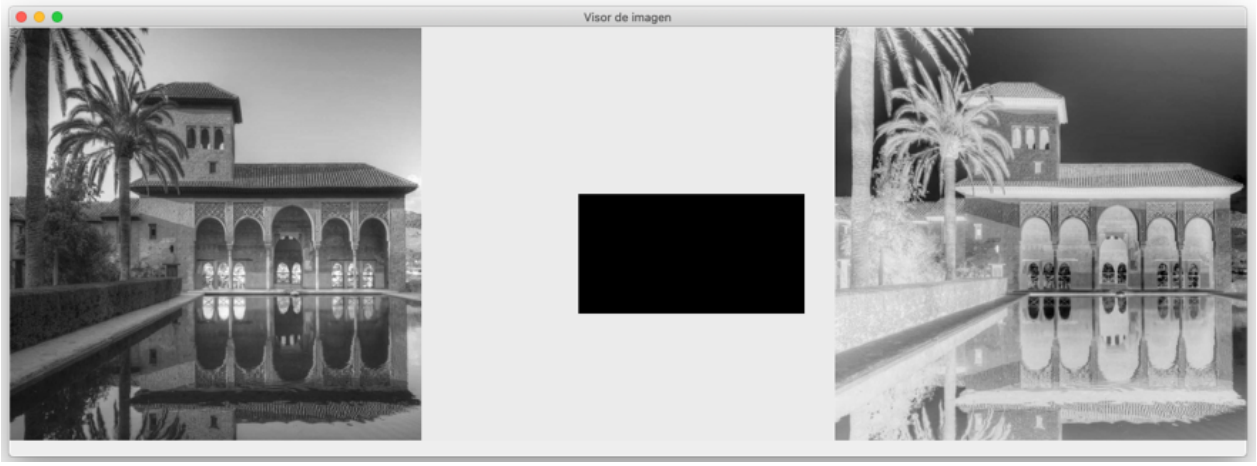

Figura 2.10 Negación de la primera imagen, la del centro solo es de relleno.

2.2.5. Operaciones relacionales

Los operadores relacionales se utilizan para expresar condiciones. Comparación entre dos imágenes con los operadores mostrados en la tabla 2.2

Tabla 2.2 Operadores relacionales entre imágenes.

Operador	Operación
==	Compara si dos imágenes son iguales (pixel a pixel en la posición (x, y)) f(x,y) == g(x,y)
<	Verifica si una imagen es menor a otra (pixel a pixel en la posición (x,y)) f(x,y) < g(x,y)
<=	Verifica si una imagen es menor o igual a otra (pixel a pixel en la posición (x,y)) f(x,y) <= g(x,y)
>	Verifica si una imagen es mayor a otra (pixel a pixel en la posición (x,y)) f(x,y) > g(x,y)
>=	Verifica si una imagen es mayor o igual a otra (pixel a pixel en la posición (x,y)) f(x,y) >= g(x,y)
¡=	Verifica si una imagen es diferente a otra (pixel a pixel en la posición (x,y)) f(x,y) ¡= g(x,y)

Este tipo de comandos (operadores) son más habituales junto a sentencias de programación como **if**, **while**, **for**, etc. Pero también se pueden utilizar con matrices (imágenes), lo que devolverá una matriz lógica (de ceros y unos) indicando los valores de las comparaciones que cumplan con las condiciones lógicas. Por ejemplo

$$A == B = C$$

$$A = \begin{bmatrix} 1, & 4, & 7 \\ 3, & 8, & 5 \\ 4, & 7, & 9 \end{bmatrix}, \text{B} = \begin{bmatrix} 0, & 3, & 7 \\ 5, & 8, & 2 \\ 2, & 7, & 6 \end{bmatrix}, \text{C=?}$$

Donde A, B y C son matrices como se muestra a continuación respectivamente

$$\begin{bmatrix} 1, & 4, & 7 \\ 3, & 8, & 5 \\ 4, & 7, & 9 \end{bmatrix} == \begin{bmatrix} 0, & 3, & 7 \\ 5, & 8, & 2 \\ 2, & 7, & 6 \end{bmatrix} = \begin{bmatrix} 0, & 0, & 1 \\ 0, & 1, & 0 \\ 1, & 1, & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 0, & 0, & 1 \\ 0, & 1, & 0 \\ 1, & 1, & 1 \end{bmatrix}$$

Ver ceros y unos en una matriz a lo mejor no tiene sentido, solo el de comparar números dentro de una matriz, y si cambiásemos esos unos por 255 para visualizar la matriz, tendríamos una imagen binarizada, es decir

$$C = \begin{bmatrix} 0, & 0, & 255 \\ 0, & 255, & 0 \\ 255, & 255, & 255 \end{bmatrix}$$

Las imágenes obtenidas de las operaciones se pueden observar en las figuras 2.11 a 2.16.

Figura 2.11 Operación de igualdad (==), entre dos imágenes.

Figura 2.12 Operación de menor que (<), entre dos imágenes.

Figura 2.13 Operación de menor o igual que (<=), entre dos imágenes.

Figura 2.14 Operación de mayor que (>), entre dos imágenes.

Figura 2.15 Operación de mayor o igual que (>=), entre dos imágenes.

Figura 2.16 Operación de diferente de (¡=), entre dos imágenes.