Desafío 1: implementación y uso de operaciones a nivel de bits

Elaborado por:

Thomas Mejía Moncada

Iván Darío Gonzalez Santana

Grupo: 06

Aníbal José Guerra Soler

Augusto Enrique Salazar

Profesores

Universidad de Antioquia “UdeA”

Facultad de Ingeniería

Programa de: ingeniería Electrónica/Telecomunicaciones

Curso: Informática II

Medellín

8 de abril del 2025

Introducción

En el cuerpo del texto se abordarán las posibles soluciones que usen operaciones a nivel de bits o técnicas aritméticas que permitan mantener la eficiencia de un programa destinado al procesamiento de imágenes distorsionadas producto de múltiples cambios a nivel de bits. El análisis y las ilustraciones graficas son incluidas para un mejor entendimiento del proceso que se quiere llevar a cabo para resolver el desafío.

Análisis textual del problema

Se tienen tres imágenes que hacen referencia respectivamente a:

1. I_D : Es la imagen final después de haber aplicado operaciones a nivel de bits a la imagen original.
2. I_m: Es una imagen aleatoria que servirá para aplicar las operaciones XOR, rotación de bits, y desplazamiento de bits sobre la imagen original o sobre la imagen resultante de otras operaciones anteriores (la llamamos Imagen Media porque se encuentra entre la imagen inicial y la imagen "resultado")
3. M: Es la imagen que hace referencia a la máscara cuya dimensión puede variar siempre y cuando sea menor a las dimensiones de la imagen inicial i_xj

La máscara como las operaciones de bits fueron usadas sobre la imagen inicial para producir una imagen distorsionada por ende debemos aplicar un proceso inverso que nos permita llegar desde la imagen distorsionada a la imagen original.

Para resolver este desafío identificamos que debemos reconocer que operación de bits fue usada sobre la I_D (esta parte la dividimos en dos, primero identificamos fue usada la máscara o si fue usada una operación a nivel de bits con I_m) para saber cual proceso fue usado en la transformación de la imagen tomaremos ayuda de la función proporcionada que retorna los pixeles en cada uno de sus componentes (RGB) de esta forma podremos aplicar una a una las posibles operaciones a nivel de bits que pudieron ser usadas a la hora de transformar la matriz, para explicar suponga que la función proporcionada F(x).

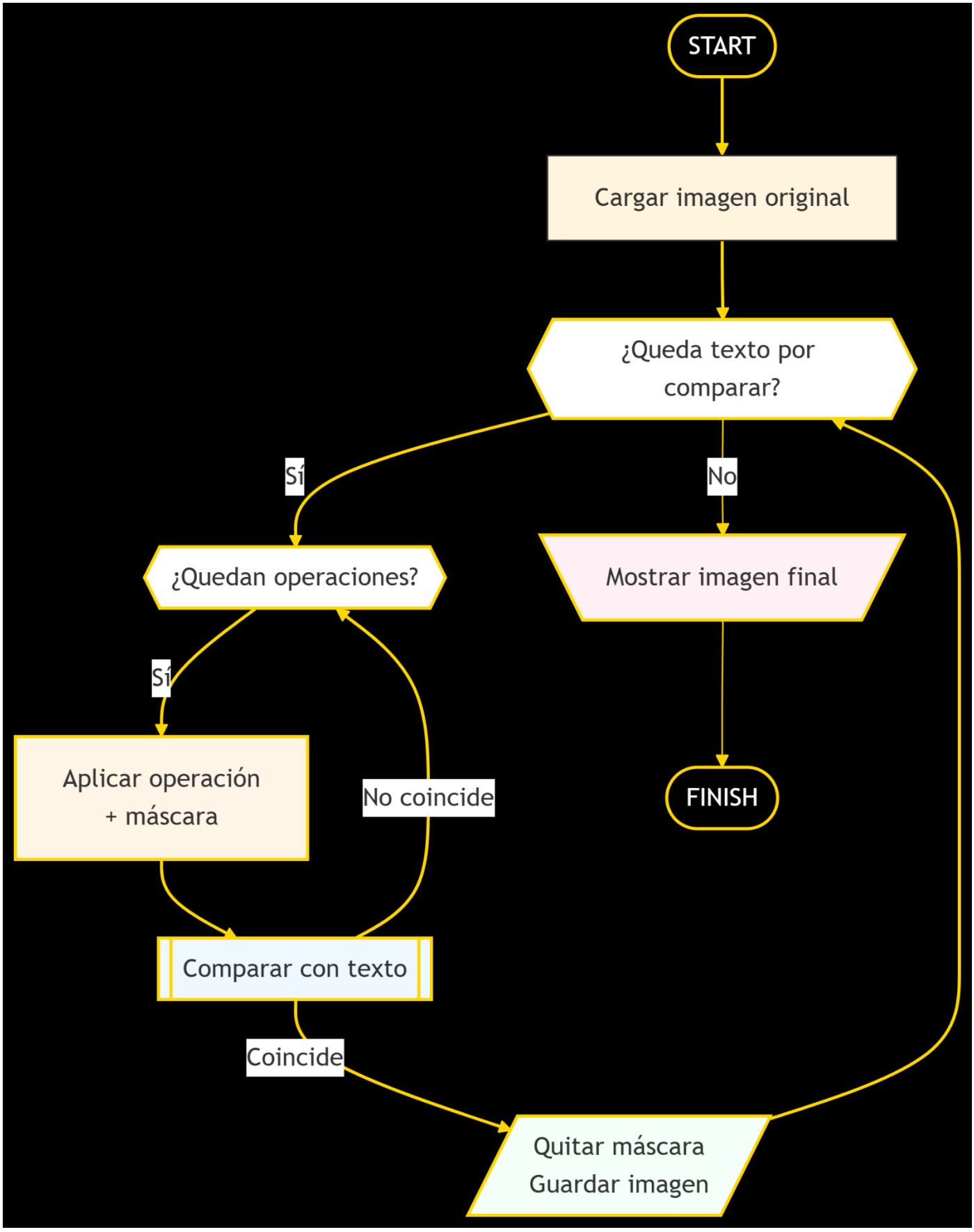
$$\mathbf{I_D \longrightarrow F(x) \longrightarrow [R,G,B]_D}$$

Con este arreglo que contiene las componentes de cada píxel de I_D hacemos lo siguiente (usaremos la palabra "bitwise" para referirnos a alguna operación a nivel de bits)

$$\mathbf{Bitwise \quad + \quad [R,G,B]_D \quad = \quad [R,G,B]_P}$$

Donde este "nuevo" $[R,G,B]_P$ se usará para ser comparado con los archivos .txt que fueron proporcionados y si todos los RGB son iguales entonces se asume que esa esa es una de las imágenes verdaderas dentro del proceso y se sigue trabajando sobre esa reasignando I_D a $[R,G,B]_P$ y así se sigue cíclicamente hasta encontrar la imagen original.

Análisis Grafico



Solución Al Problema

Como solución al problema proporcionaremos el uso de un ciclo anidado a otro ciclo, donde buscamos primero en el ciclo mas externo iterar sobre la cantidad de archivos del tipo (.txt) que hay dentro del data set. De esta forma iteramos solamente hasta tener la imagen $I_{O,}$ quiere decir que cuando no hay más (.txt) por comparar, luego en el ciclo interno se propone llevar a cabo iteraciones por cada posible operación de las transformaciones que se haya podido realizar sobre la imagen, luego se enmascara cada resultado y volvemos a la condición del ciclo donde comparamos el resultado obtenido con los pixeles del (.txt), comparando amos arreglos una vez determinemos que fue un paso correcto, removemos la máscara y guardamos salimos del ciclo y asignamos dicho resultado a nuestra nueva imagen $I_{D\ y}$ volveremos a repetir los pasos como se mencionan anteriormente mientras existan (.txt). Cuando no haya más documentos se asume que la imagen resultante es la imagen original (I_O)

Módulos

- Cargar datos
- Transformaciones (XOR, Rotaciones, Desplazamientos)
- Validaciones (validación de arreglos, validación de cantidad de txt)

Funciones por realizar

- Función para ver si hay archivos disponibles hasta encontrar la imagen final
- Función que se usara para las operaciones de trasformaciones en total serian 3 de este tipo una para cada transformación (xor, desplazamiento, rotación) que retorna el arreglo con los pixeles
- Función para el enmascaramiento retornara un arreglo con los pixeles
- Función para remover la mascara aplicada
- Función para convertir un array de enteros sin signo (unsigned int) a un array de caracteres sin signo (unsigned char).
- Función para hacer una conversión individual de char a int
- Función para la validación de los txt, comparar los arreglos

Realizando ese procedimiento se logrará pasar de una imagen altamente transformada por operaciones a nivel bits a una imagen visualmente clara por haber revertido toda la encriptación

Ejemplo:

IMAGEN ALTAMENTE TRANSFORMADA (I_D)

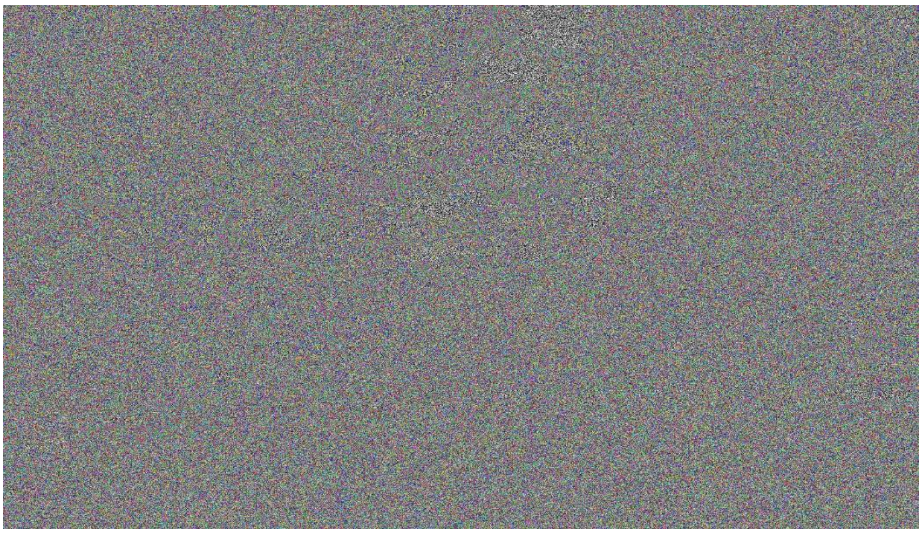


IMAGEN ORIGINAL DESPUES DE REVERITR LAS OPERACIONES A NIVEL BITS (I_O)

Conclusión

Este proceso permite revertir transformaciones complejas aplicadas a una imagen mediante operaciones a nivel de bits y enmascaramientos. A través de un sistema de ciclos que prueba combinaciones y compara con archivos de referencia, se identifica la secuencia de pasos usada para distorsionar la imagen. Al aplicar estas operaciones de forma inversa, se logra reconstruir progresivamente la imagen original. Esta solución modular y sistemática garantiza precisión, escalabilidad y una comprensión clara del proceso de desencriptación visual.

Este ejercicio demuestra cómo la lógica, la programación y la comprensión profunda de las operaciones a bajo nivel pueden ser herramientas poderosas para resolver desafíos complejos, como la recuperación de información encriptada. No se trata solo de aplicar funciones, sino de entender cómo cada transformación afecta los datos y cómo revertir ese efecto con precisión. Además, resalta la importancia de la organización modular del código y la validación paso a paso para garantizar que cada decisión nos acerque al resultado esperado. Más allá del reto técnico, esta experiencia nos enseña que incluso los procesos aparentemente caóticos pueden deshacerse con método, paciencia y razonamiento estructurado.