**Informe análisis desafio 1**

**Luis Miguel Alzate Rios**

**Jerónimo Herrera Espinosa**

**1. Introducción**

El propósito de este reto es reconstruir una imagen original a partir de una imagen modificada y una serie de enmascaramientos aplicados sobre ella. La imagen final se obtiene tras efectuar diversas transformaciones en la imagen original, y la tarea consiste en identificar esas transformaciones en el orden adecuado y revertirlas paso a paso.

Este informe detalla el análisis del problema, el enfoque secuencial inverso adoptado para resolverlo, y la forma en que cada módulo contribuye al proceso de reconstrucción.

**2. Descripción del problema**

En este reto, se nos proporciona una imagen transformada final (ID), una máscara (M), y una serie de archivos . txt (M1. txt, M2. txt, . . . , Mn. txt). Cada archivo . txt contiene los resultados de un enmascaramiento de una imagen transformada aplicada en cada etapa del proceso. Nuestra tarea es:

* Identificar qué transformaciones se aplicaron a la imagen original, en qué orden.
* Deshacer estas transformaciones de manera secuencial para reconstruir la imagen original.

Para hacerlo, debemos explorar las transformaciones posibles que fueron implementadas en la imagen original, verificar cuál se utilizó en cada etapa, y aplicar sus transformaciones inversas en el orden adecuado.

**3. Análisis del enfoque adoptado**

**Enfoque secuencial inverso**

En lugar de probar todas las combinaciones posibles de transformaciones y realizar un retroceso (lo cual sería ineficaz), hemos decidido adoptar un enfoque secuencial inverso. Esto significa que:

Comenzamos con la imagen final (ID).

Para cada archivo . txt (desde Mn. txt hasta M1. txt):

* Probamos las transformaciones posibles sobre la imagen actual.
* Verificamos si alguna transformación aplicada produce el mismo resultado que el contenido del archivo . txt.
* Cuando encontramos una transformación válida, sabemos que fue la última que se aplicó (en el proceso de distorsión), y por lo tanto es la primera que debemos deshacer.
* Aplicamos la inversa de esa transformación y pasamos al siguiente archivo . txt.
* Repetimos el proceso hasta que lleguemos a M1. txt, con lo cual habremos deshecho todas las transformaciones en orden inverso y recuperado la imagen original.

Este enfoque es más eficaz y directo, ya que no necesitamos explorar todas las combinaciones posibles de transformaciones.

**4. Descripción de los módulos**

**Módulo 1: Entrada y salida**

**Objetivo:** Cargar la imagen final transformada (ID), la imagen aleatoria (IM), y la máscara (M) desde archivos BMP, y cargar los archivos . txt con los resultados de enmascaramiento.

**Funciones:**

* loadPixels(. . . ): Carga una imagen BMP como un arreglo RGB plano.
* exportImage(. . . ): Exporta la imagen reconstruida como un archivo BMP.
* loadSeedMasking(. . . ): Carga el archivo . txt que contiene la semilla, el número de píxeles enmascarados, y los resultados del enmascaramiento.

**Módulo 2: Transformaciones**

**Objetivo:** Probar todas las transformaciones posibles sobre la imagen en cada etapa para identificar cuál de ellas produce el resultado de enmascaramiento esperado.

**Funciones:**

* aplicarXOR(. . . ): Aplica una transformación XOR entre la imagen y la imagen aleatoria.
* rotarBits(. . . ): Rota los bits de la imagen en una dirección específica.
* desplazarBits(. . . ): Desplaza los bits de la imagen en una dirección determinada.

**Módulo 3: Validación de Transformaciones**

**Objetivo:** Verificar qué transformación aplicada sobre la imagen produce el resultado del enmascaramiento conforme a los archivos . txt.

**Funciones:**

* validarEnmascaramiento(. . . ): Contrasta una imagen transformada con los resultados del archivo . txt para establecer si la transformación es válida.
* detectarTransformacion(. . . ): Evalúa las transformaciones posibles sobre la imagen y confirma cuál se alinea con el archivo . txt en ese momento. Registra la transformación identificada.

**Módulo 4: Reconstrucción**

**Objetivo:** Implementar las transformaciones inversas en el orden adecuado para restaurar la imagen original.

**Funciones:**

* aplicarTransformacionesInversas(. . . ): Ejecuta las transformaciones inversas en la secuencia correcta, comenzando desde el último archivo . txt hasta el primero.
* reconstruirSecuencial(. . . ): Organiza todo el proceso de reconstrucción, desde la identificación de las transformaciones hasta la implementación de las inversas.

**5. Conclusión**

En síntesis, el enfoque secuencial inverso que hemos adoptado resulta más eficaz que intentar todas las combinaciones posibles de transformaciones. Al utilizar los archivos . txt como guía, podemos identificar qué transformación se aplicó en cada etapa y revertirlas de manera progresiva para recuperar la imagen original.

Cada módulo desempeña una función concreta en el proceso de carga de datos, aplicación de transformaciones, validación y verificación de transformaciones, y finalmente la reconstrucción de la imagen.