Trabajo Práctico Especial

Teoría de la Información

Parte 2

Alumnos:

-Carbajo, Ezequiel:

LU: 248536

email: *eze-carbajo@hotmail.com*

- Gatica, Elias:

LU: 248480

email: *elias_g2009@hotmail.com*

Resumen

Continuando con el proyecto iniciado en la cursada 2019 de Teoría de la Información, se procedió a realizar un programa capaz de codificar y decodificar una imagen empleando estrategias estudiadas, asi como tambien lograr calcular el ruido que una imagen puede tener a causa de compresiones con pérdida (esto último implica la comparación entre la imagen real y la comprimida). Se utilizó además en la primer parte del trabajo especial, la librería JFreeChart (no incluida en el .zip de la entrega ya que cuenta con un peso que supera los límites permitidos del trabajo). Si bien no es utilizada en esta entrega, al importar la primer parte del trabajo para su uso en esta entrega, se recomienda agregarla al proyecto.

Introducción

El siguiente informe presentará el trabajo realizado por los alumnos Carbajo Ezequiel y Gatica Elías. El proyecto consistió en un programa codificado en Java capaz de recibir y procesar archivos en base a una acción dada por interfaz gráfica.

El programa ofrece las siguientes opciones al usuario:

- Comprimir una imagen a pedido del usuario.
- Descomprimir un conjunto de archivos para devolver una imagen ya comprimida.
- Comparar dos imágenes (original sin comprimir y nueva comprimida) para encontrar el ruido producido en la nueva a causa de la compresión con pérdida de la original.

Durante el desarrollo del informe se describe con mayor detenimiento cada uno de estos puntos mencionados.

Desarrollo

Compresión

Para ejecutar una comprensión de una imagen, presione el botón "Elegir imagen" y seleccione la imagen que desea comprimir. En la carpeta del ejecutable quedarán guardados todos los bloques comprimidos de la imagen y la imagen reconstruida mediante la decodificación.

El programa diseñado permite recibir una imagen en formato .bmp de ancho y alto múltiplos de 500. La imagen se fragmenta en bloques de dimensiones 500 x 500 y se realizará un tratamiento a estos, realizando una codificación utilizando los algoritmos sin pérdida previamente explicados, con el objetivo de comprimir la imagen original, almacenando como una serie de archivos de formato .huff o .rlc, según diferentes valores de entropía.

0	3	6
1	4	7
2	5	8

Imagen ilustrativa de cómo se determina el id de cada bloque

Para lograr la compresión de una imagen pueden aplicarse uno de dos métodos disponibles:

Codificación mediante Huffman:

Este algoritmo da prioridad en la codificación a aquellos símbolos que cuenten con una alta probabilidad de aparición. Estas probabilidades están dadas según la aparición de grises en un bloque. Las probabilidades de ocurrencia de cada uno se almacenan en el vector estacionario de cada bloque, el cual es utilizado para la creación de nodos que almacenan el tipo de gris y la probabilidad de aparición en el bloque, sin tener en cuenta aquellos grises cuya aparición en el bloque sea igual a 0 ocurrencias. Estos nodos posteriormente son ordenados de mayor a menor probabilidad y se utilizan para la construcción de un árbol de Huffman, mediante la combinación de manera recursiva de los nodos con menor probabilidad, asignándoles nodos padres que poseen la

suma de sus probabilidades.El uso de este árbol permite codificar cada símbolo (pixel) de la imagen uno por uno y concatenarlo en un string. El pseudocódigo se vería de la siguiente manera:

<creación de nodos con valores de grises y probabilidad a partir de vector estacionario, guardados en una lista>

<ordenamiento de la lista de mayor a menor probabilidad>

<creación del árbol de huffman>

El algoritmo de Huffman utilizado en este trabajo se considera semi-estático, ya que las probabilidades no son fijas, sino que se calculan según la fuente.

Codificación mediante Run Lenght Coding:

Este método sencillo realiza un recorrido por la imagen, pixel por pixel, detectando la repetición consecutiva de un símbolo (color). El código generado consiste en un número que indicará la cantidad de veces que aparece seguido un símbolo S, seguido del código identificatorio de dicho símbolo. Este formato permanecerá hasta terminar de recorrer toda la imagen.

A pesar de resultar mucho más sencillo, a causa de no requerir cálculos extensos como el armado de un árbol de Huffman, y poder generar un código con un solo recorrido de la imagen, normalmente termina resultando poco eficiente y ocupando más espacio que la imagen original si hay muchas variaciones en los tonos de la imagen. Hay que saber definir en qué momentos es conveniente su uso.

Descripción de encabezados:

Los bloques fueron almacenados en archivos y para su posterior decodificación se decidió guardar un encabezado, que posee una estructura diferente de acuerdo a si se codificó mediante Huffman o Run Lenght, además de los datos de la codificación de la imagen. La estructura del encabezado si se utilizó Huffman es la siguiente:

F	A	H	Px	Py	L	VE	Datos
1 bit	1 integer	1 integer	1 integer	1 integer	1 integer	255 double	L bits

F: Formato de compresión,indica si fue Huffman(valor de 1) o RLC si el valor es de 0.

A: Ancho en píxeles del bloque.

H: Alto en píxeles del bloque.

Px: Posición en x del bloque en la imagen original(siendo x el eje horizontal).

Py: Posición en y del bloque en la imagen original(siendo x el eje vertical).

L: Longitud de la cadena codificada en bits de los grises de la imagen.

VE: Vector estacionario, se utiliza en la decodificación.

Datos:Codificación de todos los grises.

F,A,H,Px,Py y L fueron convertidos a Strings de binarios, luego agregados a un arreglo de chars y posteriormente convertidos a bits. VE fue convertido de un arreglo de doubles a un arreglo de chars, y Datos fue creado como una concatenación de los códigos de grises, siendo posteriormente convertido a un arreglo de chars y finalmente a bits.

La estructura del encabezado si se utilizó RLC es la siguiente:

F	A	H	Px	Py	Rep1	Simb1	Rep..	Sim..	Repn	Simn
1bit	1 integer	1 integer	1 integer	1 integer	1 integer	1 integer	1 integer	1 integer	1 integer	1 integer

F: Formato de compresión,indica si fue Huffman(valor de 1) o RLC si el valor es de 0.

A: Ancho en píxeles del bloque.

H: Alto en píxeles del bloque.

Px: Posición en x del bloque en la imagen original(siendo x el eje horizontal).

Py: Posición en y del bloque en la imagen original(siendo x el eje vertical).

Rep1: Cantidad de repeticiones del primer símbolo.

Simb1: Valor de gris de la primer secuencia.

Repn:Cantidad de repeticiones del símbolo n.

Simbn: Valor de gris de la primer secuencia.

Decodificación:

Esta operación puede realizarse seleccionando el botón "Elegir archivos", y posteriormente eligiendo todos los archivos que representen fragmentos de la misma imagen.

La decodificación consiste en la interpretación de los archivos generados en una compresión previa, detectando de por medio el algoritmo con el que fueron codificados. La idea de la decodificación es que a partir de cada código se obtengan los fragmentos originales que componían a la imagen para después ensamblarlos y guardar en la carpeta del proyecto la imagen decodificada y rearmada.

Desgraciadamente ciertos errores en código difíciles de encontrar no nos permitieron generar correctamente las imágenes, por lo que su acción solo se limita a almacenar los fragmentos decodificados.

Transmisión de la imagen a través de un canal

Se conoce como ruido a la pérdida o equivocación en la transmisión de datos a través de un canal. La imagen marsSurfaceTp2 se considera la fuente de datos a transmitir y la imagen cS la imagen obtenida luego de su reconstrucción tras ser transmitida por un canal con ruido, el cual produce inconsistencias en la imagen. Para el cálculo del ruido producido por un canal, necesitamos conocer la probabilidad de emisión de un símbolo y(en la fuente destino) dado que se emitió un símbolo x en la fuente original(P (y/x)) y la probabilidad de emisión de un símbolo x de la fuente original (P(x)). Para ello contamos las ocurrencias de un pixel y en relación a un pixel x , creando una matriz condicional. Procedemos al cálculo del ruido, el cual está dado por la sumatoria del producto de la entropía por columna y probabilidad de cada x, dada por la siguiente fórmula:

$$H(y/x) = \sum_{x=0}^{n} px \times (- \sum_{y=0}^{n} p_{y/x} \times log_2(p_{y/x})) = \sum_{x=0}^{n} px \times hi$$

Resultados

El algoritmo de compresión utilizado por SIRDAL Ltd. devolvió una imagen con un ruido aproximado a 2. A pesar de que fue corroborado que los datos brindados en cabecera fueron bien codificados y decodificados, desgraciadamente las imágenes generadas no resultaron en la imagen original. A falta de tiempo no pudo buscarse la solución a dicho minucioso error y por ende tampoco pudo rearmarse la imagen con los bloques generados. De cualquier modo estos archivos podrán generar un archivo .bmp en caso de desearse, que quedarán almacenados en la carpeta del proyecto.

Conclusión

Se puede corroborar que los archivos codificados por RLC, a pesar de ser más livianos que el fragmento de imagen que codifican, resultan más pesados que los creados por el algoritmo de Huffman. Esto se debe a que rlc es útil si se desea codificar una fuente que presenta poca variación en sus símbolos y por ende baja entropía. Mientras más secuencias uniformes de símbolos existan en la fuente, mejor trabajara RLC. En los bloques codificados de marsSurfaceTp2 obtuvimos que Huffman presentó una mejor compresión en todos los casos.