Transmisión de imágenes en canal acotado ocupando transformada de Wavelets y código de Huffman

Roberto Melita Henríquez, Estudiante Ingeniería Civil en Informática, UACh, roberto.melita@alumnos.uach.cl
Javier Rojas Herrera, Estudiante Ingeniería Civil en Informática, UACh, javier.rojas02@alumnos.uach.cl

***Resumen*—En este documento se presenta una solución al problema de transmisión de imágenes a través de un canal con ancho de banda acotado, para éste propósito se realizó un programa en python que comprime imágenes (lossy), logrando reducir el tamaño de estas de forma considerable para que puedan ser transmitidas. Además de un decodificador, para así poder recuperar la imagen comprimida al momento de la recepción de la señal. Con este ejercicio se obtuvo una tasa de compresión de aproximadamente 4 veces el tamaño original.**

***Abstract*--This paper presents a solution to the problem of transmission of images through a channel with limited bandwidth, for this purpose a program was created with Python that compresses images (lossy), achieving to reduce the size of these in a considerable way to That can be transmitted. In addition to a decoder, in order to recover the compressed image at the moment of reception of the signal. With this exercise a compression rate of approximately 4 times the original size was obtained.**

I. Introducción

Shannon nos introduce al mundo de la comunicación a través de un modelo (Fig. 1) que contiene todos los componentes básicos que influyen en la comunicación.

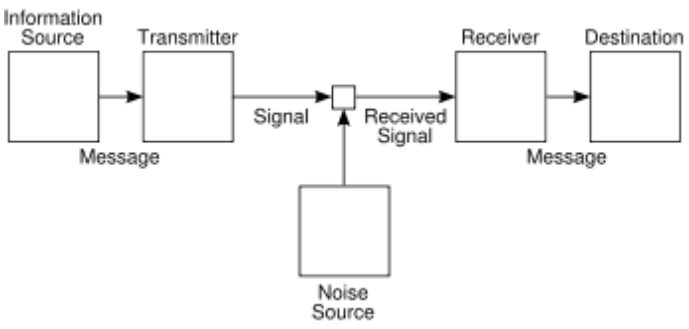

Fig. 1.Modelo de Shannon

En este trabajo se nos ha encargado centrarnos solamente en el apartado del transmisor y receptor, con una limitación del canal que tendrá un ancho de banda de 20 Mbps. Realizando un simple cálculo matemático, se puede definir una cota superior para el tamaño de las imágenes a comprimir:

Max = 20/(8*10) = 0.250 MBytes por imagen

Para poder resolver este problema se descompone el transmisor en 3 fases:

-Transformar
-Cuantizar
-Codificar

Hasta este punto tendremos las imagenes listas para ser enviadas por el canal acotado, para el receptor realizaremos las mismas 3 fases pero a la inversa:

-Decodificar
-Decuantizar
-Transformada inversa

II. Metodología

Para resolver el problema planteado, se hizo uso de herramientas matemáticas y algoritmos.

Transformada discreta Wavelet en 2 dimensiones(DWT): Esta transformada nos permite el análisis de una señal discreta en 2 dimensiones a través de la descomposición multiresolución de una imagen digital. Decidimos usar esta transformada ya que reconocidos estándares de compresión actuales la ocupan.

Transformada inversa discreta Wavelet en 2 dimensiones (IDWT): Función que nos permite volver del espacio Wavelet al espacio original de píxeles

Codificación de Huffman: Algoritmo usado para compresión de datos. El término se refiere al uso de una tabla de códigos de longitud variable para codificar un determinado símbolo, donde la tabla ha sido rellenada de una manera específica basándose en la probabilidad

estimada de aparición de cada posible valor de dicho símbolo.

Ahora pasamos a explicar cada una de las fases del programa.

TRANSFORMAR

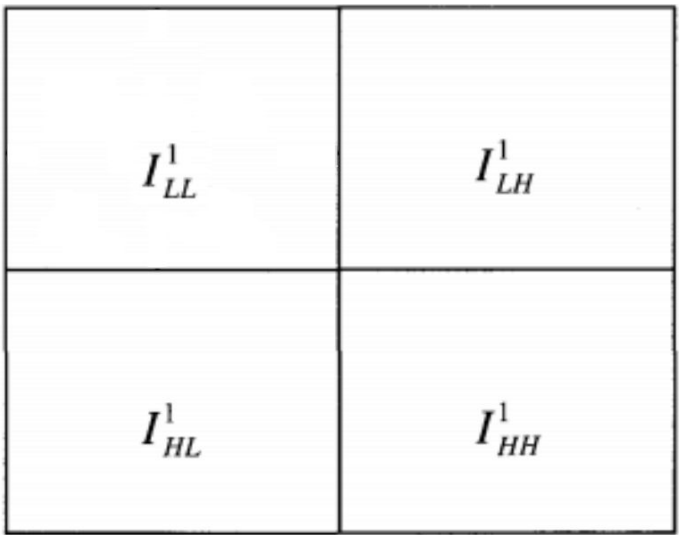

Fig. 2. Ejemplo de descomposición nivel 1 Wavelet.

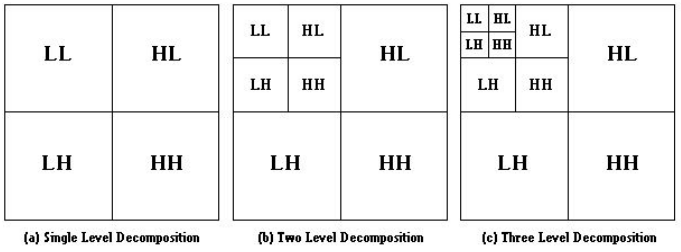

Fig. 3. Ejemplo de varias niveles de descomposición Wavelet. en 3 niveles

Para esta actividad, utilizamos la Transformada discreta wavelet en 2 dimensiones, la cual consiste en aplicar filtros de pasa baja y de pasa alta en filas y columnas a la imagen, descomponiendola en 4 sub imágenes como se muestra en la figura 2, la subimagen LL corresponde a aplicar un filtro pasa bajas en filas y posteriormente un filtro pasa bajas en columnas. La subimagen HL se obtiene al aplicar un filtro pasa altas en filas y posteriormente un filtro pasa bajas en columnas. La subimagen LH se obtiene aplicando un filtro pasa bajas en filas y posteriormente un filtro pasa altas en columnas y finalmente, la subimagen HH se obtiene al aplicar 2 filtros pasa altas, uno en filas y posteriormente en columnas, se puede volver a repetir el proceso sobre la subimagen LL, cada vez que se hace una repetición, diremos que se aumenta en 1 el nivel de descomposición, por lo tanto, si se hace este proceso de descomposición 2 veces, diremos que es una descomposición de nivel 2. En la figura 3 se pueden visualizar distintos niveles de descomposición.

Filtro pasa baja por filas (LowF): supongamos que tenemos la siguiente matriz de 2x4:

$$A = \begin{pmatrix} 2 & 4 & 6 & 8 \\ 1 & 3 & 5 & 7 \end{pmatrix}$$

El filtro pasa bajas por filas consiste que para cada fila, tomar los primeros 2 elementos y calcular una media aritmética, luego repetir el proceso para los siguientes 2 elementos, hasta no tener más elementos en la fila.

El Filtrado pasa bajas por filas de dicha matriz resulta:

$$LowF(A) = \begin{pmatrix} 3 & 7 \\ 2 & 6 \end{pmatrix}$$

Análogamente se puede realizar para el mismo proceso a la matriz A, pero por columnas:

$$LowC(A) = \begin{pmatrix} 1.5 & 3.5 & 5.5 & 7.5 \end{pmatrix}$$

Se puede observar que al aplicar un filtrado pasa bajas por filas, se obtiene una matriz con la mitad de las filas que la matriz original y al aplicar filtro pasa bajas por columnas se obtiene una matriz con la mitad de las columnas que la original.

Filtro pasa altas por filas : supongamos la misma matriz A usada anteriormente. El filtrado pasa altas por filas consiste en aplicar un filtro pasa bajas por filas, para luego hacer una diferencia entre la matriz original y el filtro pasa bajas de acuerdo a la siguiente expresión:

$$HighF_i = A_{2i} - LowF_i \quad , \quad \forall i \geqslant 0$$

Supongamos tenemos la siguiente matriz A:

$$A = \begin{pmatrix} 2 & 4 & 6 & 8 \\ 1 & 3 & 5 & 7 \end{pmatrix}$$

$$LowF(A) = \begin{pmatrix} 3 & 7 \\ 2 & 6 \end{pmatrix}$$

Se toma el primer elemento de la fila de A y se resta con el primer elemento de LowF, 2-3 =-1 . Luego se toma el tercer elemento de A y se resta con el segundo de LowF 6-7=-1, haciendo este procedimiento para cada fila:

$$HighF(A) = \begin{pmatrix} -1 & -1 \\ -1 & -1 \end{pmatrix}$$

Análogamente se puede realizar el mismo proceso pero por columnas, siendo este un filtro pasa altas por columnas HighC:

$$HighC(A) = \begin{pmatrix} 0.5 & 0.5 & 0.5 & 0.5 \end{pmatrix}$$

Se puede observar que la dimensión de HighF es la misma que la de LowF.

Regresando a lo que es la transformada wavelet discreta en 2 dimensiones, para calcular la subimagen LL, se debe aplicar a la imagen un filtro pasa bajas por filas, luego, a la matriz resultante se le debe aplicar un filtro pasa bajas por columnas. Supongamos la matriz A anterior:

$$A = \begin{pmatrix} 2 & 4 & 6 & 8 \\ 1 & 3 & 5 & 7 \end{pmatrix}$$

aplicando un filtro pasa bajas por filas, se obtiene :

$$LowF(A) = \begin{pmatrix} 3 & 7 \\ 2 & 6 \end{pmatrix}$$

y si volvemos a aplicar un filtro pasa bajas por columnas a LowF, obtenemos:

$$LowC(LowF(A)) = \begin{pmatrix} 2.5 & 6.5 \end{pmatrix}$$

siendo esta última, la matriz LL

Cuantización

Es en esta actividad es donde se produce una primera compresión, a través de una umbralización y un escalado.

Umbralización: Básicamente consiste en definir una función f(x), tal que si x es menor a un valor definido(umbral) entonces la función es 0. La utilidad de umbralizar es perder detalles que son poco visibles para los humanos (frec más altas) que en la descomposición wavelet se ve reflejado en las sub matrices LH HL y HH.

Escalado: Consiste en una función que retorna valores entre 0 y 30, la utilidad de esto, es permitir representar todos los valores de la imagen en este intervalo, permitiendo realizar de forma más sencilla la codificación de Huffman. Cabe destacar que en este proceso de escalado, se producen leves pérdidas.

Supongamos una matriz A que representa a una imagen, la escala estará dada por:

$$A[i,j,k] = A[i,j,k] * FS + FS$$

donde FS es el factor de escala, que para este trabajo utilizamos valores de 15, 25 y 50.

Codificación

Para este proceso se utiliza el método de codificación de Huffman, éste se realiza con el fin de poder comprimir los datos, esto quiere decir que podemos representar la misma información con menor cantidad de espacio requerido. La compresión ocurre gracias a que cambiamos el paradigma de que para cada símbolo de un lenguaje se utilice una codificación con un número de bits estáticos, en cambio con Huffman lo que hacemos es buscar un código que sea de un largo menor para los símbolos que tengan mayor probabilidad de ocurrencia, un punto importante es que éste código no tenga ambigüedad (que no se pueda confundir con los de los demás símbolos).

El algoritmo consiste en la creación de un árbol binario que tiene cada uno de los símbolos por hoja, y construido de tal forma que siguiéndolo desde la raíz a cada una de sus hojas se obtiene el código Huffman asociado a él. Para ello creamos un nodo hoja para cada símbolo y ordenarlos por probabilidad de ocurrencia, luego buscar los 2 nodos que tengan menor probabilidad y crear un nuevo nodo (padre) que contenga la suma de las probabilidades de sus nodos hijos, además de un camino hacia sus nodos hijos. Se vuelve a repetir este ciclo hasta que formaremos un árbol (Fig.3) que tendrá todas las conexiones entre los nodos y de él podremos extraer una tabla con todos los códigos asociados a los símbolos.

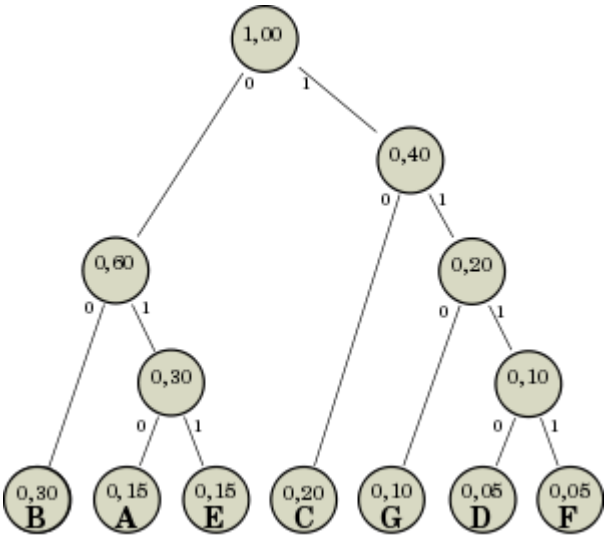

Fig. 4. Ejemplo de árbol binario de Huffman.

Para obtener el código debemos ir desde la raíz del árbol y recorrerlo hasta el símbolo requerido, guardando el 0 o 1 del camino entre nodos. Así para el árbol de ejemplo (Fig. 4) tendremos que para el símbolo ‘A’ su código será ‘010’.

Una vez obtenido este árbol extraemos una tabla con los códigos de todos los símbolos y lo descartamos.

Para nuestro ejercicio tendremos 31 símbolos, estos serán los números desde el 0 al 30. Una vez calculada la tabla de codificación procedemos a codificar en un archivo el resultado de la cuantificación. En este caso tomaremos cada una de las filas de la matriz (Imagen), las codificamos, transformamos el binario resultante a entero y lo guardaremos en un archivo “.huijse”, así para cada fila de cada canal.

Tendremos una tabla fija para estos símbolos, así que la tendremos guardada en el transmisor y en el receptor para poder cifrar y descifrar el mensaje.

Decodificación

Teniendo la tabla de codificación del proceso anterior, cargamos el archivo “.huijse”, luego iremos leyendo linea por linea el archivo, de el extraemos un número entero que transformaremos a binario, posteriormente comenzaremos a buscar coincidencias de los binarios que estamos leyendo con los de la tabla e iremos decodificando el mensaje. Esto ocurrirá hasta que leamos el carácter de salida, que indica el término de una línea, repetimos esto para cada línea del archivo. Una vez terminado tendremos la matriz (imagen) lista para ser decuantizada.

Decuantización

Para decuantizar, debemos aplicar una inversa a la escala hecha en la actividad de Cuantización.
Supongamos una matriz A que representa a una imagen escalada, inversa está dada por:

$$A[i,j,k] = \frac{A[i,j,k]}{FS} - FS$$

donde FS es el factor de escala.

No es posible desumbralizar, debido a que al aplicar la umbralización, se generan pérdidas irreparables.

Transformada Inversa

Esta actividad nos permite recuperar una imagen que se encuentra transformada(DWT), volvamos a la figura 2 donde tenemos la descomposición nivel 2 de la transformada wavelet, diseñamos un método que a partir de las 4 submatrices se puede reconstruir la matriz original. Para esto, primero definimos una operación llamada “réplica”, la cual simplemente consiste en repetir una vez cada valor, puede ser tanto en filas como en columnas, supongamos la siguiente matriz:

$$B = \begin{pmatrix} 1 & 2 \end{pmatrix}$$

una operación de replicación en filas queda:

$$ReplicaF(B) = \begin{pmatrix} 1 & 1 & 2 & 2 \end{pmatrix}$$

Análogamente una operación de réplica en columnas queda:

$$ReplicaC(B) = \begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix}$$

La idea de esta operación es aplicarla a cada submatriz, se aplica una réplica en columnas y a esa nueva matriz

replicada se le replica en filas, supongamos la siguiente matriz C:

$$C = \begin{pmatrix} 2 & 3 \\ 1 & 4 \end{pmatrix}$$

$$ReplicaF(C) = \begin{pmatrix} 2 & 2 & 3 & 3 \\ 1 & 1 & 4 & 4 \end{pmatrix}$$

$$ReplicaC(ReplicaF(C))) = \begin{pmatrix} 2 & 2 & 3 & 3 \\ 2 & 2 & 3 & 3 \\ 1 & 1 & 4 & 4 \\ 1 & 1 & 4 & 4 \end{pmatrix}$$

Lo interesante de efectuar esta operación , es que si se le realiza un filtro pasa baja por filas y luego por columnas a nuestra matriz replicada por filas y columnas, obtenemos la matriz C original planteada anteriormente.

$$LowF(LowC(ReplicaC(ReplicaF(C)))) = C$$

Una vez hecho este procedimiento para cada submatriz (LL, HL, LH, HH) hay que realizar un cierto cambio.

LL replicada: se mantiene igual.

LH replicada: los valores que se encuentren en las filas impares(contando desde 0 par) se multiplican por -1:

$$LHreplicada: \begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix}, queda \begin{pmatrix} 2 & 3 \\ -2 & 1 \end{pmatrix}$$

HL replicada: los valores que se encuentren en las columnas impares(contando desde 0 par) se multiplican por -1:

$$HLreplicada: \begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix}, queda \begin{pmatrix} 2 & -3 \\ 2 & -1 \end{pmatrix}$$

y finalmente HH replicada: los valores de columnas impares que se encuentren en una fila par y los valores de columnas pares que se encuentren en filas impar se multiplican por -1:

$$HHreplicada: \begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix}, queda \begin{pmatrix} 2 & -3 \\ -2 & 1 \end{pmatrix}$$

una vez realizado este cambio a cada submatriz replicada, se procede a realizar una suma de matrices:

$$Inversa = LLrep + LHrep + HLrep + HHrep$$

obteniendo así, la imagen original.

III. Análisis y resultados

Se realizó pruebas para distintos niveles de wavelets y distintos niveles de escalamiento, calculando el error cuadrático medio en cada uno de ellos y el peso en MB, tal y como se muestra en la figura número 303

Nivel de Wavelet	Escala (intervalos)	Peso ".huijse" MB	Error Cuadrático	Tiempo comp. Mins.
2	15	1.06	0.055	2
2	25	1.2	0.038	2.3
2	50	1.53	0.021	2.8
3	15	0.959	0.067	2.4
3	25	1.04	0.046	2.6
3	50	1.3	0.026	2.9

Tabla 1. Comparativa entre diferentes niveles de wavelets y escalas.

Se logra observar que a medida que aumentan los niveles de wavelet, aumenta la compresión junto con el error cuadrático. Por el contrario, cuando se incrementa la escala de cuantización, se disminuye la compresión junto al error cuadrático. El tiempo de ejecución varía entre 2 a 3 mins, lo que implica un tiempo de ejecución bastante aceptable.

Por lo tanto, si se desea transmitir 10 imágenes de 0,959 MB en 1 segundo, con este método se logra con un ancho de banda de 12,5 MB con tan solo un error del 6,7%.

Lamentablemente para este trabajo, para que se pueda transmitir las 10 imagenes por el canal, el tamaño de cada una debe ser menor o igual a 250 kb.

Fig. 5. Comparación entre la imagen original y una compresión usando un nivel 2 de wavelet con un escalado de 25.

Fig. 6. Comparación entre la imagen original y una compresión usando un nivel 3 de wavelet con un escalado de 50.

Fig. 7. Comparación entre la imagen original y una compresión usando un nivel 3 de wavelet con un escalado de 15.

IV. Conclusión

De este trabajo, podemos concluir que la compresión de imágenes a través de la transformada wavelet entrega niveles de compresión de hasta 4 veces manteniendo una calidad aceptable con un tiempo de ejecución moderado. Cabe destacar que realizando una optimización de la codificación de huffman, se lograrían severas mejoras en la compresión, así como también en la disminución del tiempo de ejecución.

También se debe mencionar, que el algoritmo propuesto trabaja sobre 1 núcleo, esto limita la reducción en el tiempo de ejecución, por lo tanto, si se trabaja con paralelismo con openmp o programación de GPU, se lograría reducir bastante el tiempo de ejecución .

V. Bibliografía

[1] P. Estrada “Comprimiendo datos - el algoritmo de Huffman en Python”, http://bitybyte.github.io/Huffman-coding/ .

[2] A. S. Lewis and G. Knowles, “Image compression using the 2-D wavelet transform,” IEEE Transactions on image Processing, vol. 1, no. 2, pp. 244–250, 1992.

[3] I. Bañó, “WAVELETS: UN MÉTODO DE COMPRESIÓN DE IMÄGENES”, pp .8-10,1996.

[4] G. Puetamán Guerrero y H. Salazar Escobar, “Compresión de imágenes usando wavelets”, Universidad EAFIT, 2007.