Comprimidor de Imágenes Utilizando Pirámide Laplaciana

Joaquín Farias, *Estudiante*

Resumen—En Este informe se presenta una aplicación para comprimir imágenes utilizando la descomposición de Pirámide Laplaciana. Se explicará la descomposición y la reconstrucción de la imagen ocupando este método, introduciendo un nuevo algoritmo para comprimir la información guardada de la imagen. También, se hará un análisis del error de reconstrucción con respecto al porcentaje de compresión.

Palabras clave—Comprimidor, Pirámide Laplaciana, Pirámide Gaussiana, Procesamiento de imágenes.

1 Introducción

La transformación mediante Pirámide Laplaciana es un algoritmo que descompone una imagen (o una base de datos) utilizando la **Pirámide Gaussiana**, primeramente. Esta Pirámide Gaussiana se forma aplicándole un filtro Gaussiano a la imagen para luego decimarla, así cada nivel más alto de la pirámide tiene dimensiones porcentualmente más pequeñas que su nivel superior.

La **pirámide Laplaciana** se forma haciendo una expansión de la pirámide Gaussiana y sumando la matriz del mismo nivel, así obteniendo los detalles de la imagen de forma con variación más grandes y acercando los todos los valores a cero.

2 Motivación y descripción del algoritmo de Pirámide laplaciana

El algoritmo de Pirámide Laplaciana se divide en tres partes principalmente:

1. **Pirámide Gaussiana.**
2. **Expansión de pirámide Gaussiana.**
3. **Pirámide Laplaciana.**

Estas tres funciones se relacionan directamente con funciones específicas:

1. **REDUCE.**
2. **EXPAND.**
3. **LAPLACIANA.**

Todas estas partes y funciones serán explicadas en este capítulo.

2.1 Pirámide Gaussiana

La pirámide Gaussiana se forma aplicándole un **filtro gaussiano** a la imagen de dimensiones MxN para luego decimarla.

El filtro Gaussiano ocupado se define en [3] como un filtro de 5x5 y tiene los siguientes parámetros:

$$\widehat{w}(0) = a$$

$$\widehat{w}(-1) = \widehat{w}(1) = 1/4$$
$$\widehat{w}(-2) = \widehat{w}(2) = 1/4 - a/2$$

El valor de $\boldsymbol{a}$ será de **0,4** por pruebas hechas en [3], este valor logra una óptima distribución Gaussiana como es lo requerido.

La decimación, en este caso, se hará disminuyendo las dimensiones de la imagen a la mitad. Esto quiere decir, que se borraran los pixeles 2n-1 con n de 1 hasta N/2 o M/2, dependiendo el caso.

Todo lo anterior se hace en la función **REDUCE** en el programa.

2.2 Expansión pirámide Gaussiana

Luego de formar nuestra pirámide Gaussiana, se deben expandir estas imágenes para formar una imagen borrosa un nivel más bajo de la pirámide Gaussiana. Para hacer lo anterior simplemente se hace el **procedimiento inverso de la función REDUCE**.

Para poder hacer el procedimiento inverso de la decimación se rellena con ceros la imagen G_{l-1} para poder volver a las dimensiones de la imagen G_l. Luego, se filtra esta imagen expandida con el mismo filtro gaussiano y se logra una imagen borrosa de la imagen G_l.

Todo el procedimiento anterior se hace en la función **EXPAND** de nuestro programa.

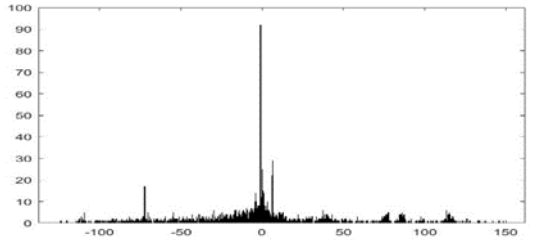
2.3 Pirámide Laplaciana

Para formar la pirámide Laplaciana se ocupan las imágenes de la pirámide Gaussiana y las imágenes expandidas de esta. Así, se ocupa la siguiente fórmula para formarla

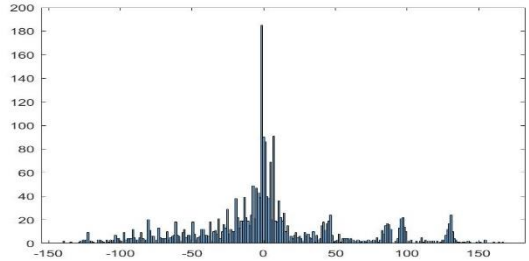
$$L_l = G_l - EXPAND(G_{l+1})$$

La función asociada a este procedimiento es llamada **LAPLACIANA** en nuestro programa.

Se nota que para reconstruir la imagen solo se necesita la imagen G de nivel más bajo y toda la pirámide Laplaciana dado que con la función EXPAND se expando las imágenes para luego ocupar la función LAPLACIANA y así formar en el nivel más bajo la imagen reconstruida, esto es comprobable algebraicamente. La función asocia-



(a)



(b)

Fig. 2. Histograma de imagen L_5. (a) Histograma de imagen original. (b) Histograma de imagen comprimida.

da a este procedimiento es llamada **RECONSTRUCCION,** de la cual se hablará más adelante.

3 ALGORITMO DE COMPRESIÓN

En este capítulo se describirá el algoritmo se ocupó para la compresión de la imagen, todo este procedimiento está asociado a la función COMPRIMIR del programa.

Para comprimir primero se debe notar que las imágenes de la pirámide laplaciana tienen un histograma donde los valores que más se repiten se acercan a cero, así es fácil darse cuenta de que una manera instintiva de comprimir estas imágenes L es disminuir el número de BINS que tiene su histograma, ya que los valores son cercanos. Este procedimiento se ejemplifica en la Figura 1, donde la imagen L_5 tiene un total de 2376 BINS y $L_{3\ comprimida}$ tiene un total de 320 BINS. Lo anterior se logra dividiendo y redondeando los valores de cada BIN.

Específicamente en este trabajo se hizo una compresión adaptativa en relación con el nivel de la matriz Laplaciana donde el porcentaje en que se dividen los L más alto va disminuyen de manera lineal. Lo anterior se describe con la siguiente formula:

$$Pc_{l+1} = Pc_l - |Pc_1/l_{Max}| \cdot l$$

Esto se hace basado en los estudios en [3], donde se afirma que mientras el laplaciano sea de mayor nivel su entropía es mayor, por lo que se emplea mayor porcentaje de BINS ocupados para la compresión.

Para asociar el número de BINS ocupados con el porcentaje de compresión que ingresara el usuario al iniciar el problema se hizo un estudio entre el número de BINS ocupados y el porcentaje de compresión llegando al grafico de la Figura 2.

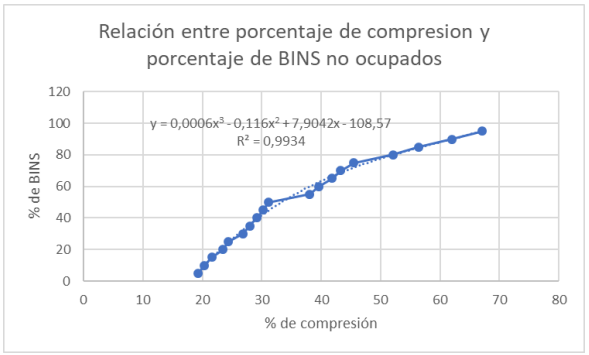


Fig. 2. Grafica del porcentaje de BINS ocupados respecto a la compresión de la imagen.

De el grafico mencionado se hacer una regresión polinómica para dar con la ecuación que se ocupara en este programa, esta es:

$$Pc = 0{,}0006Pb^3 - 0{,}11Pb^2 + 7{,}9042Pb - 108{,}57$$

Donde Pc es el porcentaje de compresión de la imagen y Pb el porcentaje de BINS no ocupados del total.

4 ALGORITMO DE CODIFICACIÓN

Para la codificación se ocupó el algoritmo de Huffman [4], el cual genera un diccionario binario para cada símbolo que se requiera. Luego de formar el diccionario se convierten todos los datos de las matrices L y la última G con este diccionario, notando que el diccionario cambia para cada imagen. Finalmente, se guarda todo una un solo vector, además de las dimensiones de cada L y G codificada para luego ocuparlas como encabezado.

Se guardan estos datos codificados en binario en un archivo .bin con la función **fwrite** de *Matlab,* además se guardan los encabezados necesarios para la decodificación en un archivo .dat con la función **save** de *Matlab.*

Todo lo anterior está asociado a la función CODIFICAR de nuestro código.

5 ALGORITMO DE DECODIFICACIÓN Y RECONSTRUCCIÓN

Para hacer el procedimiento de decodificación y reconstrucción solo se necesita hacer le procedimiento inverso de las funciones COMPRIMIR y CODIFICAR.

En el procedimiento inverso de decodificación se leen los datos del archivo .bin y los encabezados del archivo .dat para volver a formar las matrices de datos con su tamaño original, luego se les aplica la función inversa de la codificación de huffman con el diccionario guardado como encabezado. La función asociada a este procedimiento es **DECODIFICAR.**

En el procedimiento inverso de comprimir se multiplica por el cuantizador ocupado para bajar el número de

BINS para, luego, hacer el procedimiento inverso de la matriz laplaciana y llegar a la imagen original. La función asociada a este procedimiento es la **RECONSTRUCCION.**

6 PRUEBAS Y ANÁLISIS

Dentro de este capítulo se presentarán los resultados de la aplicación de nuestro programa y el análisis de error.

3.1 Prueba pirámide Gaussiana, pirámide Laplaciana e imagen Expandida

Luego de realizarse pruebas en la imagen original (Fig. 3 (a)) para encontrar las imágenes de la pirámide Gaussiana y la Laplaciana se nota que el resultado es el esperado.

Se noto también, que cuando las iteraciones de la pirámide Gaussiana son muchas, llega un límite donde la imagen de nivel más alto no tiene ningún cambio y primero se acerca al filtro Gaussiano, para luego ser una imagen negra sin información. Lo mismo ocurre para los niveles más altos de la pirámide Laplaciana.

En la Fig. 3 se muestran los resultados para el nivel 5 de ambas pirámides y de la expansión de la pirámide Gaussiana, como ejemplo.

.

3.3 Resultados y ejemplos de reconstrucción de imagen comprimida

Los resultados de pruebas hechas con una imagen con distintos niveles de compresión se muestran en la Fig. 5. Se nota que el resultado es bastante bueno, pero a medida que más se comprime la imagen el error aumenta.

4.2 Análisis de error

Para analizar el error entre la imagen original y la comprimida se utilizó el error cuadrático entre todos los pixeles de la imagen. Así, se hizo pruebas con varios valores de compresión para poder formar una gráfica de esta relación y así estimar cual sería la compresión optima.

El estudio antes dicho se refleja en el gráfico de la Fig. 5. Se nota por este grafico que un valor óptimo de compresión seria 30%, ya que el error es menor a 8%, y porque un porcentaje mayor se compresión el error pega un salto a casi 10%.

Finalmente, en la Fig. 6 se muestra la imagen original y la imagen comprimida con una compresión de 30%, es visible la buena calidad de la imagen comprimida.

6 CONCLUSIONES

Los resultados obtenidos en este informe son satisfactorios, ya que se alcanzó un buen nivel de compresión sin comprometer tanto error en la imagen comprimida con respecto a la original.

También se notó que el tiempo computacional que lleva tiene este algoritmo no es excesivamente grande, ya que solo funciona en base a iteración de funciones básicas.

Como pregunta y estudio a futura se tiene la idea de implementar una codificación RLE sobre la codificación huffman hecha. Esto con la idea de que en el sistema bi-

(a)

(b)

(c)

(d)

Fig. 4 Ejemplo del algoritmo con una imagen. (a) Imagen original. (b) Imagen de pirámide Gaussiana G5. (c) imagen expandida E5. (d) imagen de pirámide Laplaciana L5.

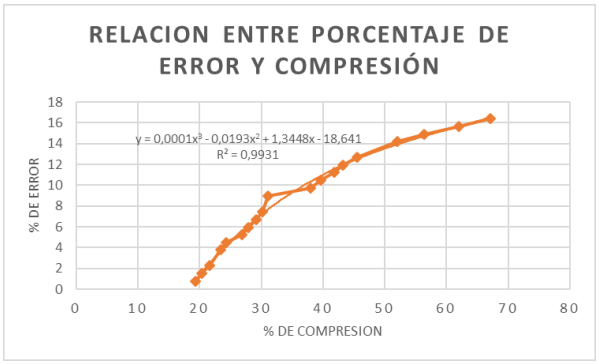


Fig. 5. Gráfico de mapeo de colores que representa el error con 1% de valores singulares ocupados.

nario solo existen dos dígitos por lo que es muy probable que haya varios dígitos iguales repetidos. Con un estudio preliminar se notó que con más de siete dígitos iguales consecutivos este método resulta útil, aunque estas pruebas son preliminares y nada concluyentes.

REFERENCIAS

[1] Boyd, S., & Vandenberghe, L. (2018). Introduction to applied linear algebra: vectors, matrices, and least squares. Cambridge university press.

[2] Norvig, P., & Russell, S. (2004). Inteligencia artificial. Editora Campus, 20.)

[3] Burt, P., & Adelson, E. (1983). The Laplacian pyramid as a compact image code. IEEE Transactions on communications, 31(4), 532-540.

[4] Knuth, D. E. (1985). Dynamic huffman coding. Journal of algorithms, 6(2), 163-180.

Joaquín Farias Muñoz Estudiante de ingeniería civil electrónica, Pontificia Universidad catolica de Valparaíso.

(a)

(b)

(c)

Fig. 6. Relación entre imagen original e imagen comprimida con un 30% de compresión. (a) imagen original. (b) Imagen con un 30% de compresión. (c) Imagen comprimida con un 50% de compresión.