ONDULETAS Y FRACTALES EN LA COMPRESION DE IMAGEN Y DE VIDEO

**Alejandro Martínez Ramírez * Alejandro Díaz Sánchez,*
** Mónico Linares Aranda ,**Javier Vega Pineda.*

* INAOE Departamento de Electrónica, Tonantzintla Puebla, México
** Instituto Tecnologico de Chihuahua, Chihuahua, México

ABSTRACT

En este articulo se muestran varios métodos de compresión de imágenes, con onduletas y fractal. Se analizan las posibilidades de aplicación de las mismas en sistemas inalámbricos, donde los aspectos de complejidad y velocidad se contemplan como parámetros directrices. Se presentan también algoritmos para implementar arquitecturas para realizar la compresión de imágenes y video

1. INTRODUCTION

La compresión de imagen y video ha incrementado su importancia con el advenimiento de sistemas multimedia. Sus aplicaciones en áreas como la enseñanza multimedia, los visores para imágenes/video, video conferencias y, recientemente, en la telefonía celular han sido de particular interes en fechas recientes. El propósito de la compresión de video e imágenes es reducir la razón de bits para la transmisión o el almacenamiento manteniendo una calidad aceptable. Se han desarrollado diferentes técnicas de compresión, como la codificación predictiva, la codificación por transformadas, la codificación mediante el vector de cuantización y la codificación fractal. De especial interés son la codificación mediante la transformada onduleta, usada en el estándar JPEG2000 para la codificación de imágenes, y la técnica de codificación fractal.

Para utilizar la transformada onduleta para aplicaciones de codificación de imagen, se necesita un proceso de codificación que incluya tres grandes aspectos: descomposición de los datos de la imagen, cuantización de los coeficientes transformados, y codificación de los coeficientes transformados cuantizados. La Figura 1 muestra un diagrama de bloques simplificado de este proceso.

Figura 1.- Diagrama de bloques de la codificación de imagen con la transformada onduleta

La descomposición de la imagen es usualmente un proceso sin pérdidas, el cual convierte los datos de la imagen desde su dominio espacial, al dominio de la frecuencia, donde los coeficientes transformados están no correlacionados. La pérdida de información se produce en el paso de cuantización, y la compresión es llevada a cabo en el paso de codificación. Para empezar la descomposición, los datos de la imagen se parten primero en cuatro subbandas, etiquetadas como LL1, HL1, LH1, y HH1, tal y como se muestra en la Figura 2 a).

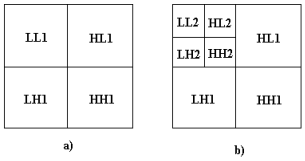

Figura 2.- Transformada onduleta 2-D. a) Primer nivel de descomposición. b) Segundo nivel de descomposición.

Cada coeficiente representa un área espacial correspondiente a un cuarto del tamaño de la imagen. Para obtener el siguiente nivel de descomposición, la subbanda LL1 es descompuesta aún más, al siguiente nivel de cuatro subbandas, tal y como se muestra en la Figura 2 b). Esta descomposición puede continuarse a tantos niveles como sea necesario o como los datos de la imagen lo permitan. Los filtros necesarios para computar la transformada onduleta discreta generalmente son filtros espejos simétricos en cuadratura (QMF, por sus siglas en inglés) [1].

Durante la cuantización, cada sub-banda es procesada de modo diferente, dependiendo de su importancia, la cual generalmente se basa en su energía o varianza [2]. Para alcanzar la razón de bits determinada o razón de compresión, deben usarse pasos de cuantización largos para cuantizar las subbandas de baja energía, y pasos pequeños para las sub-bandas de gran energía. Esto resulta en unos cuantos bits para localizar aquellas subbandas de poca energía y más bits para las de mayor energía. La mayoría de los coeficientes onduleta tienen

muy baja energía, por lo que muchos de los componentes de alta frecuencia son cuantizados a cero.

En el caso de la codificación fractal de imágenes, un sistema de funciones iteradas puede ser usado para generar una imagen única, la cual es conocida como el atractor del sistema de funciones iteradas. En otras palabras, esta imagen puede ser representada en forma más simple mediante los parámetros del sistema de funciones iteradas (IFS). Por ello, se puede usar un procedimiento inverso para generar un conjunto de transformaciones, esto es, un IFS desde una imagen, entonces, esas transformaciones o el IFS, puede ser utilizado para representar una aproximación de la imagen. El sistema de codificación de imagen puede usar los parámetros de las transformaciones en el IFS en lugar de los datos de la imagen original para almacenamiento o para transmisión. Dado que el IFS contiene una cantidad de datos muy limitada tal como los parámetros de las transformaciones, este método de codificación de imagen puede resultar en una muy alta razón de compresión. Se han propuesto tres métodos para obtener IFS [3]. Uno es el método directo que encuentra de manera directa un conjunto de transformaciones afines contractivas desde la imagen, basado en la autosimilaridad de la imagen. El segundo método consiste en particionar una imagen en objetos más pequeños cuyos IFS son conocidos. Esos IFS se usan para formar una librería. El procedimiento de codificación es buscar un IFS en la librería para cada pequeño objeto. El tercer método se denomina *IFS particionado (PIFS)*. En este método, la imagen primero se divide en bloques pequeños, y entonces se encuentra el IFS para cada bloque, mapeando un bloque grande hacia un bloque más pequeño.

En la aproximación directa, primero se particiona la imagen en bloques más pequeños no sobrepuestos, de tal manera que cada bloque es similar a la imagen completa y una transformación puede mapear la imagen completa hacia cada bloque. La transformación para cada bloque individual puede ser diferente. La combinación de esas transformaciones puede ser tomada como la IFS de la imagen dada. De esta manera se requieren menos datos para representar las IFS o las transformaciones que para transmitir o almacenar la imagen dada pixel a pixel. Para el segundo procedimiento, la clave es particionar la imagen dada en objetos cuyos IFS sean conocidos. Las técnicas de procesamiento de imagen, tales como separación del color, detección de flancos, análisis de espectro, y análisis de variación de textura se pueden usar para particionar la imagen. Sin embargo, para imágenes naturales o imágenes arbitrarias puede ser imposible o muy difícil encontrar un atractor cuyo IFS recupere la imagen original. Por ello, para la mayoría de imágenes naturales, se ha propuesto el método IFS particionado [3]. En este método las transformaciones no mapean la imagen completa hacia pequeños bloques. Para codificar una imagen, la imagen completa primero es particionada en un conjunto de bloques más grandes, que son conocidos como los bloques dominios. Los bloques dominios pueden ser superpuestos. Entonces la imagen es particionada en un número más pequeño de bloques, que son conocidos como bloques rango. Los bloques rango no se superponen y la suma de todos ellos cubre la imagen completa. En el tercer paso, se selecciona un conjunto de transformaciones contractivas. Cada bloque rango es mapeado hacia un bloque dominio con un método de búsqueda y un criterio de igualdad. La combinación de las transformaciones es usada para formar un conjunto de IFS particionado (PIFS). Los parámetros de los PIFS son transmitidos al decodificador, los bloques dominios no son transmitidos. La decodificación empieza por un fondo plano. Entonces, es aplicado el proceso iterado con el conjunto de transformaciones. La imagen reconstruida es obtenida cuando el proceso converge. Existen tres grandes pasos de diseño involucrados en los sistemas de codificación fractal de bloques de imágenes. El primero involucra las técnicas de particionamiento que incluyen el particionamiento de los bloques rangos y de los bloques dominios. El bloque dominio es mayor que el bloque rango. La forma más simple de particionar es dividir la imagen en bloques cuadrados. El segundo propósito es escoger la medida de la distorsión y un método o estrategia de búsqueda. Las medidas de distorsión comunes en la codificación de bloques de imágenes fractales son el error RMS y la diferencia absoluta máxima (MAD). El mayor parecido entre el bloque rango y el bloque dominio transformado es encontrado midiendo la distorsión RMS o la MAD. El tercer paso es la selección de un conjunto de transformaciones contractivas definidas en una forma consistente con una partición.

2. COMPRESION DE VIDEO

Un video clip puede ser considerado como un conjunto de cuadros de imágenes fijas secuenciales. Por ello, se puede aplicar a un video las mismas técnicas que se usan para las imágenes fijas. Pero aún más, en el eje del tiempo (con un desplazamiento corto en el tiempo) los cuadros adyacentes son muy similares. La redundancia entre cuadros adyacentes puede aprovecharse para obtener una compresión mucho más alta que la que se obtiene en el caso de imágenes fijas. Cuando un video es transferido digitalmente, el almacenamiento y la transmisión llegan a ser críticos, y debe tomarse ventaja de las redundancias temporales, de tal forma que el sistema de compresión pueda ser útil para aplicaciones en tiempo real [4].

Una forma de hacer la compresión es hacer una codificación cuadro por cuadro de tal manera que cada cuadro es codificado a la vez, y solamente unos cuantos cuadros adyacentes son usados como referencia para reducir la información redundante. Una técnica eficiente de hacer esto es la llamada compensación de movimiento,

la cual equivale a una compresión 2-D usando cuadros previos como imágenes de referencia, usando los códigos de esos cuadros previos como predictores para los códigos del cuadro actual en proceso de codificación.

La transformada onduleta se puede utilizar para realizar la compresión de cuadros de video de manera rápida, de forma que se produzca una secuencia rápida de cuadros. Otra forma de realizar la compresión de video es utilizando la transformada onduleta 3-D, la cual hace la compresión sobre un grupo de cuadros. En esta forma de realizar la transformada onduleta 3-D, dos dimensiones son espaciales (sobre cada cuadro) y la tercera dimensión es temporal. La transformación se aplica de igual forma en cada una de las dimensiones. En una resolución fina, esta aproximación es capaz de detectar un objeto pequeño moviéndose rápidamente (información de alta frecuencia), mientras en una resolución mas baja puede detectar objetos grandes moviéndose lentamente (información de baja frecuencia). Sin embargo, no se pueden detectar los objetos grandes moviéndose rápidamente (baja frecuencia en el espacio y alta frecuencia en el tiempo). Este problema se resuelve desacoplando la descomposición onduleta en descomposición espacial y descomposición temporal. Una descomposición onduleta 2-D ordinaria se realiza sobre cada cuadro, produciéndose una secuencia temporal de cuadros de descomposiciones onduletas 2-d. Después, una descomposición onduleta 1-D sobre cada punto (x,y) en cada nivel, a través de todos los cuadros produce la información deseada [5].

El video es dividido de forma natural en segmentos de acuerdo a los cambios de escena. En la compresión fractal cada segmento, empezando con un cuadro inicial, es codificado como una imagen fija; que es llamada cuadro intracodificado, o cuadro-I. Cada cuadro puede ser codificado usando los códigos de movimiento referenciando su cuadro precedente, el cual es llamado cuadro-P [4]. En la práctica, mas cuadros pueden ser permitidos y pueden ser sumados entre dos cualesquiera de los cuadros-I y cuadros-P, y esos cuadros adicionales son llamados cuadros bi-direccionales, o cuadros-B, véase la Figura 3. Cada cuadro-B es codificado usando la predicción desde ambos cuadros burdos que están inmediatamente antes y después de él. Un cuadro-B es insertado a menudo entre dos cuadros-P. En ese caso, para la misma región de bloque, si dos vectores de movimiento similares son dados en cada uno de los cuadros-P, entonces un vector de movimiento interpolado para el cuadro-P en la misma región de bloque es siempre esperado. Los cuadros-B tienen usualmente una razón de compresión muy alta, por ellos son codificados referenciando a ambos cuadros antes y después de él. En un sistema de compresión fractal de video bi-dimensional, los cuadros-I son comprimidos usando técnicas de compresión fija.

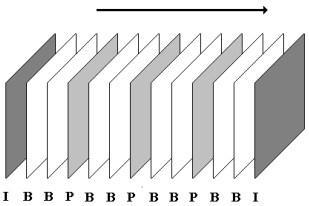

Figura 3..- Secuencia de cuadros de video.

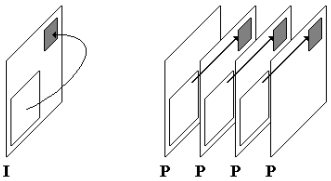

Figura 4.- Cuadro-I descompuesto en una secuencia de cuadros-P en la codificación fractal.

Por la necesidad de contar con una compresión rápida (teniendo en mente que los cuadros-I forman solamente una pequeña parte del total de los datos comprimidos en la mayoría de las implementaciones actuales), un compresor de imagen fija simplificado puede cumplir con esta necesidad. Por ejemplo, un compresor quadtree de dos niveles con un código local y uno global, colocando códigos fractales contractivos espacialmente en 2 a 1 sobre cada nivel, será suficientemente bueno en la mayoría de las aplicaciones. Así, cada cuadro-I será codificado por códigos fractales de 2 a 1.

3. METODOS PARA LA COMPRESIÓN CON ONDULETAS Y CON FRACTALES.

3.1. Métodos para la compresión con onduletas.

En la última década se han propuesto arquitecturas que varían en términos de la escalabilidad (número de coeficientes de filtro), del área, de la complejidad de control, de la latencia, y del tipo y la cantidad de memoria. Se han reportado diferentes diseños, como los sistólicos, semi-sistólicos, de dígitos en serie, etc. Algunas arquitecturas utilizan estructuras de control centralizado [6][7], y algunas otras usan control distribuido [8]. El control de señal centralizado es más fácil de implementar pero sufre problemas de escalabilidad [9]. Pocas arquitecturas 3-D han sido reportadas. Una arquitectura principal se ha introducido en [6], la cual opera con bloques de datos en lugar de renglones. El cómputo de la transformada onduleta 3-D es descompuesto en tres pasos, cada uno es una transformada en las direcciones *x*, *y*, y *z*.

La Figura 5 muestra el esquema de una arquitectura clásica para realizar la transformada onduleta de una señal 1-D, así como su reconstrucción. La arquitectura clásica

tiene la desventaja de requerir una cantidad grande de memoria debido a que una gran cantidad de muestras temporales se deben almacenar entre dos estratos sucesivos y entre procesamiento vertical y horizontal, y esta memoria limita los tiempos de acceso cuando debido a su tamaño la memoria sea externa al chip. Los filtros p y q, así como p_0 y q_0 se escogen como parejas de filtros biortogonales para lograr obtener una reconstrucción perfecta. Si la señal de aproximación y la señal de detalle son separadas recursivamente en dos subseñales, se obtiene la descomposición típica en árbol de la codificación de la DWT. La Figura 6 muestra la descomposición en árbol de Mallat en la que solamente se codifican recursivamente las señales de aproximación. En la práctica el árbol de Mallat se usa separando la señal de detalle solamente en los primeros estratos. La razón es que existe poca correlación entre las muestras en las señales de detalle, por lo que no se necesita separarlas mas que en unos cuantos estratos. La descomposición en árbol da una representación de la señal dependiente de la frecuencia. La versión de la señal que ha sido filtrada y submuestreada puede interpretarse como un filtrado de la señal original con una versión dilatada de los filtros p y q. Este escalamiento en el tiempo de los filtros es equivalente a escalar su respuesta en frecuencia, y mover proporcionalmente su centro de frecuencia. Estrato por estrato, el ancho de los filtros de análisis se duplica y correspondientemente el ancho de las bandas analizadas se divide a la mitad. La representación DWT de una señal es una unión de representación tiempo-frecuencia, y debido al escalamiento de los intervalos soportados por los filtros, se da una representación de escalamiento en el tiempo. Esta representación es capaz de enfocar los detalles y discontinuidades de la señal, lo cual es bueno para el análisis de señales no estacionarias como las de sonido e imagen. Esta representación se desempeña mejor en análisis de señales no estacionarias y en la compresión, que las representaciones de frecuencia completa que pierden cualquier información acerca del comportamiento temporal de las señales no estacionarias, y mejor también con respecto a la representación tiempo frecuencia de la DCT (transformada coseno discreta). La codificación DWT de una señal se basa usualmente en funciones de escalamiento base y onduletas separables tal que estas puedan ser llevadas a cabo iterando dos DWT 1-D ortogonales. Cada imagen se procesa primero en una dirección y luego en la otra dirección ortogonal, obteniéndose cuatro imágenes cuyo tamaño es de un cuarto del tamaño de la original. Teniendo en mente que las muestras deben ser procesadas dos veces, debido a la codificación separable 2D, la cantidad total de muestras procesadas tiende a 8/3 del número de las muestras de entrada. El resultado es que además del ancho de banda necesario para las señales de entrada y salida, se necesita

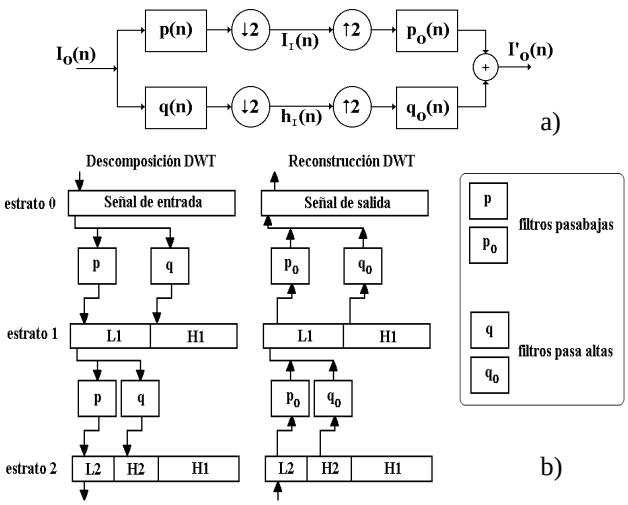

Figura 5.- Filtros biortogonales para garantizar la reconstrucción perfecta de $I_o(n)$ b) DWT 1-D con descomposición en árbol de Mallat.

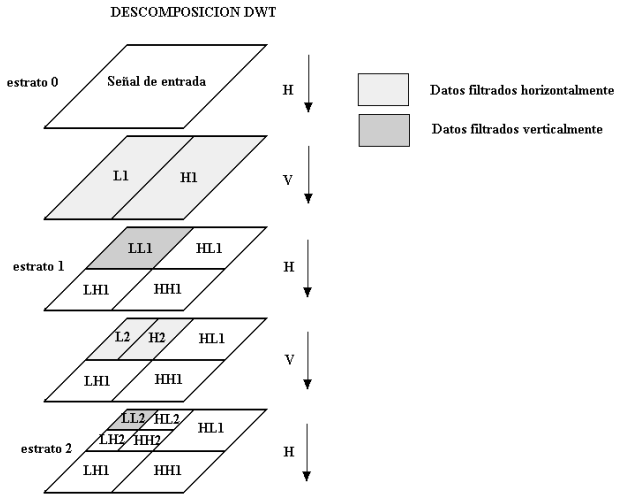

Figura 6.- DWT - 2D con descomposición en árbol de Mallat. La cantidad de datos que son filtrados tiende asintóticamente a 8/3 del tamaño de la muestra original.

un ancho de banda de 1.67 veces para las muestras temporales.

Las muestras temporales necesitan memoria extra. Si el tamaño de la memoria temporal requerida es demasiado grande, se necesitará implementarla fuera del chip, comprometiendo el desempeño. Debido a la longitud pequeña de los filtros de análisis y de reconstrucción, la carga computacional resulta pequeña comparada con la carga de la transferencia de datos de todas las operaciones de entrada y salida internas y externas. Por ello el principal cuello de botella en la codificación y en la decodificación onduleta es el manejo de los datos temporales teniéndose el compromiso con el ancho de

banda de la memoria y el tamaño de la memoria sobre el chip.

En el caso de las arquitecturas de ventanas corredizas el propósito es explotar las dependencias entre los diferentes estratos y entre el procesamiento horizontal y vertical tratando de utilizar las muestras temporales tan pronto como están disponibles. Mediante la minimización del tiempo de existencia de las muestras temporales se reduce el tamaño de la memoria temporal requerida y si tal memoria es pequeña para ser implementada en el chip, se reducirá también el ancho de banda de la memoria externa. La Figura 7 muestra a bloques como reducir el tiempo entre la creación de una muestra temporal y su utilización entre estratos intermedios.

La Figura 8 muestra la forma en la que se manejan las muestras entre el filtrado horizontal y vertical. Los filtros horizontales producen muestras a lo largo de los renglones mientras los filtros verticales necesitan muestras a lo largo de las columnas. Para producir esas columnas de muestras con los filtros horizontales se usa un conjunto de filtros horizontales, con una pareja de filtros por cada línea. De esta manera, registrando los filtros horizontales línea por línea, podrán producirse los datos temporales capaces de alimentar los filtros verticales. Para reducir el ancho de banda de la memoria, la memoria para almacenar los datos de entrada de los filtros horizontales se construye en el chip. El filtrado horizontal produce al mismo tiempo las muestras pasa-bajas y pasa altas. Así que para obtener las muestras de salida final se necesitan dos parejas de filtros verticales, para procesar en paralelo las muestras temporales procedentes del pasa-bajas y del pasa-altas que vienen del filtrado horizontal.

Si se extiende el esquema de la Figura 7 a señales 2-D, y sustituyendo cada estrato 1-D y su filtrado correspondiente con el esquema de la Figura 8, se obtiene la implementación de “ventanas deslizantes sobre todos los estratos”. En este esquema no se requiere memoria externa debido a que todas las muestras temporales son procesadas en el chip y el ancho de banda externo estará limitado al requerido para leer la señal de entrada y almacenar la señal de salida. Esta implementación requiere una gran cantidad de memoria y un sistema de registros complejo para sincronizar el filtrado en paralelo sobre todos los estratos.

Si se explota solamente el esquema de la Figura 8 para procesar un estrato a la vez, se tendrá una arquitectura más simple conocida como “ventanas corredizas estrato por estrato sobre una franja”.

Esta solución evita el almacenamiento temporal entre el filtrado horizontal y vertical, pero necesita almacenar muestras temporales entre estratos. Tiene la ventaja de ser más simple, requiere solamente tres filtros y casi la mitad de la memoria interna que la arquitectura anterior y no necesita un sistema complejo de registros entre estratos. La cantidad de memoria externa es muy grande.

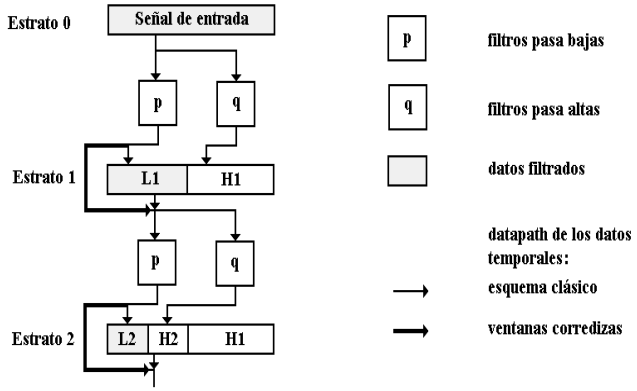

Figura 7.- Este esquema muestra que los datos temporales de los estratos L1 y L2 no necesitan ser almacenados si son leídos tan pronto como son generados.

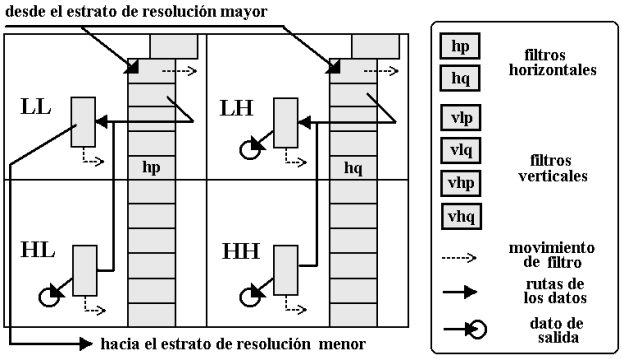

Figura 8.- Ventanas corredizas 2-D sobre un estrato.

Una solución intermedia a las dos anteriores es conocida como “ventanas deslizantes sobre N estratos” usando el mismo esquema que la “ventanas deslizantes sobre todos los estratos” pero aplicada a N de L estratos, como se muestra en la Figura 9.

Con respecto al esquema de “ventanas corredizas estrato por estrato sobre una franja” se logra reducir la cantidad memoria externa requerida y su ancho de banda, pero se necesita más memoria para procesar los primeros N estratos más grandes, y más filtros para procesar N estratos al mismo tiempo. El ancho de banda de la memoria interna es igual que los casos previos.

Puede tratar de reducirse la cantidad de memoria interna aplicando el algoritmo anterior, pero solamente a franjas de la señal de entrada, obteniéndose así la implementación de “ventanas corredizas estrato por estrato sobre N franjas”. Procesando media señal de entrada a la vez, esto es, procesando dos franjas horizontales separadamente, se utiliza la mitad de memoria, pues se implementa la ventana deslizante solo sobre la mitad de la altura, como se muestra en la Figura 10.

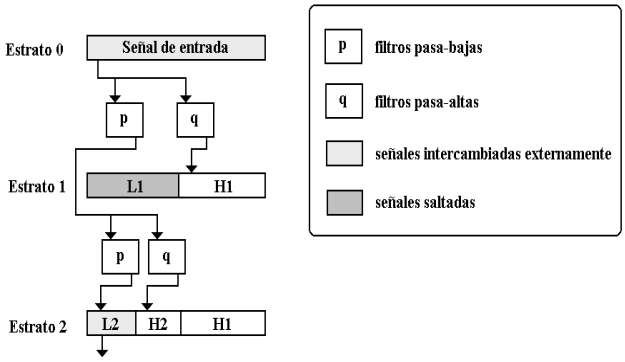

Figura 9.- Ventanas corredizas sobre 2 estratos.

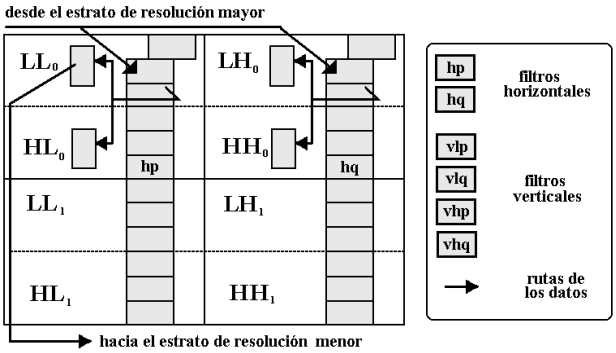

Figura 10.- Ventanas deslizantes 2-D sobre dos franjas. Se reduce la cantidad de memoria interna procesando solamente una franja de memoria a la vez.

Con N franjas se tienen N-1 bordes internos. Para procesar las líneas adyacentes entre cada franja se necesita una memoria temporal extra, y si se quiere reducir el área dentro del chip, se necesitará memoria extra fuera del chip y más ancho de banda para leer esos datos temporales externos. El número de parejas de filtros requeridos es tres veces el que se requiere para el caso de la "ventana corredizo estrato por estrato sobre una franja". Porque se procesa una franja a la vez y un estrato a la vez.

El esquema "bloque por bloque" preserva la simplicidad de la arquitectura clásica y explota la dependencia de datos como en las otras soluciones optimizadas. La idea principal es usar el esquema clásico para procesar bloques de la señal de imagen de entrada, en lugar de la imagen completa. Procesando un área local de la imagen se reduce el tiempo de existencia de los datos temporales, producidos entre el procesamiento horizontal y vertical, y entre dos estratos sucesivos. Haciendo esto se explota la dependencia de datos dentro de cada estrato y entre estratos. Se necesita una memoria temporal conocida como memoria de árbol para almacenar esos datos de muestras temporales. Para evitar cualquier efecto de bloques, debido a la separación de la señal, se necesita una memoria temporal extra, conocida como *memoria entre bloques*, para procesar los datos temporales compartidos por bloques adyacentes.

En las Figuras 11 se muestra la forma en la que se aplica el procesamiento bloque por bloque a señales 1-D y 2-D. En el caso 2-D, considerando señales más amplias en la dimensión horizontal, se extiende la memoria temporal entre bloques a lo largo de la dimensión vertical completa de la señal de entrada para almacenar muestras temporales entre dos columnas completas de bloques.De acuerdo a lo antes expuesto, la arquitectura de procesamiento bloque por bloque parece la más adecuada para aplicaciones móviles en tiempo real.

En [10] se reporta un algoritmo para la compresión escalable de video que hace énfasis en la reducción del codificador EZW (Embedded Zerotree Wavelet) basado ancho de banda de la memoria. Se hace uso de un esquema de descomposición subbanda 2-D en junto con un en bloques. En este esquema se ejecutan primero todos los niveles de la DWT 1-D vertical, y la transformada 1-D horizontal se aplica solamente a la subbanda de baja frecuencia. Se usa también un mecanismo de paralelismo basado en bloques para ayudar a reducir el número de operaciones de acceso a los coeficientes. Cada componente de color de un cuadro de video es descorrelacionado a través de la DWT 2-D, la cual genera un conjunto de coeficientes en los que la gran mayoría de ellos son cero, así que se reduce grandemente la entropía de la imagen.

3.2. Métodos para la compresión fractal de imágenes.

En cuanto a la compresión fractal, se han reportado arquitecturas en paralelo para la codificación fractal de imágenes basada en un árbol cuádruple [11]. Estas arquitecturas no utilizan una sola memoria para el almacenamiento de los bloques dominios, evitando los cuellos de botella que surgen cuando varios bloques de procesamiento necesitan acceder a la memoria al mismo tiempo. Además las transformaciones isométricas necesarias son realizadas mediante rotaciones de datos, calculándose ellas en paralelo, lo cual disminuye el tiempo de codificación. Algunas de estas arquitecturas utilizan la MAD (diferencia media absoluta) como criterio de comparación entre la imagen almacenada en los bloques rango y la codificación en los bloques dominio, logrando un tiempo de codificación más corto en comparación con otras arquitecturas. A estas puede añadirse la implementación de estrategias de búsqueda de similitudes entre los bloques rangos y bloques dominios, tales como búsquedas en cuadrantes, búsquedas locales, búsquedas en espiral, o categorización de bloques de acuerdo a su varianza [12]. Añadiendo las estrategias de búsqueda se logra reducir el tiempo de codificación de la imagen, sacrificando calidad de imagen en el proceso de reconstrucción.

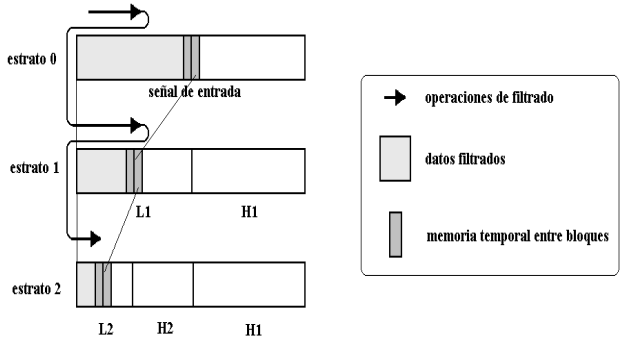

Figura 11.-Procesamiento bloque por bloque en una dimensión.

Las Figuras 12a, 12b, 12c y 12d, muestran las estrategias de búsqueda de similitudes para reducir el tiempo de codificación.

Restringiendo la búsqueda a áreas cercanas se logra reducir el tiempo de codificación. La imagen fuente puede ser seccionada en cuatro cuadrantes como se muestra en la Figura 12a. Para un bloque rango localizado en el cuadrante inferior izquierdo solamente se buscarán bloques dominios localizados en el mismo cuadrante. Como resultado de esto, el tiempo de búsqueda se reduce en un factor de cuatro, comparado con el método de búsqueda en todos los dominios. En general, la complejidad es O(nq), donde n es el número de píxeles en la imagen, y q es el número de píxeles por cuadrante. Basándose en esta misma idea se puede realizar una búsqueda local en ventanas. Otra variante es realizar una búsqueda espiral comenzando desde el bloque rango actual como se muestra en la Figura 12c. La búsqueda se detiene cuando se encuentra un bloque dominio que cumple un criterio de semejanza establecido, aunque no sea el mejor. En algunos casos, el tiempo de búsqueda se reduce drásticamente, a costa de la fidelidad de la reconstrucción de la imagen.

El grueso de los cálculos puede ser eliminado categorizando los bloques antes de la comparación. Si un bloque contiene flancos abruptos, puede evitarse tiempo si no se busca entre los bloques dominios que tienen variaciones suaves. Cada dominio es colocado en una de 72 categorías. Para hacer esto, el bloque se divide primero en cuatro cuadrantes y es orientado en una "posición canónica", como se muestra en la Figura 12d.

Una vez de haberse dividido en una de esas tres grandes categorías, los cuadrantes de cada bloque son ordenados desde la mas alta varianza a la mas baja, en 4! = 24 posibilidades en cada categoría. Los bloques rangos también son clasificados de esta forma. Cuando se busca una semejanza para un bloque rango, se busca dentro de su categoría correspondiente, y debido a que los bloques están clasificados en forma canónica, no es necesario probar para las 8 isometrías de cada bloque dominio, lográndose una reducción en el número de operaciones.

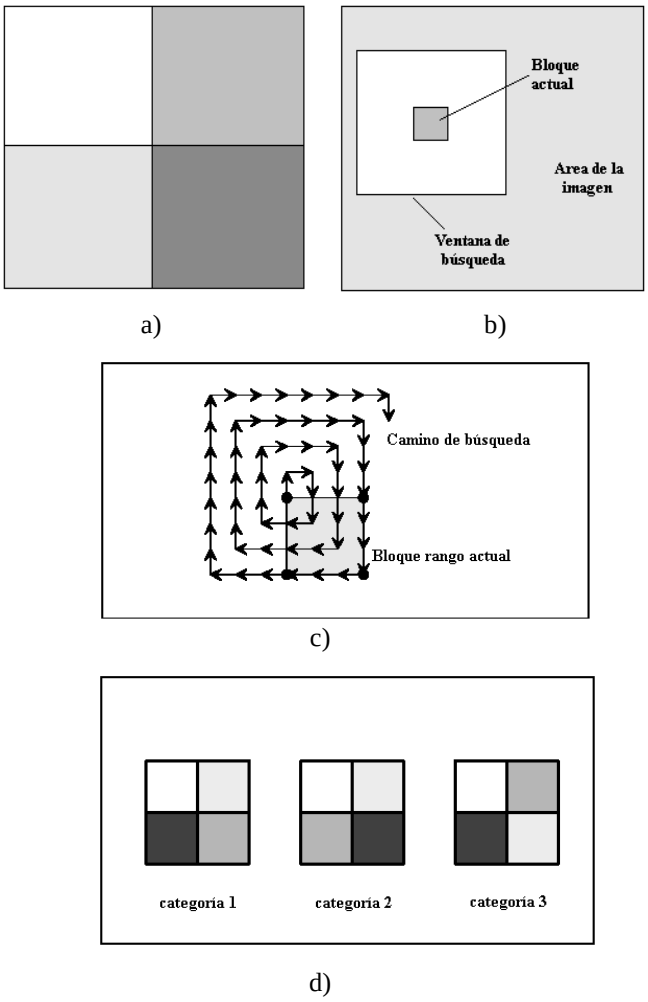

Figura 12.- Estrategias de búsqueda. a) búsqueda en cuadrantes. b) búsqueda local en ventanas.. c) Búsqueda local en espiral. d) búsqueda categorizada.

4. CONCLUSIONES.

Se han presentado varios algoritmos adecuados para implementar arquitecturas para realizar la compresión de imágenes y video, relacionados con el procesamiento con onduletas y con fractales. Los algoritmos mencionados para la compresión con onduletas buscan reducir la cantidad de memoria externa, y aprovechar al máximo la existencia de datos temporales con el propósito de disminuir la cantidad de accesos a memoria, acelerando el tiempo de codificación. En el caso de la compresión fractal, los métodos mencionados buscan disminuir la cantidad de operaciones de comparación que se tienen que realizar durante el proceso de codificación y están orientados a mantener la simplicidad requerida para poder implementar las arquitecturas correspondientes.

5. REFERENCIAS

[1] Woods,J., Ed., *"Subband Image Coding"*. Public Academic Publishers, 1991.

[2] Nollant S. Jallant and P. *"Digital Coding of Waveforms"*. Englewood Cliffs, Prentice Hall, 1984.

[3] Ning Lu. *"Fractal Image Compression"*. Signal Processing. Image Commun., 5, pp. 327-343, 1993

[4] Ning Lu. *"Fractal Imaging"* . Academic Press, 1997.

[5] Laura R. C. Suzuki, J. Robert Reid, Tomas J. Burns, Gary B. Lamont and Steven K. Rogers, "Parallel Computation of 3-D Wavelets", *ISCAS* 1995.

[6] Michael Weeks , M. Bayoumi, "3-D Discrete Wavelet Transform Architectures", *IEEE Int. Symposium on Circuits and Systems (ISCASS 98*) Monterrey, California, Mayo31 – June 3, 1998

[7] G. Knowles, "VLSI Architecture for the Discrete Wavelet Transform", *Electronics Letters*, Vol 26, No 15, pp. 1184 – 1185, 1990.

[8] Jose Fridman and Elias S. Manolakos, "Distribuited Memory and Control VLSI Architectures for 1-D Discrete Wavelet Transform", *IEEE Proc. VLSI Signal Processing VII*, La Jolla, California, pp. 388-397, October 26-28 1994.

[9] Jose Fridman and Elias S. Manolakos, "Discrete Wavelet Transform: Data Dependency Analysis and Synthesis of Distribuited Memory and Control array Architectures", *IEEE Transactions on Signal Processing*, Vol 45, No. 5, pp. 1291-1308, May 1997.

[10] Roberto Yusi Omaki, Gen Fujita, Takao Onoye, and Isado Shirakawa. "Embedded Zerotree Wavelet Based Algorithm for Video Compression". *IEEE ISSCAS* 1999.

[11] Shinhaeng Lee, Shin'ichiro Omachi and Hirotomo Aso. "A Parallel Architecture for Quadtree-based Fractal Image Coding". *Electronics Letter*, September 2000.

[12] John Kominek. "Advances in Fractal Compression for Multimedia Applications". *Department of Computer Science* ,University of Waterloo, Ontario, Canada 1995.