UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

IMPLEMENTACION ORIENTADA A OBJETOS DE UN CODIFICADOR MPEG-1 EN MATLAB, APLICANDO DIVERSAS TECNICAS DE ESTIMACION DE MOVIMIENTO

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACION

P R E S E N T A N :

RODOLFO ESTEVES JARAMILLO

M I G U E L L O P E Z T O R R E S

MAURICIO MORENO GUTIERREZ

DIRECTOR DE TESIS:
DR. BORIS ESCALANTE RAMIREZ

MEXICO, D.F. SEPTIEMBRE DE 1999

A mi madre, por su amor, por su entrega y dedicación a sus hijos, por su valía como ser humano. Éste es un triunfo tuyo, sin duda. No hay palabras que expresen mi gratitud a ti, ser humano tan maravilloso que me ha dado generosamente el don de vivir.

A mi padre, mi ejemplo, mi guía y mi espíritu. Eres sin duda el mejor y representas lo que quiero algún día llegar a ser: un hombre triunfador, íntegro, bueno, inteligente, maduro, cariñoso y responsable. Gracias por todo, seguimos juntos en el camino de la vida.

A mi hermano, por su presencia y su importante aparición en mi vida. Tu compañía, tu apoyo y tus logros también me impulsan a seguir adelante. Por todo lo que he aprendido y aprenderé de ti en lo que nos falta por andar.

A mis abuelos, por haber creado y criado ese par de padres tan sensacionales que tengo, porque siempre desearon y lucharon por la superación de sus hijos y de sus nietos.

A Samari, por todo lo que representa en mi vida. Eres una personita maravillosa, portadora de bondad, de belleza y de inteligencia. Me infundes gozo, me llenas de amor, me has hecho ver la vida diferente, me complementas. Eres médula y sustancia de este trabajo.

A Mumu, a Luis Manuel y a Toño, va este trabajo por esa amistad tan profunda y duradera. Amigos hay pocos, pero hermanos mucho menos. Comparto este con ustedes y brindo porque nuestra amistad tan fiel nos acompañe siempre.

A la familia García y Colomé, por haberme ofrecido su amor, su comprensión, su apoyo y su compañía que son tan importantes para mí. A Rosy, mujer por demás admirable y ejemplar. A Pablo, amigo, padre y maestro de excepción, y a Ana Paula, quien se ha vuelto una figura indispensable en mi vida, por los múltiples vínculos que nos unen.

A Lalo, Oscar, Juan Pablo, Pillín, Claudia, Huicho, Yoatzin, Luis, Skid, Pablito, Germán, Fer, Jaime y toda la banda ingenieril. Por su enriquecedora amistad, por su compañía, que siempre agradeceré y que apreciaré por la calidad humana e intelectual que poseen cada uno de ustedes. Son y serán, sin duda, parte inseparable de mi vida y de mi desarrollo.

A Paola, Manolo, Hiram, Jorge, Pepo, Michel y Halcón, porque saben lo mucho que significan para mí. Ustedes son actores fundamentales de muchas escenas de mi vida. Los aprecio mucho.

A la UNAM, que vive en mí, que transpiro por todos mis poros, que forma parte de mi vida, que me enriquece día a día, que admiro profundamente y que me motiva a luchar por ella y por mi país, por su excelencia académica, por su futuro, por su majestuosidad y por todo lo que representa en tantas personas que, como yo, deben tanto a ella.

Mauricio

A Trinidad Fuentes Aguilar, a quien dedico, no lo que pude haber contribuído a este trabajo, sino todo lo que haya de valioso en mi carrera.

A mis padres, a quienes les agradezco mucho más de lo que puedo expresar en palabras.

A Claudia y Alejandra, que no creo tengan idea de lo que significan para mí, y a las que tampoco se los voy a decir.

A Carmen Jaramillo de Bolaños, Concepción y Teresa Jaramillo Bolaños, nunca lejos, nunca ausentes y siempre necesarias.

A Luis Jaramillo Bolaños.

A Luis Enrique Hernández B., un amigo que nunca dejaré de agradecer a la UNAM.

Rodolfo

A mis padres:

Por todo el apoyo que me han brindado durante todos estos años, porque siempre me ayudaron a superar las situaciones más difíciles de mi vida, y siempre tuvieron mucha confianza en que podría lograr mis metas, y sobre todo por el ejemplo que me han puesto para conducirme por el buen camino y hacer de mí una persona de bien.

A mis hermanas:

Porque siempre me han brindado su apoyo y comprensión, y me motivaron siempre a terminar mi carrera y a esforzarme siempre por ser un estudiante sobresaliente.

A mis abuelitos:

Porque siempre confiaron en que podía salir adelante y llegar a ser un profesionista.

A mi tía Lucre y a mi primo Carlos:

Por sus consejos y apoyo para que obtuviera los mejores resultados de mi carrera.

A Gaby:

Por ser una persona muy especial para mí, que siempre me motivó y me impulsó a que siguiera adelante.

A mis tíos:

Tobo, David, Nino, Hipo, Berta, Nata, Min, Guille, Chayo, Rafa, Graciela y Magui.

A mis primos:

León, Marco, Mayda, Andros, Lalo, Norma, Laura, Andrés, Javier, Lucio, Lety, Paty, Paquito, Neyra, Eder,...

A mis amigos:

Erika, Huicho, Juan José y Andrés por brindarme siempre su amistad sincera.

Miguel.

Prefacio y Agradecimientos

El interés que movió a los autores a emprender este trabajo fue diverso. Algunos de nosotros estamos interesados en continuar estudios de posgrado en el procesamiento digital de señales, otros consideramos el desarrollo de un proyecto como el presente un buen ejercicio de programación e integración de lenguajes, así como probablemente nuestra primera exposición seria a la comparación del desempeño de diversos algoritmos. Asimismo, deseamos que el presente trabajo, además de constituirse como un proyecto académico interesante en el área de la ingeniería, represente una conclusión exitosa a nuestros estudios de licenciatura.

En este aspecto, cabe señalar que el énfasis de este trabajo recae, en mayor medida, en las técnicas de implementación y diseño de software, más que en la investigación en el campo del procesamiento digital de señales. Creemos que esta puntualización es necesaria pues, dado el marco en el que este trabajo se desarrolló, es fácil creer que la parte medular de este trabajo consiste en algún aspecto relacionado más directamente con el trabajo que se realiza en la División de Estudios de Posgrado de la Facultad de Ingeniería. Cabe repetir que la motivación del desarrollo del programa que este texto describe está más orientada a la aplicación y el refuerzo de los conocimientos que a lo largo de sus estudios profesionales los autores obtuvieron, que a la profundización en los trabajos en investigación de Procesamiento Digital de Señales se lleva a cabo en la antedicha dependencia. De cualquier forma, creemos que la tesis aquí presentada ha colaborado, aunque sea de forma pequeña, a la productiva serie de trabajos que, en relación al procesamiento digital de imágenes y video se desarrollan en la DEPFI.

Deseamos agradecer al Dr. Boris Escalante Ramírez, profesor de nuestra facultad, quién fungió como nuestro director de tesis en el presente trabajo. Su apoyo brindado para llevar a cabo este trabajo fue muy importante. Asimismo, agradecemos la valiosa cooperación y la abierta disposición a orientarnos en la escritura de esta tesis de licenciatura que ofrecieron nuestros sinodales: la Mtra. Laura Sandoval Montaño, el Dr. Víctor García

Garduño, el Dr. Miguel Moctezuma Flores y el Dr. Bohumil Psenicka. También queremos agradecer la cooperación de José Luis Silván Cárdenas en hacernos entender su trabajo y enfocar mejor el nuestro.

Gracias a la facultad de Ingeniería, por habernos acogido durante cinco productivos años, en los que como estudiantes recibimos nuestra formación profesional como ingenieros, y como personas conocimientos, opiniones y percepciones que serán sin duda muy importantes en el desarrollo de nuestras vidas. Nuestra facultad es, sin duda, una de las mejores escuelas de ingeniería de el país, y de la cual estamos muy orgullosos.

Agradecemos también a nuestra querida universidad, la UNAM, hoy pasando por momentos difíciles, por erigirse como nuestra casa, estancia académica y centro de convivencia enriquecedora, y por ser el proyecto cultural más importante de nuestro país. Nuestra vida ha girado, y lo seguirá haciendo, en torno a la UNAM. Es un orgullo pertenecer a esta gran comunidad, pero mayor orgullo será cuando algún día podamos contribuir en algo a la tarea que la UNAM se ha impuesto. Sin duda la admiración y fidelidad que expresamos hacia nuestra universidad es una motivación a ser profesionistas responsables, y personas conscientes y de bien.

Contenido

Lista de Figuras

Lista de Tablas

I. Introducción

El objetivo del presente trabajo de tesis es el de evaluar distintos métodos de estimación de movimiento dentro de un esquema de codificación tipo MPEG-1 elaborado en software para codificar secuencias de imágenes. El codificador recibirá una secuencia de imágenes en formato CIF, las pre-procesará para obtener imágenes indizadas y devolverá un solo archivo con extensión .MPG que cumplirá con los estándares de decodificación especificados en la norma documentada en ISO/IEC 11172, en la sección referente a capa de video.

La estimación de movimiento es una pieza fundamental en el funcionamiento del codificador, ya que genera los vectores de desplazamiento que son aprovechados por el estándar para disminuir la cantidad de información que se envía al medio de almacenamiento o al canal de transmisión. El paso siguiente a la estimación de movimiento en un codificador MPEG-1 es la compensación de movimiento, proceso que requiere los vectores previamente obtenidos. Existen otras características que le permiten al MPEG-1 comprimir dicha información: una de ellas es el submuestreo o *subsampling*, (elección ordenada de porciones de los datos), la cuantización, que se da después de transformar "bloques" de pixeles con herramientas como la transformada coseno discreta (DCT) y las mencionadas estimación y compensación de movimiento. Asimismo, al momento de codificar los datos para generar el archivo final, se elimina la redundancia en la información gracias al código de longitud variable de Huffman.

Hay que hacer notar que el estándar MPEG-1 regula las características que debe tener la *decodificación*, por lo que existe cierta libertad al momento de crear un codificador; simplemente se deben seguir las reglas básicas y se gana enorme flexibilidad al momento de implementar los módulos correspondientes. Es ahí donde intentaremos aprovechar dicha flexibilidad para asociar distintas técnicas de estimación de movimiento al sistema.

Hemos deseado, asimismo, que el presente trabajo constituya un punto medio entre dos áreas de la ingeniería: la ingeniería en computación y la ingeniería eléctrica del procesamiento de señales. Es por eso que se ha elegido como ambiente de programación el lenguaje MATLAB y se ha aplicado en la escritura del codificador el paradigma de la

programación orientada a objetos. Existe, asimismo, la inquietud de congeniar al ambiente MATLAB con el lenguaje de programación "C". La decisión anterior se derivó de la imposibilidad que presenta el MATLAB para llevar a cabo la escritura de datos a nivel de bits, cuestión fundamental en la parte final de la codificación. El sistema presentado puede verse como un híbrido en el que los módulos programados en MATLAB representan la entrada y control del proceso y los archivos programados en "C" reciben la información pertinente para llevar a cabo la codificación de bajo nivel.

Creemos que en un mundo que observa vertiginosos cambios en las comunicaciones y que día a día muestra dramáticos avances en la tecnología debidos a rápidas evoluciones, principalmente en el área del cómputo, de la electrónica y en general, en la ingeniería, siempre son justificables los trabajos que intenten colaborar aunque sea un poco con el enriquecimiento de una pequeña parte del acervo tecnológico.

Pretendemos, asimismo, que el presente trabajo pueda resultar de utilidad en proyectos posteriores, por lo que se ha intentado dar un carácter de "modularidad" al mismo, haciendo fácil su análisis y permitiendo así su fácil implante en posibles proyectos posteriores.

Durante la elaboración de este proyecto de tesis de licenciatura hemos encontrado varios problemas y nos hemos enfrentado con situaciones probablemente invisibles al emprender su implementación. Trataremos de expresarlas y de explicar las soluciones que propusimos. Además, resumiremos y sintetizaremos la teoría relevante sobre la que se apoya el desarrollo práctico del trabajo.

I.1 Breve Historia y Fundamentos del Video.

Hablaremos a continuación un poco sobre la evolución que ha presentado el video a lo largo de los últimos años, hablando un poco sobre su difusión y su grabación. En los párrafos siguientes hacemos un breve recorrido por la historia del video en televisión.

Los primeros métodos utilizados en transmisiones de televisión y en grabaciones eran analógicos, y los formatos de las señales estaban determinados esencialmente por los requerimientos del monitor, es decir, del tubo de rayos catódicos; de esta forma, el receptor

debería ser lo más simple posible y en su construcción debería estar considerada la menor cantidad de tubos de vacío (bulbos). Gracias, en parte, al desarrollo de la grabación magnética de audio durante la segunda guerra mundial, se percibió la necesidad de que existiera grabación en televisión. Otra de las causas que motivaron este avance fue la variedad de husos horarios en los Estados Unidos: sin la ventaja de la grabación, los programas populares de televisión tenían que ser representados en vivo varias veces para que el público pudiera verlos en horas "pico" en cada zona horaria [Watkinson, 1994].

Hay que mencionar que los métodos de grabación analógica se han convertido ya en una tecnología madura y han llegado ya a límites en ocasiones determinados por el conocimiento actual de las leyes de la física. Los esfuerzos por lograr refinamientos en esta técnica producen actualmente pequeños avances.

Habrá que hablar un poco sobre las enormes diferencias que representan los sistemas de video analógicos y los digitales. Defenderemos, finalmente, la existencia de los sistemas de video digitales y ampliaremos un poco la información sobre ellos.

En un sistema analógico, la información es interpretada gracias a la variación infinita de un parámetro continuo, como podría ser el voltaje en un cable o la fuerza, la frecuencia o el flujo magnético en una cinta. En una grabadora, la distancia a través del medio es un continuo análogo al tiempo, es decir, la información contenida en ella se encuentra distribuida de manera continua y sigue un orden espacio-temporal; para acceder a una información determinada hay que recorrer el medio (cinta). No importa en qué punto de la longitud de la grabación examinemos, siempre encontraremos un valor para la señal grabada. Ese valor puede, por sí solo, cambiar con una resolución infinita dentro de los límites físicos del sistema [Watkinson, 1995].

Esas características son las principales debilidades de las señales analógicas. Dentro del ancho de banda permitido, *cualquier* forma de onda es válida. Si la velocidad del medio no es constante, una forma de onda válida cambia a otra forma de onda también válida; no podemos detectar errores basados en tiempo en un sistema analógico. En suma, un error en voltaje simplemente cambia un voltaje válido en otro; en estos sistemas el ruido no puede ser separado. Podríamos sospechar algo acerca del ruido, pero ¿cómo haríamos para saber qué proporción de la señal recibida es ruido y qué proporción estadística es original?. Si la

función de transferencia de un sistema[1] no es lineal, existe distorsión, pero las formas de onda distorsionadas son todavía válidas; así que tampoco es fácil detectar distorsión en un sistema analógico.

Una de las características de los sistemas analógicos es que las degradaciones no pueden ser separadas de la señal original, por lo que no puede existir ningún tipo de "limpieza" posterior (filtrado analógico). Al final de un sistema determinado, la señal arrastra la suma de todas las degradaciones introducidas para cada etapa por la que pasó. Esto establece un límite al número de etapas a través de las cuales pasa una señal antes de que sea imposible de usar.

Las señales de video son formas de onda eléctricas que permiten que imágenes en movimiento sean transportadas de un lugar a otro. La observación del mundo real con el ojo humano es realmente una imagen bidimensional en la retina. Esta imagen cambia en el tiempo, de forma que la información básica es tridimensional. Con dos ojos, estaremos hablando de una imagen estereoscópica[2] [Watson & Ahamuda, 1985].

Una forma de onda eléctrica es una función de una variable unidimensional, ya que representa un voltaje cambiante en el tiempo. Para poder transportar información de imagen tridimensional a través de un cable bidimensional, es necesaria la exploración (scanning). En vez de intentar transferir la brillantez de todas las partes de una imagen al mismo tiempo, la exploración transfiere la brillantez de un solo punto que se mueve en el tiempo.

En los sistemas de televisión la exploración consiste en rápidos barridos horizontales combinados con un barrido vertical más lento, de manera que la imagen es explorada en líneas. Al final de cada barrido vertical, o cuadro (frame), el proceso reinicia [Watkinson, 1994].

El proceso de exploración convierte la resolución de la imagen al dominio de la frecuencia. Entre más alta sea la resolución de la imagen, se necesitarán más líneas para dar la resolución vertical. La proporción de línea se incrementa junto con el número de ciclos de modulación que se necesita que sean llevados en cada línea. Si la razón de cuadro (frame

[1] La función de transferencia de un sistema resulta ser la transformada (de Laplace, Fourier, etc.) de la función de salida dividida entre la transformada de la función de entrada del mismo, trabajada desde el punto de vista de un dominio distinto al del tiempo (por ejemplo, la frecuencia, en caso de Fourier) [Oppenheim & Willsky, 1994].

[2] el video estereoscópico es posible con la utilización de equipo adecuado, aunque esto está restringido a aplicaciones especializadas y no ha sido explotado el término en transmisión masiva.

rate) permanece constante, el ancho de banda crece cuadráticamente a la resolución [Inglis, 1990].

Dentro de los tipos principales de video analógico de color, se encuentra el RGB (red, green, blue), en el cual las señales correspondientes a estos tres colores básicos emergen de los sensores de la cámara necesitando un ancho de banda completo, ya que cada una de las tres señales tiene el mismo espectro. El formato RGB se usa cuando se requiere máxima precisión, generalmente para la producción de imágenes estáticas. RGB es raramente usado en grabación de video en tiempo real por su alto costo. Es posible obtener un ahorro en ancho de banda utilizando trabajo con diferencia de color, como el que MPEG utiliza. Si se obtiene una señal de luminancia, o intensidad de brillo, con una suma ponderada (ya que el ojo no es igualmente sensitivo a los tres colores primarios) de R, G y B; dicha señal necesitará ancho de banda completo pero las señales de diferencia de color, llamadas *crominancias*, Cr y Cb requerirán menos ancho de banda. Combinando Cr y Cb en un esquema de modulación de portadora obtenemos una transmisión de color en el mismo ancho de banda que el monocromo. Esto será explicado con más detalle en capítulos posteriores.

Las señales de diferencia de color no necesitan el mismo ancho de banda que Y, ya que la agudeza del ojo humano no se extiende a la visión de colores. La mitad, o inclusive la cuarta parte del ancho de banda serán suficientes, dependiendo de la aplicación. El ajuste cromático es mucho más preciso con un amplio ancho de banda de diferencia de color, y es por lo que los sistemas de producción utilizan la mitad del ancho de banda de luminancia para las señales de diferencia de color. El ancho de banda total es entonces dos tercios del necesitado por RGB.

Para lograr una difusión de televisión de color compatible con monocromo en un canal sencillo, los sistemas NTSC, PAL y SECAM entrelazan en el espectro de la señal monocromática una sub-portadora que lleva dos señales de diferencia de color en un ancho de banda restringido. La intención es que dicha sub-portadora sea invisible en la pantalla de un aparato de televisión monocromático. A un sistema de color como éste se le conoce como sistema de video compuesto (composite video), y la sub-portadora modulada es llamada *chroma* [Watkinson, 1995].

I.2 ¿Por qué video digital?.

Uno de los conceptos vitales a considerar es que el video digital es simplemente una forma alternativa de llevar una forma de onda de video. Un sistema de video digital ideal tiene, desde el punto de vista operacional, el mismo objetivo que un sistema analógico ideal: los dos tienen una función de transferencia que les permita reproducir a la salida una señal idéntica a la de la entrada, es decir, sin distorsión, errores, ni pérdidas. Uno sólo necesita comparar los equipos analógicos y digitales de alta calidad que hay, con las mismas señales, para darse cuenta de qué tan transparentes pueden llegar a ser ambos sistemas modernos. No es necesario recalcar que las condiciones ideales nunca se presentan, de forma que ambos equipos quedan lejos de lo deseado, pero el equipo digital simplemente queda más cerca de lo ideal que el analógico y a un menor precio o, si el diseñador así lo desea, puede tener el mismo rendimiento que el analógico pero a un precio muy bajo. Este detalle es importante: la electrónica digital ha crecido en todos los aspectos principalmente debido a que la producción de dispositivos de alta (LSI) o muy alta escala de integración (VLSI) es cada vez más barata y masiva. La tremenda oferta de estos dispositivos en el mercado los hace muy accesibles para la construcción en serie de complicados sistemas digitales de alta calidad. Es asombroso ver cómo ha crecido la cantidad de aparatos electrodomésticos que están controlados por electrónica digital en los últimos 10 o 15 años. Ni hablar de los sistemas de cómputo, que han sufrido severas revoluciones y transformaciones en pocos lustros; pero regresemos al video digital.

A pesar de que hay un gran número de maneras en las que las formas de onda de video pueden ser representadas digitalmente, hay un sistema, conocido como Modulación de Código de Pulsos (PCM) que es prácticamente de uso universal. PCM trabaja así: en vez de ser continuo, el eje del tiempo se valora en forma discreta. La forma de onda no es trabajada como una representación continua, sino que se va valuando a intervalos regulares. Este proceso es llamado muestreo y la frecuencia con la que se lleva a cabo se llama frecuencia de muestreo (Fs). Cada muestra, de todas formas, varía infinitamente como lo hizo la forma de onda original. Los dispositivos analógicos de muestreo son bien conocidos en video. El Sensor Acoplado por Carga (CCD) de una cámara de televisión es un ejemplo. Para completar la conversión a PCM, cada muestra es entonces representada con alta

precisión por medio de un número discreto en un proceso conocido como cuantización [Andleigh & Thakrar, 1996].

En los sistemas de televisión, la imagen que entra en el sensor de la cámara será continua en tiempo y continua en dos dimensiones espaciales, correspondientes al alto y ancho del sensor. En los sistemas de video analógicos, el eje del tiempo es muestreado con cuadros, y el eje vertical es muestreado en líneas. El video digital simplemente agrega un tercer proceso de muestreo entre las líneas.

Hay una conexión directa entre el concepto de muestreo temporal, donde la entrada cambia con respecto al tiempo a cierta frecuencia y es muestreada a otra frecuencia, y el muestreo espacial, donde una imagen cambia un número determinado de veces por unidad de distancia y es muestreada a otro determinado número de veces por unidad de distancia. La conexión entre estos dos conceptos es el proceso de exploración (scanning) ya mencionado. La frecuencia temporal se puede obtener multiplicando la frecuencia espacial por la velocidad de la exploración [Watkinson, 1994]. En el dominio espacial nos referiremos a resolución, mientras que en el dominio temporal nos referiremos a respuesta en frecuencia.

Aunque cualquier tasa de muestreo suficientemente alta puede ser usada para el video, es más común hacer que la tasa de muestreo sea un múltiplo entero de las dimensiones de la línea. Las muestras son, entonces, tomadas en el mismo lugar en todas las líneas. Si hacemos esto así, una imagen monocromática digital sería un arreglo rectangular de puntos en los cuales se guarda un número correspondiente a la brillantez en él. Los puntos se conocen como elementos de imagen (picture elements), generalmente abreviados como pels o pixeles. Con la colocación cercana de los pixeles se espera que el observador perciba una imagen continua. Obviamente, entre más fino sea el espaciamiento entre pixeles, más grande será la resolución de la imagen y mayor cantidad de información se necesitará para almacenar la imagen, con ello aumentará también el costo del proceso.

Si lo que se desea es representar una imagen coloreada, se tienen que superponer tres capas de muestras, cada una correspondiente a cada color primario. En este caso, el pixel ya no es simplemente un escalar que representa la brillantez del punto, sino un vector que describe de alguna forma la brillantez, el matiz (hue) y la saturación de ese punto en la imagen; aunque hay varios modelos para lograr el coloreado.

Para producir imágenes en movimiento, la forma más sencilla es simplemente crear un mecanismo para que el valor de cada pixel pueda ser actualizado periódicamente. Esto da como resultado efectivo un arreglo tridimensional, donde dos de los ejes son espaciales y el tercero es temporal [Inglis, 1990].

Ya que es posible representar cualquier forma de onda analógica, el video compuesto también puede digitalizarse. En el ADC (Convertidor Analógico-Digital), se intenta siempre librar la inestabilidad temporal, de forma que todas las muestras sean tomadas en intervalos exactos. Es claro que si existiera cualquier error subsecuente basado en el tiempo, cambiarían los instantes en los que llegarían las muestras y el efecto podría ser detectado. Cuando las muestras lleguen a algún destino con una base de tiempo irregular, el efecto se podría eliminar guardando las muestras temporalmente en memoria y leyéndolas después usando un reloj local estable. Este proceso se llama corrección de la base de tiempo (timebase correction[3]) [Watkinson, 1994] y todos los sistemas de video digital que se precien de tener calidad deben utilizarlo, ya que el error no sólo es reducido, sino que es eliminado.

Esencialmente, el video digital lleva la forma de onda original *numéricamente*. El número de la muestra es análogo al tiempo, que por sí mismo es análogo a la posición a través de la pantalla, y la magnitud de la muestra es (en el caso de la luminancia) análogo a la brillantez en el punto apropiado de la imagen. De hecho, la sucesión de muestras en un sistema digital es realmente *análogo* a la forma de onda original. Esto podría sonar como una contradicción y es por lo que muchas autoridades prefieren manejar el término "video numérico", como se podría inferir del francés *numérique*.

Como ambos ejes de la forma de onda digital son discretos, la forma de onda puede ser fielmente restaurada desde esos números de la misma forma que sería dibujada en un papel para gráficas. Si requerimos gran precisión, simplemente escogeremos un papel con cuadrícula más pequeña. De esta forma, más números se necesitarían y cada uno de ellos podría variar dentro de un amplio rango.

En términos sencillos, la forma de onda de video es transferida en una grabadora digital como si el voltaje hubiera sido medido en intervalos regulares con un medidor

[3] Este proceso es también utilizado en el diseño de sistemas digitales con otro tipo de aplicaciones, para sincronizar la comunicación entre procesos con la ayuda de un dispositivo "temporizador".

digital y las lecturas hubieran sido escritas en un rollo de papel. La tasa a la que las medidas fueron tomadas y la exactitud del medidor son los únicos factores que determinan la calidad, ya que una vez que un parámetro se expresa con un número discreto, cualquier serie de ese tipo de números se puede transportar sin pérdidas de precisión.

Los humanos insistimos en el uso de números expresados en base 10, habiendo evolucionado dentro de este sistema. Pero, como sabemos, el sistema numérico mínimo utilizado con mayor frecuencia es el binario, que tiene sólo dos dígitos, 1 y 0. Los dígitos binarios son rápidamente relacionados con la palabra "bits". Son eficientemente transferidos con circuitos cambiando de estado "on" a "off" y viceversa. Con dos estados solamente, hay poca posibilidad de error.

Hay dos formas en las que las señales binarias pueden ser usadas para llevar muestras: transmisión serial (PCM) y transmisión en paralelo. La transmisión en paralelo implica que cada dígito del número binario es llevado en un cable diferente, como en la comunicación normal entre una PC y una impresora. El estado de los cables cambia con la tasa de muestreo. El uso de varios cables es algo incómodo, particularmente cuando hablamos de largas distancias; pero se puede implementar el uso de un solo cable con transmisión serial. Aquí, los dígitos son enviados sucesivamente por el mismo medio: ésta es precisamente la definición de la modulación de código de pulsos (PCM). Es obvio que aquí la frecuencia del reloj tendrá que ser más alta que la tasa de muestreo [Andleigh & Thakrar , 1996].

La transmisión de video con este esquema es ventajosa, ya que el ruido y el error "timebase" son eliminados, pero su problema actual es que, por ejemplo, una señal de diferencia de colores de alta calidad requiere entre dos y tres millones de bits por segundo de tasa de transmisión, por lo que el video digital será una herramienta común cuando tasas como la mencionada sean fácilmente manejables. Más y más aplicaciones estarán disponibles cuando las formas de reducción de tasas de transmisión sean económicas. Aquí es donde entra la importancia de tener altos índices de compresión en una aplicación digital tan "pesada" como es el video. Los archivos de video "de alta calidad" comúnmente encontrados en Internet, son difíciles de obtener debido a su bromosidad y generalmente la duración del mismo es de unos pocos segundos. Creemos, por lo expuesto anteriormente, que toda la investigación que se genere en relación a nuevos métodos de compresión de

este tipo de datos es importante, además de que es fácilmente extrapolado a otras áreas del conocimiento digital similares a la presente.

Ya hemos hablado de las ventajas de una señal binaria, ahora hablemos de las ventajas que representa que una señal, como la de video, sea digital.

La representación digital eficiente de señales de imagen y video ha sido objeto de intensos trabajos el los últimos 20 años. La tecnología digital de codificación de video se ha convertido en un campo maduro y se han desarrollado ya productos de una gama amplia de aplicaciones recientes, como el VOD ("Video on demand"), la distribución de televisión digital (DTV) y la de alta definición (HDTV) y los servicios de base de datos multimedia. Con el creciente interés comercial en las comunicaciones en video, aparece la imperiosa necesidad de trabajar en la compresión de imágenes y video.

Con una señal digital, la calidad de reproducción en un sistema de video digital bien diseñado es independiente del medio, y sólo depende de la calidad de la conversión.

La conversión de video al dominio digital ofrece tremendas oportunidades que son denegadas a los sistemas analógicos.

Cuando una grabación digital es copiada, los mismos "números" aparecen en la copia: no es una aproximación, es un *clon*. Si la copia es indistinguible por cualquier medio del original, no ha habido pérdidas de generación. Las grabaciones digitales pueden ser copiadas indefinidamente sin pérdida de calidad.

En el mundo real, todo tiene un costo, y uno de los puntos fuertes de la tecnología digital es el bajo costo. Si el proceso de copiado no genera pérdidas de calidad, las grabadoras digitales no tienen que ser mucho mejores que lo necesario para sobreponerse a ese problema. Sólo necesitan ser adecuadas en la primera generación, cuya calidad será mantenida vez con vez. Ya no hay necesidad, además, de que exista el enorme consumo de cintas de las grandes grabadoras analógicas profesionales; cuando la información consiste solamente de números discretos, éstos pueden ser densamente "empacados" en el medio sin temor a pérdidas de calidad. Si algunos bits cayeran en errores por ruido u omisión, la corrección de errores restauraría los valores originales. Las grabaciones digitales ocupan menos espacio que las analógicas por la misma o mejor calidad.

La manufactura de los circuitos digitales cuesta menos. La circuitería de dos estados que maneja lenguaje binario puede ser integrada más densamente que la analógica; más

funcionalidad en el mismo chip. Los circuitos analógicos son construidos a partir de un conjunto de componentes que tienen una gran variedad de tamaños y formas, y por ende son caros de ensamblar y de ajustar. La circuitería digital posee diseños de componentes estándar, lo que significa que son más fáciles de ensamblar en equipo automático, y además, se necesita muy poco o nulo ajuste.

Una vez que el video está en el dominio digital, se convierte en datos, que son indistinguibles de cualquier otro tipo de datos. Una gran ventaja es que los sistemas y técnicas desarrollados en otras industrias y para otros propósitos pueden ser utilizados para el video digital. El equipo computacional se encuentra disponible a bajos costos porque el volumen de producción es mucho más grande que el del equipo de video profesional [Watkinson, 1994]. Las lectoras de disco y las memorias desarrollados para computación pueden ser puestas a la disposición del video digital. Un procesador de palabras adaptado para manejar muestras de video se convierte en una estación de trabajo. Parece que no tiene sentido estar esperando a que una cinta sea reembobinada cuando la cabeza lectora de un disco puede acceder a los datos en milisegundos.

Una de las ventajas obvias de la información digital y a la que debemos, en gran parte, el haber desarrollado esta tesis, es la capacidad de procesamiento *por software*. Una vez que poseemos las secuencias, imágenes digitales somos libres para a través de sencillos módulos programados en algún lenguaje, como MATLAB, procesar o transformar la información de byte a byte y obtener los resultados deseados.

Las redes de comunicaciones que fueron desarrolladas para manejar información pueden eficientemente llevar video digital y su correspondiente audio a distancias indefinidas sin pérdida de calidad. La difusión de televisión digital hace uso de estas técnicas para eliminar los problemas de interferencia y difuminación de las transmisiones analógicas. Al mismo tiempo, se hace un uso eficiente del ancho de banda disponible [Andleigh & Thakrar , 1996].

El equipo digital puede tener además, sus propios programas de auto diagnóstico. La máquina detecta sus propias fallas, y el costo de hallar un posible error manualmente en un circuito puede ser mayor que la tableta en sí. De esta forma, los costos de mantenimiento caen. Una pequeña operación puede no necesitar todo un staff de mantenimiento: un contrato de servicio es suficiente. Una organización más grande podría aún necesitar un

staff de mantenimiento, pero serían pocos en número y sus esfuerzos estarían orientados más a los sistemas que a los dispositivos en sí.

I.3 Contexto del video digital.

El contexto en el que el video digital se encuentra inmerso es muy variado y además bastante creciente. Como una herramienta relativamente reciente, ha ido adquiriendo importancia dentro del mundo tecnológico. Al video digital lo podemos localizar en distintas aplicaciones particulares, pero de forma general lo ubicaremos en el conjunto de la tecnología multimedia. Entendamos, así, multimedia como un mundo extendido en casi todas las áreas del conocimiento humano, como el arte, la medicina, el diseño, la educación, la ingeniería, las humanidades y muchas más.

El cómputo multimedia ha emergido como un área de investigación exhaustiva en los últimos años. Los sistemas computacionales en multimedia han abierto un gran rango de aplicaciones potenciales combinando varias fuentes de información, como voz, gráficas, animación, imágenes, audio y video. Si hacemos una extrapolación de la influencia de la tecnología multimedia, podemos decir que se trata de la conjunción de tres grandes industrias: la computación, las comunicaciones y las industrias de difusión masiva.

Los esfuerzos en investigación y desarrollo en el cómputo multimedia han sido divididos en dos áreas. La primera es investigación, y gran parte de la energía dedicada a esta área se ha centrado en las estaciones de trabajo autónomas y sus sistemas de software asociados, así como composición de música, aprendizaje ayudado por computadora y video interactivo. De cualquier forma, la combinación de cómputo multimedia y sistemas distribuidos ofrece un gran potencial.

La característica principal de los sistemas multimedia es que incorporan información continua, como voz, video y gráficas animadas [Andleigh & Thakrar, 1996]. Esto implica que los sistemas multimedia tienen que manejar la información con estrictos requerimientos de tiempo y a una tasa alta. El uso de información continua en sistemas distribuidos denota la necesidad de una transferencia de datos estable en largos periodos de tiempo.

El video digital se justifica en el ambiente de los sistemas distribuidos, en el que interactúa como parte de un sistema complejo de multimedia.

Los principales componentes de un sistema multimedia distribuido son:

- Los proveedores de información (contenido).
- Una red de área amplia (WAN).
- Clientes multimedia.

Los usuarios de multimedia se conectan a los servidores de información a través de una red consistente de switches y un medio de transmisión. El medio de transmisión puede ser un cable coaxial o un canal de fibra óptica. Las compañías de cable ya tienen una gran red de cable coaxial con el propósito de mantener una entrega de video de calidad. Las compañías telefónicas, por su lado, han mejorado sus antiguas redes con nuevas líneas de fibra óptica [Hardman, van Rossum et al., 1993].

Sin duda, el candidato más fuerte para consolidarse como la esperada "supercarretera" de la información es la Internet. Con su asombroso crecimiento en los últimos años ha dejado ver que es posible pensar en una transmisión de medios en tiempo real en poco tiempo. Ya es común la transmisión de video de baja calidad en tiempo real en la "telaraña", así como el tráfico masivo de información multimedia (voz, audio, imágenes y video). La limitante actual es, fundamentalmente, la baja tasa de transmisión debida a los reducidos anchos de banda.

Sin embargo, una futura unificación de los sistemas de cable y de teléfono pudiera reducir la importancia de Internet. En el futuro, las arquitecturas locales de cable y teléfono serán casi idénticas. Ambos sistemas probablemente se sincronizarán y se multiplexarán. Una red híbrida coaxial-fibra óptica podría transmitir en forma full-duplex voz, datos y televisión digital por cable. Las redes backbone, como el servicio público conmutado de teléfono o las redes privadas de cómputo podrían ser conectadas a la oficina central de teléfonos o a la terminal del servicio de TV de cable.

Muchas aplicaciones como correo de video, videoconferencia y sistemas colaborativos de trabajo, requieren multimedia en red. En estas aplicaciones, los objetos multimedia se guardan en un servidor y se reproducen en los sitios locales de los clientes. Dichas aplicaciones pueden requerir datos multimedia de difusión a gran escala o el acceso a grandes depositorios de información.

Los sistemas multimedia sugieren una gran cantidad de aplicaciones. La siguiente tabla especifica algunas de las aplicaciones existentes actualmente que incluyen la adquisición, procesamiento o despliegue de video digital.

Aplicación	Tipo de Datos	Funciones Necesarias
Sistemas Médicos	Video (telefonía), Imágenes	Adquisición de datos, Comunicación
Educación/Capacitación	Audio, Video, Imágenes, Texto	Búsqueda, Interactividad
Agencias de Viajes	Audio, Video, Imágenes, Texto	Búsqueda en Video, Comunicación
Publicidad	Video, Imágenes	Composición de Imágenes, Realce
Correo Electrónico de Video	Audio, Video, Imágenes, Texto	Comunicación
Catálogos Electrónicos de Consumo	Audio, Video, Texto	Búsqueda en Video
Distribución de Video Casero	Audio, Video	Búsqueda en Video
Bienes Raíces	Audio, Video, Imágenes, Texto	Búsqueda en Video, Comunicaciones
Sistemas de Información Turística	Audio, Video, Imágenes, Texto	Búsqueda en Video
Información Masiva en Medios de Comunicación	Audio, Video, Imágenes, Texto	Comunicación, Difusión
Colaboración Electrónica	Audio, Video, Texto	Videoconferencia, Comunicación
Industria Cinematográfica	Audio, Video	Almacenamiento, Búsqueda en Video
Televisión Interactiva	Video, Audio	Interactividad, Comunicación

Tabla 1.1. El Video Digital en el mundo actual.

Dos importantes aplicaciones que se hallan hoy en día en uso y que se explicarán brevemente debido a que están fundamentados en el video digital son los sistemas multimedia de videoconferencia y los sistemas multimedia sobre demanda (on-demand), como son la televisión interactiva y los sistemas de distribución de noticias.

Los sistemas de conferencia multimedia permiten a un número determinado de participantes intercambiar información multimedia a través de redes de voz y datos. Cada participante tiene una estación de trabajo multimedia unida a las otras estaciones de trabajo gracias a redes de alta velocidad. Asimismo, cada participante puede enviar y recibir video, audio y datos y puede llevar a cabo ciertas actividades colaborativas. La conferencia multimedia utiliza el concepto del *espacio virtual compartido* (shared virtual workspace), que describe la parte del despliegue replicada a cada estación de trabajo [Hardman, van Rossum et al., 1993].

Los avances logrados en los sistemas multimedia distribuidos han afectado significativamente el desarrollo de los servicios "sobre demanda", como el entretenimiento interactivo, la distribución de noticias en video, los servicios de renta de video y las bibliotecas digitales multimedia. Muchas compañías se han dado cuenta de que las redes de alta velocidad de fibra óptica, aunadas al cómputo avanzado y a las técnicas de compresión, podrían dar lugar pronto a la generación y entrega de películas digitales. En los últimos años se han formado varias alianzas entre compañías de cable, teléfono y computación con el objetivo de cubrir las aplicaciones del video sobre demanda. La autopista de la información, audio y video digitales comprimidos, juegos y películas interactivos, acceso a bases de datos y servicios varios: todo en la televisión. Esto cambiaría en cierto modo la cultura y los hábitos de las próximas generaciones.

Un sistema de televisión interactiva (ITV) bien diseñado debe ser capaz de cubrir los siguientes servicios interactivos: televisión básica, televisión de suscripción, pago por evento, video sobre demanda, compras, educación, prensa electrónica, audio digital, transacciones financieras y juegos mono y multiusuario. Algunas otras aplicaciones soportables por el ITV incluyen guía de programación electrónica, navegadores electrónicos y de internet y sección amarilla [Watkinson, 1995].

Otro ejemplo de sistema multimedia sobre demanda es el de producción y difusión de noticias, cuyo objetivo principal es el de generar un servicio de información interactivo ágil y eficaz.

Hay todavía muchos retos que involucran a los sistemas multimedia y a su relación con el video digital que están siendo o serán investigados para el futuro crecimiento de dichos sistemas. Las aplicaciones multimedia hacen tremendas demandas de recursos en el software y en el hardware de las computadoras. Es por esto que una de las líneas de acción es la de crear cada vez más poderosas estaciones de trabajo multimedia, que a su vez requerirán cada vez de mejores sistemas operativos multimedia (MMOSs) y avanzadas interfaces de usuario. Un MMOS debe manejar información continua proveyendo multitarea, fácil expandibilidad, acceso a la información independientemente del formato y soporte para sincronía en tiempo real. El usuario espera del sistema multimedia que la información esté disponible rápida y fácilmente, por lo que las interfaces de usuario deben ser altamente sofisticadas e intuitivas [Hardman, van Rossum et al., 1993]. La sana integración de la interfaz de usuario al nivel operativo puede eliminar muchos problemas para los desarrolladores de software de aplicación.

Otros retos en el área de investigación incluyen el desarrollo de nuevos algoritmos de compresión en tiempo real, técnicas de indización y recuperación de audio y video, grandes dispositivos de almacenamiento y sistemas de manejo de información multimedia. Las tareas permanentes son el refinamiento de la alta velocidad, el desarrollo de redes determinísticas con baja latencia y baja agitación y la búsqueda de nuevos algoritmos de sincronización de multimedios [Andleigh & Thakrar, 1996].

I.4. Sistema de Visión Humana:

A continuación se hablará un poco del sistema de visión humana, incluyendo la formación de imágenes en el ojo y su capacidad de discriminación y de adaptación a la iluminación. Es importante conocer al menos algunos conceptos básicos de este tema ya que muchas técnicas de procesamiento y compresión de imágenes están basadas en ellos.

El ojo humano es casi esférico, con un diámetro medio aproximado de 20 mm. Está rodeado por tres membranas: la córnea y la esclerótica, que constituyen la cubierta exterior,

la coroides y la retina. La córnea es un tejido resistente y transparente que cubre la superficie anterior del ojo. En prolongación de la córnea, la esclerótica es una membrana opaca que encierra el resto del globo ocular.

La coroides, debajo de la esclerótica, contiene una red de venas que constituyen la "alimentación" del ojo, y en su extremo anterior está dividida en cuerpo ciliar y diafragma o iris. El iris es el que se abre o cierra para controlar la cantidad de luz que entra al ojo. El cristalino, justo debajo de la córnea, absorbe aproximadamente el 8% del espectro luminoso visible, con una absorción ligeramente superior en las longitudes de onda más cortas. Tanto la luz infrarroja como la ultravioleta son absorbidas de forma apreciable, por lo que en cantidades excesivas pueden dañar al ojo.

La membrana más interna del ojo es la retina, que recubre la totalidad de la pared posterior. Cuando el ojo está correctamente enfocado, la luz de un objeto exterior al ojo forma su imagen en la retina. La visión del objeto se debe a una distribución de receptores de luz repartidos por la superficie retiniana, hay dos clases de ellos: conos y bastones. En cada ojo hay aproximadamente entre 6 y 7 millones de conos principalmente en la región central de la retina (fóvea) y son muy sensibles al color. Gracias a ellos percibimos detalles finos porque cada uno está conectado a su propia terminal nerviosa. A la visión de los conos se le llama fotópica o visión de luz brillante. Los bastones son mucho más: entre 75 y 150 millones distribuidos en la retina. Su mayor área de distribución y el hecho de que comparten varios la misma terminal nerviosa, reducen la cantidad de detalle discernible por estos receptores. No están implicados al color y son sensibles a niveles de iluminación bajos (visión escotópica) [Rodríguez & Woods, 1992].

Debido a que las imágenes digitales se representan como un conjunto de puntos brillantes, la capacidad del ojo de discriminar entre diferentes niveles de iluminación es una consideración importante para presentar los resultados del procesamiento de la imagen. El rango de niveles de intensidad de luz a los que es capaz de adaptarse el sistema visual humano es enorme. Asimismo, la capacidad del ojo humano de discriminar entre cambios de iluminación para cada nivel específico de adaptación también es de considerable interés [Graham, 1965]. Se ha demostrado que la iluminación percibida por el ojo humano no es una simple función de la intensidad. Primeramente, se dice que el sistema visual tiende a sobrevalorar o infravalorar la intensidad cerca de los límites de dos regiones con

intensidades diferentes. Asimismo, en el fenómeno denominado *contraste simultáneo*, se nota que la iluminación percibida de un área no depende únicamente de su intensidad, sino del contexto en que se halla [González & Woods, 1992]. Para mostrar con mayor claridad las características básicas de la anatomía del ojo, se presenta el siguiente esquema:

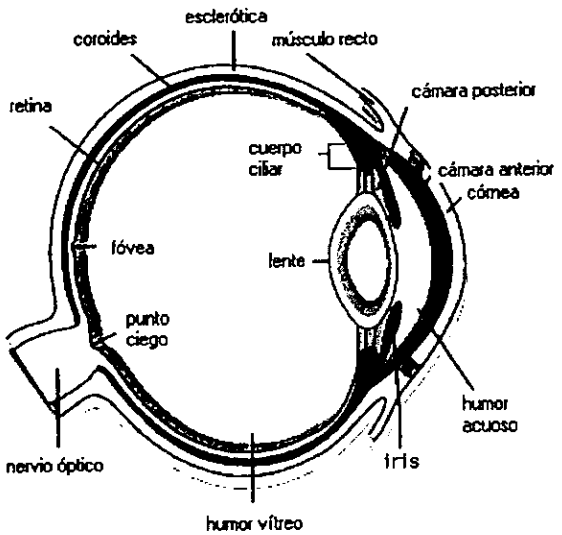


Figura 1.1 Anatomía básica del ojo humano

Algunos modelos de procesamiento, y más en particular, de compresión, explotan ampliamente las características del sistema de visión humana anteriormente mencionadas. Dichas ventajas se explicarán con detalle un poco más adelante, en capítulos posteriores, al hablar de los métodos de procesamiento y compresión.

I.5. Papel de los Estándares.

La estandarización de una tecnología es vital para la intercomunicación de aplicaciones y equipo, y el video digital no es la excepción. El video digital es una

tecnología hasta cierto punto nueva, pero que nació inmersa en el contexto de grandes industrias con tecnologías razonablemente maduras, como la de las resoluciones de despliegue de video en la industria de la computación, estándares de grabación en el de la televisión y protocolos de comunicación estándares en la de telecomunicaciones. El video digital liga y acerca a estos tres campos, y, con ellos, a sus estándares. Es, pues, natural que la estandarización del video sea considerado crucial para su aceptación por cualquiera de estas tres industrias.

En realidad, en la industria de la televisión, el video digital tanto para fines de edición ya tiene tiempo que se usa, debido a su facilidad de manipulación, conservación de la información a través de un proceso de grabación repetitivo (como el que sufre cada grabación en las diversas etapas de su producción), como para la conversión entre formatos (PAL/SECAM - NTSC, por ejemplo). Quizá el primer estándar utilizado fue la recomendación 601 del CCIR (Comité Consultivo Internacional para el Radio), que tuvo mucha difusión sobre todo en la industria de la televisión.

En la economía globalizada de hoy en día, el papel de los estándares es cada vez más importante. Sin embargo, de poco o nada servirán si el conocimiento que se tiene de ellos o de los beneficios de la estandarización es poco.

II. El Procesamiento de Imágenes en el Video Digital

En este capítulo trataremos de integrar los conceptos referentes al procesamiento digital de imágenes como parte fundamental del procesamiento del video digital. El video digital, como ya se dijo, puede ser entendido como secuencias de imágenes digitales, por lo que la ciencia dedicada a procesar a éstas es de suma importancia para los trabajos abocados al video digital, como el nuestro. Hablamos de las ventajas que la representación digital de las señales de video trae consigo, a saber, robustez y facilidad de incorporar diversos servicios sobre una misma línea de transmisión. Explicaremos los conceptos básicos de la teoría de procesamiento digital de imágenes y desarrollaremos con mayor profundidad los temas de mayor relevancia. En este capítulo abundaremos también en las características de la representación digital, y de su procesamiento en la computadora.

II.1 Aspectos Fundamentales:

El procesamiento digital de imágenes –la manipulación de imágenes a través de una computadora- es un avance tecnológico de desarrollo relativamente reciente en términos de la antigua fascinación de los humanos por el estímulo visual. Comprende un amplio rango de hardware, software y recursos teóricos. Es un campo creciente en el que confluyen e interactúan varias disciplinas [Castleman, 1996].

A continuación se explicarán los conceptos básicos del procesamiento digital de imágenes, para después abundar en aspectos de particular importancia para el desarrollo del presente trabajo.

Hablemos un poco de los elementos que intervienen en el procesamiento digital de imágenes. En su nivel más básico, se necesita una computadora que pueda procesar imágenes y dos dispositivos especiales de entrada/salida: un digitalizador de imágenes y un monitor. Debido a que las computadoras trabajan con información numérica más que pictórica, las imágenes deben ser convertidas a forma numérica antes de llevar a cabo cualquier proceso computacional.

La representación de la imagen en la computadora puede ser tan simple como un arreglo rectangular de números, producto de la división que se hizo de la imagen real en pequeños elementos de imagen, o pixeles. A cada celda del arreglo se asocia el valor de brillantez de la imagen en ese preciso punto. Al proceso de conversión explicado se le llama digitalización. En cada localidad de la imagen, se muestrea[1] y se cuantiza[2] la brillantez u obscuridad de dicha región. Cuando se ha hecho esto para todos los pixeles, la imagen es representada como un conjunto rectangular de números enteros. Cada pixel tiene una dirección entera y un valor entero llamado *escala de gris*[3]. Una vez hecho esto, nuestra imagen es un candidato para ser procesada digitalmente.

Un sistema completo de procesamiento de imágenes cuenta con un dispositivo adecuado de almacenamiento que guarda la imagen producida por el digitalizador. En respuesta a instrucciones dadas por el operador, la computadora llama y ejecuta programas de procesamiento desde bibliotecas especiales. Durante la ejecución, la imagen de entrada es "leída" por la computadora de forma ordenada. Posteriormente al proceso, se genera la imagen de salida, pixel a pixel, y es guardada en un dispositivo de almacenamiento de información.

A lo largo del proceso los pixeles pueden ser modificados a discreción, ya que las únicas limitantes en el trabajo son la potencia del método utilizado y los recursos del sistema de cómputo. Después de ser procesado, el producto final se muestra con la ayuda de un proceso exactamente inverso al explicado líneas arriba: el nivel de gris de cada pixel se usa para determinar la brillantez de dicho punto en una pantalla o dispositivo de despliegue cualquiera.

La imagen procesada es, entonces, visible y entendible para la interpretación humana.

Si abundamos en la correspondencia entre imágenes en el proceso completo, obtendremos un ejemplo de la explicación derivada del siguiente esquema:

---

[1] Muestreo, en el caso de imágenes digitales, es llevar a cabo la medición del nivel de brillantez o de gris en cada localidad de pixel. Generalmente se lleva a cabo con un dispositivo sensor de imagen que produce un voltaje proporcional a la intensidad luminosa presente en la escena [Shapiro & Rosenfeld, 1992].

[2] Cuantización es la representación de un valor medido con un número entero. Los sensores de imágenes están generalmente seguidos de convertidores analógico-digital (ADC), un circuito electrónico que genera un número proporcional a cierto voltaje [Castleman, 1996].

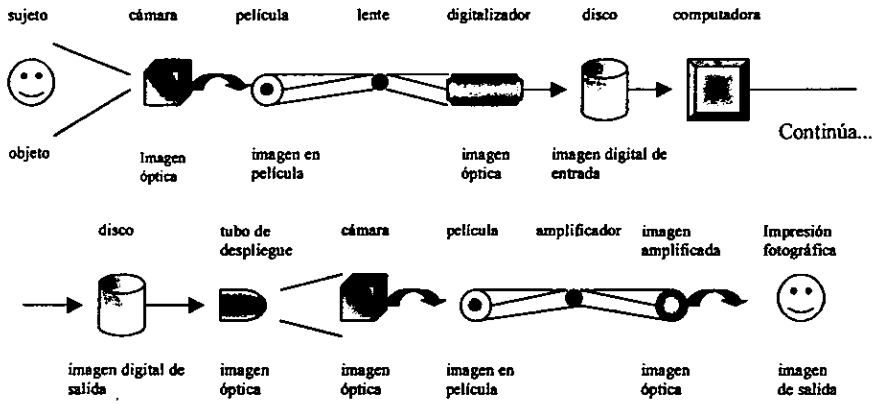


Figura 2.1. Sistema de Procesamiento de Imágenes

Una imagen contiene información descriptiva del objeto que representa. Una fotografía muestra esta información en una forma que permite al observador visualizar por sí mismo al objeto retratado en ella. Una definición más formal del procesamiento digital de imágenes puede ser la sujeción de la representación numérica de un objeto a una serie de operaciones con el objetivo de obtener un resultado deseado o de llevar a cabo alguna meta perceptible. Habría que restringir también la definición de imagen digital a algo así como: "una función bidimensional muestreada y cuantizada que ha sido generada por medios ópticos, muestreada en un arreglo o red rectangular igualitariamente espaciada, y cuantizada a intervalos de amplitud iguales" [Castleman, 1996].

Queremos presentar a continuación varios temas en los que interviene el procesamiento digital de imágenes transformándolas y analizándolas con fines diversos. Nótese que los procedimientos y explicaciones que se expondrán tienen gran relación con los objetivos que persigue esta tesis, habiendo utilizado la mayoría de ellos en alguna parte de su implementación

[3] Posteriormente se hablará de las imágenes de color, o de varias bandas

II. 3. Histogramas y Operaciones en Imágenes Digitales.

Una herramienta muy sencilla y útil en el procesamiento digital de imágenes es el histograma, que muestra la representación de los niveles de intensidad de una imagen. El histograma de cualquier imagen contiene mucha información, y hay algunos casos en los que ciertos tipos de imágenes están completamente especificadas por su histograma. Generalmente al representarlo tenemos para cada nivel de intensidad (o de gris) su correspondiente ordenada que da el número neto de pixeles que lo tienen. A continuación se muestra una imagen en blanco y negro y su correspondiente histograma:

Imagen "Lena"

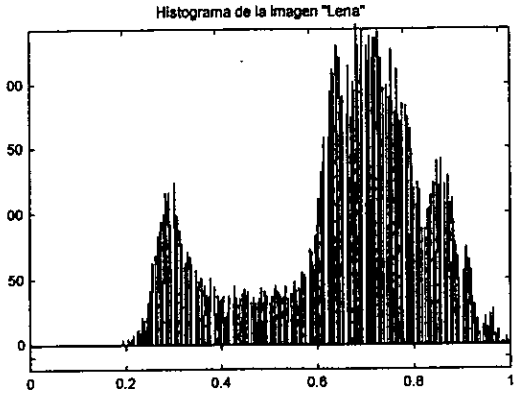


Figura 2.2. Imagen "Lena" y su correspondiente histograma de niveles de gris

En el histograma se muestran las ocurrencias de cada nivel de gris. El 1 corresponde al blanco puro y el 0 al negro puro.

Frecuentemente es muy útil construir histogramas de dimensiones superiores a uno. Esto es particularmente útil en imágenes de color [Novak & Shafer, 1992]. Por ejemplo, un histograma bidimensional puede ser una función de dos variables en las que una sea el nivel de intensidad en una imagen en colores rojos y la otra el nivel de intensidad en colores azules. Su valor en la coordenada (DR, DB) es el número de pares de pixeles correspondientes que tienen su valor de intensidad en rojo DR y en azul DB, como se puede

ver, para cualquier imagen multiespectral se pueden generar histogramas en los que el número de variables en la función dependerá del número de bandas presente en la señal.

El uso de histogramas como un método sencillo para procesar y comprimir imágenes es muy utilizado, ya que es un estimador que ayuda a poder reducir la cantidad de datos útiles en una imagen descartando toda la información espacial, además de ser un agente sujeto a transformaciones de todo tipo: reducción de tonos, ecualización, rotación, etc. Algunos ejemplos reportados en la literatura en los que el histograma es utilizado son:

- Revisión de parámetros de digitalización: El histograma es un indicador simple de si una imagen está correctamente escalada dentro del rango disponible de niveles de intensidad o no. Generalmente se desea que una imagen digital haga uso de todo o de casi todo el rango de niveles, de lo contrario no se obtiene una cuantización óptima [Castleman, 1996].
- Selección del umbral de frontera: El análisis de contornos o curvas de nivel nos provee de una manera muy efectiva de establecer las fronteras de un objeto simple dentro de una imagen. A esta técnica se le llama *thresholding*[4]. Esta sencilla técnica es ampliamente utilizada en el reconocimiento de patrones.
- Densidad óptica integrada: Dado un histograma podemos determinar la densidad óptica integrada de la imagen (IOD) sin siquiera verla. Es un útil estimador de la "masa" de una imagen y se define como:

$$IOD = \int_0^a \int_0^b D(x,y)dxdy$$

donde a y b delimitan la región de la imagen. Cuando la imagen consiste de un objeto obscuro situado en un fondo de nivel cero de grises, el IOD refleja una combinación del área y la densidad del mismo [Castleman, 1996].

Las operaciones que se aplican a las imágenes pueden dividirse en tres grandes rubros: operaciones puntuales, operaciones algebraicas y operaciones geométricas.

- Operaciones Puntuales: Las operaciones puntuales toman como entrada una imagen digital y devuelven una imagen de salida modificada de tal manera que cada valor de

[4] que, abusando un poco del idioma español, puede ser traducido como "umbraleo" o "determinación del umbral"

pixel de ésta depende solamente del nivel de intensidad del correspondiente pixel de entrada. Estas operaciones tienen generalmente otros nombres más particulares, como realce de contraste, adelgazamiento de contraste y transformación de tonos. Las operaciones puntuales modifican el histograma de una imagen de una forma predecible. Pueden ser vistas como transformaciones a pixeles. Más formalmente, una operación puntual que toma una imagen de entrada A(x,y) hacia una de salida B(x,y) se puede expresar como sigue [Pratt, 1978]:

$$B(x,y) = f[A(x,y)]$$

La operación puntual queda totalmente especificada por la transformación de intensidades *f(D)*, que determina el mapeo de la entrada hacia la salida. Dichas operaciones, dependiendo de la forma de su especificación, pueden ser lineales o no lineales. Algunas aplicaciones de los operadores puntuales son: calibración fotométrica (un equivalente en software a la digitalización), realce del contraste (dar más interés a algún área del histograma), calibración de despliegue (generalmente para adecuación a dispositivos de salida), líneas de contorno (como ya se explicó en párrafos anteriores) y el llamado *clipping*, que limita el rango de definición de los valores de los pixeles. Varias de estas aplicaciones pueden, asimismo, verse modificadas para lograr otro tipo de objetivos, como eliminación de redundancias y compresión, aspectos medulares de la presente tesis.

- Operaciones Algebraicas: Las operaciones algebraicas producen una imagen de salida que puede ser la suma, diferencia, producto o cociente de dos imágenes pixel a pixel. Para el caso de sumas y productos más de dos imágenes pueden verse envueltas en el proceso. Hay algunas aplicaciones, por ejemplo, para las sumas de imágenes que involucran el promediado de varias imágenes en la misma escena, para analizar redundancias y diferencias así como para eliminar los efectos de ruido aditivo aleatorio. La resta es muy útil para la detección de movimiento en secuencias de imágenes, para retirar algún efecto aditivo no deseado (como un patrón periódico de ruido). La multiplicación, entre otras cosas, sirve para generar efectos de enmascaramiento (por ejemplo, para eliminar ciertas partes de una imagen). La división es muy útil en la

producción de imágenes radiométricas[5] que son importantes en el análisis de color y de imágenes multiespectrales.

- Operaciones Geométricas: Las operaciones u operadores geométricos cambian las relaciones espaciales entre los objetos en una imagen digital. Se requieren dos algoritmos separados para llevar a cabo tal acción: uno que defina la transformación espacial propiamente (que especifique el "movimiento" de cada pixel), y el otro que haga la interpolación de niveles de intensidad. Esto es necesario debido a que, en general, posiciones enteras (x, y) en la imagen de entrada pueden devolver posiciones no enteras en la de salida.

  Hay varios tipos de transformaciones geométricas, pero son tan complejas como el modelo a diseñar así lo requiera. La definición general, y más simple, de una operación geométrica es [Castleman, 1996]:

$$g(x,y) = f(x',y') = f[a(x,y),b(x,y)]$$

donde *f(x, y)* es la imagen de entrada y *g(x, y)* la de salida. Las funciones *a(x, y)* y *b(x, y)* únicamente especifican la transformación espacial. Nótese que al agregar situaciones como rotación, escalamiento, etc., el orden y la forma de la transformación se complicará.

II.4. La Transformada de Fourier:

La transformada de Fourier es una poderosa herramienta en el análisis de sistemas lineales. Permite cuantificar los efectos de sistemas digitalizadores, elementos de muestreo, amplificadores electrónicos, filtros convolutivos, ruido y elementos de despliegue. No entraremos a discusión profunda en las bases matemáticas de la transformada, así que nos limitaremos a definirla en su faceta continua [Oppenheim & Willski, 1994]:

La transformada de Fourier de una función unidimensional f(t) se define como:

$$\Im\{f(t)\} = F(s) = \int_{-\infty}^{\infty} f(t)e^{-j2\pi st}dt$$

La transformada de Fourier es una transformación integral lineal que, en el caso general, toma una función compleja de *n* variables reales y la transforma en otra función compleja de *n* variables reales. La transformada inversa de Fourier *F(s)* se define como:

[5] o relativas al espectro

$$f(t) = \Im^{-1}\{F(s)\} = \int_{-\infty}^{\infty} F(s)e^{j2\pi st} ds$$

La transformación es recíproca, por lo que cada función o señal tiene su *par* correspondiente. Para cada función *f(t)*, la transformada de Fourier *F(s)* es única, y viceversa.

Lo que más nos interesa es el caso de las imágenes, por lo que en los aspectos prácticos, tenemos que utilizar una variante de la transformada de Fourier, la transformada discreta de Fourier (DFT). Dicha transformada es fácilmente derivada de la expansión de la serie de Fourier, que se muestra a continuación [Castleman, 1996]:

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos(2\pi \frac{n}{T} t) + \sum_{n=1}^{\infty} b_n sin(2\pi \frac{n}{T} t)$$

donde

$$a_n = \frac{2}{T} \int_{-T/2}^{T/2} f(x)\cos(2\pi \frac{n}{T} x)dx \quad \text{y} \quad b_n = \frac{2}{T} \int_{-T/2}^{T/2} f(x)sin(2\pi \frac{n}{T} x)dx$$

esto representa una función periódica con periodo T gracias a la ayuda de dos secuencias infinitas de coeficientes reales.

Una vez teniendo la expansión de la serie de Fourier, podemos obtener la expresión reducida de la transformada discreta de Fourier. Si $\{f_i\}$ es una secuencia de longitud N, como la que se obtiene al tomar muestras de una función continua a intervalos iguales, entonces la DFT es la secuencia $\{F_n\}$ dada por:

$$F_n = \frac{1}{N} \sum_{i=0}^{N-1} f_i e^{-j2\pi \frac{n}{N} i}$$

y la DFT inversa es:

$$f_i = \frac{1}{N} \sum_{n=0}^{N-1} F_n e^{j2\pi \frac{i}{N} n}$$

donde $0 \le i, n \le N-1$ son índices. Afortunadamente, la DFT, para nuestros fines, es muy cercana a la transformada de Fourier continua, con algunas pequeñas consideraciones son esencialmente equivalentes. Esta flexibilidad nos permite bastante holgura al momento de diseñar procesos. Podemos, por ejemplo, emplear el enfoque de la transformada continua para obtener una solución en un procesamiento de una imagen y después implementar dicha solución con el enfoque discreto.

Afortunadamente existen una clase de algoritmos que reducen el número requerido de operaciones relativas a la transformada discreta de Fourier al orden de *Nlog2(N)* [Bracewell, 1986]. Éstas algoritmos son llamadas transformadas rápidas de Fourier (FFT). *N* debe ser factorizable en un producto de enteros pequeños. La mejor eficiencia y las implementaciones más simples resultan cuando N es una potencia de 2 ($N=2^p$), donde p es un entero). La transformada rápida de Fourier trabaja representando la definición de la DFT como un producto de matrices, y aprovecha la simetría de la misma para factorizarla en un producto de N por N matrices, dividiendo así bastante el número de operaciones a realizar. El factor con el que la FFT reduce la carga computacional es:

$$\frac{N^2}{N\log_2(N)} = \frac{N}{\log_2(N)}$$

este valor se incrementa con N, y para N=1,024, la FFT es aproximadamente 100 veces más eficiente que la implementación directa.

Nótese que las escuetas definiciones que se han dado pertenecen sólo al caso de señales unidimensionales. El salto a las señales bidimensionales (como las imágenes digitales) es muy sencillo, por ejemplo, la DFT en dos dimensiones quedaría así:

$$G(m,n) = \frac{1}{N}\sum_{m=0}^{N-1}\sum_{k=0}^{N-1} g(i,k)e^{-j\frac{2\pi}{N}(mi+nk)}$$

Pero su manipulación como tal no es estrictamente necesaria, ya dicha expresión puede ser dividida en dos: operaciones verticales y horizontales, aprovechando así las dos dimensiones espaciales (en el caso de imágenes digitales). De esta forma, estaremos sólo calculando dos veces una DFT unidimensional. Algunas eficientes implementaciones de la DFT bidimensional utilizan este método de separación.

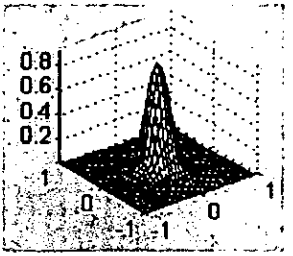
La transformada discreta de Fourier es una poderosa herramienta en el procesamiento de imágenes, en el diseño de filtros digitales y en el tratamiento digital de señales. Como se verá mas adelante, la transformada de Fourier es ampliamente utilizada como agente de correlación de fases, un método de estimación de movimiento. También resultó de útil aplicación al llevar a cabo filtrados de todo tipo sobre imágenes. Respecto a los filtros digitales hablaremos un poco en los siguientes párrafos.

II.5. El Filtrado en Imágenes Digitales:

El filtrado es sumamente importante en la teoría de sistemas y señales. Permite el diseño y aplicación de elementos que "moldean" las señales de entrada a gusto del diseñador. Como en todas las implementaciones físicas, existen limitaciones al reflejar los resultados prácticos contra los teóricos, pero se intenta acercarse lo más posible al filtro ideal. En este trabajo sólo tocaremos el tema de los filtros digitales, ya que los analógicos no encuentran relación directa con el campo de las imágenes digitales. Para llevar a cabo el análisis de un filtro hay que examinar su comportamiento en los dominios temporales y de frecuencia, sobre todo. A continuación mencionaremos algunos sencillos filtros que tiene aplicaciones diversas al mundo de las imágenes:

- Filtros paso-bajas. Muy frecuentemente, una señal o imagen tiene la mayoría de su energía en el rango de frecuencias bajas y medias de su espectro de amplitud. En las frecuencias más altas, la información de interés está a veces ocultada por el ruido. De esta forma, un filtro que reduce la amplitud de las componentes de alta frecuencia puede reducir sustancialmente los efectos del ruido. Hay varios tipos de filtro paso bajas. Dentro de los más simples, se hallan el de "caja", que lleva a cabo un promediado local convolucionando la señal con un pulso rectangular $\Pi(x)$. Entre más grande sea el *kernel*, o filtro de caja, más severa será la ecualización. Este tipo de filtro es muy útil para hacer difuminaciones sencillas en imágenes, variando los parámetros adecuados del filtro. Otro tipo de filtro paso bajas simple es el triangular, que utiliza el pulso triangular $\wedge(x)$ como un filtro paso bajas de respuesta a impulso. A esto se le suele llamar un filtro de promedio ponderado. En dos dimensiones, toma la forma de una pirámide. Para evitar problemas, se puede obtener el efecto del filtro triangular aplicando dos veces el filtrado de caja. Debido a la simplicidad del filtro de caja, esto puede ser más eficiente computacionalmente que usar la señal $\wedge(x)$. De hecho, utilizando tres o más veces sucesivas el pulso rectangular $\Pi(x)$ se pueden emular filtros que, como la función Gaussiana, tengan un comportamiento mucho más suave en el dominio de la frecuencia [O'Neil, 1994]. Debido a que la transformada de una función Gaussiana es también una Gaussiana, esta función da lugar a un filtro paso bajas con funcionamiento muy suave y fino en ambos dominios. Puede, por supuesto,

ser implementado por medio de una convolución en el dominio del tiempo o en el espacio, o por la multiplicación en el dominio de la frecuencia. A continuación se muestra un ejemplo de filtrado paso bajas, en el que la señal es una imagen de una flor. Abajo se halla la gráfica de la respuesta en frecuencia del filtro. Nótese que difumina severamente la nitidez de la imagen, debido a la severidad del filtrado:



Filtro paso bajas

Imagen original

Imagen difuminada

Figura 2.3. Filtro paso bajas y sus efectos

- Filtros paso banda: En algunos casos, los componentes deseados o no deseados de una imagen ocurren predominantemente en diferentes rangos de frecuencia del espectro. Cuando los componentes son separables en esta forma, una función de transferencia que pase o que detenga frecuencias particulares puede ser muy útil. Un filtro paso banda, de manera general, se construye de la siguiente forma: se selecciona una función unimodal no negativa *K(s)* y se convoluciona con un impulso par en la frecuencia s_0. Esto da como resultado una función de transferencia paso banda. La función de transferencia está dada por:

$$G(s) = K(s) * [\delta(s - s_0) + \delta(s + s_0]$$

y la respuesta a impulso es:

$$g(t) = 2k(t)\cos(2\pi s_0 t)$$

- Filtros paso altas: El término filtro *paso altas* generalmente se usa para describir una función de transferencia que es unitaria en frecuencia cero y se va incrementando conforme la frecuencia sube. Dicha función puede ya sea nivelarse en algún valor mayor que la unidad o, más comúnmente, caer hacia cero en altas frecuencias. En el último caso, el filtro paso altas es un caso particular de filtro paso banda, con la restricción de que debe tener ganancia unitaria en frecuencia cero.

  En la práctica es algunas veces deseable tener menos de uno en la ganancia en frecuencia cero, esto para reducir el contraste en las componentes grandes y lentamente variables de la imagen. Si la función de transferencia pasa a través del origen, es llamada *filtro Laplaciano*. Se puede producir una función de transferencia que realce las altas frecuencias expresándola como la diferencia de dos Gaussianas de diferentes anchuras [Castleman, 1996]:

$$G(s) = Ae^{-s^2/2\alpha_1^2} - Be^{-s^2/2\alpha_1^2} \qquad A \geq B, \alpha_1 > \alpha_2$$

  Nótese que la Gaussiana gruesa en el dominio de la frecuencia produce una Gaussiana angosta en el dominio del tiempo y viceversa. A continuación se muestra la respuesta en frecuencia típica de un filtro paso altas bidimensional aplicable, por ejemplo, a las imágenes digitales:

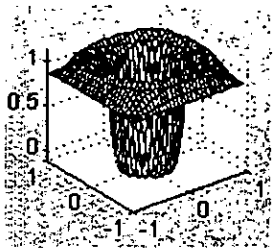


Figura 2.4. Filtro paso altas

Las aplicaciones de los filtros paso altas en el procesamiento digital de señales son varias, y entre las más utilizadas está la detección de bordes. Un filtro paso altas, al solo dejar pasar las frecuencias más altas, elimina las regiones espaciales que observen mayor homogeneidad. Un correcto filtrado paso altas devolverá, en el dominio espacial, una imagen en la que se conserven las regiones donde hubo mayores y más bruscos cambios en luminosidad: regiones con bordes.

Mencionaremos, sólo a modo de breviario, que existen gran cantidad de procedimientos para crear filtros digitales y que hay gran cantidad de filtros digitales que por su efectiva aplicación, son reiterativamente revisados en la bibliografía. Muchos de ellos son utilizados en el campo de las imágenes, como el filtro de Wiener[6], el de Sobel, el de Prewitt, etc.

II.6. Transformadas y Wavelets.

La transformada discreta de Fourier (DFT), de la que ya se ha hablado en el presente capítulo, es solo una de las tantas transformaciones lineales discretas que han probado ser muy útiles en el procesamiento digital de señales. Las imágenes de interés normalmente ocurren en su forma continua y deben ser vistas de esa manera. Como estamos limitados al uso de una representación discreta de una imagen continua, la mayoría del procesamiento requiere que tengamos a la mano consideraciones de sub-muestreo y de interpolación mientras procesamos la información discreta. Algunas aplicaciones, sin embargo, nos

[6] que es un filtro que principalmente funge como el clásico filtro lineal de reducción de ruido

permiten tratar a la imagen digital como una entidad discreta, sin atender particularmente a su posible origen continuo.

Una de esas aplicaciones es la compresión de imágenes. En este caso uno desea codificar una imagen en un formato de información más compacto, ya sea con pérdidas, sin pérdidas, o solo con un tolerable nivel de pérdidas.

De manera general, una transformada discreta bidimensional lineal que lleva de la matriz **F** de N x N a la matriz transformada **G** (también de NxN), se define de la siguiente forma:

$$G_{m,n} = \sum_{i=0}^{N-1}\sum_{k=0}^{N-1} F_{i,k}\Im(i,k,m,n)$$

donde i, k, m y n son variables discretas que van de 0 a N-1 y $\Im(i,k,m,n)$ es la función *kernel*[7] de la transformación. Dicha función puede ser vista como una matriz de bloques de $N^2 \times N^2$ que tiene N filas de N bloques donde cada uno de ellos es una matriz de N x N. Los bloques son indizados por m y n y los elementos de cada sub-matriz de $N \times N$ con i y k. De forma matricial, la transformada puede ser escrita así:

$$\mathbf{G = TFT}$$

Donde T es una matriz unitaria, llamada la matriz kernel de la transformada, como antes. La transformada inversa es

$$\mathbf{F = T^{-1}GT^{-1} = T^{*t}GT^{*t}}$$

Y recupera a **F** exactamente.

La diferencia principal entre dos transformadas unitarias es la elección de las funciones *basis*, o básicas, que vienen a ser las filas de **T**. Las filas de la matriz *kernel* forman un conjunto de vectores básicos para un espacio vectorial N-dimensional. Las filas son ortonormales. Estas son las características principales de las funciones básicas. Cualquier vector en el espacio vectorial puede ser expresado como una suma ponderada, o una combinación lineal de vectores básicos, como nos dice el álgebra lineal [Speziale & Solar, 1985].

Mencionaremos a continuación algunos ejemplos de transformadas lineales importantes.

Transformadas sinusoidales:

[7] también se le suele llamar función semilla, o función médula

Dentro del grupo de las transformadas sinusoidales entra la ya comentada transformada discreta de Fourier (DFT), por lo que no se extenderá su explicación.

- Transformada Coseno Discreta (DCT): La transformada coseno discreta bidimensional se define como sigue [Furht et al, 1990]:

$$G_c(m,n) = \alpha(m)\alpha(n)\sum_{i=0}^{N-1}\sum_{k=0}^{N-1} g(i,k)\cos\left[\frac{\pi(2i+1)m}{2N}\right]\cos\left[\frac{\pi(2k+1)n}{2N}\right]$$

donde los coeficientes son $\alpha(0) = \sqrt{\frac{1}{N}}$ y $\alpha(m) = \sqrt{\frac{2}{N}}$ para $1 \le m \le N$

también como la DFT, la DCT puede ser calculada con un algoritmo rápido. A diferencia de la DFT, la DCT tiene sus valores en el dominio real. Dicha transformada ha encontrado muchas aplicaciones en el procesamiento digital de imágenes, sobre todo en lo que a compresión se refiere, ya que se puede llevar a cabo una selección de los coeficientes que se conservarán en la imagen transformada, de forma que se eliminen ya sea los correspondientes a altas o bajas frecuencias. En el caso de la compresión, generalmente se hace un recorte de coeficientes para sólo dejar el coeficiente de DC (frecuencia cero) y algunos de baja frecuencia, así se reduce el ancho de banda de la imagen al antitransformar sin dañar demasiado la calidad de la misma. Esto se explicará un poco más adelante en el capítulo al hablar más a fondo de los tipos de compresión que hay.

- Transformada Seno Discreta (DST): A diferencia de otras transformadas sinusoidales, la DST es calculada más convenientemente para $N=2^p-1$, don de p es un entero. La DST también tiene un algoritmo de implementación rápida, y algunas propiedades de ésta han demostrado ser de utilidad en asuntos de compresión de imágenes, aunque en menor medida que la DCT, ya que la DST tiene a dispersar la energía mientras que la DCT tiene a concentrarla, por lo que su análisis para esos fines es más natural.
- Transformada de Hartley [Pratt, 1978]: En 1942, Hartley introdujo una transformada integral continua como una alternativa a la transformada de Fourier [Castleman, 1996]. Bracewell definió después una transformada discreta unitaria análoga basada en la transformada de Hartley. La DHT es una alternativa computacional a la DFT. Hay un algoritmo rápido en la transformada de Hartley también. Para aplicaciones de filtrado lineal –particularmente si el *kernel* es simétrico- la DHT puede significativamente reducir la carga computacional, ya que evita la aritmética compleja.

Existe otro tipo de transformadas lineales, las basadas en ondas rectangulares. Muchas de las transformadas de interés en el procesamiento digital de imágenes utilizan funciones base que son variaciones de la onda cuadrada más que de ondas sinusoidales. En general, éstas son más fáciles de calcular, ya que muchas de las operaciones de multiplicación se vuelven triviales. A continuación mencionaremos brevemente algunas de ellas:

- Transformada de Hadamard [Pratt & Andrews, 1969]: Esta transformada es una transformación unitaria simétrica y separable que tiene solamente elementos –1 y 1 en su matriz *kernel*. Existe para $N=2^n$, donde n es un entero. Para generar la matriz de bloques de cualquier número N, se puede partir de la matriz de bloques siguiente:

$$\frac{1}{\sqrt{N}} H_N = \frac{1}{\sqrt{N}} \begin{bmatrix} H_{N/2} & H_{N/2} \\ H_{N/2} & -H_{N/2} \end{bmatrix}$$

Las funciones de Hadamard son también llamadas funciones de Walsh. Es por esto que la transformada de Hadamard es también conocida como transformada de Walsh.
Otras funciones rectangulares de amplio uso son la de Haar y la de Slant.

- Transformada Karhunen-Loéve [Pratt, 1978]: La siguiente ecuación, donde la matriz **A** define una transformación lineal que genera un nuevo vector y de cualquier vector **x**, define una transformada unidimensional llamada Karhunen-Loéve (K-L):

$$\mathbf{Y = A(x - m_x)}$$

Donde **A** está construida de tal manera que sus filas son los eigen-valores[8] de $\mathbf{C_x}$. El vector transformado, **y**, es un vector aleatorio con media cero. Su matriz de covarianza está relacionada con **x** por:

$$\mathbf{C_y = AC_xA^t}$$

Ya que las filas de **A** son eigen-valores de $\mathbf{C_x}$, $\mathbf{C_y}$ es una matriz diagonal que tiene los eigen-valores de $\mathbf{C_x}$ a lo largo de su diagonal.

La capacidad de reducción de dimensión que tiene la transformada K-L la hace muy útil para la compresión de imágenes. Las imágenes multiespectrales, por ejemplo, tienen muchos valores de niveles de gris en cada pixel, cada nivel de gris correspondiente a una banda espectral diferente. Entonces, una imagen multiespectral de 1000x1000 en 24 canales puede ser vista como un conjunto de un millón de vectores de 24 elementos cada uno (los pixeles). La propiedad de reducción de la K-L

[8] o valores característicos

puede ser aplicada a este conjunto de vectores. Ya que la correlación entre las diferentes bandas espectrales de una imagen multiespectral tiende a ser alta, muchos de los 24 eigen-valores serán pequeños. Esto significa que la pila de 24 imágenes monocromáticas puede ser representada con un pequeño error con solamente las principales imágenes componentes. Cada una de ellas es calculada como la suma ponderada, o la combinación lineal, de las 24 originales. Además, cada imagen en el conjunto original puede ser reconstruida, aproximadamente, como una combinación lineal de las pocas imágenes de componentes principales. Esto simplifica grandemente el almacenamiento y la distribución de, por ejemplo, imágenes tomadas desde satélites.

Como la transformada de Fourier, las transformadas unitarias en general expanden una imagen como una suma ponderada de imágenes básicas. El proceso de transformación determina los coeficientes de peso para dicha suma, mientras que la transformada inversa arma la imagen a partir de la expansión de las imágenes básicas. El filtrado basado en transformadas, como ya se dijo anteriormente, supone la modificación de los coeficientes de peso antes de llevar a cabo la reconstrucción de la imagen vía la antitransformada. Con el filtrado lineal, la transformada es la de Fourier y la modificación es llevada a cabo multiplicando al espectro por una función de transferencia. En un caso de filtrado más general, la matriz de coeficientes es modificada (por algún medio) y la transformada inversa produce la imagen filtrada.

Como es de suponerse, lo que establece las diferencias entre el comportamiento de las distintas transformadas es la naturaleza de los vectores básicos (y de sus imágenes básicas resultantes). Por ejemplo, la contaminación por ruido sinusoidal aparece de manera muy compacta en el dominio de una transformada sinusoidal, por lo que es fácilmente eliminado convirtiendo ciertos coeficientes a cero. Las transformadas de onda rectangular serían menos adecuadas para este objetivo, ya que la contaminación no sería tan separable de la señal en el dominio de la transformada. Por dar otro ejemplo, la transformada de Haar es un buen candidato para encontrar bordes y líneas verticales y horizontales, ya que varias de sus imágenes básicas satisfacen exactamente dichas características.

Un interés considerable se ha venido dando en los últimos años con el afán de generar de nuevas técnicas de transformación que toquen específicamente los problemas de la compresión de imágenes, la localización y detección de bordes y regiones, y el análisis de texturas. Estas técnicas han encontrado un método novedoso de estudio, las transformaciones *wavelet*. Trataremos de explicar brevemente sus fundamentos en la siguiente sección y enumeraremos algunas de sus aplicaciones directas en el campo de las imágenes digitales.

Recordemos que la transformada de Fourier usa, como sus funciones básicas ortonormales, ondas sinusoidales, que son llamadas así porque recuerdan las olas del oceáno u ondas propagándose en otro medio. Para una transformación integral, estas funciones se extienden hacia el infinito en ambas direcciones. Los vectores básicos de la transformada discreta de Fourier son también distintos de cero a lo largo de todo el dominio, esto significa que no tiene *soporte compacto*.

De forma distinta, las componentes de señales transitorias son diferentes de cero sólo en un corto intervalo. Igualmente, algunas características importantes en las imágenes (bordes, por ejemplo) son altamente localizados en posiciones espaciales. Dichos componentes no son parecidos a ninguna de las funciones básicas de Fourier, y por lo tanto no están representados de manera compacta en los coeficientes de la transformada. Esto hace que la transformada de Fourier y algunas otras, como las que hemos mencionado, sean representaciones sub-óptimas para comprimir y analizar señales e imágenes que contengan componentes transitorios o altamente localizados.

Los ingenieros y matemáticos han explorado varios acercamientos utilizando transformadas que tengan funciones básicas de duración limitada. Son ondas de duración limitada y fueron bautizadas con el nombre de *wavelets*. Las transformadas basadas en ellas son llamadas transformaciones *wavelet*[9].

La literatura que habla del procesamiento de señales contiene muchos trabajos que analizan a las señales en términos de un espacio bidimensional tiempo-frecuencia. Dichos estudios preceden directamente al desarrollo de la teoría de las transformadas wavelet, y ahora se puede decir que ya "caben" en el mismo marco moderno de trabajo. De acuerdo a él, cada componente transitoria de una señal mapea a una posición en el plano tiempo-

frecuencia que corresponde a la frecuencia predominante de dicha componente y a su tiempo de ocurrencia. En el análisis de imágenes, el espacio es tridimensional y puede ser visto como una pila de imágenes. Un componente localizado aparecerá primeramente en el nivel de la pila que corresponde a la frecuencia predominante de dicho componente.

Este tipo de estudio comenzó con la transformada de Fourier de Gabor, y dio lugar a la transformada de tiempo corto de Fourier (STFT), para después ver nacer la famosa codificación de sub-banda (*subband coding*) [Castleman, 1996].

A pesar de que las transformadas por wavelets son relativamente nuevas en la escena del procesamiento digital de imágenes, ya tienen ganado un campo importante en las aplicaciones prácticas, como son:

- Compresión de imágenes: La transformada wavelet discreta descompone una imagen en un conjunto de imagenes ortonormales cada vez más pequeñas. Además, mientras el histograma de niveles de gris de la imagen original puede tener cualquier forma, los relativos a imágenes transformadas por wavelets son comúnmente unimodales y simétricos con respecto al cero. Esto simplifica el análisis de las propiedades estadísticas de la imagen. Hay ocasiones en las que uno puede ya sea cuantizar fuertemente o de plano eliminar totalmente aquellos coeficientes que tienen valores muy pequeños. Mallat [Mallat, 1991] y otros han estudiado la posibilidad de reconstruir una imagen a partir únicamente de las localizaciones de los cruces con cero de su transformada wavelet. Sabiendo de antemano que la reconstrucción perfecta de imágenes es generalmente imposible, muchas de ellas pueden ser adecuadamente aproximadas por este código de alta compactación.
- Realce de imágenes: La transformada wavelet discreta (DWT), descompone una imagen en componentes de distinto tamaño, posición y orientación. Así como en el filtrado lineal en el dominio de la frecuencia de Fourier, uno puede alterar la amplitud de coeficientes en el dominio de la transformada wavelet antes de obtener la antitransformada. Esto puede acentuar selectivamente componentes interesantes a costa de los no deseados.

[9] suelen ser llamadas también ondeletas, del francés ondelette, debido a la gran cantidad de literatura francesa que trata el tema

- Fusión de imágenes: La fusión de imágenes combina dos o más imágenes registradas del mismo objeto en una sola que es más fácil de interpretar que cualquiera de las originales. Esta técnica encuentra aplicación en la interpretación de imágenes multiespectrales, así como en aplicaciones médicas, donde imágenes del mismo cuerpo son obtenidas a partir de distintas modalidades.

II.7. Compresión.

Hemos dejado casi hasta el final del presente capítulo, a manera de conclusión, el tema que consideramos que es el más importante por el enlace que representa con nuestro trabajo práctico: la compresión de imágenes. Ésta se lleva a cabo eliminando la información innecesaria o redundante.

La información de las imágenes contiene una cantidad considerable de información que es redundante y mucha que es irrelevante, haciéndolos candidatos directos para las modernas técnicas de compresión de imágenes. Las técnicas de compresión de información explotan las redundancias inherentes e irrelevantes transformando un archivo de datos en uno de tamaño menor del que el original puede ser reconstruido más tarde (exacta o aproximadamente). La razón entre los tamaños de los dos archivos (el radio de compresión) especifica el grado de compactación.

Algunos algoritmos de compresión de datos son *con pérdidas*, mientras que otros no lo son. Un algoritmo *sin pérdidas* elimina sólo la información redundante, de manera que se puede recuperar la imagen exacta después de descomprimir el archivo. Un método de compresión con pérdidas elimina también información considerada irrelevante y esto permite sólo una reconstrucción aproximada del original, más que un duplicado exacto. Como es de suponerse, los métodos de compresión con pérdidas logran radios de compresión más altos. Para las imágenes digitales, una pequeña pérdida de fidelidad es casi siempre un trato aceptable para lograr un muy alto nivel de compactación. Algunas imágenes, así como los programas ejecutables, no toleran generalmente alguna alteración en sus datos, en estos casos se aplicaría la compresión sin pérdidas.

Los tiempos requeridos para la compresión y descompresión de datos no son despreciables. Los algoritmos que logran la compactación mas densa no son generalmente los más rápidos, así que hay que tomar una decisión adecuada en cada caso.

1. Compresión sin pérdidas.

Los algoritmos de compresión de datos sin pérdidas caen en dos grandes categorías: las técnicas basadas en diccionario y los métodos estadísticos. Las técnicas basadas en diccionario generan un archivo comprimido conteniendo códigos de longitud revisada (de usualmente 12 a 16 bits), cada uno de los cuales representa una secuencia particular de bytes en el archivo original. Un ejemplo que cae dentro de esta clasificación es la codificación por longitud de la corrida (RLE).

RLE es el tipo más simple de compresión basada en diccionario. Las imágenes – particularmente las que tiene pocos niveles de gris- suelen contener regiones de pixeles adyacentes, todos con el mismo nivel de gris o de color. En una imagen almacenada línea a línea, las series de pixeles que tienen el mismo valor se llaman *corridas*. Se puede guardar sólo un código especificando ese valor, seguido de la longitud de la corrida, lo que es mejor que simplemente almacenar el mismo valor varias veces seguidas. Funciona muy bien en ciertos tipos de imágenes, como las que tienen fondos homogéneos.

Los métodos estadísticos implementan la compresión de datos representando caracteres frecuentemente ocurrentes en el archivo con menos bits de los que realmente tienen. Este es el tipo de análisis que Samuel Morse utilizó cuando definió el código internacional del telégrafo.

Dentro del estudio de los métodos estadísticos se encuentra intrínseca la teoría de la información, tan comentada en la literatura, y de la que hablaremos muy brevemente [González & Woods, 1996].

Supongamos que tenemos una fuente sin memoria de mensajes que usa un alfabeto $\{a_k\}$, $k= 0, 1, ..., K-1$. Aquí, el a_k son los símbolos de tal alfabeto. Supóngase entonces que la probabilidad de ocurrencia de cada símbolo es conocida y denotada como $P(a_k)$. En un mensaje que proviene de una fuente sin memoria, el ordenamiento de los símbolos en el mensaje es irrelevante, sólo su presencia en el mensaje importa.

Shannon [Shannon, 1948] definió una medida de la información impartida por la ocurrencia de el símbolo a_k en el mensaje como:

$$I(a_k) = -\log[P(a_k)]$$

La medida es satisfactoria porque a) entre menos probable sea un símbolo, con mayor información contribuye al mensaje, y b) la información en un mensaje es la suma de la información con la que han contribuido los símbolos que lo componen.

La entropía de la fuente del mensaje está definida por:

$$H = E\{I(a_k)\} = -\sum_{k=0}^{K-1} P(a_k)\log[P(a_k)]$$

y especifica el contenido promedio de información contenido por símbolo de los mensajes generados por la fuente. La entropía de un mensaje es no negativa y llega a su máximo valor cuando todos los símbolos son igualmente probables. Si escogemos 2 como la base del logaritmo, las unidades de entropía serán entonces *bits por símbolo*.

El código de Huffman [Huffman, 1952], introducido en 1952, es un método estadístico sin pérdidas que siempre encuentra un código de longitud variable con la menor redundancia posible. Usa un árbol de codificación binaria para representar los valores que tienen mayor ocurrencia en pocos bits y los valores con menor ocurrencia con más bits.

2. Compresión con pérdidas.

La cuantización es un método ampliamente utilizado para lograr compresión con pérdidas. Hay dos tipos de cuantización: escalar y vectorial.

- Cuantización Escalar: Es una de las maneras más simples para reducir el volumen de datos. Esto se lleva a cabo forzando a que la imagen tenga un número menor de niveles de gris. Cuando la entrada cae entre dos umbrales de decisión, la salida es colocada en el nivel representativo correspondiente. Lloyd y Max [Max, 1960] mostraron que, para una imagen que tiene un función de densidad de probabilidad dada, evidenciada por su histograma, el esquema de cuantización que minimiza el error cuadrático medio introducido por la cuantización tiene las siguientes propiedades interdependientes: 1) cada umbral de decisión cae exactamente a la mitad del camino entre dos niveles representativos adyacentes, y 2) cada nivel representativo cae en el centroide de la sección de la función de densidad de probabilidad que está entre dos umbrales

sucesivos de decisión. Esto establece un sistema de ecuaciones que normalmente deben ser resueltas iterativamente para determinar los umbrales de decisión y sus niveles representativos.

- Cuantización Vectorial: Es un procedimiento más moderno por el cual cada vector perteneciente a la señal de entrada (por ejemplo, bloques de pixeles pertenecientes a una imagen digital) es referenciado, por su similitud, o por algún criterio de cercanía, con algún símbolo correspondiente a un *libro de códigos*. El libro de códigos es generado a través de un proceso iterativo y cuenta con un número limitado de vectores representativos que se han obtenido con secuencias de imágenes de entrenamiento, esto con el objetivo de contar con un conjunto de vectores que puedan "suplir" a los vectores originales de la señal de entrada. El resultado es una compresión muy alta debido a que cada vector de la señal de entrada es simplemente sustituido por un índice (perteneciente al libro de códigos) para después ser reconstruida por un decodificador que posea el mismo libro de códigos. Según la teoría de la tasa de distorsión, la cuantización vectorial, a diferencia de la escalar, tiende a acercarse bastante a la optimalidad. A continuación se muestra una imagen original y su correspondiente imagen cuantizada vectorialmente usando un libro de códigos de 256 vectores y vectores de 4x4 pixeles.

  La compresión lograda, en este caso, es de 1:16:

Imagen Original

Imagen Cuantizada Vectorialmente

Figura 2.5. Efectos de la cuantización vectorial en la imagen "Lena"

Compresión por transformada: Como ya se dijo, una de las aplicaciones más utilizadas de las transformadas discretas es la compresión de imágenes. Combinadas con otras técnicas, permiten la transmisión, almacenamiento y despliegue de imágenes y secuencias de video que de otra forma serían poco prácticas.

Supongamos que tenemos un grupo de imágenes que serán codificadas en una representación compacta de datos. Podemos transformar las imágenes, descartar los coeficientes que sean cercanos a cero, y cuantizar severamente aquellos que son pequeños, para concentrar nuestra transmisión de datos y nuestros recursos de almacenamiento solo en los coeficientes que contienen mas información sobre la imagen. Cuando la imagen sea reconstruida más tarde, muy poco contenido importante se habrá perdido. Esta técnica explicada a grandes rasgos se llama codificación de imágenes por transformada, y que tiene las siguientes consideraciones importantes:

- Codificación de bloques: En muchos de los casos la imagen es dividida en bloques (típicamente de 8x8 o de 16x16 pixeles) y cada uno de ellos es transformado por separado. Esto simplifica el proceso de transformación. Sin embargo, puede aparecer el llamado "efecto de bloquificación", si después de cuantizar y eliminar coeficientes aparecen divisiones notorias entre los bloques en la imagen reconstruida.
- Localización de bits: Hay que establecer cuántos bits serán usados para codificar cada uno de los coeficientes que resultan de la transformación de un bloque de imagen.
- Consideraciones de calidad de la imagen: Siempre hay un "estira y afloje" entre el radio de compresión logrado y la cantidad de información perdida en la codificación. Normalmente, el ojo humano es el último juez que decidirá si la pérdida de información es aceptable o no. De esta forma, podemos afirmar que hay componentes objetivas y subjetivas en el proceso de diseño.
- Selección de la transformada: Hay que analizar qué tanto una transformada en particular soportará el tipo de compresión deseado. Esto depende intrínsecamente de la transformada y de la naturaleza de las imágenes que se vayan a comprimir. La eficiencia de un esquema de codificación depende de la carga computacional en los pasos de codificación y decodificación, así como el grado de compresión obtenido. Ya se ha hablado un poco de la gran variedad de transformadas que hay, en qué se

distinguen unas de otras y qué tan práctica o impráctica resulta su implementación. El resto está sujeto al diseñador.

8. El Video Digital:

Aquí viene el gran enlace con el estándar MPEG: la introducción de las ténicas de procesamiento de imágenes al video digital. Explicaremos un poco la política de procesamiento del video digital, habiendo hecho ya un recorrido por el tratamiento de las imágenes individuales.

Es bien conocido el hecho de que la representación digital de una señal analógica de video que ocupa unos cuantos megahertz de ancho de banda alcanza tasas de bits que superan los 100 [Mbps] para lograr una calidad similar y la tarea básica de los protocolos de comunicación es garantizar que los mensajes enviados a través de una red lleguen a su destino inalterados. Dichas exigencias resultan prohibitivas para muchos procesadores y redes. Esta situación fue reconocida a principios de la década de los 70 y hasta la fecha la compresión de imágenes es un campo de investigación que no ha perdido interés.

Como ya se dijo en páginas anteriores, la compresión se logra aprovechando la redundancia que existe en la información que se posee: en el caso del video, existe redundancia tanto espacial (en un cuadro de la película) como temporal (entre varios cuadros).

El primer tipo de redundancia está muy estudiado, sobre todo por la comunidad de compresión de imágenes fijas y a su eliminación se le denomina "compresión intra". En el segundo se emplean técnicas como la "predicción" de movimiento de regiones en un cuadro con base en el cuadro anterior, y se envía únicamente el error de predicción, técnicas conocidas como "compresión inter" o bien "no-intra". Debemos aclarar en este punto, que estamos hablando de dos tipos de redundancia: de codificación y psicovisual. La primera es, como ya se dijo, un concepto estadístico, independiente del tipo de información de que se trate, mientras que la segunda se deriva de los estudios que se han hecho en el sistema de visión humano (HSV por sus siglas en inglés). Al mecanismo encargado de eliminar el primer tipo de redundancia en el video digital se le denomina "codificador entrópico". Los

estándares de video por lo general parten de una representación de la imagen ("modelo del codificador") que permite la explotación de las propiedades y redundancia de las señales por un codificador entrópico.

- Representación digital de la señal de video [RI 1], [RI 2]:

Un video trata de representar el movimiento tridimensional en una proyección bidimensional, dando por resultado una imagen variante con el tiempo, es decir, una señal de tres variables independietes, dos espaciales y una temporal. El video digital muestrea esta señal en tiempo y espacio, dando lugar a una serie de imágenes fijas, denominadas cuadros, cada uno de los cuales está representada despliegue, como un arreglo de pixeles (picture elements), cada uno de los cuales tiene un valor numérico que identifica completamente su color ante el sistema de despliegue. Las características de los compresores hacen ver que esta representación no es la más adecuada para fines de procesamiento estadístico de la imagen. A continuación, estudiaremos la representación antes mencionada, y la manera como puede transformarse (sin pérdida) a un formato más adecuado para su análisis con fines de compresión.

Por lo general, una imagen a color es representada en términos de los componentes que la forman, es decir, está formada por la composición aditiva de los valores de ciertos parámetros (tres). En términos más técnicos, se dice que la señal de video es una combinación lineal, donde la base la forman tres funciones de frecuencia adecuadamente elegidas. Existen una gran variedad de bases: RGB (para monitores), HSB (para televisores), CMYK (para impresoras), etc. Existen funciones que traducen una representación a otra.

En el procesamiento de video la representación más comúnmente empleada es la de luminancia-crominancia. La luminancia representa el brillo de la imagen, mientras que la crominancia se ocupa del color. Las razones para la elección de esta base son psicovisuales: el HSV es más sensible a la luminancia que a la crominancia, lo que permite representar esta última dimensión a menor resolución sin grandes pérdidas perceptibles. Para pasar de un esquema de codificación RGB a uno de luminancia-crominancia, se utilizan las siguientes ecuaciones:

R= luminancia (Y) + diferencia de color de rojo (C_r o V)

Y=0.3R+0.6G+0.1B

B= Y + U; Cr=(V/1.6)+0.5

G= Y + V; (U/2)+0.5

y, dado que C_r + C_b + C_g = constante, no necesitamos más que dos componentes de crominancia, por lo general C_r y C_b. Además, como el HSV es más sensible a la luminancia que a las crominancias, por lo general se submuestrean estas componentes en una razón de 2 a 1. Este submuestreo se puede hacer de varias maneras, pero por lo general se promedian los valores de crominancia correspondientes a los cuatro bloques en esa posición. A un grupo de 4 bloques (de 8x8) de luminancia y 2 de cada una de las crominancias se les conoce como MCU (minimum coded unit), y por lo general se utilizan como el elemento decodificación y decodificación.

Como hemos dicho, la señal de video presenta frecuentemente redundacia (o correlación) en las dimensiones espacial y temporal. La compresión de una imagen fija puede lograrse utilizando compresión sin pérdida pero este tipo de compresión ha demostrado no ser suficiente para alcanzar tasas de bits transmitibles, por lo que se ha de recurrir a la eliminación de la redundancia subjetiva, es decir, echar mano a la información que se dispone del HSV para eliminar datos visualmente imperceptibles, lo que reducirá la cantidad de datos a enviar sin dañar severamente la calidad de la señal (compresión con pérdida).

La mayor parte de los estándares de transmisión de video utilizan las mismas técnicas: Codificación Predictiva de diferencias entre muestras (DPCM -Digital Pulse Code Modulation) y la codificación por transformada, de la que ya se habló anteriormente.

- Codificación Predictiva de diferencias entre muestras (DPCM -Digital Pulse Code Modulation):

Esta es una de las técnicas más simples, pues consiste en recorrer una imagen en orden de despliegue (raster, es decir, de izquierda a derecha y de arriba a abajo). Cada pixel está representado por un número de 8 bits, pero el valor real del pixel no se transmite, pues el

codificador "predice" el valor probable del pixel, basado en los pixeles ya enviados, enviando únicamente el error de predicción (diferencia entre el valor del pixel y su valor predicho) posiblemente cuantizado. Dado que la mayor parte de las imágenes son espacialmente redundantes, pixeles vecinos están altamente correlacionados, y el error de predicción es generalmente pequeño. Todavía puede mejorarse la compresión usando códigos de longitud variable. Debe tomarse en cuenta que la eficiencia de codificación depende de la exactitud del predictor. El predictor más simple consiste en utilizar el valor del pixel apenas transmitido; mejores predictores se logran promediando regiones cada vez más grandes alrededor del pixel en cuestión. Un esquema todavía mejor es adaptar el predictor a las características de la imagen.

- Codificación por transformadas en video digital:

Las transformadas son un recurso muy utilizado en el campo de la compresión de imágenes fijas. Por lo general se aplican las transformadas a bloques de pixeles, más que imágenes completas. Entre más grandes sean estos bloques, mayor es la eficiencia de compresión. También se obtiene mayor compresión si se eligen bloques de forma y tamaño variable, pero, en general, estas dos soluciones elevan la complejidad computacional de los algoritmos que calculan las transformadas.

La DCT actúa sobre bloques (generalmente cuadrados de 8 o 16 pixeles de lado, de preferencia los primeros pues proporcionan el mejor equilibrio en eficiencia de compresión y complejidad computacional), los cuales son transformados, para posteriormente ser cuantizados y reordenados según su frecuencia. El bloque con mayor concentración de energía es el primero, el de DC (esquina superior izquierda), y es el de menor frecuencia espacial (contiene el promedio escalado de la totalidad de la imagen, reflejando que cos(0)=1). La frecuencia horizonal aumenta hacia la derecha, y la vertical hacia abajo. Un ordenamiento en zigzag en general ordenará los coeficientes de acuerdo a su frecuencia, es decir, a la perceptibilidad de ese coeficiente por el HSV. Para un bloque típico, una buena parte de los coeficientes será cero, y los valores más grandes se hallarán alrededor del coeficiente de DC. Un paso de cuantización posterior cambia la amplitud de la señal para adecuarlas al conocimiento que se tiene del HSV, ponderando cada coeficiente por un valor

mayor que uno, y redondeando el resultado de la división al entero más cercano. Los coeficientes de ponderación varían de acuerdo a la frecuencia en la posición correspondiente, y son el resultado de una multitud de experimentos con un gran número de imágenes. Estas matrices también toman en cuenta si se está representando luminancia o crominancia, si el bloque se está codificando inter o intra y la posición del bloque, ya que algunos bloques tienen que ser codificados más precisamente que otros, sobre todo cuando se trata de gradientes suaves, donde pequeñas inexactitudes se hacen evidentes.

- Eliminación de la redundancia temporal.

La predicción por compensación de movimiento es una herramienta poderosa para reducir la redundancia temporal entre cuadros, por lo que es muy utilizada en esquemas de compresión como el MPEG 1 y 2 como técnica de predicción para la codificación DPCM temporal. El concepto de compensación de movimiento está basado en la estimación de movimiento entre cuadros, es decir, si todos los elementos en una escena están desplazados en el espacio, el movimiento entre cuadros puede describirse con razonable precisión si se limita el número de parámetros de movimiento (en adelante llamados vectores de movimiento), por ejemplo, mandando un solo vector por cada bloque de 16x16 pixeles. Esta representación de baja resolución tiene, por supuesto, sus bemoles, pues resulta en una compensación inadecuada y varios artefactos de bloques se hacen visibles (sobre todo en los bordes de los bloques). En el siguiente capítulo describiremos en detalle diversos algoritmos de estimación de movimiento, identificando sus ventajas y desventajas. Baste decir aquí, entonces, que a pesar de que el modelo de casamiento de bloques es simple, tiene severas limitantes, la primera de ellas que no puede calcular las rotaciones o deformaciones de bloques de un cuadro al otro, así como discontinuidades en el campo de movimiento.

La codificación de video hace uso posteriormente de las ya mencionadas técnicas de codificación entrópica (códigos de longitud variable y codificación por longitud de la corrida) para lograr compresión de canal y adecuar la información para ser almacenada o transmitida.

- Códigos de longitud variable

La codificación de longitud variable (VLC) aprovecha el hecho de que algunos valores ocurren con mayor frecuencia que otros, máxime después de que cada cuadro ha sido objeto de un proceso de predicción, transformación y cuantización. En particular, estos procesos resultan en una predominancia de ceros o de valores cercanos a cero en los coeficientes de DCT. Es natural comprender que si se asignan cadenas cortas como código de valores frecuentes, se obtendrá una reducción en la tasa de bits considerable (dependiendo, por supuesto, de la distribución estadísticas de los valores en la DCT).

- Codificación por longitud de corrida (RLC)

Como ya se ha dicho, el efecto de las técnicas de codificación es reducir lo más posible el número de datos que tienen que ser transmitidos. En el caso de los coeficientes de DCT, donde la gran mayoría es cero o tienen un valor muy cercano, es frecuente hallar cadenas razonablmente largas de 0s. Puede favorecerse este efecto recorriendo los coeficientes de DCT en zig-zag (técnica que se explicó arriba y se justificó por argumentos psicovisuales). Se ha comprobado que existen beneficios en asignarle un código a una cadena de determinada longitud de un símbolo que transmitir un código dicha longitud para ese símbolo, técnica que se conoce como codificación por longitud de corrida.

III. Estimación de Movimiento

III.1. Conceptos Básicos.

El movimiento relativo entre un sistema de visión y el ambiente en observación causa movimiento en la imagen, que resulta ser el movimiento del ambiente proyectado en el plano de imagen del sistema visor [Mitchie & Bouthemy, 1996]. A la proyección de cada punto visible del ambiente puede ser asociado un vector de velocidad, y el campo de velocidades es conocido como *flujo óptico* [Gibson, 1950].

La estimación y análisis de movimiento en secuencias de imágenes digitales juega actualmente un papel muy importante dentro de los paradigmas de la computación visual. Algunos ejemplos son:

- Detección de movimiento en escenas. El movimiento en la imagen puede ser indicativo del movimiento en la escena observada. En sistemas de visión estáticos, la detección de movimiento en la escena se logra directamente de la detección de movimiento en la imagen. En sistemas en movimiento hay que compensar el movimiento de la cámara.
- Segmentación de objetos. El movimiento en imágenes puede ser utilizado para identificar regiones de la imagen correspondientes a diferentes objetos en movimiento y para distinguirlos del contexto.
- Rastreo de objetos (tracking): Consiste en el seguimiento de objetos en una escena a partir de su comportamiento dinámico actual.
- Estimación de profundidad y movimiento real de objetos físicos: La distancia relativa de un objeto a la cámara y su velocidad física pueden ser cuantitativamente relacionadas con la posición en la imagen y el flujo óptico[1].

Actualmente, los estudios enfocados al análisis de movimiento en imágenes están motivados por muchas aplicaciones. Algunos de los dominios en los que el análisis de movimiento ha alcanzado un avance especial son:

[1] El flujo óptico, definido más formalmente por Horn y Schunk es la distribución de velocidades aparentes en el movimiento de patrones de brillantez en una imagen. Puede surgir de el movimiento relativo entre los objetos y el espectador [Horn, Schunk, 1981].

1. Televisión. Para codificar imágenes de video con compensación de movimiento.
2. Robots móviles. Conferir a los robots la capacidad de navegar autónomamente en ambientes parcial o totalmente desconocidos.
3. Imágenes de satélite. Particularmente en meteorología, estimando movimiento de nubosidad y estableciendo patrones de viento.
4. Aplicaciones militares. Para el seguimiento de objetivos y navegación autónoma de vehículos.
5. Imágenes biomédicas. Análisis automático de movimientos cardiacos, estudio de movimentos humanos, etc.
6. Monitoreo de espacios. Detección de intrusos, revisión de tráfico urbano, etc.
7. Realidad virtual e interfaces. Interpretación de movimientos faciales (gestos) para el diseño de nuevas interfaces hombre-máquina.

Las aplicaciones en las que intervienen los ejemplos anteriores son diversas, pero el estándar MPEG utiliza la estimación de movimiento primordialmente para *comprimir*. La meta de el análisis de movimiento en este caso es explotar la redundancia espacio-temporal para reducir la tasa de transmisión, siempre y cuando se conserve cierta calidad en las imágenes reconstruidas [Mitchie & Bouthemy, 1996]. El principio utilizado es el siguiente: si se logra detectar el movimiento entre escenas (imágenes digitales), la cantidad de información que el codificador tendrá que enviar será menor, ya que sólo atenderá a los objetos en movimiento, dejando el resto de la imagen *igual*. Éste es un principio muy explotado en los sistemas actuales de compresión de video. La diferencia radica en cómo se estima el movimiento.

A diferencia de la compensación de movimiento, que es un aspecto normativo del estándar MPEG, la estimación de movimiento no lo es. La compensación de movimiento se refiere al uso de vectores de movimiento en la codificación y decodificación de la secuencia, mientras que la estimación de movimiento se refiere a la determinación de dichos vectores de movimiento.

En el codificador la diferencia entre la imagen "fuente" y la predicción se codifica; en el decodificador esta diferencia es decodificada y agregada a la predicción para obtener la salida decodificada. El asunto importante es que el codificador debe estimar los

desplazamientos antes de codificarlos en el flujo de bits; mientras que el decodificador sólo los leerá.

La estimación de movimiento es absolutamente necesaria en cualquier implementación completa de un codificador MPEG, pero los procedimientos para obtenerla no están normados. Lo que es normativo son las reglas y constantes especificados en el estándar, como por ejemplo, que sólo habrá un vector por macrobloque en imágenes P[2], dos vectores por macrobloque en imágenes B, los tamaños de los vectores y la restricción a meros movimientos traslacionales [Mitchell, Pennebaker, Fogg & Le Gall, 1996].

Las técnicas actuales de estimación de movimiento son muy diversas y parten de distintas fuentes teóricas. El estándar MPEG se centra en modelos sencillos que involucran simplificaciones significativas. En primer lugar, sólo asume movimientos traslacionales y dicho movimento se presenta en bloques de pixeles, no en pixeles individuales. Considera, además, que los bloques no son deformables, sino que mantienen su forma cuadrangular al ser trasladados.

Las técnicas de estimación de movimiento para movimientos traslacionales simples pueden ser clasificadas en dos rubros principales: pixel-recursivas y *block-matching* (comparación de bloques) [Furht, Smoliar & Zhang, 1990]. Las primeras se refieren a algoritmos que son programados en sistemas donde los vectores de movimiento pueden variar de pixel a pixel. Las técnicas de *block-matching* son, a grandes rasgos, las aplicables a ambientes donde un solo vector de movimiento es asignado a un bloque de pixeles de tamaño determinado[3].

Si deseamos determinar el vector de desplazamiento óptimo de una predicción, habría que hacer una búsqueda exhaustiva sobre otros posibles desplazamientos para encontrar el mejor valor. Esta búsqueda, sin embargo, supone que se tienen los recursos computacionales necesarios para llevar a cabo tal tarea. La estimación de movimiento es uno de los aspectos de MPEG que resulta más demandante desde el punto de vista de la complejidad computacional, por lo que se ha echado mano de algoritmos rápidos, que reducen la calidad de la búsqueda encontrando valores que probablemente no sean los mejores, pero que reducen dramáticamente la carga computacional [Andleigh & Thakrar,

[2] En capítulos posteriores se hablará con detalle de esta nomenclatura relativa a tipos de imágenes MPEG-1.
[3] Generalmente llamado "Macrobloque", como se verá después.

1996]. El presente capítulo explora distintas posibilidades de estimar movimientos en secuencias de imágenes aplicando algoritmos de distintos tipos, que se explicarán con detalle en sus respectivos subcapítulos.

III.2. Block-Matching.

A pesar de que el estándar MPEG-1 no especifica un método de estimación de movimiento en particular, la estimación de movimiento basada en bloques se vuelve una opción natural. El movimiento de bloques es ampliamente utilizado también en varias aplicaciones distintas de video digital, ya que además de la facilidad que presenta para ser implementado el software, tiene muy poca complejidad al llevarlo a hardware [Gharavi, H. & Mills, M., 1990].

Para entender el modelo de movimiento de bloques, suponemos que la imagen está compuesta de bloques móviles. Para el caso de este estudio, consideraremos movimientos traslacionales en dos dimensiones únicamente. Observe la siguiente figura.

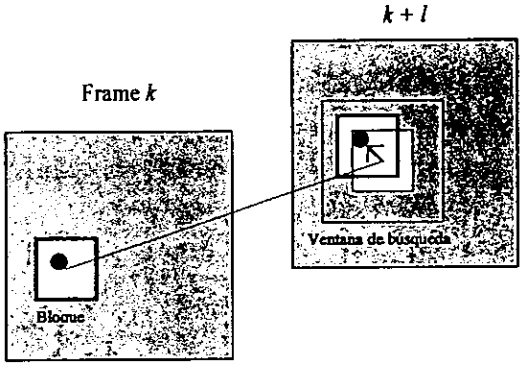


Figura 3.1 Block-Matching

El modelo de movimiento traslacional de bloques *s* puede ser simplificado de la siguiente manera, donde *d1* y *d2* son los componentes del vector de desplazamiento (y están cuantizados al entero más cercano), *k* es el frame o cuadro correspondiente y n_1, n_2 son las componentes del pixel en cuestión:

$$s(n_1, n_2, k) = s(n_1 + d_1, n_2 + d_2, k + l)$$

Esto es porque un bloque B en el "frame" k, de tamaño $N \times N$ centrado en el pixel n = (n_1, n_2) es modelado como una versión recorrida pero del mismo tamaño de el bloque en el "frame" $k + l$, para l entera.

Para el caso presentado, la compensación de movimiento puede ser lograda copiando la información de escala de grises o de colores del bloque correspondiente en el "frame" $k + l$ con un criterio pixel a pixel. La popularidad del método de compensación y estimación de movimiento basados en el modelo de bloques traslacionales se origina en los bajos requerimientos que hay que considerar para representar el campo de movimientos, ya que se necesita sólo un vector por bloque, y la prestancia para ser implementados en sistemas VLSI[4] de bajo costo. De cualquier forma, todo tiene un precio y estos modelos tienen defectos al hacer acercamientos bruscos en escena, en movimientos rotacionales y en deformaciones locales de objetos. También se puede llegar a presentar un efecto de "bloquificación" en las escenas de aplicaciones con tasas muy bajas de bits, debido a que algunos bloques adyacentes pueden no coincidir del todo. Asimismo, la mayoría de los métodos de estimación de movimiento involucrados, no son sensibles a cambios de luminosidad de la escena [Furht, Smoliar & Zhang, 1990] y presentan dificultades al encontrarse con los problemas de oclusión y de apertura, que se explican a continuación.

El problema de oclusión [Watson & Ahamuda, 1985] surge cuando un objeto perteneciente a la escena, al llevar a cabo un movimiento tridimensional, cubre a otro(s) en el campo de percepción visual del sensor, que son relevantes para el análisis de movimiento bidimensional de la misma.

El problema de apertura [Hildret, 1983] surge directamente del hecho de que el movimiento aparente que puede medirse para cualquier porción local de una estructura fuertemente orientada (un borde, por ejemplo), es solo la componente de movimiento ortogonal a la dirección local.

[4] VLSI son los circuitos electrónicos digitales de muy alta integración: Very Large Scale of Integration.

La búsqueda de posibles movimientos es usualmente limitada a una región de dimensiones $N_1 + 2M_1$ x $N_2 + 2M_2$ llamada "ventana de búsqueda" por razones computacionales. Los distintos algoritmos basados en "block-matching" difieren en:

- El criterio de error, o de coincidencia (ej., mínima diferencia absoluta, máxima correlación de cruz, mínimo error cuadrático, etc.)
- La estrategia de búsqueda (tres pasos, búsqueda en cruz, etc.)
- La determinación del tamaño del bloque (jerárquico, adaptivo, etc.)

Criterio de error o de coincidencia:

Los principales criterios de error o de coincidencia incluyen el de máxima correlación cruzada, el mínimo error medio cuadrático (MSE), la mínima diferencia absoluta (MAD), y el conteo máximo de pixeles coincidentes (MPC).

En el criterio MSE, evaluamos una función que podría tener la siguiente forma:

$$MSE(d_1,d_2) = \frac{1}{N_1 N_2} \sum_{(n1,n2)\in B} [s(n_1,n_2,k) - s(n_1+d_1,n_2+d_2,k+1)]^2$$

donde B denota un bloque N_1 x N_2, para un conjunto de vectores de movimiento candidatos (d_1, d_2).

De forma similar, el criterio MAD, que habla de la mínima diferencia absoluta, se define como:

$$MAD(d_1,d_2) = \frac{1}{N_1 N_2} \sum_{(n1,n2)\in B} |s(n_1,n_2,k) - s(n_1+d_1,n_2+d_2,k+1)|$$

Dicho de paso, éste es el criterio favorito de las implementaciones VLSI. Sin embargo, es bien conocido que el rendimiento del criterio MAD se deteriora conforme el área de búsqueda crece, ya que aparecen varios mínimos locales [Furht, Smoliar & Zhang, 1990].

Una alternativa mencionada es la del conteo de máxima concordancia de pixeles (MPC). En este acercamiento, cada pixel dentro del bloque B es clasificado como un pixel concordante o discordante de acuerdo a:

$$T(n_1, n_2; d_1, d_2) = \begin{cases} 1 & si \quad |s(n_1,n_2,k) - s(n_1+d_1,n_2+d_2,k+1)| \le t \\ 0 & si \quad cualquier\ otro\ caso \end{cases}$$

donde t es un umbral predeterminado. Entonces, el número de pixeles concordantes dentro del bloque está dado por:

$$MPC(d_1, d_2) = \sum_{(n1,n2)\in B} T(n_1, n_2; d_1, d_2)$$

Ya que lo que deseamos obtener es el valor de (d_1,d_2) que da el número más alto de pixeles concordantes [Ghanbari 1990].

Procedimientos de Búsqueda:

Para encontrar el bloque que represente la mejor opción de desplazamiento se necesita evaluar el criterio de coincidencia en todos los vectores de movimiento de los pixeles candidatos (n1, n2). Esto podría ser llevado a cabo por el método exhaustivo, o de búsqueda completa, pero debido a que este algoritmo evalúa todos los valores de (d1, d2) en cada pixel, da como resultado un consumo de tiempo altísimo.

Para reducir la carga computacional, reducimos el área de búsqueda a

$$-M_1 \le d_1 \le M_1$$

y

$$-M_2 \le d_2 \le M_2$$

a la que se le suele llamar "ventana de búsqueda", centrada en cada pixel para el que se vaya a estimar un vector de movimiento.. M_1 y M_2 son números enteros predeterminados.

Los procedimientos de búsqueda que presentaremos en esta sección del capítulo [Furht, Smoliar & Zhang, 1990] se refieren al enfoque de *block-matching* solamente y han sido programados en MATLAB para integrarlos al sistema. La correlación de fase y los métodos jerárquicos híbridos se explican con más detalle en los subcapítulos consecuentes.

- *Búsqueda exhaustiva reducida*:

La búsqueda exhaustiva, como ya se dijo, es un método de búsqueda muy simple pero computacionalmente intensivo. Evalúa la función de costo en cada localidad de la ventana de búsqueda, y es por esto por lo que se llama reducida. No evalúa todas las localidades en

la imagen, como tendría que ocurrir en una búsqueda exhaustiva completa. Sin duda, es el método que arroja los mejores resultados.

- *Búsqueda por los "Tres Pasos":*

Este algoritmo de búsqueda es uno de los algoritmos rápidos más simples que existen. Consiste en calcular la función de costo (criterio de error o coincidencia) en el centro de la ventana de búsqueda y en ocho localidades circundantes y equidistantes al mismo. Generalmente se utilizan en las direcciones cardinales principales (N, S, E, O, NE, NW, SE, SW). La localidad que genera el resultado de coincidencia mayor (o de menor costo[5]) se convierte en el centro de la búsqueda siguiente –2° paso-, y se reduce el área de búsqueda a la mitad. Lo mismo pasa con el tercer paso. Ilustraremos el presente algoritmo con el siguiente esquema:

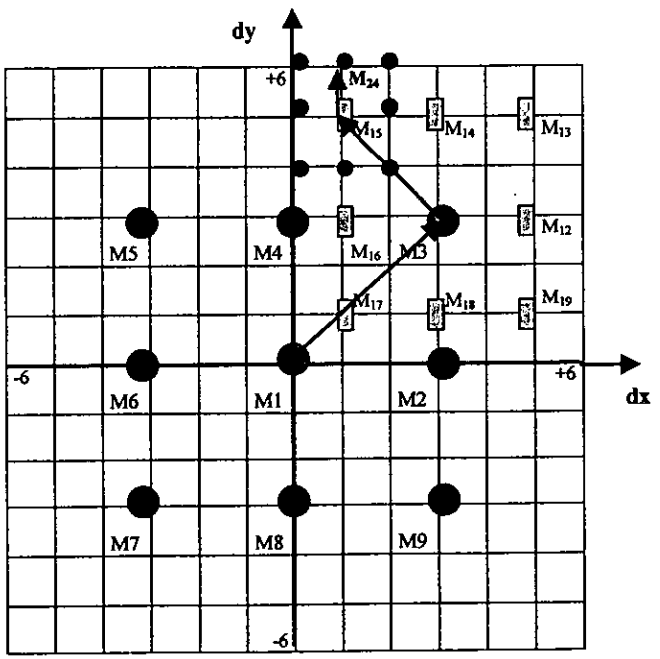


Figura 3.2 Algoritmo de Búsqueda por los "Tres Pasos"

Paso 1: Se calculan los nueve valores de la función costo MAD, que incluyen la localidad central (M_1, M_2, M_3, M_4, M_5, M_6, M_7, M_8, M_9). Si suponemos que el mejor valor apareció en M_3, éste se convierte en el centro de búsqueda del siguiente paso.

Paso 2: Se vuelven a calcular nueve funciones de costo, esta ocasión para M_3 y sus ocho localidades circundantes, pero se reduce la distancia radial a la mitad. Los puntos correspondientes a este paso son M_{11}, M_{12}, M_{13}, M_{14}, M_{15}, M_{16}, y M_{17}, M_{18}, y M_{19},

Paso 3: En el último paso, volvemos a seleccionar la localidad donde se encontró el mejor valor de la función de costo como centro, reducimos el radio de búsqueda a la mitad otra vez y el punto en el que exista la mejor opción de costo, será el vector de movimiento, que parte del centro de la búsqueda en el paso 1. En este ejemplo, el vector de movimiento {dx, dy} resultó ser {1, 6}.

El número máximo de cálculos de la función costo es 9 x 3 = 27, mucho menor que en una búsqueda exhaustiva, donde tendríamos 169 llamadas.

- *Algoritmo de búsqueda por "Dirección Conjugada"*:

Este algoritmo para estimar vectores de movimiento, es una adaptación del método iterativo de dirección conjugada. En este enfoque se ve como un algoritmo de búsqueda en un solo ciclo. La figura que aparece más adelante puede ser de utilidad para comprender mejor el proceso de búsqueda.

La característica principal del método es que establece la dirección de búsqueda paralelamente a alguno de los ejes coordenados, cada valor es ajustado mientras el otro es revisado, para llegar finalmente a la obtención de un vector final de búsqueda, donde se tendrá el ajuste fino.

El algoritmo consiste de los siguientes pasos. (el ejemplo está basado en el esquema siguiente):

[5] Generalmente se utiliza el criterio MSD.

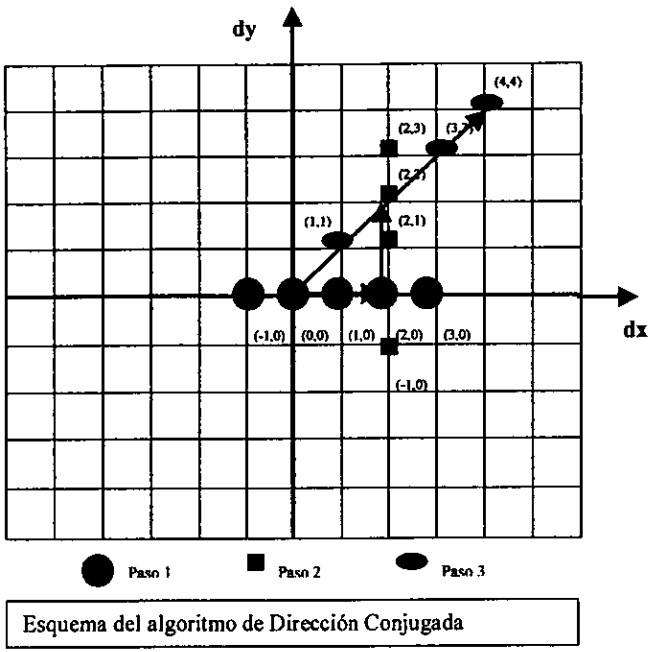


Figura 3.3 Algoritmo de "Dirección Conjugada"

Paso 1: Los valores de la función costo (MAD, por ejemplo) se calculan en la dirección *dx*, hasta que se halla el mínimo. El cálculo sigue el orden siguiente: (a) M(0,0), M(1,0), y M(-1,0); (b) si el mínimo es M(1,0), se calcula y evalúa MAD para M(2,0), y sigue el proceso. El paso termina cuando es hallado un mínimo en la dirección *dx*. Para el caso de la figura, el mínimo es M(2,0).

Paso 2: La búsqueda continúa en la dirección *dy* a partir del punto crítico en *dx*, por lo que en este caso, calcularíamos las funciones de costo en M(2,-1) y M(2,1). Siguiendo el proceso indicado en la figura, encontramos un mínimo en la dirección *dy* en M(2,2).

Paso 3: La dirección de la búsqueda es ahora el vector que conecta el punto de inicio (0,0) con el mínimo obtenido por los pasos 1 y 2, es decir, (2,2). Las siguientes funciones de

costo se calculan y evalúan después: M(1,1) y M(3,3), y de la misma forma que los pasos anteriores, hasta hallar un mínimo en esta dirección. En el ejemplo de la figura el mínimo es M(4,4), por lo que el vector final de movimiento es:

$$\{dx = 4, dy = 4\}$$

Podría darse el caso en el que los vectores *dx* y *dy* no constituyeran un cuadrado como en el ejemplo anterior, es decir, que no fueran de igual magnitud. En dado caso, los pixeles más cercanos al vector obtenido entre el inicio (0,0) y dicho punto serán los responsables de determinar el mínimo del tercer paso.

- *Algoritmo de búsqueda logarítmica en 2-D:*

El presente algoritmo hace uso, preferentemente, de la función de costo MAD y lleva a cabo una búsqueda logarítmica en dos dimensiones a lo largo de una dirección virtual de distorsión mínima (DMD) con los datos de la ventana de búsqueda. Existe un parámetro, llamado umbral, que será de mucha utilidad para determinar el grado de coincidencia de bloques, como se verá con detalle el los siguientes párrafos. El procedimiento puede ser descrito con la ayuda de los pasos explicados a continuación:

Paso 1: Se calcula la función MAD para la localidad *dx*=0, *dy*=0, osea, el punto M(0,0) y se compara este resultado con el umbral T preestablecido (un valor común es 4 con 255 tonos de gris). Si M(0,0) < T no hubo un movimiento detectable y la búsqueda termina aquí. Si es M(0,0) es mayor, continúa el paso 2a.

Paso 2a: Se calculan las siguientes cuatro funciones de costo, $M_1(4,0)$, $M_2(0,4)$, $M_3(-4,0)$ y $M_4(0,-4)$, y se determina el valor mínimo para compararlo con M(0,0), asignando así a M':

$$M' = min(M_1, M_2, M_3, M_4) < M(0,0)$$

Si el valor M' es menor que M(0,0), hay que ir al paso 3, de otra forma, el valor es comparado con el umbral T. Si M' < T, éste dato M' es el mínimo y la búsqueda termina. Si es mayor, el algoritmo continua en el paso 2b.

Paso 2b: Suponiendo que en el paso anterior, el 2a, el mínimo M' fue encontrado en $M_1(4,0)$, entonces se trabaja con las dos posiciones vecinas: $M_5(4,4)$ y $M_6(4,-4)$, como se ve

en la figura adjunta. Se llevan a cabo otra vez las revisiones de valor mínimo y de umbral T, y si se llega al mínimo, el procedimiento finaliza. En caso contrario, continúa en el paso 3.

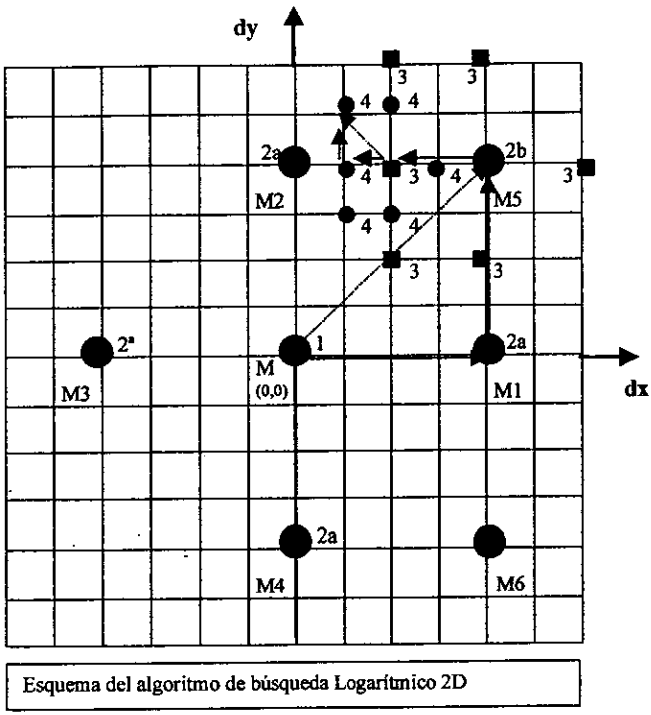


Figura 3.4 Algoritmo "Logarítmico 2D"

Paso 3: Si suponemos que la nueva localidad mínima es $M_5(4,4)$, continúa una búsqueda similar (como la de los pasos 2a y 2b), con la diferencia de que el radio de búsqueda es dividido entre dos. En nuestra figura, el nuevo mínimo es ahora M(2,4).

Paso 4: El radio de búsqueda es, de nuevo, reducido a la mitad y se lleva a cabo una búsqueda final (pasos 2a y 2b). Como resultado, el mínimo (dx, dy) aparece. En el caso del esquema explicativo, el vector es (1, 5).

- *Búsqueda en Cruz*:

El método de búsqueda en cruz es otra estrategia de tipo logarítmico, en la que en cada paso hay cuatro localidades de búsqueda, que son los puntos finales de una figura de equis (X) o de cruz (+) [Ghanbari, 1990]. La figura que se presentará a continuación ilustra el caso de la figura en cruz.

La distancia entre los puntos de búsqueda se reduce si se llega a la mejor coincidencia en el centro de la figura o en la frontera de la ventana de búsqueda. Hay muchas variantes del algoritmo de búsqueda en la literatura [Gharavi & Mills, 1990]. Recientemente, han sido propuestos algoritmos más eficientes de este tipo que utilizan las correlaciones espaciales y temporales entre los vectores de movimiento de los bloques. Veamos la figura siguiente, en la que observamos 4 pasos:

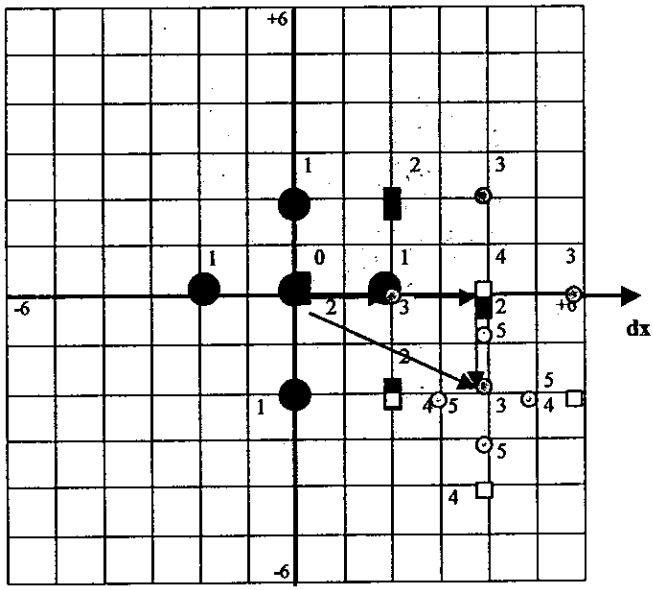


Figura 3.5. Algoritmo de búsqueda en "Cruz"

1) Se evalúa la cruz en el centro y se halla la mejor concordancia en el brazo derecho de la misma, por lo que éste adopta el centro de la nueva cruz.
2) Al evaluar la nueva cruz, la mejor coincidencia en la función de costo ocurre en el brazo inferior, que será ahora en centro del siguiente paso.
3) De nueva forma evaluamos las posiciones (N, S, E, O) y obtenemos la mejor opción en E.
4) En el presente paso, la mejor ocurrencia fue el valor del centro, por lo que reducimos el área de la ventana de búsqueda y volvemos a evaluar los cinco puntos. El resultado final se halla en la parte inferior de la última búsqueda. Al ya no poder reducir más la ventana de búsqueda, el proceso termina.

III.3. Correlación de Fase (phase-correlation).

El que a continuación se explicará brevemente es un método alternativo de estimación de movimiento. Se encuentra clasificado en otra área debido a que no trabaja con ventanas de búsqueda, como los algoritmos presentados en la sección anterior; sino que recibe 2 entradas: dos bloques de la misma posición correspondientes a un cuadro presente y uno anterior. Las técnicas basadas en correlación suponen típicamente la conservación de la distribución local de las intensidades

El método de correlación de fase, como su nombre lo indica, trabaja en el dominio de la frecuencia, no en el dominio espacial, y para ello se vale de la Transformada de Fourier. A *grosso modo*, diremos que dos patrones de brillantez en imágenes, siendo uno una versión trasladada del otro, tienen representaciones en el dominio de Fourier que difieren sólo en su fase [Mitchie & Bouthemy, 1996].

Si llevamos a cabo la transformada de Fourier del modelo básico de movimiento discreto, dado por:

$$s(n_1, n_2, k) = s(n_1 + d_1, n_2 + d_2, k + l)$$

donde d_1 y d_2 son los componentes del desplazamiento, k es el bloque actual y (n_1, n_2) es el pixel actual, quedaría:

$$S_k(f_1,f_2) = S_{k+1}(f_1,f_2)\exp\{j2\pi(d_1f_1 + d_2f_2)\}$$

denotando $S_k(f_1f_2)$ la transformada de Fourier en 2 dimensiones del bloque k con respecto a las variables espaciales n_1 y n_2. Entonces, la diferencia de las 2 fases de Fourier en dos cuadros respectivos quedaría así:

$$\arg\{S(f_1,f_2,k)\} - \arg\{S(f_1,f_2,k+1)\} = 2\pi(d_1f_1 + d_2f_2)$$

esto, para el caso de movimiento traslacional, define un plano en las variables (f1, f2). Entonces, el vector de movimiento entre bloques se puede estimar a partir de la orientación del plano [Murat, 1990].

El método de correlación de fases estima el desplazamiento relativo entre dos bloques de imágenes gracias a una función normalizada de correlación cruzada calculada en el dominio espacial de dos dimensiones de Fourier. Está también basado en el principio de que un movimiento relativo en el dominio espacial da como resultado un término lineal de fase en el dominio de Fourier. A continuación explicaremos los aspectos relativos a la implementación de este modelo y a la obtención de la función de correlación de fase.

La función de correlación cruzada entre los bloques k y $k+1$ se define como

$$c_{k,k+1}(n_1,n_2) = s(n_1,n_2,k+1) * s(-n_1,-n_2,k)$$

donde * denota la operación de convolución en dos dimensiones. Si aplicamos la transformada de Fourier en ambos lados, obtendremos la expresión del espectro cruzado de potencia (cross power-spectrum):

$$C_{k,k+1}(f_1,f_2) = S_{k+1}(f_1,f_2)S_k^*(f_1,f_2)$$

Normalizando $C_{k,k+1}(f_1,f_2)$ por su magnitud obtenemos la fase del espectro de energía cruzada:

$$\tilde{C}_{k,k+1}(f_1,f_2) = \frac{S_{k+1}(f_1,f_2)S_k^*(f_1,f_2)}{|S_{k+1}(f_1,f_2)S_k^*(f_1,f_2)|}$$

Tomando su fase y antitransformando en dos dimensiones por Fourier obtendremos la función de correlación de fase:

$$c_{k,k+1}(n_1, n_2) = \delta(n_1 - d_1, n_2 - d_2)$$

se observa que la función de correlación de fase consiste en un impulso cuya localización da el vector de desplazamiento.

Detalles de implementación: Para implementar la función de correlación de fase en un algoritmo de computadora, hay que reemplazar a la transformada de Fourier por la transformada *discreta* de Fourier (DFT), por lo que hay que seguir los siguientes pasos para generar la función:

1. Calcular la DFT de dos dimensiones de los bloques respectivos de los cuadros k y $k+1$.
2. Calcular la fase de la función del espectro de energía cruzada.
3. Obtener la antitransformada de dos dimensiones de $\tilde{C}_{k,k+1}(f_1, f_2)$ para encontrar la función de correlación de fase, $c_{k,k+1}(n_1, n_2)$.
4. Detectar la localización de los picos en la función de correlación de fase.

Idealmente, esperamos observar un impulso sencillo en la función de correlación de fase indicando el desplazamiento relativo entre los dos bloques. En la práctica, varios factores contribuyen a la degeneración de la función de correlación de fase y que tenga varios picos; y son el uso de la transformada DFT de dos dimensiones en vez de la transformada de Fourier, la presencia de más de un objeto moviéndose dentro del bloque y la existencia de mucho ruido de observación [Murat, 1990].

A continuación se muestra la representación tridimensional de una función típica de correlación, en la que podemos ver la correlación más alta cerca del origen. En este caso particular, el movimiento presentado fue [x=4, y=0].

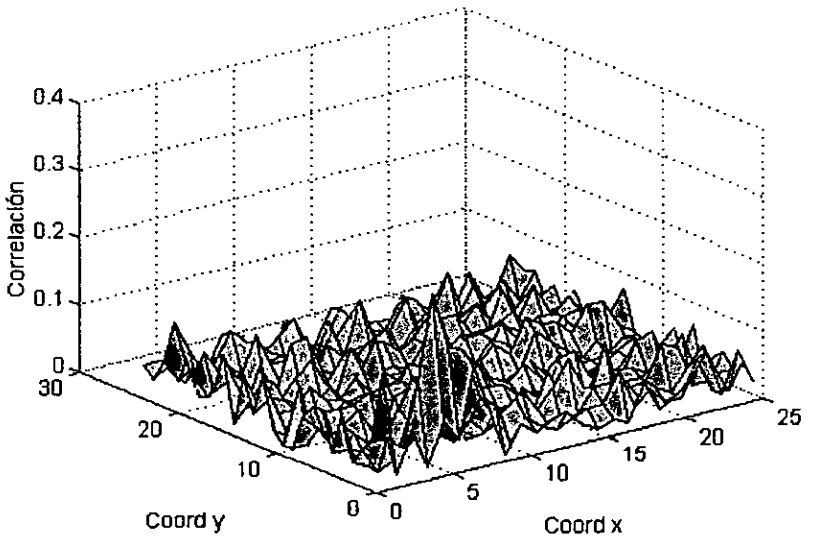


Figura 3.6 Representación 3D de una función típica de correlación

La determinación del tamaño del bloque es uno de los más importantes parámetros en cualquier algoritmo de estimación de movimiento y éste no es la excepción. La selección del tamaño del bloque generalmente involucra una negociación entre dos requerimientos en conflicto: la ventana debe ser lo suficientemente grande como para estimar vectores de movimiento grandes, y por el otro lado, debe ser lo suficientemente pequeña como para que el vector de desplazamiento permanezca constante dentro de la ventana.

III. 4. Estimación de Movimiento Jerárquica.

Las representaciones de imágenes[6] en varias resoluciones pueden ser usadas conjuntamente con los métodos de "block-matching" o de correlación de fases para obtener una estimación de movimiento más precisa. A esta representación se le llama representación "pirámide", ya que adopta esa forma la estructura de cuadros que conforme

[6] Entendidas como cuadros de una secuencia.

se acerca a la parte más baja su resolución aumenta (imagen original), y por el contrario, mientras más se acerca a la cúspide tiene una menor resolución, producto de filtrados de tipo paso-bajas y sub-muestreos.

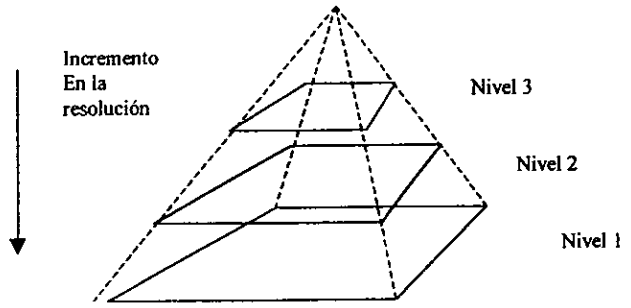


Figura 3.7 Representación piramidal de niveles de resolución

El enfoque que este tipo de algoritmos presenta es muy interesante y genera resultados efectivos. La obtención de imágenes en distintos niveles de resolución y de distintos tamaños espaciales abre un abanico de posibilidades en cuanto a análisis toca.

- *Block-matching jerárquico*:

La idea básica de "block-matching" jerárquico es la de llevar a cabo la estimación de movimiento en cada nivel de resolución, comenzando con el más bajo [Bierling, 1988]. Estos niveles de baja resolución sirven para dar un vector "burdo" de estimación de movimiento usando bloques relativamente grandes. Hay que hacer notar que el tamaño relativo del bloque puede ser medido como el tamaño del bloque normalizado por el tamaño de la imagen en un determinado nivel de resolución.

La estimación lograda por el nivel de resolución más bajo se pasa al siguiente nivel (un poco más alto) de resolución como una predicción inicial. El nivel de resolución más alto sirve para hacer el ajuste fino del vector de desplazamiento. En los niveles de

resolución más altos se pueden ir utilizando ventanas relativas más pequeñas, ya que estamos comenzado con una estimación relativamente buena.

En la práctica se puede omitir el sub-muestreo, ya que la pirámide contiene imágenes que son del mismo tamaño pero cada vez más difuminadas conforme vamos hacia arriba.

Un conjunto típico de datos para el método jerárquico en tres niveles es el siguiente:

NIVEL	3	2	1
Tamaño del filtro	10	5	3
Desplazamiento máximo	+31, -31	+7, -7	+1, -1
Tamaño del bloque	64	16	4

Tabla 3.1 Una propuesta de configuración para el método de búsqueda jerárquico

En nuestro caso hemos implementado el método jerárquico en conjunción con el proceso de búsqueda de los "Tres Pasos" y "Cruz", utilizando los valores de la tabla anterior. El algoritmo consiste en filtrar paso-bajas la imagen original en tres niveles: baja, media y alta resolución. El proceso de los "Tres Pasos" o de la "Cruz" es llamado en cada nivel de resolución comenzando por el más bajo con una ventana grande (64x64). El vector estimado se manda al siguiente nivel de resolución (medio) para comenzar allí, pero la ventana es más chica, de 16x16. Finalmente, se llama al nivel más alto de resolución para el ajuste fino, teniendo allí una ventana de 4x4 y un movimiento restringido sólo a (–1, +1). El vector final es el que el algoritmo devuelve.

- Estimación jerárquica con Correlación de Fases:

Otro algoritmo que da buenos resultados al ser implementado en software es la combinación de un método que trabaja en el dominio de la frecuencia, como el ya

explicado método de Correlación de Fases y múltiples resoluciones espaciales [Erkan, Sezan & Erdem, 1993]. Como ya se explicó, el método de correlación analiza en el dominio de Fourier dos bloques de pixeles de las mismas dimensiones. MPEG requiere que dichos bloques (o macrobloques) sean de tamaño 16x16 para cumplir con el estándar. Es decir, por cada macrobloque tendremos que enviar un vector de movimiento. Esto en apariencia entorpece el funcionamiento de la estimación jerárquica, sin embargo puede hacerse lo siguiente:

1) Se comienza con una ventana de búsqueda grande. Como los dos bloques deben ser del mismo tamaño para que se pueda llevar a cabo la convolución, tendremos una ventana de búsqueda inicial de, digamos, 32x32 y un macrobloque de las mismas dimensiones y localizado en las mismas coordenadas que la ventana, sólo que perteneciente al otro *frame*. Llevamos a cabo la estimación a ese nivel difuminando ambos bloques con un filtro digital paso bajas grande. Se recomienda el uso de flitros de, digamos, 5x5. De esta forma la máxima correlación se encontrará en un área grande tomando en cuenta solamente los rasgos más significativos de la imagen, los que no fueron eliminados por el filtrado.
2) Centramos nuestra nueva búsqueda en la localidad a la que apuntó el vector de movimiento en el paso anterior, y esta vez recortamos los dos bloques (de los *frames* originales) para que sean de tamaño menor, digamos, 24x24. En esta ocasión el filtro paso bajas tendrá que ser menos severo con las imágenes, efecto que lograremos reduciendo las dimensiones del mismo. Una sugerencia puede ser llevarlo a 3x3. Como en el paso 1, se hace la correlación de fases y se obtiene el vector resultante.
3) Para el tipo de implementación aquí presentada, éste sería el último paso[7]. Nuestros bloques finalmente adoptarán el tamaño deseado, 16x16, pero esta vez los "recortaremos" en la posición a la que nos han llevado los dos pasos anteriores. En el último paso se puede utilizar un filtrado muy suave o bien no valerse de ninguno, es decir, usar la porción de la imagen original. Como es

[7] Ya que el número de niveles de resolución es libre y depende de el grado de exactitud que se requiera y de la cantidad de recursos disponibles para llevar a cabo más operaciones computacionales por ciclo.

natural, el movimiento en este paso es más fino y solamente afectará en algunos pixeles al resultante. La suma de los tres vectores será el vector final de desplazamiento.

III.4. La Transformada Polinomial.

En esta sección del capítulo dedicado a la estimación de movimiento introduciremos un concepto reciente relacionado con el tema. Se trata de la utilización de una herramienta matemática llamada "transformada polinomial" en la estimación de movimiento entre secuencias digitales de imágenes. Nos pareció de suma importancia citar este método, aunque no haya sido implementado para el codificador en su primera versión, ya que representa un trabajo de tesis [Silván, 1998], anterior al nuestro que propone precisamente la estimación de movimiento utilizando la transformada polinomial y que fue desarrollado en la DEPFI..

La idea principal es la siguiente: se busca un método para estimar movimiento en secuencias de video a partir de un modelo de representación en tres dimensiones basándose en transformadas polinomiales. Este acercamiento analiza localmente la secuencia de video por medio de ventanas espacio-temporales. El procedimiento permite recuperar el flujo verdadero en regiones libres del problema de apertura[8] y estima la componente normal de desplazamiento para patrones orientados que sean detectados dentro de cada ventana. Hay que mencionar que la variación del tamaño de las ventanas en el espacio y en el tiempo es útil para un análisis eficiente de todos los tipos de movimiento.

El método entra dentro de los llamados "métodos basados en energía"[9] y está basado en un modelo de percepción visual. Incorpora tanto multiresolución como derivadas de funciones *gaussianas*, que son el modelo más ampliamente aceptado de visión primitiva, y formaliza los conceptos en una herramienta matemática conocida como transformada polinomial.

En el dominio espacial, la transformada polinomial analiza la imagen por regiones con la ayuda de ventanas gaussianas traslapadas y dentro de cada ventana lleva a cabo una

[8] Problema que limita a todos los métodos expuestos anteriormente en este capítulo

expansión polinomial con funciones de Hermite. Los operadores de este análisis son equivalentes a derivadas enésimas de gaussianas. La multiresolución se logra introduciendo esta expansión en un esquema piramidal. Para el caso de análisis de video, se ha extendido el concepto al dominio temporal, pero con la diferencia de que en este dominio los polinomios de Hermite no son perceptualmente relevantes. Si se tienen movimientos lentos, éstos serán más fáciles de identificar utilizando tamaños de ventana temporal más grandes y ventanas más pequeñas en el dominio espacial; por el contrario, ventanas temporales pequeñas combinadas con ventanas espaciales de mayor tamaño darán como resultado una identificación más adecuada de movimientos rápidos.

El siguiente esquema explica el funcionamiento de la transformada polinomial y de su correspondiente antitransformada:

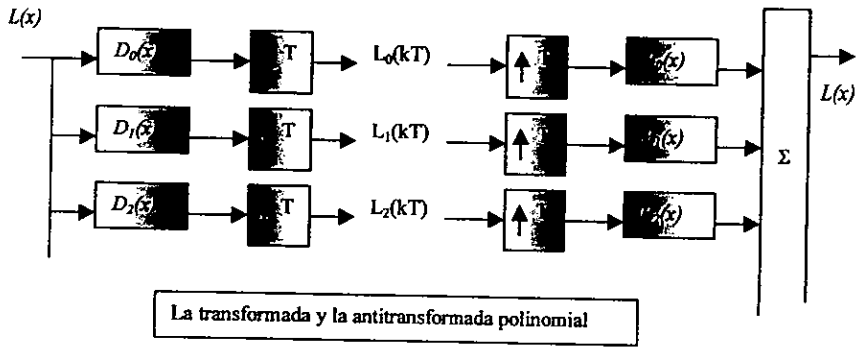


Figura 3.8 La transformada y antitransformada polinomial

Los bloques que se hallan a la izquierda corresponden a la transformada. Una señal *L(x)* entra al sistema y es convolucionada con los distintos filtros $D_0(x)$, $D_1(x)$, ... $D_n(x)$, dados por:

$$D_n(x) = G_n(-x) \cdot V^2(-x)$$

[9] Los métodos que utilizan filtros espacio-temporales orientados para muestrear la función de espectro de potencia y detectan el plano de impulso se conocen como métodos basados en energía.

esto indica que dichos filtros son a su vez producto de la multiplicación de un miembro obtenido a partir de los polinomios base[10] (vectores base), que son $G_n(x)$ y un término obtenido de la ventana local predefinida, que es $V(x)$.

Posteriormente, se lleva a cabo un muestreo dado por múltiplos de T (periodo de muestreo de la señal) para obtener el mapeo final de la señal original $L(x)$ a los coeficientes polinomiales $L_n(kT)$. Aquí hemos transformado ya la señal y nuestro resultado es el mencionado grupo de coeficientes polinomiales. Esto significa que la señal original dentro de la ventana puede ser expresada en términos de los coeficientes de la expansión, como se muestra a continuación:

$$\int_{-\infty}^{+\infty} L^2(x) \cdot V^2(x - kT)dx = \sum_{n=0}^{\infty} L_n^2(kT)$$

La antitransformada polinomial, o transformada inversa, consiste en regenerar la señal original a partir de los coeficientes obtenidos. El proceso es exactamente el contrario: hay que interpolar los coeficientes { $L_n(kt)$; donde k es entero } con la función de patrón $P_n(x)$ y hacer la sumatoria sobre todos los niveles u órdenes n.

La función de patrón Pn(x) se obtiene de la siguiente forma:

$$P_n(x) = \frac{G_n(x)V(x)}{W(x)}$$

donde $Gn(x)$, como ya se dijo, son las funciones polinomiales básicas de orden n, $V(x)$ es la ventana local y $W(x)$ es la función de ponderación, obtenida de calcular la siguiente sumatoria:

$$W(x) = \sum_{k} V(x - kT)$$

La introducción de esta herramienta como agente de detección de movimiento es definitivamente novedosa. De la teoría explicada anteriormente, los autores derivan un comportamiento local de la intensidad de las imágenes basándose en la representación que

[10] Recordemos que, en un espacio vectorial, los elementos pertenecientes a la base, y de los cuales se derivan el resto de los vectores por combinaciones lineales, deben formar un conjunto generador y ser linealmente independientes. En este caso, las polinomiales básicas son ortonormales entre sí.

la transformada polinomial brinda para obtenter una recuperación de flujo en una y dos dimensiones. Para el caso bidimensional, se trabaja con patrones libres del problema de apertura, como podrían ser regiones texturizadas y esquinas. El caso unidimensional analiza una extensión del acercamiento mencionado para recuperar flujo normal a partir de patrones orientados. Remitimos de nuevo al lector a revisar el desarrollo correspondiente en el trabajo de tesis citado. De cualquier forma, en el presente trabajo incorporamos el método de estimación de movimiento a nivel teórico para compararlo ligeramente con los métodos tradicionales y ampliamente comentados en la literatura.

IV. El Estándar MPEG

IV.1 Historia de MPEG, propósito y alcances.

Para estudiar la creciente necesidad de representar eficientemente señales digitales de imagen y video, se formó, desde hace varios años, el Grupo de Expertos en Imágenes en Movimiento, "Moving Pictures Experts Group" (MPEG), dedicado principalmente a diseñar, desarrollar y revisar estándares de codificación. Los estándares de codificación de video MPEG-1 y MPEG-2 han atraído la atención mundial recientemente, junto con el crecimiento de la tecnología electrónica VLSI (muy alta escala de integración) y la implementación en software de los mismos, que ya se encuentra disponible comercialmente.

La reducción de los anchos de banda de las señales de televisión digitalizada ha sido un tópico de investigación y desarrollo por más de 30 años. Los propósitos por los que la investigación se ha llevado a cabo son varios, pero todos comparten la misma idea básica. Si se tiene un medio digital de distribución medio y se puede llevar la cifra de más de 200 Mbits/s de tasa de transmisión que la televisión PCM requiere a un valor que podría ser manejado económicamente con la tecnología de procesamiento de señales, se habría cumplido la meta de crear o inclusive mejorar substancialmente los productos y servicios. La tasa de transmisión de alrededor de 1.5 Mbit/s fue el valor sobre el que tres industrias manifestaron un interés común al final de los años 80: la de electrónica de consumo con el disco compacto, la de difusión con el radio digital y la de telecomunicaciones con el ISDN y el ASDL de banda angosta. El mérito de MPEG fue su habilidad para identificar ésta oportunidad y lograr que las tres empresas trabajaran conjuntamente. El grado de colaboración internacional sin precedentes que se dio involucrando a cientos de investigadores produjo como resultado un estándar MPEG-1 de muy alta calidad técnica. Todavía hoy, después de muchos años e incontables implementaciones no se ha hallado un solo error en la especificación del estándar. Los dispositivos MPEG-1 se cuentan por millones en todos los continentes. El inicio de los años 90 trajo consigo la oportunidad de que apareciera interés general en un nuevo estándar: el MPEG-2, para aplicaciones en casi

todas las industrias; distribución satelital, redes terrestres y de CATV, ISDN de banda amplia, discos y cintas digitales, etc.

Comercialmente, la estandarización internacional de protocolos y sistemas de comunicación con video ha ayudado a cumplir dos propósitos: interoperabilidad y economía de escala. La compatibilidad entre equipo de comunicación con video de distintos vendedores es una característica que el usuario desea y que a los manufactureros de equipo les gusta. Esto hace más atractivo el comprar y usar equipo de comunicaciones con video porque permite el intercambio internacional de video a gran escala a través de medios de almacenamiento o a través de redes de comunicación. Desde el principio de los años 80, comenzaron un gran número de actividades internacionales de estandarización de audio y video dentro del Comité Consultivo de Telefonía Internacional (CCITT), seguidos del Comité Consultivo Internacional de la Radio (CCIR), y la Organización Internacional de Estandarización/Comisión Internacional de Electrotécnica (ISO/IEC). MPEG se estableció en 1988 dentro del trabajo del Comité Técnico Conjunto de Tecnología de la Información de el ISO/IEC (JTC 1) con el mandato de desarrollar estándares para la representación codificada de imágenes en movimiento, audio asociado y su combinación en el uso en almacenamiento y recuperación en medios digitales de almacenaje con una tasa de transmisión de 1.5 Mbits/s. El estándar fue apodado MPEG-1 y fue publicado en 1992. La inquietud del grupo fue de extender más tarde las condiciones necesarias para desarrollar los algoritmos apropiados para la compresión de audio y video en MPEG-2 enfocando esta vez el estándar a una serie de aplicaciones audiovisuales que requerían una tasa de transmisión substancialmente más alta y que no habían sido mencionadas o cubiertas satisfactoriamente en el estándar MPEG-1. Específicamente, a MPEG-2 le fue entregada la tarea de proveer calidad de video no menor a la de NTSC/PAL[1] y hasta calidad de CCIR 601 con tasas de transmisión entre 2 y 10 Mbits/s. Las aplicaciones nacientes, tales como la distribución de televisión digital por cable, servicios de base de datos a través de red vía ATM (Modo de Transmisión Asíncrona), aplicaciones de video grabación digital en cinta (VTR), y distribución digital satelital y terrestre, fueron beneficiadas con la creciente calidad que se esperaba resultaría del recién surgido estándar MPEG-2. El estándar MPEG-2 fue liberado en 1994.

[1] NTSC son las siglas de "National Television System Committee" y PAL significa "Phase Alternating Line".

Las técnicas de compresión de video descritas en MPEG-1 y MPEG-2, desarrolladas y estandarizadas por el grupo MPEG han dado como fruto importantes y exitosos estándares de codificación de video en todo el mundo, con un creciente número de conjuntos de circuitos integrados VLSI y de productos comerciales MPEG-1 y MPEG-2. Un factor que ha sido clave de éxito es la estructura genérica de los estándares MPEG, cubriendo una amplia gama de aplicaciones y de parámetros específicos en las mismas. Para soportar el enorme rango de perfiles de las aplicaciones, pueden ser especificados por el usuario un gran número de parámetros de entrada que incluyen un tamaño flexible de la imagen y la razón de cuadros (frame rate). Otro factor importante es el hecho de que el grupo MPEG sólo estandarizó las estructuras de los decodificadores y los formatos de las cadenas de bits. Esto otorga un alto grado de libertad a los fabricantes para optimizar la eficiencia del código o la calidad del video a una tasa de transmisión dada desarrollando innovadores algoritmos de codificación incluso después de que se finalizó la descripción del estándar.

Anticipando la rápida convergencia de las industrias de telecomunicaciones, computación y TV/Cine, el grupo MPEG inició oficialmente una nueva fase de estandarización en 1994, conocida como MPEG-4, con el objetivo de estandarizar los algoritmos y herramientas para la codificación y la representación flexible de información audiovisual que cumpla con los retos que las futuras aplicaciones multimedia y sus ambientes demanden. En particular, MPEG-4 ataca la necesidad de un acceso universal y robusto a la información en ambientes propensos a error, alta funcionalidad interactiva, la codificación de información natural y sintética, así como alta eficiencia en compresión. Las tasas de transmisión buscadas para el estándar MPEG-4 se encuentran entre 5 y 64 kbits/s para aplicaciones en redes conmutadas de teléfono (PSTN), ya sean móviles o públicas y cerca de 4 Mbits/s para aplicaciones de televisión y cine. La publicación oficial del estándar se dio a conocer en octubre de 1998 y en unos meses más será un estándar internacional.

La estructura de MPEG-4 se fundamente en el éxito comprobado de tres áreas de desarrollo: la televisión digital, las aplicaciones interactivas de gráficos (contenido sintético) y el multimedia interactivo (distribución y acceso en WWW) y proveerá los elementos tecnológicos estandarizados que permitirán la integración de la producción, distribución y paradigmas de los tres campos. La segunda versión de MPEG-4 está en

camino y planea ser aprobada en diciembre de 1999. Contendrá modificaciones finas con respecto al campo de aplicación y tratará de completar algunos aspectos de su antecesor.

El lector se preguntará el porqué de la ausencia de un proyecto MPEG-3. Dicho proyecto fue planeado con miras a la televisión digital de alta definición (HDTV), pero los avances logrados en el estándar MPEG-2 demostraron que cubriría con eficiencia también dicho rubro. Es por esto que nunca se llegó a la realización de dicho proyecto.

La noticia más reciente en la historia de MPEG es que la organización estandarizadora comenzó en octubre de 1996 la generación de un nuevo proyecto llamado MPEG-7, que trabajará bajo el nombre de "Interfaz de Descripción de Contenido Multimedia", para solucionar problemas a solicitudes de contenido y contexto en las distintas aplicaciones multimedia. Dicho estudio extenderá las limitadas capacidades de los descriptores actuales que identifican el contenido de información multimedia existente hoy en día, e incluirá una gran cantidad de tipos adicionales de datos. En otras palabras, MPEG-7 especificará un conjunto estándar de descriptores que podrán ser usados para determinar los diversos datos de la información multimedia. MPEG estandarizará un lenguaje para especificar esquemas de descripción (DDL, Description Definition Language). Se espera que sea aprobado en julio del 2001.

IV.2. Descripción del Codificador.

El documento ISO/IEC 11172, que describe al estándar MPEG-1 contiene cinco partes. La parte 1 cubre la capa MPEG de sistema, la parte 2 describe el video MPEG y la parte 3 especifica el audio MPEG. La parte 4 se refiere a las pruebas referentes al mismo y la parte 5 es un modelo de referencia de software para el MPEG-1.

- Capa de Sistema:

La capa de sistema de MPEG tiene la tarea básica de combinar uno o más flujos de bits de audio y de video comprimidos en una sola cadena de bits. Define la sintaxis del flujo

de datos que se preocupa por la sincronización, el control del tiempo y la interacción de las cadenas de bits de audio y video.

Desde la perspectiva de los sistemas, un flujo de bits MPEG está hecho de una capa de sistema y de capas de compresión. La capa de sistema provee una envoltura para las capas de compresión. Dichas capas contienen la información con la que se alimentará a los decodificadores de audio y de video, mientras que la capa de sistema tiene los controles para demultiplexar las capas de compresión, que se encuentran entrelazadas. La siguiente figura muestra el sistema MPEG en un diagrama de bloques:

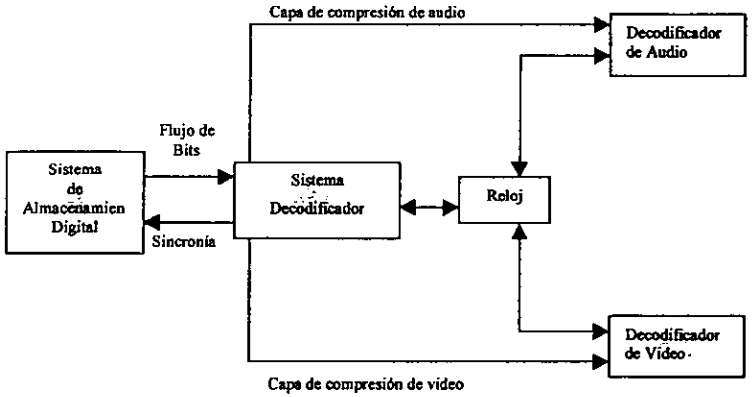


Figura 4.1 Esquema de bloques del codificador MPEG

La cadena de bits MPEG está compuesta de una secuencia de grandes paquetes que a su vez están subdivididos en más paquetes, como se muestra en la siguiente figura:

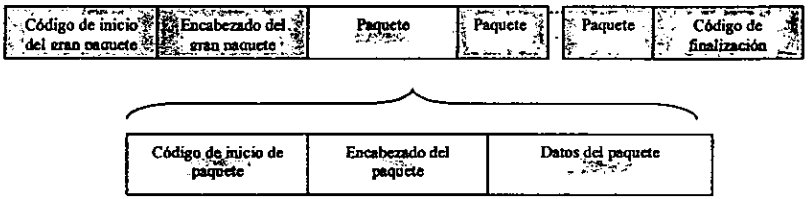


Figura 4.2 Distribución de paquetes en la cadena de bits MPEG

Cada paquete tiene una cadena única de inicio de 32 bits y un encabezado de paquete, seguidos por uno o más paquetes de información. Cada paquete consiste también del código de inicio de paquete (otra secuencia única de 32 bits) y su encabezado, seguidos de datos empaquetados (información comprimida de audio o video). El decodificador de sistema analiza este flujo de bits (los códigos de inicio del gran paquete y de los paquetes pueden ser detectados sin necesidad de decodificar la información) y alimenta a los decodificadores apropiados con información ya sea de audio o de video junto con el correcto manejo del tiempo y de la sincronía.

El sistema MPEG utiliza un decodificador idealizado llamado "Decodificador Objetivo de Sistema" (STD). Este decodificador idealizado interpreta los encabezados de los grandes paquetes y de los paquetes, entregando los flujos de bits elementales a los decodificadores apropiados, ya sea de audio o de video. Hay que hacer notar que podría estar presente más de un decodificador de cada tipo.

Una de las tareas principales de este decodificador idealizado es la prevención del *overflow* y del *underflow* en el búfer. Los requerimientos del búfer están descritos en términos del decodificador más que del codificador; por esta razón el *overflow* en el búfer ocurre cuando el decodificador no quita la información lo suficientemente rápido, por ejemplo, cuando la compresión es muy buena. De manera contraria, el *underflow* en el búfer ocurre cuando el decodificador borra la información muy rápidamente, como podría ser en el caso de un proceso de codificación que produce mucha información por imagen.

En el STD los bits para una unidad de acceso (una unidad de acceso de imagen o de audio) son removidos del búfer instantáneamente en el momento que así lo desea el DTS (decoding time stamp) del flujo de bits. También existe dentro del flujo de bits el llamado PTS (presentation time stamp); mientras que el *underflow* y el *overflow* del búfer están controlados por el DTS, la sincronización entre audio y video está controlada por el PTS. En algunos puntos de la secuencia de video, el DTS y el PTS son idénticos, y sólo se usa el PTS.

Tomando como base la información en la cabecera del flujo de bits, el sistema sabe cuándo conmutar la cadena de bits al búfer 1, o al búfer 2. Los bits que no estén destinados a estos búfers son mandados instantáneamente al búfer de control de sistema. Los bits son

eliminados en bloques llamados "unidades de acceso". Los DTS (o los PTS, si los DTS no se necesitan) en el flujo de bits determinan los tiempos de transferencia y presentación. En este modelo idealizado la transferencia, decodificación y presentación ocurren instantáneamente; en los sistemas reales algunos retrasos adicionales pueden llegar a ocurrir.

Un paquete MPEG tiene una longitud definida en su cabecera. Las longitudes de los paquetes están típicamente estructuradas para llenar los requerimientos del medio de almacenamiento o de transmisión digital, así que no necesariamente se deben a las unidades de acceso del audio y del video creadas por los codificadores. De cualquier forma, un paquete sólo puede contener un tipo de información comprimida.

- Audio MPEG:

La codificación de audio MPEG define tres capas de complejidad creciente y de calidad subjetiva[2]. Soporta tasas de muestreo de 32, 44.1 y 48 kHz. A una tasa de 16 bits/muestra el audio sin compresión requeriría más o menos 1.5 Mbits/s. Después de la compresión, las tasas de bits para canales monofónicos están entre 32 y 192 Kbits/s; las tasas de transmisión para canales estereofónicos están entre 128 y 384 Kbits/s.

Las técnicas de codificación de audio MPEG se valen en gran medida de las propiedades psico-acústicas del oído humano. Así como hay un umbral en el sentido de la vista para los patrones observables, hay también un umbral dependiente de la frecuencia en la percepción de estímulos audibles. Hay un efecto conocido como "enmascaramiento simultáneo", en el que la presencia de una señal de audio puede "enmascarar" la percepción de una señal más pequeña. Un efecto temporal de enmascaramiento ocurre también inmediatamente antes y después de la aparición de una señal de audio dominante. Todos los efectos mencionados anteriormente son de mucha utilidad en los modelos de codificación de audio que utiliza MPEG.

El sistema de audio MPEG primeramente segmenta al audio en ventanas de 384 muestras de anchura. Las capas I y II usan un banco de filtros para descomponer cada

[2] MPEG utiliza el término "capa" en dos sentidos: como un término descriptivo para la división del flujo de bits en un sistema, audio o video MPEG y para distinguir entre los tres algoritmos de audio.

ventana en 32 sub-bandas, cada una con una anchura de aproximadamente 750 Hz (para una tasa de muestro de 48 kHz). Como es típico para la codificación de sub-bandas, cada sub-banda es modificada de forma que la tasa de muestreo por sub-banda sea 1.5 kHz y haya 12 muestras por ventana. Se utiliza una transformada rápida de Fourier (FFT) de la entrada de audio para calcular el umbral global de enmascaramiento para cada sub-banda, y a partir de esto se escoge un cuantizador uniforme que entregue la mínima distorsión a la tasa de bits requerida. Las capas I y II son muy similares, pero la capa II logra mejor rendimiento utilizando una FFT de alta resolución, una más fina cuantización y una manera más eficiente de mandar los factores de escala para las sub-bandas.

Uno de los problemas derivados de la cuantización es la aparición de pre-ecos. Los pre-ecos pueden ocurrir cuando un sonido agudo, de percusión está precedido por silencio. Cuando la señal es reconstruida, los errores debidos a la cuantización tienden a ser distribuidos sobre el bloque de muestras, causando así una distorsión audible anterior a la señal actual. En una ventana de 8 ms, los pre-ecos no son suprimidos completamente por el enmascaramiento temporal.

El control de pre-ecos es una parte importante de la capa III de la codificación de audio MPEG. La capa III agrega una descomposición de las sub-bandas con una transformación coseno discreta (DCT) para obtener una subdivisión mucho más fina de la frecuencia. La capa III también agrega cuantización no uniforme (señales más grandes pueden enmascarar grandes errores de cuantización), codificación de entropía, y conmutación dinámica entre ventanas. Esta última característica provee mejor resolución temporal, lo que ayuda a un mejor control de los pre-ecos.

- Video MPEG:

La capa de video en MPEG está específicamente diseñado para la compresión de secuencias. Una secuencia de video es simplemente una serie de imágenes tomadas a intervalos muy pequeños en el tiempo. Exceptuando el caso particular de un cambio de escena, dichas imágenes tienden a ser similares a la siguiente. Intuitivamente, un sistema de compresión de video debe tomar en cuenta esta característica para aprovecharlo en términos

de eficiencia. Las técnicas de compresión que utilizan información de otras imágenes en la secuencia son comúnmente llamadas técnicas inter-cuadro (interframe).

Cuando se da un cambio de escena (y en ocasiones por otros motivos), no funciona la técnica inter-cuadro y el modelo de compresión cambia. En este caso el modelo de compresión debe ser estructurado para tomar ventaja de la similitud que existe en un región dada de la imagen con respecto a las áreas adyacentes de la misma. Las técnicas de compresión que sólo usan información de una imagen sencilla son llamadas técnicas intra-cuadro (intraframe).

Estas dos técnicas de compresión, inter-cuadro e intra-cuadro están en el corazón del algoritmo de compresión de video de MPEG.

La capa más externa de un flujo de bits de video MPEG es la capa de secuencia de video. Excepto por cierta información crítica de sincronía en la capa de sistema MPEG, una secuencia de bits MPEG de video está completamente autocontenida. Es independiente de otros flujos de bits de video (o de audio).

Cada secuencia de video está dividida en uno o más grupos de imágenes, y cada grupo de imágenes se compone de una o más imágenes de uno de tres distintos tipos; "I", "P" y "B". Las imágenes "I" (I es por intra-codificadas) se codifican independientemente, totalmente libres de referencias a otras imágenes en el grupo. Las imágenes "P" y "B" son comprimidas codificando las diferencias entre la imagen en cuestión y alguna imagen de referencia, que puede ser del tipo "P" o del tipo "I", explotando las similitudes que puedan existir entre ellas.

Las imágenes "P" (imágenes codificadas con predicción) obtienen predicciones de imágenes "I" o "P" precedentes en la secuencia, mientras que las imágenes "B" (imágenes codificadas con predicción bidireccional) requieren predicciones de las imágenes "I" o "P" precedentes y/o siguientes en la secuencia. Diferentes regiones de las imágenes "B" pueden usar distintas predicciones, y pueden predecir de imágenes precedentes, imágenes posteriores, de las dos o de ninguna. Similarmente, las imágenes de tipo "P" pueden también predecir de imágenes anteriores o simplemente no utilizar la predicción. Si no se utiliza la predicción, esa región de la escena es codificada usando técnicas "intra". En un grupo cerrado de imágenes, las de tipo "P" y las de tipo "B" son predichas sólo de otras

imágenes en el grupo de imágenes; en un grupo abierto de imágenes, la predicción puede ser hecha con imágenes externas al grupo.

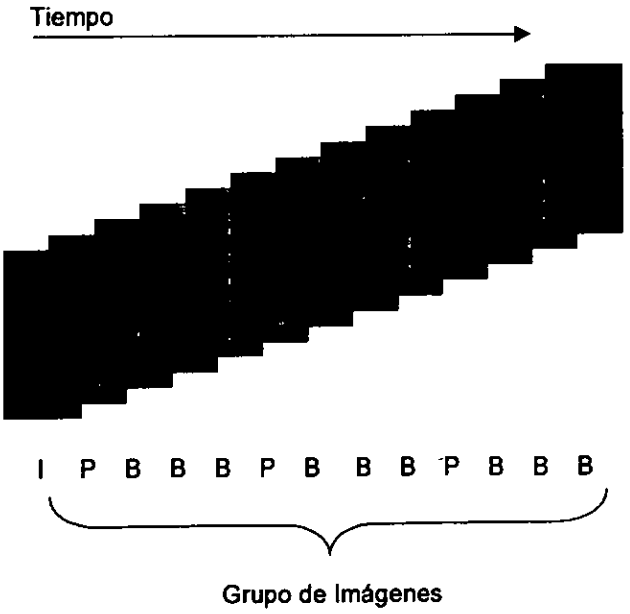


Figura 4.3 Arreglo típico de imágenes en el estándar MPEG

Un arreglo típico de imágenes para un grupo y que es utilizado en varias aplicaciones es el siguiente (figura anterior):

I-B-B-B-P-B-B-B-P-B-B-B-P-...

Como MPEG utiliza información de imágenes futuras dentro de la secuencia, el orden de codificación, el orden en el que las imágenes comprimidas se encuentra en el flujo de bits, no es el mismo que el orden de despliegue (el orden en el que las imágenes se presentarán a un espectador). El orden de codificación es el orden en el que las imágenes

deberán ser decodificadas por el decodificador. El grupo típico de imágenes descrito en el párrafo anterior tendría como orden de codificación el siguiente:

I-P-B-B-B-P-B-B-B-P-B-B-B-...

La estructura básica de construcción en una imagen MPEG es el "macrobloque", ilustrado en la siguiente figura. El macrobloque consiste de un arreglo de muestras de luminancias (tonos de grises) de 16 x 16 junto con un bloque de muestras de 8 x 8 para cada uno de los componentes de color (crominancias).

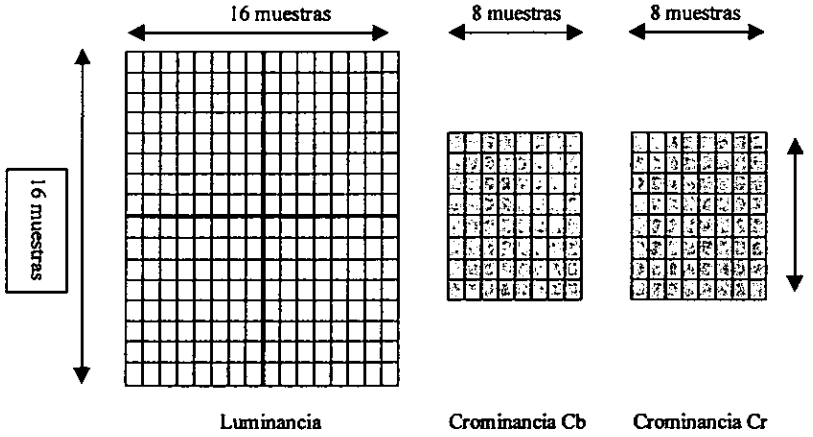


Figura 4.4 Formación de un "macrobloque"

El arreglo de muestras de 16 x 16 de las luminancias está realmente compuesto de cuatro bloques de 8 x 8, y estos bloques de 8 x 8 son las unidades de información con las que se alimentan los modelos de compresión.

La imagen MPEG no es simplemente un arreglo de macrobloques. Más aún, está compuesta de "rebanadas" (slices), donde cada rebanada es una secuencia contigua de macrobloques en orden horizontal de barrido (como se leen las líneas de este texto), empezando en una dirección o posición específica de la imagen especificada en el encabezado de la rebanada. Las rebanadas pueden continuar de una fila de macrobloques a la siguiente en MPEG-1. Esta estructura de rebanadas, entre otras cosas, permite gran

flexibilidad en cambios de señalización en algunos de los parámetros de la codificación. Esto se necesita tanto para optimizar la calidad para una tasa de transmisión dada y para controlar la misma. A pesar de esto, algunas implementaciones de codificadores ignoran la estructura de rebanadas por causar esto una complicación computacional del algoritmo.

En el corazón de las codificaciones "intra" e "inter" en MPEG se encuentra la transformada coseno discreta (DCT). La DCT tiene ciertas propiedades que simplifican los modelos de codificación y hacen al código eficiente en términos de medidas de calidad perceptuales.

Básicamente, la DCT es un método para descomponer un bloque de información en una suma ponderada de frecuencias espaciales. En el caso particular de bloques de 8 x 8, obtenemos 64 patrones de frecuencias espaciales. Cada uno de esos patrones espaciales de frecuencia tiene un coeficiente correspondiente, que es la amplitud necesaria para representar la contribución de ese patrón de frecuencia espacial en el bloque de información que está siendo analizado. En otras palabras, cada patrón de frecuencia espacial se multiplica por su coeficiente y los resultantes 64 arreglos de 8 x 8 de amplitud son sumados, cada pel[3] por separado, para reconstruir el bloque de 8 x 8.

Si sólo los coeficientes de baja frecuencia de la DCT son distintos de cero, la información en el bloque varía lentamente con la posición. Si están presentes altas frecuencias, las intensidades del bloque cambian rápidamente de pel a pel. Este comportamiento cualitativo se entiende al observar la componente de DC, o frecuencia cero, que resulta ser el promedio del valor de todo el bloque y al observar después algún término de alta frecuencia, que se vería prácticamente como un tablero de ajedrez de 8 x 8. Algo interesante es que la transformada DCT de dos dimensiones puede ser dividida en dos transformadas DTC, una en la dirección de cada eje y de esta forma se facilita mucho el análisis de las frecuencias.

Cuando se calcula la DCT para un bloque de pels, es deseable representar a los coeficientes de altas frecuencias espaciales con menor precisión. Esto se hace gracias a un proceso llamado cuantización. Un coeficiente DCT es cuantizado dividiéndolo por un entero positivo diferente de cero, llamado el valor de cuantización, y redondeando el cociente de esta división –el coeficiente DCT cuantizado- al entero más cercano. Entre más

[3] Pel significa "picture element", o elemento de imagen. También es conocido como *pixel.*

grande sea el valor de cuantización, menor la precisión del coeficiente DCT cuantizado. Coeficientes de baja precisión pueden ser transmitidos al decodificador con menos bits. El uso de grandes valores de cuantización para altas frecuencias espaciales permite al codificador descartar selectivamente actividad de alta frecuencia espacial que el ojo humano no puede percibir. Los valores por los que se dividen los coeficientes obtenidos de la DCT forman la llamada "matriz de cuantización", misma que el estándar JPEG recomienda. Esta matriz fue hallada a partir de una serie exhaustiva de experimentaciones llevadas a cabo por el grupo de expertos en diferentes tipos de imágenes, tratando de cuantificar la importancia relativa de cada uno de los coeficientes dentro de un análisis psico-visual. La cuantización en MPEG-1 es variable, y la variación consiste simplemente en *escalar* la matriz de coeficientes de cuantización en determinadas ocasiones; con este efecto, se puede conseguir una cuantización más severa o más suave, dependiendo del factor de escala. Hay que hacer notar que el resultado final de la cuantización es el de obtener un número considerable de valores que por ser tan pequeños, puedan ser considerados como ceros y así conservemos sólo los coeficientes más relevantes de cada bloque sin perder la información contextual importante.

La DCT y la cuantización ponderada visualmente de la misma son partes fundamentales del sistema de codificación y compresión de MPEG.

Como se dijo anteriormente, un macrobloque está compuesto de cuatro bloques de muestras (monocromáticas) de luminancias de 8 x 8 y dos bloques de 8 x 8 de muestras de crominancias. Las muestras de crominancia representan el color en términos de la presencia o ausencia de rojo y azul para una intensidad de luminancia dada. Estos bloques de 8 x 8 de información son la unidad procesada por la DCT. La razón por la que se usa una resolución más baja en los bloques de crominancia es porque el ojo humano tiene una capacidad de resolución mejor para altas frecuencias en luminancia que en crominancia.

La DCT se constituye en una herramienta que genera varias ventajas desde el punto de vista de la compresión de datos. Primeramente, en el caso de la codificación "intra", los coeficientes DCT están casi completamente decorrelacionados, esto es, son independientes de los demás, y por lo mismo, pueden ser codificados independientemente. Esto hace posible diseñar un algoritmo relativamente simple (llamado modelo de codificación) para poderlos codificar. La decorrelación es de gran interés teórico y práctico en términos de la

construcción del modelo de codificación. De cualquier forma, el rendimiento de la codificación está más profundamente influenciado por la cuantización visual ponderada.

En la codificación "no-intra" (codificando la diferencia entre la imagen actual y una imagen ya transmitida) la DCT no mejora grandemente la decorrelación, ya que la señal de diferencia obtenida restando la predicción de una imagen similar está ya de por sí bastante bien decorrelacionada. De todas formas, la cuantización es una manera muy poderosa de comprimir para controlar la tasa de bits, aún si la decorrelación no mejora mucho con la DCT. En la práctica, la DCT trabaja muy bien para las codificaciones "intra" e "inter".

Si hay movimiento en la secuencia, se puede obtener una mejor predicción codificando las diferencias relativas a las áreas que están corridas con respecto al área que está siendo codificada; a este proceso se le llama compensación de movimiento. El proceso que se lleva a cabo para determinar los vectores de movimiento en el codificador se llama estimación de movimiento.

Los vectores de movimiento que describen la dirección y cantidad de movimiento de los macrobloques son transmitidos al decodificador como parte del flujo de bits. Es entonces cuando el decodificador sabe qué área de la imagen de referencia fue usada para cada predicción, y suma la diferencia decodificada con esta predicción de movimiento compensado para obtener la salida. El codificador debe seguir el mismo procedimiento cuando la imagen reconstruida será usada para predecir otras imágenes. El proceso de reconstrucción del codificador es comúnmente llamado "decodificación local".

Los coeficientes DCT cuantizados son codificados *sin pérdidas*, por lo que el decodificador puede reconstruir precisamente los mismos valores que le fueron enviados. Para el MPEG, una técnica cuasi-óptima de codificación basada en código de Huffman modificado se usa para generar las tablas de código de longitud variable necesarios para esta tarea. Los códigos de longitud variable se requieren para lograr alta eficiencia en la codificación, ya que los códigos más cortos serán asignados a los eventos más altamente probables y su entropía sería menor.

Los coeficientes son arreglados de acuerdo a una secuencia unidimensional llamada "zig-zag". El ordenamiento zigzag se muestra en la siguiente figura, donde se puede ver que esta revisión ordena a los coeficientes aproximadamente de acuerdo al criterio de frecuencia espacial ascendente. Como ya se había mencionado, la cuantización visualmente

ponderada descarta fuertemente a las altas frecuencias espaciales, sólo algunos coeficientes de baja frecuencia son distintos de cero en una transformación típica. Como regla general podríamos decir que entre más coeficientes valgan cero, será mejor la compresión. Consecuentemente, es muy importante que el modelo de codificación use *símbolos* (combinaciones particulares de información de coeficientes DCT) que permitan una codificación eficiente de DCT's que contengan muchos coeficientes con valor nulo. Uno de estos símbolos es un EOB (end-of-block), que le indica al decodificador que todos los coeficientes restantes del bloque son cero.

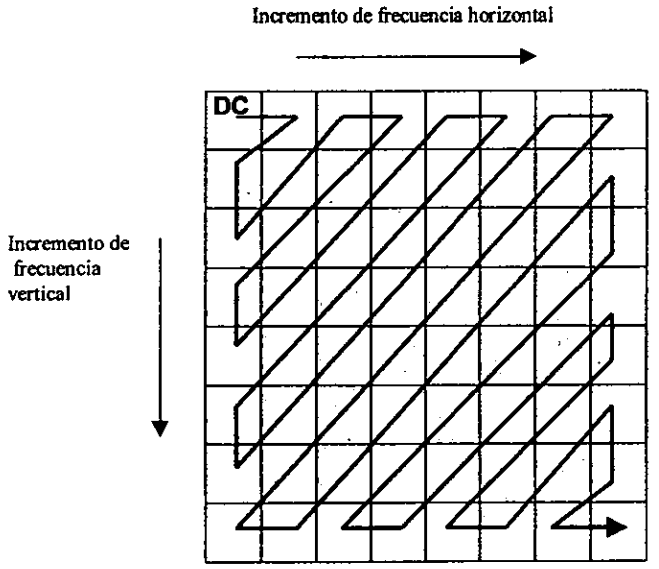


Figura 4.5 Barrido en forma de "zig-zag" de los coeficientes de la DCT

La DCT es utilizada para codificar información de imagen tanto de tipo "intra" y de tipo "no intra", pero las propiedades son realmente muy diferentes en los dos ambientes de codificación. Se utilizan diferentes tablas de cuantización en compresión "no intra", y el redondeo se hace de forma distinta. La información que va a ser transformada es muy diferente, por lo que se usan diferentes conjuntos de símbolos y distintas tablas de código.

La codificación de los coeficientes de DC de los macrobloques sucesivos se lleva a cabo con la ayuda de un código diferencial de Huffman modificado, se toma el primer coeficiente como referencia, y se codifican sólo las diferencias del actual con el siguiente, aprovechando la alta correlación que existe entre macrobloques adyacentes. Por el otro lado, los coeficientes de AC luego de haber sido acomodados por el método "zig-zag" en un vector fila, son codificados utilizando tablas categorizadas de símbolos con baja entropía, tomando en cuenta los ceros precedentes (que suelen aparecer en las altas frecuencias) y el "rango" del coeficiente mismo, obtenido de una tabla que los categoriza de acuerdo a su magnitud, ya que es más común encontrar coeficientes de orden pequeño (1, 2, 4, etc.) que valores mayores con las matrices de cuantización recomendadas. De esta forma, el flujo de bits a ser enviado finalmente, es un flujo bastante práctico y eficiente, resaltando el aspecto de que la compresión por codificación entrópica es sin pérdidas (losless), con el objetivo de que el decodificador reciba la información y sea absolutamente capaz de reconstruir los coeficientes de la DCT y poder antitransformarlos para regenerar la secuencia.

Es importante reconocer que el MPEG es potencialmente un sistema altamente asimétrico. Los decodificadores son relativamente simples, ya que únicamente siguen las direcciones codificadas dentro del flujo de bits. Los codificadores, por lo contrario, son mucho más complejos que los decodificadores y deben tener mucho más inteligencia. Entre otras cosas, los codificadores deben identificar las áreas en movimiento, determinar los vectores óptimos de movimiento, deben controlar la tasa de bits, deben controlar el manejo de datos en los búfers de forma que no se presente *overflow* ni *underflow*, deben determinar donde cambiar la cuantización, decidir cuándo un bloque determinado puede ser simplemente repetido, deben tomar la decisión de cuando utilizar las técnicas "intra" o "inter" de codificación y variar todos estos parámetros y decisiones dinámicamente para maximizar la calidad deseada para la tasa de transmisión requerida. La estrategia en la toma de decisiones para implementar un codificador MPEG es probablemente el aspecto más complejo y menos publicado en el área.

- Algo sobre MPEG-2

MPEG-2, como es sabido, es la continuación de los esfuerzos de MPEG-1 y está enfocado principalmente a lograr más altas tasas de transmisión, más grandes tamaños de imagen y cuadros entrelazados[4]. MPEG-2 ha sido construido con base en las capacidades de MPEG-1. De hecho, los decodificadores MPEG-2 deben al menos estar capacitados para decodificar la mayoría de los flujos de bits codificados en MPEG-1.

Han sido definidos cinco diferentes perfiles para el video MPEG-2, especificando conjuntos particulares de capacidades que van desde las funciones cercanas al MPEG-1 hasta las avanzadas técnicas de compresión de video que se requieren para la HDTV. Uno de los perfiles de MPEG-2, el principal (MP), es curiosamente una extensión directa de MPEG-1. De manera escueta, este perfil introduce el concepto de campos de video y de cuadros entrelazados. Se extiende el concepto de cuantización variable, se puede configurar la precisión de la DCT y se define una nueva tabla de codificación. Por otro lado, el estándar espera más alta resolución, más alta tasa de bits y más alta razón de cuadro. También se crecen los rangos de los parámetros especiales, como ancho y alto de la imagen, tasa de bits y tamaño de los búfers. Otra innovación es que es posible la grabación de información de "copyright". Existe la posibilidad de transmitir características de despliegue, incluyendo un corrimiento para ver un subconjunto de la imagen en una ventana.

Los sistemas MPEG-2 incluyen todas las ventajas de MPEG-1 con la adición de mejoras que les permiten operar en ambientes más propensos a errores. En suma, se pueden manejar más cadenas de audio y video que no tienen que estar ligadas por una base común de tiempo. Los sistemas MPEG-2 todavía utilizan los conceptos de gran paquete y paquete, pero los combinan en modos que son tanto compatibles como incompatibles. Algunos de los modos incompatibles soportan aplicaciones MPEG-2 en el Modo de Transmisión Asíncrona (ATM).

El audio MPEG-2 es compatible "hacia atrás" con el MPEG-1. A pesar de que se permiten tres canales más aparte de un *sub-woofer* para generar un ambiente mejorado de sonido envolvente, un decodificador de audio MPEG-1 puede extraer y decodificar dos

[4] Llamados en la literatura original "Interlaced frames"

canales de audio. También, un decodificador de audio MPEG-2 puede hacer la labor de decodificación de los dos canales de audio MPEG-1. El requerimiento de que sea compatible "para atrás" implica algunos compromisos de eficiencia de código.

El video MPEG-1 tuvo como enfoque principal a las aplicaciones de medios de almacenamiento digital y se actuó en consecuencia para diseñar el formato de video, suponiendo bajas tasas de error e ignorando la televisión entrelazada. MPEG-2 tiene la intención de ser un estándar genérico y por lo tanto es mucho más amplio. El video MPEG-2 mantiene toda la sintaxis del video MPEG-1, pero utiliza extensiones para agregar flexibilidad adicional y más funciones. Las extensiones "escalables" se adicionan para proveer flujos de información de video con múltiples resoluciones para una potencial coexistencia entre la TV normal y la HDTV. Otras extensiones escalables permiten a la cadena de datos ser dividida en dos piezas. Otras también ofrecen alguna flexibilidad temporal para que no todos los cuadros tengan que ser reconstruidos.

- Algo sobre MPEG-4.

Antes de que MPEG-2 fuera concluido, un nuevo proyecto, el MPEG-4, había comenzado. Había sido encaminado inicialmente a tasas muy bajas de transmisión, pero con el tiempo su enfoque cambió hacia la alta compresión. Las prioridades más altas fueron asignadas a la interactividad basada en el contenido y al "acceso universal", incluyendo redes inalámbricas propensas al error. La manipulación basada en el contenido, la escalabilidad y la edición son características esperadas. El estándar MPEG-4 proveerá un lenguaje de descripción sintáctica (MSDL). Se espera que este lenguaje describa cómo analizar y procesar los flujos elementales de información (que podrían ser MPEG-1, MPEG-2 o inclusive objetos sintéticos bi-dimensionales o tri-dimensionales). Las piezas restantes estarán disponibles en Internet y podrán ser "bajadas" de un conjunto abierto de herramientas y algoritmos. La configuración e instalación de dichas piezas se hará de la forma "plug-in".

V. Implementación del codificador

Un codificador consta de dos partes: codificador de fuente (o modelo entrópico), y de canal. Cada una de estas partes ya ha sido discutida en capítulos anteriores. En este capítulo discutiremos en algún detalle cada una de ellas según existen en el presente trabajo, su diseño, implementación y posibles mejoras.

V.1 Diseño:

Es posible que esta fase sea considerada trivial en la implementación de un estándar. En nuestro caso, nada está más lejos de la verdad. MPEG es un estándar de decodificación, y muchas de los algoritmos ocupados en el codificador son dejados a la discreción de los programadores. Además, se debieron hacer consideraciones ajenas a las cuestiones propias del estándar: puesto que el objetivo de la tesis es proporcionar una versión fácilmente mantenible y extensible, estas cuestiones debieron tomar un papel preponderante en cada una de las decisiones de diseño.

Elección de lenguaje:

Dichas consideraciones sugerían que el uso de un lenguaje como C no sería recomendable. Si bien la velocidad y solidez de C son universalmente reconocidas, su legibilidad es pobre para programadores con poca experiencia en el lenguaje. C++ presentaba un problema semejante. Era también necesario tomar en cuenta que el programa que presentamos aquí es parte de la investigación que se lleva a cabo en la División de Posgrado, y que es deseable que se integrara con trabajos existentes y posteriores. Como gran parte de dicho trabajo se elabora haciendo uso del ambiente de programación MatLab, por no hablar que la manipulación numérica de matrices (de las cuales las imágenes pueden considerarse parte) es la especialidad de MatLab, el mismo representaba la solución natural al problema de la elección del lenguaje. Más adelante profundizaremos en la incidencia de MatLab en el presente proyecto.

Es aconsejable, sin embargo, el uso de C en cualquier proyecto que tenga que interactuar a bajo nivel con el sistema operativo (en nuestro caso la entrada/salida a nivel de bits), por lo que una combinación de ambos ambientes nos pareció la solución más viable.

- Matlab

Matlab es un ambiente de desarrollo fuertemente orientado a la manipulación numérica. Diseñado por Cleve Moler de la Universidad de Stanford a finales de los años 70, este lenguaje presenta facilidades a ingenieros y físicos equivalentes a las que los diversos intérpretes de comandos (*shells*) de UNIX ofrecen a los administradores de sistemas y usuarios del sistema. Mathworks, la compañía que lo publica, ha definido MatLab como un sistema interactivo para el cálculo (cómputo) y la visualización, que integra el análisis numérico, el álgebra de matrices, el procesamiento de señales y los gráficos en un ambiente interactivo.

Como lenguaje de programación, MatLab presenta algunas características diferentes a lenguajes más tradicionales (procedurales ``duros" como C o Perl). Esto se ha atribuido en gran medida a su línea de evolución: adición de características de gran popularidad entre los usuarios al *corpus* de lenguaje, así como de desarrollos en la implementación de lenguajes de programación universalmente considerados útiles, todo ello con el propósito de ser compatibles con las versiones anteriores del lenguaje.

Hasta recientemente, el único tipo de datos en MatLab era la matriz: un número o escalar, es considerado una matriz de 1x1, mientras que una cadena es un vector de caracteres. A partir de la versión 5, se añadieron celdas (que extienden la idea de matriz permitiendo ser indicadas por cadenas, además de números), estructuras (parecidas a las de C, o a los registros de Pascal) y objetos. Estos tipos de datos pueden remediar en alguna medida los defectos en el desarrollo de sistemas mayores mencionados arriba. De la última estructura, los objetos, hacemos uso extensivo en el programa que presentamos en este escrito.

Entre las características poco comunes a las que nos referíamos arriba, la más evidente es la marcada preferencia a las operaciones sobre matrices elemento por elemento, pero sin hacer uso de los ciclos *for* a los que los proyectos en C o Pascal nos han

acostumbrado. El ciclo *for* existe, por supuesto, pero su uso es tan gravoso en los recursos de la máquina que pronto se hace lo posible por prescindir de su utilización.

El parecido entre Matlab y los lenguajes de "shell" y orientados al procesamiento de texto, como Perl también tiene sus aspectos negativos: el uso para el que están diseñados es programas rápidos y cortos, con poca o ninguna comunicación con un sistema mayor, lo que no facilita el desarrollo de ese tipo de sistemas. Evidencia de ello es el manejo de variables globales, lo suficientemente aparatoso para descalificarlas como la solución rápida que representan en lenguajes tradicionales. En este sentido, MatLab obliga a mantener un diseño poco acoplado, lo que facilita el mantenimiento del programa, pero no su desarrollo.

Los programadores de MatLab pronto admitieron que había muchas cosas que otros lenguajes hacían mejor que MatLab, por lo que prefirieron proveer un escape tanto al sistema operativo como una interfaz de programación en C, que agregar características al lenguaje original. De estas facilidades también hacemos uso extensivo, pues toda escritura al archivo se hace a través de dicha interfaz.

Elección de plataforma:

En cuanto a la plataforma, teníamos la opción de programar para estaciones de trabajo Sun o en PCs. La experiencia que podemos tener como programadores se ha alcanzado en estas últimas, además de que existen implementaciones semejantes a la nuestra, cuyo código podíamos estudiar y aprovechar. Asimismo, desde el principio tuvimos claro el objetivo de hacer el codificador fácilmente extensible y modificable.

Programación orientada a objetos:

Las ciencias y la ingeniería de computación son razonablemente jóvenes comparadas con otras disciplinas. Podría decirse que es por ello que se ven sujetas a "modas" con mayor facilidad que dichas disciplinas. Durante los años 70, la programación estructurada era considerada el estado del arte, aunque no fue firmemente establecida sino hasta una década después. Paralelamente, se desarrollaba un nuevo estilo de programación, la orientada a objetos (OO), que no alcanzaría gran difusión entre los programadores sino hasta la década

de 1980, específicamente con la introducción del lenguaje C++, y el estrellato con la inusitada popularidad del lenguaje de programación Java, en 1994 y hasta la fecha.

Sin embargo, creemos que este trabajo no es sólo un nombre más que apuntar a la lista de proyectos que usan objetos. La decisión de usarlos se debió principalmente a la naturalidad con que un esquema sintáctico puede trasladarse a la metodología OO. Conviene mencionar que MPEG es un especificación puramente sintáctica, pues describe piezas de datos encapsuladas por cabeceras, que a su vez están encapsuladas, etc. Las clases están claramente definidas por cada uno de los elementos sintácticos que deben manipularse. Los métodos son aquellas operaciones pertinentes a cada elemento (semántico) del codificador. Así, por ejemplo, si la DCT y el modelo de codificación operan sobre macrobloques, dicha operación puede implementarse como un solo método sobre esa clase.

Se le llama objeto a una instancia de una clase determinada. Las clases y los objetos sirven para agregar nuevos tipos de datos y nuevas operaciones a un lenguaje de programación. La clase de una variable describe la estructura de dicha variable e indica el tipo de operaciones y funciones que se pueden aplicar sobre esa variable. En otras palabras, lo anterior es semejante a definir en un lenguaje de programación, un tipo de datos diferente a los que ya contiene, para después utilizar variables del nuevo tipo de datos. En este caso, la variable definida sería una instancia de la clase o tipo de datos nuevo. La frase "Programación Orientada a Objetos" describe un enfoque para escribir programas que hagan énfasis en el uso de clases y objetos,

El concepto de objetos en el codificador se aprovechó para definir objetos como: película, imagen (frame) y macrobloque (MB). En MPEG-1 se tienen los siguientes elementos: película (secuencia), grupo de imágenes (GOP), imagen (frame), rebanada de macrobloques (slice), macrobloque (MB) y bloque. Como se puede observar, los elementos antes listados tienen una relación entre sí, ya que el primer elemento mencionado se compone de varios elementos del tipo siguiente y así sucesivamente, por ejemplo: la película es un conjunto de imágenes, cada una de las cuales se compone de un slice y así sucesivamente.

En Matlab, una clase se reconoce porque el nombre del directorio que contiene los métodos que aplican sobre una clase es igual al nombre de la clase precedido por una @. Entre los métodos que contiene el directorio se debe encontrar el constructor de dicha clase. El nombre del constructor es igual al nombre de la clase y al del directorio sin la @. La función del constructor es crear los objetos inicializando la estructura de datos y asignar la etiqueta de clase.

En el presente trabajo no se creó un objeto para cada uno de los elementos que forman parte del flujo de datos MPEG-1, nosotros manejamos únicamente 3 objetos que son la pelicula, el frame y el macrobloque, porque con ellos fue suficiente para manejar de una manera rápida y fácil los datos. Se prescindió de la creación de un objeto GOP, porque dicha división en grupos de imágenes es sólo utilizada en la decodificación de películas grandes, con el fin localizar rápidamente una escena en especial de la película.

Como nuestro objetivo no es utilizar el codificador para películas muy grandes, sólo manejamos un GOP. También para obtener una mayor facilidad en el manejo de los datos, no creamos un objeto slice que agrupa de forma general un renglón de macrobloques, sino que sólo usamos un slice por imagen, con fines de sincronización en la barrida (raster) de un aparato de despliegue. El no utilizar estos objetos no quiere decir que no los estemos tomando en cuenta, ya que en el estándar forzosamente aparecen estos 6 elementos, lo que nosotros hacemos es mandar los datos de GOP una sola vez en el encabezado del mismo nombre, al inicio de la secuencia y los datos de slice en su encabezado al inicio de cada imagen.

Lo que sí hicimos para una mejor claridad y manejo de los datos, fue crear una "clase abstracta", frame, con subclases de imágenes, que son framei, framep y frameb. Lo anterior se hizo para no tener un solo objeto para los tres tipos de imágenes y que quedaran campos vacíos en el objeto. Por ejemplo un framep sólo tiene referencia pasada y no futura, mientras que un frameb tiene las dos referencias, por lo tanto si se manejara un solo objeto, el framep tendría siempre vacía la referencia futura y el framei tendría ambas referencias vacías, por lo tanto optamos por tener una clase frame que contenga los campos comunes a los tres tipos de imágenes y subclases framei, framep y frameb que contengan solamente los campos específicos para cada tipo de imagen, sabiendo que por ser una subclase,

además de todos los campos particulares de la subclase, tendrá todos los campos de la clase padre debido a la característica de herencia que poseen los objetos.

Los beneficios obtenidos al utilizar objetos en la implementación, se observan aparte de en la facilidad de manejo de los datos, en que el tamaño de las imágenes que componen la película no es fijo como sucede al utilizar la película de Matlab, sino que el modelo de objetos trabaja con el tamaño real de la imagen.

Al utilizar el formato de película que proporciona Matlab, nos obliga a utilizar un tamaño fijo para las imágenes de la secuencia, que es de 432 por 304, no importando el tamaño de la imagen, ya que la estructura de Matlab para hacer películas no depende del tamaño de la imagen, sólo depende de la resolución a la que se encuentre configurado el equipo con el que estemos trabajando. Lo que se obtiene como ganancia al utilizar los objetos, es que como se trabaja con el tamaño real de la imagen, se tiene una mayor compresión al escribir al archivo.

Trabajo anterior:

MPEG-1 es un estándar ya ampliamente aceptado en la industria. Prueba de ello es la gran variedad de programas que existen para el procesamiento (codificación y decodificación) y análisis de un flujo de datos MPEG. La vastedad de estos recursos hace casi inevitable que existieran programas con objetivos semejantes a los nuestros. De entre los que encontramos, los que más se parecen a lo que queríamos fueron el MPEGv1.2, del Portable Video Research Group en la Universidad de Stanford, y el MPGWrite, una adaptación del codificador de dominio público mpeg_encode.

El primero es un codec de dominio público, compatible con las versiones de agosto (Santa Clara) y diciembre (París) de 1991 del estándar. Este grupo de investigación no tiene como objetivo principal el desarrollo de implementaciones de los estándares de video, por lo que su programa no es muy eficiente. En todo caso, su código es más legible que el de nuestra otra referencia, que se describe a continuación.

El segundo es un codificador/decodificador de gran difusión, completamente implementado en software y muy portable. Escrito originalmente por Lawrence A. Rowe, Kevin Gong, Eugene Hung, Ketan Patel, Steve Smoot y Dan Wallach, del Berkeley Plateau

Research Group, en UCB, fue modificado por David Foti de Mathworks, quien le agregó una interfaz para MatLab 4. Posteriormente, Stephen Bond, de la Universidad de Kansas, lo adaptó al API de MatLab 5. Esta última versión, si bien no es parte de la distribución estándar de MatLab, está disponible en la sección dedicada a las contribuciones de la comunidad de programadores del sitio FTP de Mathworks, y al parecer es muy popular entre los usuarios de este ambiente de desarrollo.

A primera vista podría parecer que un programa de las características descritas cubre completamente las necesidades que nos proponíamos satisfacer con el nuestro. Por diversas razones no es así. Quizá la más importante de éstas (definitivamente no la única) es que el mpeg_encode está completamente escrito en C, lo que hace difícil el aprovechamiento del trabajo anterior en la División de Posgrado, que, como ya se ha dicho, está escrito en su mayor parte en MatLab. Además, el código está diseñado para lograr una alta eficiencia en el mayor número posible de plataformas, por lo que la legibilidad del mismo sufre en alguna medida. Asimismo, no puede ignorarse el hecho de que parte del código fue inhabilitado por el segundo esfuerzo de programación (que hizo el código compatible con el formato de películas de MatLab) lo que no mejora la legibilidad, y es fuente constante de perplejidades al lector, pues hay regiones en el programa original aparentemente necesarias pero comentadas (por ejemplo, todos los módulos de lectura, que manejan automáticamente la conversión de distintos tipos de imágenes vía la biblioteca netlib).

En todo caso, mucho del código en este último programa era muy aprovechable tanto para estudio como para inclusión. La licencia (estilo BSD) nos permite hacer uso y redistribución del código modificado provisto que se conserve el letrero de derechos de autor original, lo cual hemos cumplido puntualmente.

No se piense por ello que todo el trabajo de esta tesis consiste en la copia indiscriminada de trabajo ajeno. Casi la totalidad del codificador fue reescrita en MatLab, con la excepción de algunos módulos en la codificación de canal. Vale la pena mencionar que en este problema (específicamente en lo que se refiere a la construcción de las tablas del código de Huffman) la implementación de Berkeley da una solución particularmente elegante, proveyendo en la distribución un archivo de texto que es preprocesado por un

script de Perl y genera un archivo .h y otro .c, que son compilados con el resto de la fuente para dar origen al programa ejecutable. El hecho de que MatLab incluya un intérprete de Perl en su distribución estándar hace particularmente conveniente este tipo de solución.

V.2 Codificación:

Nuestro codificador consta de un conjunto de scripts de MATLAB y bibliotecas dinámicas escritas en C y compiladas bajo el sistema operativo Windows (95 y 98), organizadas en una jerarquía de objetos que reflejan las principales entidades en el proceso de codificación.

El archivo principal del codificador, que corresponde a la entrada del programa es el script mpeg.m, el cual recibe un conjunto de archivos especificados por una cadena (prefijo), su tamaño horizontal y vertical, el patrón de codificación(que determina el número de imágenes I, P y B que tendrá la secuencia de imágenes), el nombre del archivo del archivo de salida, el algoritmo de estimación de movimiento (especificado como una cadena), las escalas de cuantización correspondientes a imágenes I, P y B, el número de imágenes a codificar y de forma opcional, se puede especificar a partir de que número de imagen tome para empezar a codificar, esto es porque por ejemplo si se tiene un directorio donde se encuentran las imágenes de la secuencia sphere, esto se le pasa como prefijo, el programa ordena todos los archivos que empiecen con dicho prefijo, y si por ejemplo se tienen 50 imágenes que van del 1 al 50, se puede especificar, si es que a mí me interesa así, que tome a partir de la imagen 10 el número de imágenes que se le especifique.

Dicho programa se encarga de llamar al constructor del objeto película, el cual se encarga de determinar el tipo de imagen que le corresponde al frame actual de acuerdo al patrón de codificación, en este constructor es donde se hace la tabla de dependencias mencionada anteriormente. Dependiendo del tipo de frame al que corresponda la imagen en cuestión, se llama al constructor de esa subclase, el cual llama al constructor de la clase padre, para así tener los datos comunes a los tres tipos de frames en el padre y los datos particulares en el hijo(subclase). Para los frames i solo se tiene como dato particular la tabla

de cuantización, para los frames p, se tienen como datos, la dependencia pasada, los vectores de movimiento y la escala de cuantización, mientras que para los frames b, se tienen las dependencias pasada y futura, los vectores de movimiento y la tabla de cuantización. Se tiene especificada la tabla de cuantización de forma particular, aunque para imágenes p y b es la misma, porque para las imágenes i se usa una tabla de cuantización intraframe, y con que uno de los tipos de imagen tenga una característica diferente, ya es necesario especificar los datos de forma particular.

En el constructor frame (clase padre de framei, framep y frameb), es donde se leen las imágenes, se transforman al formato YUV (YCbCr luminancia-crominancia-crominancia), quedando tres matrices paralelas, la primera corresponde a las luminancias (Y), la segunda y tercera a las crominancias (Cb y Cr respectivamente). Como ya se dijo anteriormente, por cada macrobloque se toma un bloque de luminancias de 16x16 y un bloque de crominancias de 8x8 para Cb y otro también de 8x8 para Cr. Como las matrices de crominancias son del mismo tamaño que la matriz de luminancias, entonces que da un bloque de 16x16 para cada macrobloque, el cual tiene que ser muestreado, para obtener el bloque de 8x8 que se requiere, nosotros estamos tomando como muestra, la parte central del bloque de 16x16 para obtener el bloque de 8x8. El constructor de frame también llama al constructor de macrobloque, que guarda las luminancias y las crominancias de forma independiente, ya como bloques , que es la parte más pequeña que se considera en el flujo de datos MPEG. Debido a que separamos el bloque de luminancias de 16x16 como una celda que contiene 4 bloques de 8x8, y las crominancias ya tienen el tamaño de un bloque, ya no es necesario que tengamos un constructor para el objeto bloque. Hasta este momento ya tenemos organizada la información de una buena manera para poder procesarla y mandar a escribir al archivo MPEG.

Una vez que tenemos todos los datos en los objetos, es necesario tener métodos que puedan manejar la información contenida en los objetos, un método es también una función, que se encuentra en el mismo directorio que el constructor del objeto y es necesario porque un objeto solo lo puede leer alguno de sus métodos, es decir que solo se puede tener acceso al contenido de los objetos desde alguno de sus métodos.

Una vez construidos todos los objetos, ahora sí inicia el trabajo de procesar la información y después mandar a escribir al archivo ayudándose de funciones mex escritas en 'C', que nos permiten realizar la escritura de bajo nivel al archivo bit por bit. Fue necesaria la comunicación con 'C' porque Matlab sólo permite la escritura byte a byte, pero como las tablas de codificación de los datos son de longitud variable y son generalmente menores de un byte, no es posible escribir bytes completos, porque además de que obtendríamos una menor compresión, ese no es el problema más grave ya que el verdadero problema es que la codificación toma en cuenta el número de ceros que hay entre cada código de longitud variable, y con ello obtendríamos grandes errores al momento de la decodificación del archivo escrito.

El archivo principal mpeg.m, una vez creados los objetos, se encarga de llamar a las funciones que escriben los encabezados de cada uno de los 6 elementos del flujo MPEG ya mencionados y de llamar a las funciones que pasan a la imagen por la DCT(Transformada Coseno Discreta), y la cuantizan.

- Codificación de imágenes I

Como se menciona en capítulos anteriores, las imágenes I se codifican de forma independiente de los demás frames, ya que no dependen ni de referencias anteriores ni de referencias futuras. La codificación *intra*, quiere decir que todos los datos necesarios para la codificación se encuentran en el mismo frame y que no se requiere de estimación de movimiento para mandar a escribir los datos al archivo. Para codificación de imágenes *intra*, se utiliza la siguiente tabla de cuantización:

8,16,19,22,26,27,29,34

16,16,22,24,27,29,34,37

19,22,26,27,29,34,34,38

22,22,26,27,29,34,37,40

22,26,27,29,32,35,40,48

26,27,29,32,35,40,48,58

26,27,29,34,38,46,56,69

27,29,35,38,46,56,69,83

Tabla 5.1 Coeficientes de cuantización recomendados

Una vez escritos los encabezados de secuencia, GOP, picture y slice, el encabezado que sigue es el de macrobloque, el cual contiene diferentes datos, dependiendo del tipo de codificación que se esté haciendo, intraframe ó interframe.

En la codificación de imágenes I, en el encabezado de macrobloque, únicamente se especifica el tipo de imagen I con un número 1, el incremento en la dirección del macrobloque, que siempre es 1 para este tipo de imágenes, ya que en imágenes I, no es posible saltarse macrobloques, además de que se especifica que se trata de un macrobloque de tipo intra. Una vez escrito el header de macrobloque, se procede a codificar los 6 bloques que corresponden a cada macrobloque, que son 4 bloques de 8x8 para las luminancias y un bloque de crominancia cb y otro de crominancia cr.

Para iniciar la codificación de cada bloque perteneciente a un macrobloque, lo que se hace en el código de MatLab es tomar el frame completo, luego ir tomando cada macrobloque, pasar cada bloque por la DCT, cuantizar (dividir el resultado de la DCt entre la tabla de cuantización correspondiente, ya sea intra o interframe) los valores resultantes y separar el coeficiente 1,1 del arreglo que se obtiene al aplicar la DCT y cuantizar. Esto de hace para cada bloque, y dichos coeficientes se denominan coeficientes de DC, los cuales contienen la mayor cantidad de la información para una imagen que se codifique como intra. Después de pasar cada bloque por la DCT y cuantizarlo, se recorren los coeficientes en forma de zigzag y quedan los 63 coeficientes restantes, llamados de AC, listos para ser codificados.

Una vez que ya se separaron los coeficientes de DC de los de AC, se tiene que hacer una resta del valor actual de DC con el valor predicho de DC, esto es, que al inicio del

frame, los valores predichos tanto para luminancias (ydc_pred) como para crominancias (cbcd_pred y crdc_pred) son puestos a 128, y entonces al tomar el primer valor de DC, se le resta el valor predicho que es 128 en ese momento, el siguiente valor predicho será el valor actual del coeficiente de DC. Cabe hacer notar que en nuestro caso como no estamos codificando auxiliándonos con la película de MatLab, todos los valores de DC, tienen que ser divididos entre 8 para compensar que no estamos usando dicho formato de película, ya que de no hacerse ésta división no se observa nada en el archivo de salida con formato MPEG al tratar de decodificarlo. Si numeramos los bloques de luminancia del 0 al 3 y los de crominancia como 4 y 5 para el primer macrobloque de la imagen, y del 6 al 9 para luminancias del segundo macrobloque y 10 y 11 las crominancias se verían de la siguiente manera:

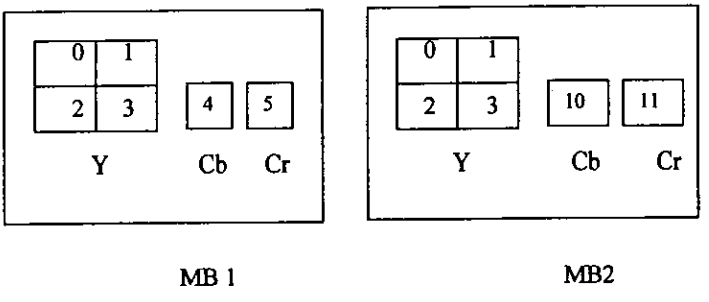


Figura 5.1 Numeración de bloques de luminancia y crominancia

Para hacer las restas de coeficientes de DC, se resta el valor de DC del bloque 0 dividido entre 8 con el valor predicho inicial que es de 128, luego el siguiente valor predicho será el resultado de la división del valor actual de DC entre 8 y así se prosigue, al llegar al bloque 3 no se continúa con el 4, ya que el 4 pertenece a crominancias, sino que el siguiente bloque que se toma es el 6, y se divide el valor de DC entre 8 y se le resta el valor predicho que dio como resultado el último bloque de luminancias del macrobloque 1 que en nuestro caso es el bloque 3, y así se procede hasta el final con las luminancias, lo mismo

sucede para las crominancias, además que se toman por separado las crominancias Cb de las Cr.

Ya que se tienen los valores de DC después de las restas(denominado DPCM) y los de AC almacenados en campos del objeto, lo único que falta es mandar a codificar los valores de DPCM y los de AC, lo cual se hace de manera distinta. Los valores que se encuentran en el campo DPCM se codifican con tablas de Huffman, mientras que los valores de AC se codifican mediante RLE (Longitud de la Corrida de zeros). Esto último que se refiere a codificación de Huffman y RLE se hace desde 'C' ayudándonos de las funciones mex, para poder escribir a nivel de bits, los valores resultantes de la codificación al archivo de salida.

- Estimación de movimiento

Gran parte del codificador se hizo con la extensibilidad en mente. En ningún lugar es más obvia esta forma de programar que en las funciones que ``administran'' los algoritmos de estimación de movimiento. Puede muy bien escribirse una función que reciba un bloque y una ventana de búsqueda y devuelve una matriz con un vector de movimiento por macrobloque para que pueda funcionar con el codificador, con sólo especificar el nombre. La mayor parte de los programas que los implementan pertenecen a la categoría de "block matching'", aunque también proporcionamos una implementación del algoritmo de correlación de fases. Como dijimos arriba, la extensión del mecanismo de estimación es inmediato, la función me_driver se ocupa de aplicar una función que reciba un bloque y una ventana de búsqueda y devuelva una matriz de vectores de movimiento (un vector por macrobloque), para lo cual sólo se necesita pasarle a dicha función el nombre de la implementación alternativa del nuevo algoritmo.

Los métodos de estimación de movimiento fueron programados independientemente y sobre el mismo "molde". Las funciones de estimación reciben, en todos los casos, el bloque que será sujeto de búsqueda y la ventana de búsqueda correspondiente. Generalizando a los valores estándar de MPEG-1, los bloques serán de 16x16 (macrobloques).

Como ya se dijo anteriormente, la determinación del tamaño de la ventana de búsqueda es uno de los factores más delicados en la programación de cualquier método de estimación de movimiento. Algunos de los métodos programados (casi todos los de block-matching) trabajan preferentemente con ventanas de búsqueda de 36x36 pixeles, los métodos jerárquicos, que trabajan en distintos niveles de resolución, reciben ventanas de búsqueda mucho mayores, ya que en el primer paso trabajan con regiones espaciales grandes con baja resolución, y van progresivamente reduciendo las dimensiones espaciales y aumentando la resolución de la imagen. Debido a esto, la decisión fue tomar una ventana inicial de 48x48 pixeles. Con la correlación de fases no hay problema, ya que ésta recibe bloques de igual tamaño para funcionar.

Ya que estamos hablando del método de correlación de fases, sería conveniente aclarar que al momento de programarlo, se tienen que considerar algunos puntos importantes: en primer lugar, el uso de la transformada de Fourier tradicional es reemplazada por su versión computacionalmente eficiente, la transformada discreta rápida de Fourier o DFT bidimensional., lo que trae como consecuencias algunos efectos de frontera, como la aparición de picos en la función de correlación de fases localizados en los bordes de la imagen; éstos causados por objetos que desaparecen en el límite de la misma. También debemos al uso de la DFT bidimensional una pequeña "fuga"[1] de valores en el dominio espectral al aparecer vectores de movimiento con magnitudes no enteras.

Aunado al problema intrínseco de la implementación de los métodos, está el esfuerzo por minimizar el consumo de tiempo en cada uno de ellos, ya que serán módulos utilizado exhaustivamente por el codificador. Se hizo lo posible por vectorizar el código y por registrar los "cuellos de botella" inmersos en cada función. La vectorización, en la mayoría de los casos, se logra introduciendo asignaciones directas con el operador *colon* (:) de Matlab.

Se dijo en párrafos anteriores que la función *me_driver* era la encargada de administrar y coordinar el correcto análisis de los bloques de cada *frame*, haciendo las veces de *wrapper*, o envolvente. Dicha función devuelve el conjunto de vectores de

[1] Dicho efecto de "fuga" es conocido en la literatura como "spectral leakage" [Murat, 1990].

movimiento correspondientes al tamaño completo del *frame*. El siguiente esquema ejemplifica de manera más clara su funcionamiento:

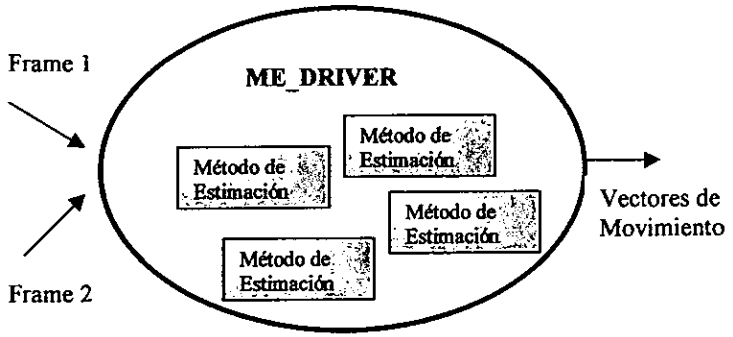


Figura 5.2 Funcionamiento del módulo ME_DRIVER

- Codificación de imágenes P

La codificación de imágenes P, es más compleja que la codificación de imágenes I, ya que la primera requiere de estimación y compensación de movimiento. Para la codificación de imágenes P, lo que se hace es que una vez que se tienen los objetos y que el patrón de codificación indica hacer codificación de una imagen P, lo que se hace es escribir los encabezados de picture y slice que corresponden a una imagen P y mandar llamar al driver de estimación de movimiento correspondiente al algoritmo indicado, ya que hay un driver para estimar movimiento con el algoritmo de correlación de fases, otro para el método jerárquico y otro para los demás métodos. Se tienen éstos tres drivers diferentes, ya que como se dijo anteriormente, el tamaño de la ventana de búsqueda es diferente según el método de estimación que se vaya a utilizar. En el driver de estimación de movimiento, lo que se hace es ir estimando movimiento para cada macrobloque, e inmediatamente mandar a escribir al archivo de salida, el procedimiento que se sigue con cada macrobloque es el siguiente:

Para codificación de imágenes P, se toman dos imágenes, una que es la referencia pasada y otra que es la imagen actual, entonces para codificar cada macrobloque lo que se hace primeramente es verificar si con el movimiento cero, es suficiente para codificar el macrobloque y en caso de ser así, ya no es necesario utilizar el método de estimación de movimiento, lo que se hace para decidir si el movimiento cero es suficiente o no, es tomar el macrobloque correspondiente en la misma posición, tanto del frame actual como de la referencia pasada, las luminancias de estos macrobloques se mandan a una función donde se calcula la mínima diferencia absoluta, y si dicha diferencia no rebasa un umbral que es 256, se decide que el movimiento cero es suficiente, ya que esto indica que los macrobloques son muy parecidos.

En caso de que el movimiento cero no sea suficiente, se calculan las componentes en "x" y "y" del vector de movimiento para ese macrobloque, utilizando para ello los mismos bloques de luminancias que se mandaron a la función que calcula la diferencia mínima absoluta. Una vez obtenidos los vectores de movimiento, lo que se hace es la compensación de movimiento, que consiste en colocarse en la imagen referencia en la misma posición en que se encuentra el macrobloque de la imagen actual, sumarle el vector de movimiento obtenido a la posición actual y recortar un bloque de luminancias de 16x16 en la posición resultante de sumar la posición actual con las componentes en "x" y "y" del vector. Una vez que se recortó este bloque, se manda junto con las luminancias del macrobloque actual, a una función que calcula la diferencia mínima absoluta entre los bloques de luminancias y decide si el bloque obtenido vía la estimación y compensación de movimiento es más parecido al bloque actual o bien se parece más el bloque con movimiento cero al actual, lo cual indica que el movimiento cero es mejor que el movimiento estimado, por lo que si este es el caso, las componentes del vector de movimiento deben regresarse a cero.

Ya que se ha hecho lo anterior, lo que procede es compensar las crominancias del macrobloque utilizando la mitad del valor de las componentes del vector, ya que se sabe que los bloques de crominancias son de 8x8 y no de 16x16 como las luminancias. Una vez que se tienen listos los bloques de la imagen actual y los bloques de movimiento compensado tanto de luminancias como de crominancias, lo que se hace es restar a los macrobloques actuales los macrobloques de movimiento que corresponden a la imagen

referencia, y ésta diferencia que se le denomina error, es lo que se pasa por la DCT, se cuantiza y se recorre en zigzag para después codificarse con el RLE.

Antes de mandar a codificar al RLE, primero se obtienen los valores principal[2] y residual[3] del vector de movimiento, lo que se conoce como codificación del vector, dichos valores principal y residual, dependen directamente del rango de búsqueda de los métodos de estimación, y en caso de que el vector de movimiento exceda el máximo posible permitido para el rango de búsqueda establecido, lo que se hace es sumar el rango de búsqueda al vector y así dejarlo dentro del rango permitido para los vectores de movimiento. Por ejemplo si el rango permitido es -32 a 31 y alguna componente del vector me indica un valor de -34 lo que se hace es sumar el rango de búsqueda que en este caso es 64 y obtenemos ahora una componente de 30, la cual ya cae en el rango permitido y puede ser codificada.

La codificación de vectores de movimiento es diferencial, y es necesario guardar los valores de movimiento del macrobloque actual ya que son necesarios para codificar el vector de movimiento siguiente. Lo que se guarda son las componentes del vector actual, porque para codificar el vector de movimiento siguiente se ocupa un desplazamiento que se obtiene al sumar los valores de las componentes de movimiento actuales con los valores de las componentes anteriores, el valor de las componentes anteriores es igual a cero si se trata del primer macrobloque de la imagen.

La tabla de cuantización para imágenes P es una tabla de cuantización interframe, ya que se utilizan varias imágenes y es igual a un bloque de 8x8 con valor 16 en todos sus elementos, el cual se multiplica por la escala de cuantización que en general es 10. Cuando se cuantiza cada bloque que compone al macrobloque en cuestión, se revisa si todos los elementos del bloque después de la cuantización son cero, en caso de ser así no se incrementa el valor del patrón de codificación de canal, y en caso contrario, dicho valor se incrementa en 32 unidades para el primer bloque de luminancias, en 16 para el segundo, en 8 para el tercero y en 4 para el cuarto, además de que se incrementa en 2 unidades para el bloque de crominancias Cb y en una unidad para el bloque Cr.

---

[2] cociente de la división de la componente del vector entre el fcode(vale 1 si el rango de búsqueda es -16 a 15 y 2 si el rango es -32 a 31)

Una vez que se ha codificado el vector de movimiento se obtuvo el error entre macrobloques, se pasó dicho error por la DCT, se cuantizó y se recorrió en zigzag, tenemos todo listo para mandar estos datos a una función mex que se llama write_macroblok_header, que recibe éstos datos, escribe el header de macrobloque, donde se especifica que se trata de una imagen P, que se usa movimiento hacia adelante con sus respectivos valores principal y residual, y el incremento de macrobloque. Una vez escrito el header se mandan los valores del error entre macrobloques al RLE y se escriben los datos codificados al archivo de salida y se termina con la codificación de macrobloques P.

Para la codificación de macrobloques hay tres formas diferentes de escribir el header, y también podría ni siquiera escribirse dicho encabezado en caso de que el movimiento sea cero y el patrón de codificación de canal también lo sea, en cuyo caso solo se incrementa en uno la dirección del macrobloque y con esto el decodificador entiende que se ha saltado este macrobloque. La única situación en donde no se puede saltar un macrobloque es cuando es el primer ó el último macrobloque de un slice, y en este caso sí se debe codificar dicho macrobloque. Las tres formas de escribir el header son las siguientes:

Una forma de escribir el header es cuando es el primero ó el último macrobloque del slice, en cuyo caso no se escribe nada referente al movimiento, ni al patrón de codificación de canal.

La segunda forma de escribir el header es cuando se tiene un vector de movimiento igual a cero pero el patrón de codificación de canal es diferente de cero, en este caso sí se escribe el patrón de codificación en el header de macrobloque, pero tampoco se escribe nada referente al vector de movimiento.

La tercera forma de escribir el header es la más completa y la que involucra más información, ya que es cuando no se trata ni del primer ni del último macrobloque del slice, el movimiento es diferente de cero y el patrón de codificación de canal también lo es, en cuyo caso se escribe en el header el valor del patrón de codificación de canal y todos los datos referentes al vector de movimiento.

[3] residuo de la división de la componente del vector entre el fcode.

- Codificación de imágenes B

La codificación de imágenes B tiene mucha semejanza con la codificación de imágenes P y solo varía en lo siguiente:

Para este tipo de codificación también se requiere estimación y compensación de movimiento, pero se usan otros drivers de estimación, ya que aquí, se utilizan tres imágenes, una referencia pasada, una referencia futura y la imagen actual, al igual que para las imágenes P, se tienen tres drivers distintos según el algoritmo de estimación, ya que varía el tamaño de la ventana de búsqueda.

Ya que se han escrito los headers de picture y slice, y se ha llamado al driver de estimación, aquí lo que se hace diferente a las imágenes P es que no se verifica si el movimiento cero es suficiente, sino que se tiene una variable que guarda si el movimiento del macrobloque anterior es hacia adelante, hacia atrás o es movimiento interpolado, ya que estas tres son las formas de movimiento que se pueden elegir para imágenes B. Al principio de cada imagen, ésta variable se inicializa con movimiento hacia adelante, y lo que se hace en lugar de ver si el movimiento cero es suficiente es determinar si el movimiento anterior es suficiente; aquí también al iniciar una imagen se ponen los movimientos anteriores en cero. Aquí se llama a una función que lo que hace es determinar si el macrobloque compensado[4] con el movimiento anterior al calcularle la diferencia mínima absoluta en relación con el macrobloque actual, no se rebasa el umbral que es 512, de ser así se decide que el movimiento anterior es suficiente, y se incrementa la dirección del macrobloque en una unidad y se pasa al siguiente macrobloque sin estimar movimiento ni escribir nada al archivo. En caso de que el movimiento anterior no sea suficiente, se calculan los vectores de movimiento hacia adelante y hacia atrás, se compensan y recortan en la imagen que corresponde (imagen de referencia pasada para movimiento hacia adelante e imagen de referencia futura para movimiento hacia atrás), después de esto se determina cuál de las tres opciones de movimiento es más parecida con el macrobloque de la imagen actual, para lo cual se calcula la diferencia mínima absoluta entre el macrobloque compensado hacia atrás con el actual, el macrobloque de movimiento compensado hacia adelante también con el

[4] La compensación de movimiento para imágenes B es igual a la compensación de movimiento que se explicó para imágenes P.

actual y el macrobloque de movimiento interpolado y el macrobloque actual. El macrobloque de movimiento interpolado se obtiene al promediar los valores de los macrobloques de movimiento compensado hacia adelante y hacia atrás, una vez obtenidas las tres diferencias mínimas absolutas, se escoge la menor de ellas y se indica en la variable mode, qué tipo de movimiento se va a utilizar para codificar el macrobloque en cuestión. En caso de que se determine que el movimiento hacia adelante es mejor, se procede igual que en las imágenes P, solo que la escala de cuantización utilizada para este tipo de imágenes es en general 25, y ya no se utiliza el vector de movimiento hacia atrás, si resulta que el movimiento hacia atrás es mejor, el vector de movimiento de movimiento hacia adelante ya no se necesita y solo se trabaja con el vector hacia atrás y la referencia futura, además de que se indica en el header que solo se está usando movimiento hacia atrás, ya que en dicho header se especifica si el movimiento es hacia adelante poniendo en la variable motionf un uno y en la variable motionb un cero, si el movimiento es hacia atrás se ponen los valores anteriores negados, y si se determina que el movimiento interpolado es mejor, se especifica poniendo un uno en cada una de las variables motionf y motionb.

Una observación más que cabe señalar es que para este tipo de imagen, también se codifican los vectores de forma diferencial, pero, se tienen en variables diferentes los movimientos anteriores hacia atrás, y hacia adelante, lo cual quiere decir que si el primer macrobloque de una imagen b se codificó como movimiento hacia adelante, el siguiente como movimiento hacia atrás y el tercero nuevamente como movimiento hacia adelante, el movimiento anterior a utilizar es el del anterior macrobloque que se haya codificado como movimiento hacia adelante, es decir, con los valores del primer macrobloque de la imagen.

Para este tipo de imágenes, también lo que se codifica de cada macrobloque aparte del vector de movimiento es la diferencia entre el macrobloque actual y el de movimiento compensado, es decir, el error es el que se codifica. La codificación de macrobloques es igual que en las imágenes P y también se codifican con el mismo RLE que dichas imágenes. La codificación del vector de movimiento se hace igual que para las imágenes P, solo cambia lo mencionado en el párrafo anterior.

Para imágenes B solo hay un tipo de header de macrobloque y únicamente varía si se trata de movimiento hacia adelante, hacia atrás ó interpolado.

- Interfaz al usuario (UI)

La Interfaz: Como sabemos todos, una interfaz es un dispositivo que permite al usuario interactuar de forma amigable, sencilla y total con un sistema. En programas como el nuestro, la interfaz es la capa más externa, y cumple con las funciones de enlace con el usuario. El diseño de la misma es relativamente sencillo, fue llevada a cabo en el lenguaje MatLab, donde existen funciones de manejo de objetos visuales como cajas de diálogo, botones de acción, menús *pop-up*, etc. que hacen más sencilla la programación de la misma (relacionando objetos visuales con sus atributos y con eventos). Decidimos sólo tomar en cuenta para el diseño de la interfaz los parámetros más relevantes al funcionamiento del codificador. A continuación se muestra la pantalla de interfaz:

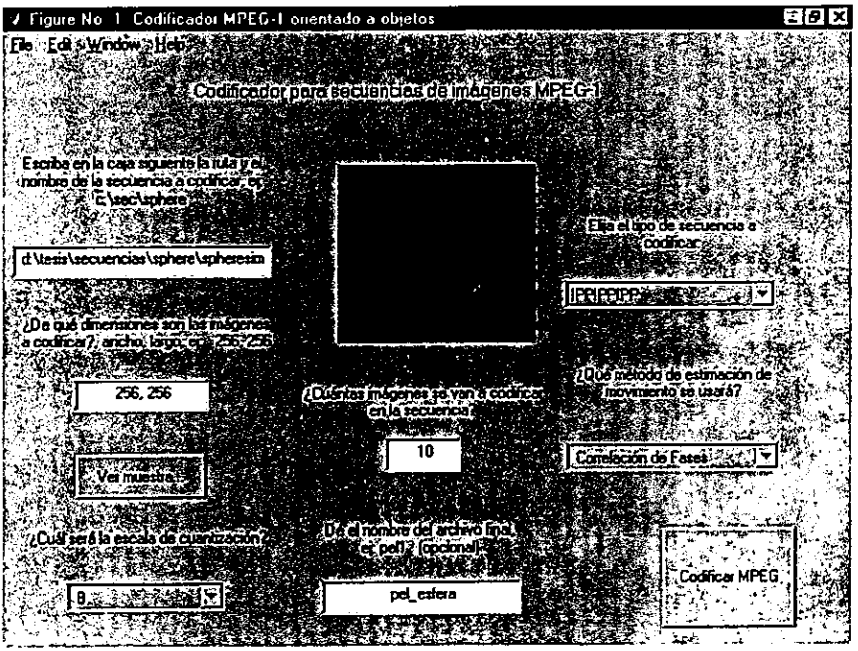


Figura 5.3 Pantalla de la interfaz de usuario

En ella se pueden notar las cajas de diálogo y los menús *pop-up* en los que el usuario interviene para indicar los parámetros deseados al momento de generar el archivo .MPG. La primera caja de diálogo (arriba, izquierda) recibe la ruta donde se halla la secuencia que se va a codificar, la segunda caja de diálogo acepta las dimensiones de las imágenes que intervienen en la secuencia[5]. Con el botón que se encuentra justo debajo de esta caja (botón ver muestra) aparecerá en la parte central de la pantalla la imagen muestra de una de las pertenecientes a la secuencia en cuestión, generalmente la primera de ellas. Debajo del botón de muestra se encuentra un menú *pop-up* que permite elegir la escala de cuantización a la que se va someter la película. Los valores que aparecen allí son los valores *default* del estándar MPEG. Posteriormente se pregunta cuántas imágenes se codificarán en la secuencia; esto asume que las imágenes de la secuencia que se hallen dentro del mismo directorio cumplen con una cierta sintaxis, dada por:

nombre_generico,número

como en las imágenes spheresim1, spheresim2,... spheresim20

En la parte baja central de la pantalla se pide al usuario que teclee un nombre de archivo final, es decir, el archivo producto de la codificación que llevará la extensión .MPG y que será decodificable con un decodificador estándar MPEG-1 que reciba la capa de video. El siguiente menú *pop-up* (arriba, derecha) permite seleccionar la clase de secuencia que se desea obtener en términos de los tipos de imágenes involucradas (imágenes I, P, etc.). Recordemos que de esto depende fuertemente el tiempo de codificación necesario, la compresión y el resultado cualitativo al que se llegará en el archivo .MPG final. Otro parámetro probablemente más importante es la elección del método de estimación de movimiento. En este menú pop-up se dan a escoger los métodos que han sido programados para esta tesis. Hay que recordar que los métodos de estimación de movimiento varían mucho en su funcionamiento intrínseco, en el tiempo de cálculo y en los resultados finales. El último botón, como su etiqueta lo dice, codifica la película deseada con los parámetros que han sido especificados en la interfaz.

[5] Es obvio decir que todas las imágenes de una secuencia que se hallen en el mismo directorio deberán tener las mismas dimensiones.

Herramientas utilizadas:

Durante el desarrollo de este proyecto, echamos mano de un conjunto de herramientas que por su utilidad y efectividad, vale la pena documentar aquí. Es necesario aclarar que no se trata aquí de describirlas con toda amplitud, y que cada una de estas herramientas se encuentra descrita en algún detalle en el manual de MatLab, sino describir el uso que les dimos en el curso del presente proyecto. Asimismo, fue necesario desarrollar pequeños programas cuya utilidad y generalidad trasciende los límites del presente programa, por lo que los describiremos aquí en beneficio de quien se anime a usarlos.

PROFILER.-Permite obtener una idea de la eficiencia relativa de distintas partes de un programa. Este tipo de información es muy útil para mejorar el rendimiento de alguna función de la que hace uso el programa en cuestión, ya sea mediante su reescritura en C, o la vectorización de su código (la técnica al parecer más socorrida entre los programadores de MatLab).

DEBUGGER.-El debugger (depurador) de MatLab permite fijar puntos de pausa (breakpoints), donde se interrumpe la ejecución del programa en cuestión y se devuelve el control al intérprete, sin perder el contexto (conocido en MatLab como espacio de trabajo, o workspace) ni la información en la pila, que permite establecer el flujo de invocaciones a la función en cuestión. Dichos puntos pueden establecerse explícitamente (indicando el número de línea), o en respuesta a ciertos eventos (errores o advertencias). La versión 5.2 ofrece además la particularidad de desplegar en una pequeña ventana el valor de todas las variables en el espacio de trabajo (contexto) particular de la llamada, si se tiene abierto el editor.

- Comunicación MatLab-C

MatLab es un lenguaje extensible: además de los scripts en el propio lenguaje, puede hacerse uso de una interfaz de programación (API), que permite tanto manejar el ambiente MatLab desde un programa en C como utilizar C como ambiente de desarrollo para funciones del ".m", indistinguibles de estas últimas.

Fueron este último tipo de funciones de las que hicimos uso. Para ello, basta escribir una función llamada mexFunction, que es el punto de entrada (gateway) al programa. Esta función recibe cuatro parámetros, que representan uno y otro lado de la asignación con matrices (que, como hemos dicho, hasta recientemente eran el único tipo de datos disponible en MatLab). Una vez que se tiene una función de este tipo, debe compilársele como biblioteca compartida (DLL en Windows). No es necesario escribir los archivos de opciones y recursos que permitirán al compilador efectuar esta operación. El script mex, provisto de manera estándar con MatLab permite hacer uso de un archivo

La manipulación de matrices dentro del programa en C no es trivial, pero si muy sencilla. Se hace uso extensivo de estructuras del tipo mxArray, y de un conjunto de funciones (que constituyen la API) para manipularlas. El proceso es muy similar cuando se trata de manipular celdas, estructuras u objetos de MatLab.

El hecho de que MatLab llame a las funciones mex en el mismo espacio facilita la comunicación entre procesos. No es necesario hacer uso de archivos temporales, semáforos u otras técnicas de IPC específicas a un sistema operativo. Basta pasar los argumentos a una función.

- Funciones auxiliares

Se escribieron rutinas que leen imágenes indizadas (indicadas) estándares de MatLab y hacen la conversión al espacio luminancia-crominancia. Esto era necesario cuando contábamos únicamente con la versión 5.1 de Matlab. Con la versión 5.2 de MatLab se incluyó una función de resultados equivalentes, ligeramente más rápida por lo que ya no resultó muy útil la función que implementamos.

Hoy en día están en boga los generadores de documentación. Del ambicioso sistema WEB de Donald Knuth, que generó un estilo de programación conocido como ``programación literaria" a modernos generadores de documentación y referencias cruzadas, como el javadoc, incluido en las herramientas de desarrollo de Java hay muchas diferencia, pero éstos pueden considerarse un paso en la dirección correcta hacia aquél. Para MatLab existe un generador semejante a javadoc escrito por Jeffrey Kantor, que

nosotros utilizamos con la finalidad de generar documentación escrita con cierto formato, y hacer referencias cruzadas de diferentes módulos de nuestro programa. Para poder aprovecharlo también en la impresión de código fuente en nuestra tesis, hubo necesidad de modificarlo algún tanto, para posibilitar el uso de hojas de estilos, tan en boga en nuestros días. Dichas modificaciones permiten tener control en el formato de algunos niveles del código: palabras reservadas, comentarios, comentarios descriptores (los que aparecen entre la palabra function y la primera línea de código), constantes interconstruidas y cadenas. Dicho control se puede ejercer de dos maneras: bien modificando el archivo de hoja de estilos provisto (default.css), o bien utilizando el modificador -s añadido a mat2html para establecer la nueva hoja de estilos.

También se escribió una interfaz minúscula a la biblioteca regex de GNU, un casador de expresiones regulares. Las expresiones regulares representan un pequeño lenguaje para describir cadenas. Son ampliamente utilizadas en y por multitud de aplicaciones y utilidades. A pesar de esto, la sintaxis de este lenguaje no es estándar ni con mucho común, y muchas herramientas implementan su propia variante. De éstas, la más común es la puesta en boga por el lenguaje de programación Perl (con mucho la más completa).

Regex provee la misma funcionalidad básica, pero una sintaxis ligeramente distinta. En nuestra interfaz, debe pasársele a la función match tres argumentos, la cadena a analizar, una expresión regular y 1 o 0 dependiendo si se quiere o no que se tome en cuenta la condición de mayúsculas o minúsculas en la prueba. Como puede verse, esto no es sino la punta del iceberg en lo que puede hacer este paquete de expresiones regulares, pero nos sirvió para algunas decisiones que hubieran sido, en el mejor de los casos, tediosas de escribir si se opta por el lenguaje estándar

VI. Conclusiones y resumen

En este capítulo presentamos los resultados de diversas evaluaciones al codificador objeto de esta tesis. Algunas de éstas reflejan la calidad relativa de los estimadores de movimiento, mientras que otras se abocan a la implementación en particular.

Aparte de las gráficas comparativas que se presentan abajo, conviene recordar aquí algunos puntos de la implementación que hacen de éste un trabajo razonablemente original. Matlab no está diseñado específicamente para sistemas grandes, sino para pequeños programas en un dominio específico. Mucho de los filtros que ofrecen diversos toolkits (por ejemplo, los lectores de imágenes en formatos JPEG, GIF y TIFF incluidos en el toolkit de procesamiento digital de imágenes) están desarrollados completamente en C, que, como lenguaje de menor nivel (entendiéndose por esto que está "más cerca" de la máquina) facilita la programación de tales proyectos. Esta facilidad, tristemente, no se extiende a la tarea de darle mantenimiento a los programas. Es muy sencillo perderse en los detalles de código escrito en C, a menos que el diseño del programa en cuestión proporcione capas de abstracción claramente definidas, y en ocasiones ni de esta forma es posible.

La facilidad de desarrollo no es la única ventaja de C,: la velocidad del programa resultante se ve favorablemente impactada, pues, aunque no se use un compilador especialmente adecuado a la generación de código en la arquitectura objetivo, se elimina la carga del paso de interpretación a la ejecución. Dicho esto, conviene recordar que los objetivos del programa que aquí presentamos son otros, empezando por la facilidad de mantenimiento y el diseño sencillo que permita reutilizar la mayor parte del código en proyectos semejantes.

MPEG es un estándar que evoluciona, o que se adapta a las cambiantes necesidades de un público creciente. MPEG-1 se ocupa de aplicaciones de video adecuadas a ser almacenadas en medios ópticos o magneto-ópticos, donde el ancho de banda es muy grande

porque la transmisión es a través de canales exclusivos y las distancias son mínimas. MPEG-2 propone una solución a ese problema de transmisión, y por lo tanto contempla aplicaciones con anchos de banda menos abundantes. MPEG-4 (publicado en noviembre de 1998) está más enfocado a aplicaciones, pues su principal preocupación es la edición, la manipulación de la película y artefactos en la misma, así como la navegación eficiente. Por último MPEG-7, todavía en estudio, pretende lograr una interfaz que permita la búsqueda de figuras en una películas, proporcionando un análisis hasta cierto punto semántico de las películas.

Los problemas que cada una de estas encarnaciones del estándar enfrentan refleja el estado y las necesidades de las personas inmersas en estos campos. Se considera que la importancia del problema de la transmisión se ve en buena parte disminuido por la existencia de mecanismos de transporte mucho menos restrictivos (transmisión por compañías de cable, satélites de órbita baja y arquitecturas de redes como ATM), pero no deja de ser importante. Es probable que la necesidad de compresión de datos deje de ser de importancia apremiante en un futuro no lejano, pero es casi seguro que esa importancia nunca sea tan poca que el problema pueda ignorarse. Es notable el aumento de la producción de información en distintos formatos que se ha venido dando desde la introducción de Internet a las posibilidades (económicas e intelectuales) de cada vez mayor número de personas, por lo que la idea de que estas nuevas tecnologías de transporte de alta velocidad no sean capaces por sí solas de solucionar todas las consecuencias de la enorme exigencia sobre las redes es una hipótesis razonable.

Asimismo, las técnicas de compresión con pérdida han demostrado ser una alternativa viable a las anteriores, en términos de calidad de la salida. Es eminentemente ingenieril la convicción de que no vale la pena transmitir lo que el hombre no puede percibir, si esta operación descarga elementos del sistema que corren el riesgo de convertirse en cuellos de botella.

Una idea que se ha ido presentando con cada vez mayor claridad a lo largo del desarrollo de este trabajo, y que no puede ser denominado conclusión, pues en justicia es más una percepción, es la naturalidad con que un esquema sintáctico se modela en el paradigma orientado a objetos. Esta noción ha sido muy estudiada (sobre todo en textos de desarrollo de compiladores), pero creemos que su importancia no puede ser ignorada. En nuestro trabajo, fue más deseo de usar las nuevas facilidades de Matlab (específicamente los objetos) lo que nos llevó a aplicar este estilo de diseño que la consciencia de lo poderoso de la idea arriba expuesta.

VI.1 Una nota final acerca del lenguaje Matlab.

Este trabajo sirvió también para evaluar las posibilidades del sistema Matlab y su lenguaje en el desarrollo de programas significativamente más grandes y gravosos en los recursos de la máquina que los que habíamos tenido contacto hasta ahora. En retrospectiva, las dificultades que presenta el lenguaje para hacer uso de recursos muy socorridos pero probablemente no muy favorables al programa final resultan en una codificación más limpia y mantenible. Asimismo, es innegable la naturalidad con la que conceptos necesarios para el manejo de imágenes se expresan en el paradigma de matrices de Matlab. Probablemente la introducción de rutinas en C al paso de escritura al archivo podría haberse alejado en alguna medida, con la consecuente degradación en el desempeño. A juzgar por la cantidad y la complejidad de la codificación de los programas que nos sirvieron de documentación, la decisión de implementar más algoritmos de estimación en C para incluirlos en dicho programa habría resultado en un retraso significativo en los tiempos de desarrollo.

VI.2 Resultados.

Hemos identificado un número significativo de variables en la configuración de cualquier codificador. En nuestra implementación, algunas de estas son accesibles a través de la interfaz de usuario, o bien manipulando la llamada a la función mpeg. Es sobre éstas

donde actúan nuestros programas de prueba, variando un parámetro a la vez y graficando los resultados.

Los resultados arrojados por dichos programas dependen en gran medida en la arquitectura y configuración del equipo en el que se les corrió, por lo que se proporciona esa información.

En lo que respecta al análisis de la eficiencia relativa de los algoritmos de estimación de movimiento, aceptamos que una prueba estilo "benchmark" no es ni con mucho la más informativa, no digamos ya rigurosa. Sin embargo, y al estar probados en lo que suponemos son igualdad de condiciones, creemos que la idea que proporcionan las cifras con respecto al desempeño relativo (por más burdas que puedan ser), no se alejarán mucho de los resultados de un análisis formal.

Nuestras secuencias de prueba en este análisis son las llamadas "stefan" y "sphere". La secuencia "stefan" está en formato NTSC (352 x 240 pixeles) y escala de grises en un rango de 0 a 256. Cada una de sus imágenes mide por lo tanto 84,480 [bytes]. Esta secuencia muestra a un tenista (Stefan Edberg) llevando a cabo una jugada en un partido; hay gran variedad de texturas y movimiento general de la escena. La secuencia denominada "sphere" es de tipo sintético (fue diseñada con una computadora) y mide 256 x 256 pixeles. Su escala de grises es también de 0 a 256, por lo que cada una de sus imágenes es de 65,536 [bytes]. Esta secuencia muestra dos esferas con iluminación simulada cuya única variación es en el tamaño, aumentando y disminuyendo el mismo de forma regular.

La primera gráfica ilustra la eficiencia relativa en la compresión lograda por los diversos métodos de estimación en tres imágenes de la secuencia "stefan".

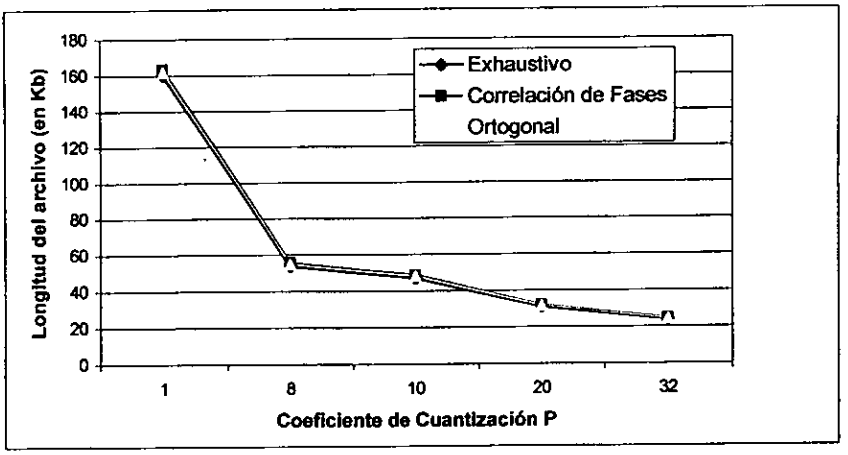


Figura 6.1 Gráfica de cuantización contra longitud de archivo. Secuencia "stefan"

Como puede verse, la diferencia entre algoritmos de estimación no es tan significativa como la variación del coeficiente de cuantización en las imágenes P, con la consecuente degradación en la calidad de la imagen. Conviene ilustrar dicha degradación, y para ello presentaremos sus efectos en una sola imagen "I" de la secuencia:

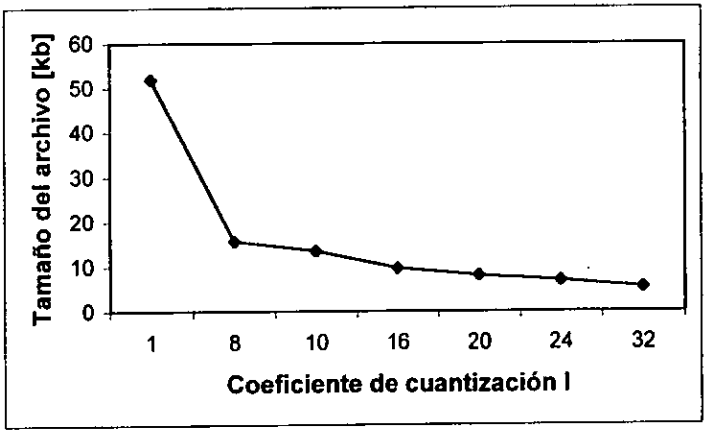


Figura 6.2 Gráfica de coeficiente de cuantización "I" contra tamaño de archivo. "stefan"

En las imágenes de la siguiente página se debe observar el efecto que la cuantización imprime en la codificación de la misma; una cuantización muy severa como la que se logra con un coeficiente de 32 (el más alto utilizado) da lugar a una imagen demasiado vaga, en la que resisten solamente los coeficientes de DC y ocasionalmente algún coeficiente de AC que haya sido notablemente importante en cuanto a su magnitud. Por el otro lado, cuantizaciones de baja severidad como la que se logra con un coeficiente de 8 dejan una imagen de una calidad muy buena, pero con una compresión mucho menor que con el coeficiente de cuantización 32.

La subsecuente degradación se ilustra en las siguientes imágenes para la secuencia "stefan" (con coeficientes de cuantización de 8, 16 y 32, respectivamente):

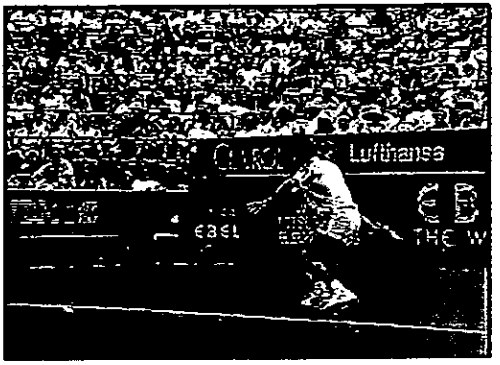


Figura 6.3 Imagen "I" de la secuencia "stefan" cuantizada con un coeficiente 8

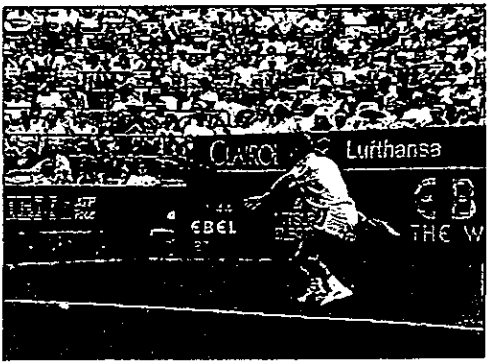


Figura 6.4 Imagen "I" de la secuencia "stefan" cuantizada con un coeficiente 16

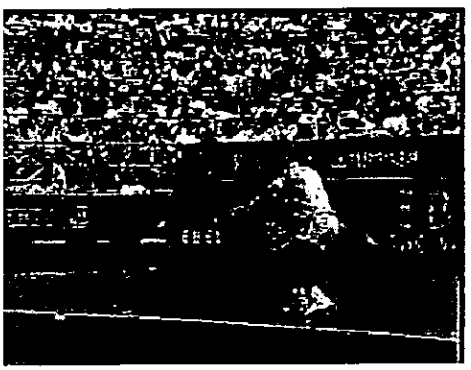


Figura 6.5 Imagen "I" de la secuencia "stefan" cuantizada con un coeficiente 32

Observemos a continuación los mismos efectos de cuantización aplicados en nuestra otra secuencia de trabajo, la secuencia "sphere".

Figura 6.6 Imagen "I" de la secuencia "sphere" cuantizada con un coeficiente 8

Figura 6.7 Imagen "I" de la secuencia "sphere" cuantizada con un coeficiente 16

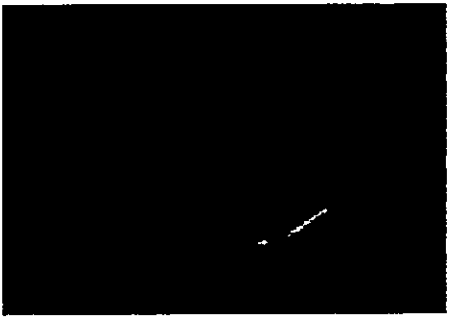


Figura 6.8 Imagen "I" de la secuencia "sphere" cuantizada con un coeficiente 32

Siguiendo en el tema de la cuantización de las imágenes de la secuencia, podemos hablar un poco de calidad de video. Hemos adoptado dos maneras de determinar la calidad de una película MPEG. La primera de ellas es subjetiva y está basada en la apreciación por parte del usuario del producto final, es puramente individual e involucra aspectos complicados de cuantificar dentro de los *frames* de la película. De forma coloquial, se dice que una imagen está más "bonita" o de mayor calidad que otra. El otro enfoque para determinar la calidad de la película es el objetivo, el cuantificable, y para ello hemos decidido utilizar una función SNR[1], programada en Matlab. Cabe aclarar que los resultados que generan los dos métodos pueden ser distintos. En algunas ocasiones factores como la nitidez provocan que una imagen sea mejor vista que otra, a pesar de que su relación señal a ruido con la original sea más baja que la otra.

Hemos mandado como parámetros a dicha función un cuadro original de la secuencia de video y un *frame* perteneciente a una imagen "I" que corresponde al cuadro original mencionado, aunque ya codificado. La función devuelve la razón señal a ruido en decibeles. Los resultados de la aplicación de la función en frames de la secuencia "sphere" arrojaron los siguientes resultados: (se utilizó la función en varias parejas de cuadros y se promedió el SNR final)

Escala de Cuantización	Relación Señal a Ruido (SNR) [Db]
8	45.2954
32	39.4256

Tabla 6.1 Relación señal a ruido para distintas cuantizaciones, secuencia "sphere"

Los resultados obtenidos indican que la calidad de los cuadros "I" que conforman la película es bastante buena en la cuantización con coeficiente 8 (más de 45 [Db]) y muy aceptable en la cuantización con coeficiente 32, ya que éste valor rondaba los 40 [Db]. Si a esto aunamos que la compresión lograda con cuantizaciones tan altas es muy grande,

[1] SNR se debe a las siglas en inglés de "Signal to Noise Ratio", osea, razón señal a ruido.

tenemos una herramienta muy poderosa de compresión. La razón por la que no se aplicó este análisis a imágenes P y B radica en que no existe una imagen original contra la cual compararlas, ya que dichas imágenes, como sabemos, son predichas.

Toca el turno a la secuencia "stefan". La tabla siguiente indica los resultados obtenidos al aplicar la función SNR a cuantizaciones logradas con coeficientes 8 y 32 en una imagen "I" original de la secuencia.

Escala de Cuantización	Relación Señal a Ruido (SNR) [Db]
8	35.7550
32	27.4891

Tabla 6.2 Relación señal a ruido para distintas cuantizaciones, secuencia "stefan"

Como era de esperarse, la calidad objetiva en la secuencia "stefan" disminuye con respecto a la esfera. El valor para la cuantización con 8 está por encima de los 35 [Db], lo cual hace a una imagen medianamente buena, y la cuantización con 32 deja una imagen regular que queda inclusive por debajo de los 30 [Db]; aunque si a calidad subjetiva nos vamos, consideramos que ambas imágenes cumplen con el requisito al que las hemos sujetado, sobre todo la primera de ellas, que consideramos que conserva en gran parte la nitidez original.

Integremos a la discusión de resultados a los métodos de estimación de movimiento, parte medular del presente trabajo de tesis. A continuación se muestran resultados de distintas medidas del desempeño de los métodos y su relación con otros indicadores importantes de la codificación. Las siguientes gráficas ilustran la efectividad de los algoritmos de estimación cuando se utilizan en secuencias con más cuadros, incluyendo imágenes B. Asimismo, se observa el resultado de aplicar diferentes patrones a una secuencia constante (en la primera gráfica el patrón es IBBP, mientras en la segunda es IBP).

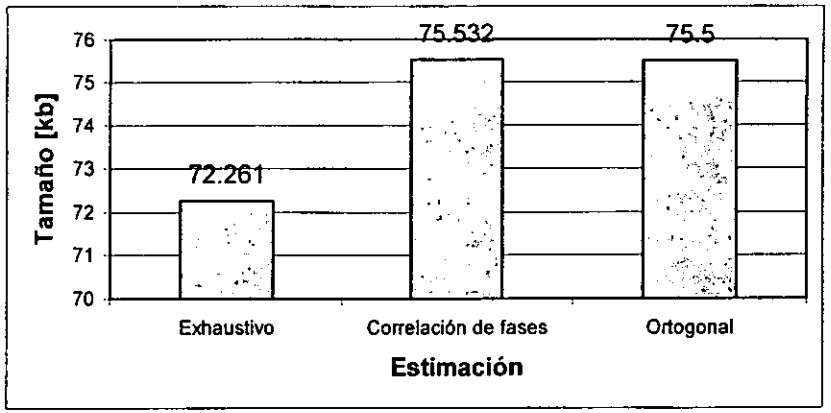


Figura 6.9 Comparación de algunos métodos de estimación de movimiento y tamaño del archivo con una secuencia IBBP

Como se verá a continuación, la influencia del tipo de secuencia a codificar es determinante en el tamaño final del archivo, y la elección del método de estimación de movimiento varía también, pero en menor medida, la longitud del mismo.

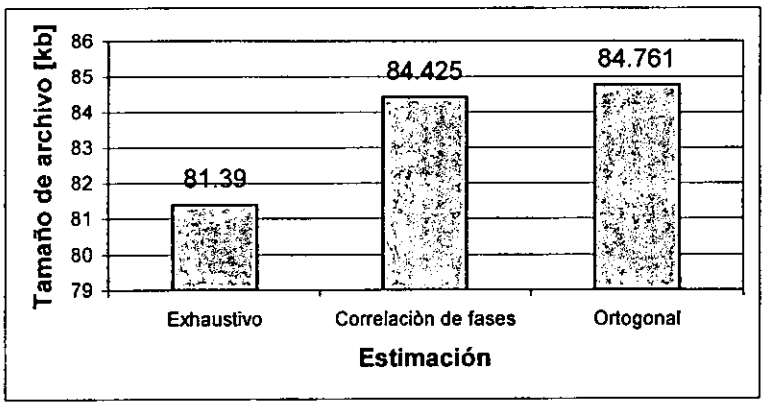


Figura 6.10 Comparación de algunos métodos de estimación de movimiento y tamaño del archivo, secuencia IBP

Hablemos un poco de tiempos de codificación y tamaños de archivo. Sin duda son dos aspectos muy importantes en el desempeño de un codificador de este tipo, y aunque son directamente dependientes de la arquitectura en la que se codifique la película, sus medidas relativas se mantienen para efectos de comparación.

La arquitectura en la que se llevaron a cabo dichas pruebas es la de una computadora personal con procesador Pentium Celeron a 266 MHz, 32 Mb en memoria RAM y 512 Kb de memoria Caché. La versión, como se sabe, del ambiente de programación MatLab utilizada es la 5.2

A continuación se muestran dos tablas en donde se anota una comparación entre los diversos algoritmos de estimación de movimiento. Para llenar estas tablas se utilizó el codificador mpeg-1 presentado en esta tesis con los valores de *default* para la cuantización (8, 10 y 25 para imágenes I, P y B respectivamente), se usó la secuencia de "stefan" con 9 imágenes y el patrón de codificación IBP; ambas tablas se encuentran ordenadas ascendentemente, la primera de ellas tiene acomodados los algoritmos de acuerdo al tamaño final del archivo que contiene la película (archivo) mpeg. La segunda muestra los algoritmos ordenados de acuerdo al tiempo que tarda en codificarse la secuencia usando el método de estimación dado:

Algoritmo de estimación	**Longitud del archivo (en Kb)**	**Tiempo de codificación (en min)**
Correlación de fases	109.667	5.45
Búsqueda en cruz	110.241	13.14
Exhaustivo	111.716	11.45
Dirección conjugada	111.848	10.58
Ortogonal	114.99	5.25
Tres pasos	116.77	10.92
Jerárquico	120.614	33.3

Tabla 6.2 Comparación entre algoritmos de estimación de movimiento, tamaños de archivo y tiempos de codificación, ordenada ascendentemente por longitud final del archivo codificado

Algoritmo de estimación	Longitud del archivo (en Kb)	Tiempo de codificación (en min)
Ortogonal	114.99	5.25
Correlación de fases	109.667	5.45
Dirección conjugada	111.848	10.58
Tres pasos	116.77	10.92
Exhaustivo	111.716	11.45
Búsqueda en cruz	110.241	13.14
Jerárquico	120.614	33.3

Tabla 6.3 Comparación entre algoritmos de estimación de movimiento, tamaños de archivo y tiempos de codificación, ordenada ascendentemente por tiempo de codificación

Las gráficas correspondientes a las tablas anteriores se muestran a continuación, para dar una estimación gráfica más clara de la comparación.

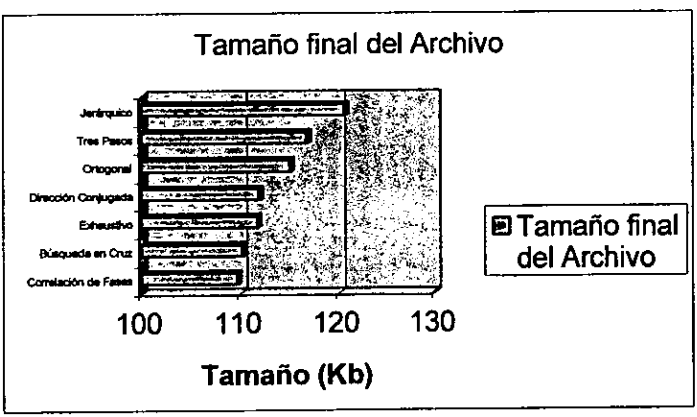


Figura 6.11 Gráfica comparativa entre métodos de estimación y tamaño del archivo

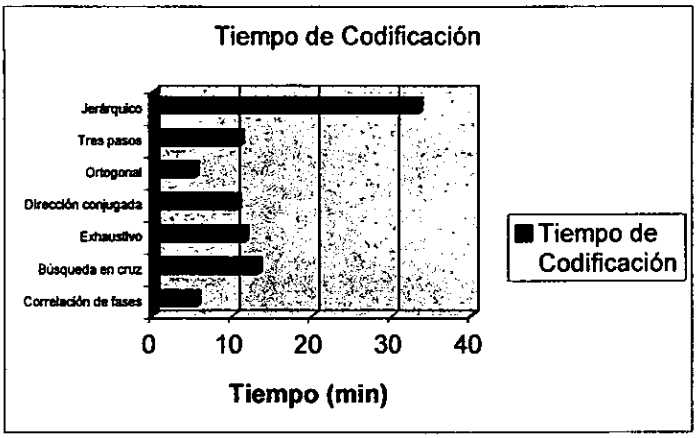


Figura 6.12 Gráfica comparativa entre métodos de estimación y tiempo de codificación del archivo

De lo anterior se puede concluir que el algoritmo de correlación de fases es un buen estimador, aunque también se ha comprobado que no es muy bueno para codificar secuencias que contengan muchas imágenes. Asimismo, podemos constatar la eficiencia de compresión del método más pesado computacionalmente, el exhaustivo, que resultó menos largo en tiempo de codificación que otros y dio lugar a un archivo no muy grande. Un punto a su favor es que se adecua a la codificación de largas películas debido a que hay poca pérdida de calidad en las imágenes B y P gracias a la correcta estimación de movimiento. El llamado método jerárquico, que trabaja en múltiples niveles de resolución, es el que más tiempo máquina consumió. También genera curiosamente el archivo con mayor longitud, por lo que es el método menos adecuado para esta secuencia. Lo anterior lo atribuimos al trabajo pesado que lleva a cabo el método jerárquico en las búsquedas coincidentes en las ventanas multiresolución. Hay que considerar que los resultados presentados aquí, varían con el tipo de secuencia que se desea codificar. La secuencia "Stefan" es una secuencia que es considerada como compleja, debido a la tremenda variedad de texturas y a los movimientos rápidos que presenta la cámara que lleva a cabo el seguimiento al tenista.

Dichas secuencias han resultado en la codificación de archivos más grandes y más lenta que aquellas que tienen poco movimiento entre tramas y texturas poco complicadas.

A continuación se lleva a cabo un análisis similar al visto con "stefan" utilizando nuestra segunda secuencia de trabajo, poco después mostraremos convenientemente resultados comparativos entre ambas secuencias. Para llenar las tablas que se muestran a continuación se utilizó el codificador MPEG-1 con los valores de default para la cuantización (8, 10 y 25 para imágenes I, P y B respectivamente), se usó la secuencia "sphere" con 10 imágenes y el patrón de codificación IBP, ambas tablas se encuentran ordenadas en orden ascendente, la primera de ellas tiene ordenados los algoritmos de acuerdo al tamaño final del archivo que contiene la película mpeg, la segunda, muestra los algoritmos ordenados de acuerdo al tiempo que tarda en codificarse la secuencia usando ese método de estimación.

Algoritmo de estimación	**Longitud del archivo [en Kb]**	**Tiempo de codificación [min]**
Correlación de fases	9.224	5.5952
Dirección conjugada	9.934	5.0513
Ortogonal	10.572	5.0980
Exhaustivo	11.096	7.6997
Búsqueda en cruz	11.235	7.7930

Tabla 6.4 Los métodos de estimación de movimiento, la longitud final de archivo y el tiempo de codificación para la secuencia "sphere", ordenados por longitud creciente.

Algoritmo de estimación	**Longitud del archivo [en Kb]**	**Tiempo de codificación [min]**
Dirección conjugada	9.934	5.0513
Ortogonal	10.572	5.0980
Correlación de fases	9.224	5.5952
Exhaustivo	11.096	7.6997

Búsqueda en cruz	11.235	7.7930

Tabla 6.5 Los métodos de estimación de movimiento, la longitud final de archivo y el tiempo de codificación para la secuencia "sphere", ordenados por tiempo creciente

Para facilitar la visualización relativa de los datos que presentamos arriba, mostraremos a continuación un par de gráficas obtenidas con dichos valores.

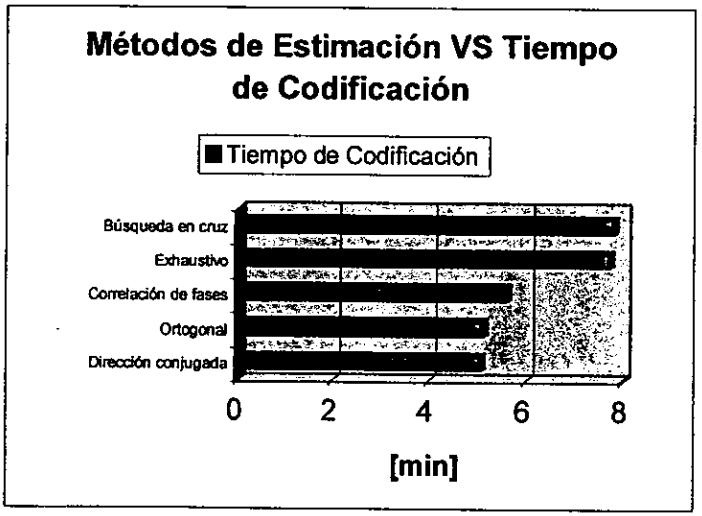


Figura 6.13 Gráfica comparativa de Métodos de Estimación y Tiempo (secuencia "sphere")

Recordamos que en las gráficas solamente aparecen algunos de los métodos de estimación de movimiento, debido a que la inclusión del de los tres pasaos y del jerárquico alterarían las escala original a la que deseamos mostrarlas.

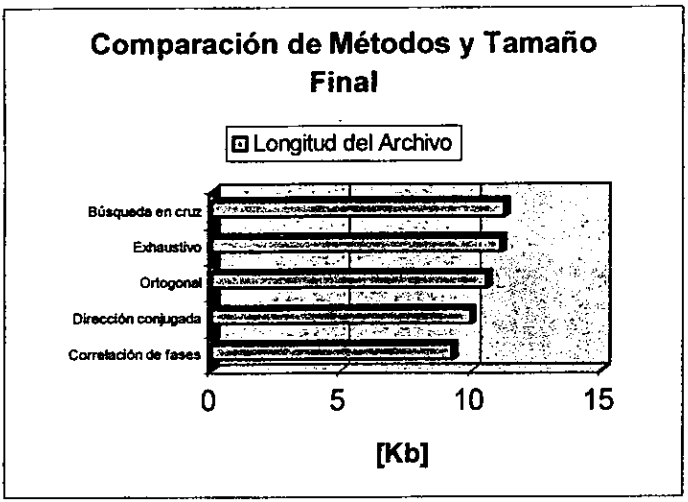


Figura 6.14 Gráfica comparativa de Métodos de Estimación y Tamaño (secuencia "sphere")

De lo anterior se puede concluir que el método de dirección conjugada representa un buen algoritmo para este tipo de secuencia, en donde se encuentra crecimiento de regiones en un objeto, ya que con dicho método se obtiene una muy buena compresión y se invierte poco tiempo en la realización de las codificaciones. Sin embargo, el hecho de que la longitud del archivo sea pequeña se refleja, aunque sea de una manera pequeña, en la pérdida de calidad de la película mpeg1, el eterno forcejeo entre calidad y cantidad de compresión al que se enfrenta frecuentemente esta área de la ingeniería. Regresando al caso de las secuencias en análisis, creemos que lo más importante es obtener una calidad aceptable logrando una buena compresión, lo que equivaldría a un punto a la mitad del camino. Se puede jugar con los distintos niveles de cuantización y los métodos de estimación de movimiento programados para adecuar la codificación de cierta secuencia a las características particulares que le deseemos dar, probablemente necesitamos mucha calidad en el archivo MPEG, o en ocasiones lo que impera es la alta compresión. La elección del método de estimación de movimiento, como se ha mencionado, va fuertemente ligada a las características intrínsecas de la secuencia.

Para las tablas anteriores, no se tomaron en cuenta los algoritmos de tres pasos ni el jerárquico, debido a que no funcionan muy bien para este tipo secuencias, no precisamente porque el tamaño del archivo final sea muy grande, ya que es comparable con los tamaños obtenidos con los otros algoritmos, sino que el tiempo de codificación es mucho mayor que cualquiera de los anteriores, por lo que evitamos introducirlo a fin de no alterar la escala de tiempos que se ha venido manejando en el presente capítulo. Como sabemos, el método jerárquico utiliza recursividad en tres niveles de resolución ayudándose del método de los tres pasos; con esto se explica que dicho algoritmo sea el más lento, aún más que el de los tres pasos simple.

La explicación expuesta en el párrafo anterior nos lleva a concluir la dificultad que presentan estos dos métodos para estimar movimiento en secuencias en las que intervenga algún crecimiento de regiones y oclusiones en figuras sintéticas. Al analizarlos detenidamente, hemos observado que en la mayor parte de los conjuntos de ventanas de búsqueda, los métodos obtienen campos de velocidades poco homogéneos, más bien caóticos, a diferencia de los otros métodos, que detectan en mayor o menor medida el crecimiento de las esferas y la oclusión presente en la secuencia. Esto lleva a que generalmente recurran tanto los tres pasos como el jerárquico, al número mayor de operaciones buscando la mejor coincidencia, que ocasionalmente está equivocada.

Un aspecto sin duda fundamental que no por hallarse en esta altura del capítulo es menos importante que cualquiera de los anteriores, es el de las tasas de compresión y de transmisión de los archivos MPEG codificados. Como sabemos, la tasa de compresión refleja la razón o cociente entre el tamaño original, sin comprimir, de la secuencia, y el tamaño final del archivo MPEG. Esta tasa suele variar de secuencia en secuencia, y esto es normal debido a que el codificador MPEG es flexible en este sentido: hay secuencias en que son mucho más evidentes las redundancias espaciales y temporales, y hay secuencias en las que se tiene que enviar una cantidad mayor de información en cada trama porque existe gran variación espacio-temporal entre ellas.

La tasa de compresión, se obtuvo al saber el tamaño original en Kb que resulta de juntar los tamaños de todas las imágenes de la secuencia antes de comprimir y esto se divide entre el tamaño del archivo comprimido (con el cálculo para cada algoritmo). Con esto intentamos dar al lector una medida un poco más asimilable que el simple y llano tamaño final del archivo MPEG, que no dice mucho en términos de compresión. En este cálculo de la tasa de compresión despreciamos los bytes escritos en el archivo MPEG que involucran su información estructural (headers), ya que son de tamaño despreciable con respecto a la totalidad de la información relativa a las imágenes codificadas.

El tamaño de la secuencia (de 10 imágenes) de "sphere" utilizada, sin comprimir es de 640 Kb, ya que cada imagen mide 64 Kb, mientras que el tamaño del archivo final, utilizando el método de estimación de movimiento que comprime menos para esta secuencia, es alrededor de 57 veces más pequeño, aunque hay que hacer notar que hay un precio que hay que pagar por esta compresión y está implícito en la calidad final de la película.

Para el cálculo de la tasa de transmisión deberemos recurrir a la especificación del estándar, donde se dice que la transmisión deberá llevarse a cabo a 30 [*frames/s*]. De esta forma, y considerando las tasas de compresión que se muestran en la siguiente tabla, se obtuvo la tasa de transmisión correspondiente.

Algoritmo de estimación	**Longitud del archivo (en Kb)**	**Tasa de Transmisión (Kb/s)**	**Tasa de compresión (veces)**
Correlación de fases	9.224	27.67	69.38
Dirección conjugada	9.934	29.80	64.42
Ortogonal	10.572	31.72	60.53
Exhaustivo	11.096	33.29	57.67
Búsqueda en cruz	11.235	33.71	56.96

Tabla 6.6 Longitud y tasas de compresión y transmisión para la secuencia codificada "sphere"

Regresando al análisis de tasas de compresión y de transmisión de la secuencia "stefan", podemos valernos de las siguientes tablas.

La secuencia "stefan", como ya se dijo, está en formato NTSC (352 x 240 pixeles) y escala de grises en un rango de 0 a 256. Cada una de sus imágenes mide por lo tanto 84,480 [bytes]. La secuencia codificada fue de 9 *frames*, por lo que estaríamos hablando de 760,320 [bytes] de información sin comprimir.

Algoritmo de estimación	**Longitud del archivo (en Kb)**	**Tasa de Transmisión (Kb/s)**	**Tasa de compresión (veces)**
Correlación de fases	109.667	357.14	6.93
Búsqueda en cruz	110.241	359.22	6.89
Exhaustivo	111.716	363.44	6.81
Dirección conjugada	111.848	363.97	6.80
Ortogonal	114.99	374.43	6.61
Tres pasos	116.77	380.18	6.51
Jerárquico	120.614	392.86	6.30

Tabla 6.7 Longitud y tasas de compresión y transmisión para la secuencia codificada "stefan"

Es notable la diferencia en términos de compresión y tasa de transmisión entre las dos secuencias, aspecto atribuible a la distinta constitución de ambas. Por un lado, la secuencia "sphere" cuenta con redundancias espaciales y temporales enormes, la similitud entre cuadros adyacentes es tremenda y la distribución de tonos de grises en cada frame está de manera muy uniforme y en grandes regiones, lo que favorece la eliminación de coeficientes de AC en la transformada coseno discreta. Deseábamos incluir estas comparaciones para evidenciar el poder de compresión del MPEG en ambientes favorables y su comportamiento adecuado en secuencias difíciles, como la de "stefan". Aquí, como ya se dijo, se suma la gran variedad de texturas dentro de cada cuadro (lo que provoca la aparición de muchas frecuencias altas) y la menor correlación entre tramas, lo que causa menos redundancia temporal. La compresión promedio lograda fue de alrededor de 7:1, pero observamos en experimentos posteriores que en otros tipos de secuencia menos complicada (como las llamadas "interview", "hall", etc.) la compresión lograda era mayor. Nos limitamos en este capítulo a reportar resultados de las dos secuencias mencionadas por

la dificultad que representaría diseñar un método evaluatorio con distintas secuencias. Lo que se desea es mostrar cualitativamente y cuantitativamente pruebas de desempeño del codificador, y con dos secuencias radicalmente distintas como las utilizadas se pudo lograr el objetivo.

VI. 3 Análisis final de los métodos de estimación.

Correlación de fases: Este método es muy bueno para cuando se trata de secuencias reales como es la de "stefan", que no es una imagen sintética, pero en la cual hay mucho movimiento, y resulta medianamente bueno en compresión de secuencias sintéticas como la esfera, en donde hay crecimiento de objetos, aunque si nos vamos a la calidad, no es tan bueno, sobre todo en las orillas de las imágenes que componen la secuencia. Funciona muy bien en ambientes con mucha textura y poco movimiento de objetos, ya que al haber movimientos bruscos o de gran magnitud tiende a perder el valor correcto en el pico de la correlación. Es un método sumamente rápido debido a que su estructura es radicalmente distinta a la de los otros métodos, como se explicó en el capítulo III. Ha resultado bastante satisfactoria, tomando en cuenta su novedad, la inclusión de la correlación de fases en el esquema MPEG, ya que se ha comportado muy bien en la mayoría de las secuencias codificadas.

Exhaustivo: Este es un método de estimación muy bueno para codificar secuencias reales, así como ocurre con el de correlación de fases, ya que donde hay crecimiento de objetos, tiene problemas. La compresión que se obtiene con este método, podríamos decir que es mediana comparada con la que se obtiene con los demás métodos. Sin embargo, hay ventajas fuertes que posee el método exhaustivo: al tratarse del método que predice mejor el movimiento logra imágenes P y B de muy buena calidad objetiva y subjetiva y favorece la correcta compresión de las secuencias. Como es de esperarse, el tiempo que consume es mayor que en la mayoría de los métodos, pero es una buena elección en secuencias con movimientos rápidos, finos y complicados. No es tan recomendable su aplicación en

secuencias sencillas, pudiendo en éstas integrar algunos de los otros métodos de estimación de movimiento.

Ortogonal: Es de los algoritmos más rápidos para codificar secuencias junto con el de correlación de fases, o quizás sea el más rápido, no importando del tipo de secuencia de que se trate. Sin embargo, se obtiene una compresión regular y la calidad de la película no es muy buena. Podemos decir que, algorítmicamente, es el método más sencillo de los programados, pero esto desgraciadamente se paga con errores en la estimación de movimiento. Si se desea velocidad, el método cumple con creces, pero si la calidad de la película final es lo que realmente deseamos, no es muy recomendable su utilización. Funciona mejor en secuencias sintéticas.

Dirección conjugada: Éste método involucra mayor complejidad computacional que el ortogonal, pero gana mucho en compresión y calidad. Dio buenos resultados en la mayoría de las secuencias experimentadas, arrojó niveles de desempeño aceptables en todas las líneas, por lo que podemos decir que es un método "promedio" en su calidad. Es un algoritmo muy bueno para codificar secuencias de crecimiento como es la de la esfera, ya que comprime mucho y rápido, la calidad subjetiva de las películas puede decirse que es regular.

Búsqueda en cruz: Este es un algoritmo lento, con el que se obtienen mejores resultados en cuanto a compresión y calidad en secuencias como la de stefan, y no es bueno para las secuencias que presentan crecimiento de objetos. No es precisamente exacto en la estimación de movimiento, ya que es una variación del ortogonal. Sin embargo, se comportan de manera muy distinta en las secuencias probadas. Lleva a cabo más operaciones por ventana que algunos otros métodos, como el de dirección conjugada, por lo que consume más tiempo.

Tres pasos: Este algoritmo es medianamente rápido en secuencias como la de "stefan", aunque la compresión y la calidad no son tan buenas, pero en secuencias como la

de la esfera, resulta sumamente lento y no es posible compararlo con los demás algoritmos para estas secuencias. La enorme variación en tiempos para este método se debe a que el número de operaciones por iteración puede ser muy pequeño o muy grande, dependiendo de la convergencia del cálculo dentro de alguna ventana de búsqueda. Esperábamos mejores resultados en algunas líneas para este método, sin embargo, a pesar de que consume mucho más tiempo en ocasiones que otros métodos, la calidad del archivo final MPEG es mejor.

Jerárquico: De acuerdo a las pruebas realizadas este fue el algoritmo con el que se obtuvieron los peores resultados, ya que tiene casi los mismos problemas que el de los tres pasos, pero es mucho más lento, ya que se vale de dicho algoritmo y es llamado tres veces para cada estimación de un bloque. Hay que hacer notar que se podría refinar todavía mas el uso del algoritmo modificando inteligentemente los parámetros intrínsecos del mismo (niveles de resolución, algoritmo de "batalla" dentro de cada nivel, tamaños de las ventanas de búsqueda, etc.) de acuerdo a la secuencia que se esté codificando, pero este proceso sería muy lento en sí y traería consigo otro tipo de análisis mas profundo que el que enfrenta esta tesis. La multiresolución resultó útil solo en cierto tipo de secuencias, ya que en algunas el filtrado paso-bajas para el análisis burdo del primer nivel de resolución alteraba de manera grave el curso de la búsqueda arrojando una dirección equivocada, sobre todo en zonas con alta frecuencia, con texturas complicadas y con muchos bordes presentes.

VI.4 Resumen.

Los resultados anteriores nos permiten suponer que los objetivos escritos del trabajo se cumplieron en gran medida. La comparación entre los métodos de estimación de movimiento integrados satisfactoriamente dentro del estándar de codificación se logró con los algoritmos programados para ese fin. Si bien el programa tiene algunas pequeñas deficiencias (específicamente en términos de tiempo de ejecución), es una implementación mantenible de un codificador MPEG-1 estándar. Queremos insistir en la modularidad lograda en el codificador, misma que permite que futuras adiciones o modificaciones al

mismo sean llevadas a cabo con relativa facilidad. Muchas de nuestras pruebas fueron hechas con un reproductor de MPEG independiente de Matlab, además del propio lector de formato MPEG que los usuarios de Matlab han hecho disponible en la sección de contribuciones de la compañía Mathworks. Asimismo, la extensión del codificador con nuevos métodos de estimación, si bien no es trivial, sí es mucho más sencilla que la modificación del codificador antes mencionado, además de que la programación se hace en un lenguaje más adecuado a la tarea que C. Creemos, además, que hemos obtenido un producto innovador al haber explotado el aspecto sintáctico del estándar MPEG-1 desarrollando nuestra programación en un enfoque orientado a objetos, escrito en el ambiente de Matlab.

Referencias

Andleigh, P. & Thakrar, K., 1996, *Multimedia Systems Design*, Prentice Hall PTR, New Jersey, USA.

Castleman, K., 1996, *Digital Image Processing*, Prentice-Hall, New Jersey, USA.

Bierling, M., 1988, "Displacement estimation by hierarchical block-matching", *Proc. Visual Comm. And Image Proc.*, SPIE vol 1001, pp. 942-951.

Bracewell, R.N., 1986, *The Fourier Transform and its Applications*, McGraw-Hill, New York, USA.

Black, M., 1968, "El Laberinto del Lenguaje", Monte Ávila Editores, Caracas, Venezuela.

Erkan, Y., Sezan, M. & Erdem, T., 1993, "A hierarchical phase-correlation method for motion estimation", *Proc. Conf. on Info. Scien. and Systems*, Baltimore, MD, pp. 419-424.

Foley, Van Dam et Al., 1996, "Introducción a la Graficación por Computador", Addison Wesley Iberoamericana, México D.F, México.

Furht, B., Smoliar, S. & Zhang, T., 1990, Video and Image Processing in Multimedia Systems. Kluwer Academic Publishers, Boston, USA.

Ghanbari, M,. 1990, "The cross-search algotithm for motion estimation", *IEEE Trans. Commun.*, vol 38, pp. 950-953.

Gharavi, H. & Mills, M., 1990, "Block Matching motion estimation algorithms: New results", *IEEE Trans. Circ. And Syst.*, vol. 37, pp. 649-651.

Gibson, J.J., 1950. *The perception of the visual world.* Houghton Mifflin, Boston, MA.

González, R. & Woods, R., 1992. *Digital Image Processing.* Addison-Wesley, Massachusetts, USA.

Graham, C. H., 1965. *Vision and Visual Perception.* John Wiley & Sons, New York, USA.

Hardman, L, van Rossum, G. et al., 1993. "Structured Multimedia Authoring", ACM Siggraph Proceedings.

Hildreth, E. C., 1983, "The computation of velocity field", MIT Artificial Intelligence Laboratory Memo.

Huffman, D., 1952, "A Method for the Construction of Minimum Redundancy Codes", *Proc. IRE*, 40.

Inglis, A., 1990, "Video Engineering", Mc Graw Hill, USA.

Mallat, S., 1991, "Zero-Crossings of a Wavelet Transform", *IEEE Trans.* IT-37 (4).

Max, J. "Quantizing for Minimum Distortion", *IRE Trans. Info. Theory*, IT-6.

Mitchell, J. L., Pennebaker, W. B., Fogg, C. E., Le Gall. D. J., 1996, "MPEG Video Compression Standard", Chapman & Hall, USA.

Mitchie, A., & Bouthemy, P., 1996, "Computation and analysis of Image Motion: A Synopsis of Current Problems and Methods". *International Journal of Computer Vision* 19(1), 29-55.

Mitra, S., Sicuranza, G., 1992, "Multidimensional Processing of Video Signals", Kluwer Academic Publishers, Norwell, Massachusetts, USA.

Murat, N., 1990, "Digital Video Processing", Prentice Hall Signal Processing Series, USA.

Novak, C.L. & Shafer, S. A., 1992, "Anatomy of a Color Histogram", *Proceedings of the 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society Press, Los Alamitos, USA.

O'Neil, P. V., 1994, "Matemáticas Avanzadas para Ingeniería", volumen 2, editorial CECSA, México, D.F., México.

Oppenheim, A. & Willsky, A., 1994. "Señales y Sistemas". Prentice Hall, México, D.F.

Pratt, W., 1978, "Digital Image Processing". John Wiley & Sons, Washington, USA.

Pratt, W. & Andrews H., 1969, "Hadamard Transform Image Coding", Proc. IEEE, 57-68.

Shannon, C., 1948, "A Mathematical Theory of Communication", *Bell System Tech.* Journal, 27.

Shapiro, L. & Rosenfeld, A., 1992. "Computer Vision and Image Processing", Academic Press, San Diego, USA.

Sikura, T., 1997, "MPEG Digital Video-Coding Standars", *IEEE Signal Processing Magazine*. IEEE.

Silván J.L., 1998, *Estimación de Movimiento en Secuencias de Imágenes por medio de Transformadas Polinomiales*, Trabajo de Tesis de Licenciatura, UNAM, México.

Speziale, L. & Solar, E. 1985, "Apuntes de Álgebra Lineal", editorial Limusa, México, D.F.

Watkinson, J., 1994, "An Introduction to Digital Video", editorial Focal Press, USA.

Watkinson, J., 1995, "The Art of Digital Video", editorial Focal Press, USA.

Watson, A. & Ahamuda, A., 1985. "Model of Human Visual motion sensing", *Journal of Optical Society of America* (A2).

[RI 1]: http://jura2.eee.rgu.ac.uk/dsk1/dgivid/

[RI 2]: http://cs4sun.cs.ttu.edu/lakhani-doc/compression_papers/video_tutorial