Compresión de video MPEG. Análisis de algoritmos de Codificación Paralela

Ms. Claudia Cecilia Russo[1]*, Lic. Hugo D. Ramon*[2] *, Ing. Armando De Giusti*[3]

L.I.D.I. (Laboratorio de Investigación y Desarrollo en Informática)[4]
Facultad de Informática
Universidad Nacional de La Plata

Resumen

Se presenta el análisis de dos clases de algoritmos paralelos para compresión de video MPEG. El estudio se justifica ya que la enorme cantidad de cálculo computacional involucrada hace que un codificador por software secuencial sea lento en la codificación de imágenes de tamaño razonable (por ejemplo, 0.1-0.3 frames por segundo).

En el trabajo se proponen y discuten dos clases de algoritmos paralelos: El primer método codifica espacialmente partes de la imagen en paralelo, mientras que el segundo codifica grupos de imágenes (Groups of Pictures, GOP) en paralelo.

Básicamente se estudia el impacto de ambas técnicas sobre la actividad (overhead) de comunicaciones en la arquitectura y el desbalance de carga resultante de cada una de ellas.

Al análisis conceptual realizado se agregan propuestas de arquitecturas paralelas para la implementación de los algoritmos y sus ventajas/desventajas. Asimismo se deja abierta la posible migración a hardware o firmware de los algoritmos.

Palabras clave: Compresión de video, MPEG, Algoritmos paralelos, Eficiencia algorítmica.

[1] **Profesor Adjunto DE, Facultad de Informática, UNLP E-mail: crusso@info.unlp.edu.ar**
[2] **Profesor Adjunto DE, Facultad de Informática, UNLP E-mail: hramon@info.unlp.edu.ar**
[3] **Investigador Principal CONICET, Prof. Tit. DE, Facultad de Informatica, UNLP**
E -mail: degiusti@info.unlp.edu.ar
[4] **LIDI - Calle 50 y 115 - (1900) La Plata - Buenos Aires – Argentina TE-FAX = 54 21 22 7707**
WEB :http//lidi.info.unlp.edu.ar

Introducción

El algoritmo de compresión de video MPEG es la base de muchas aplicaciones de multimedia y de comunicaciones actuales y futuras como el video CD, Video a Demanda (VOD) y Disco de Video Digital (DVD). Los estándares MPEG describen una sintaxis para los datos y especifican los pasos de decodificación, sin embargo, no especifican el codificador. Aunque los codificadores MPEG se han llevado a cabo en hardware, un enfoque más flexible es usar codificación por software.

Esta solución tiene la ventaja que en la fase de diseño del codificador se pueden hacer y evaluar los cambios fácilmente, y que en la fase de producción del codificador una modificación requiere una actualización de software más que un rediseño del hardware.

Más aun, un codificador de software puro puede llevar a cabo un análisis de secuencias de imágenes más sofisticado para maximizar la calidad del material de video a una tasa de datos dada. Un codificador de software secuencial es bastante lento para codificar imágenes de tamaños razonables. Esto es porque la compresión de video MPEG requiere una enorme cantidad de calculo computacional. En particular la Transformada Discreta del Coseno (DCT), la Estimación de Movimiento (ME) y la Compensación de Movimiento (MC) son de computo intensivo.

Es de esperar que la tasa de codificación puede aumentar usando un sistema paralelo y/o distribuido.

Desafortunadamente, aun si se usan máquinas poderosas, el sistema puede no lograr un procesamiento en tiempo real (codificar 25-30 frames por segundo) porque varios factores pueden degradar la performance.

El ancho de banda de la red y las comunicaciones de E/S en particular imponen una carga crítica. Como generalmente las secuencias de imágenes se leen de archivos de computadora el tiempo de transferencia de disco de un frame forma un límite superior a la tasa de codificación. La fluctuación de la carga en la red y en las máquinas también afecta la performance del sistema. En realidad, no siempre se requiere una codificación de tiempo real, para muchas aplicaciones es satisfactorio el procesamiento “off-line” (por ejemplo, CD, VOD), pero también debe hacerse dentro de una cantidad de tiempo razonable. Teniendo todo en cuenta, es esencial una buena estrategia del balance de carga, mientras que el “overhead” de comunicaciones (cantidad de inicios de comunicaciones, cantidad de datos transferidos) debe mantenerse lo más bajo posible.

Otra forma de implementar un sistema paralelo es usar una supercomputadora que contenga muchos nodos procesadores. En este caso, se puede lograr una mejor performance. Sin embargo, una aplicación escrita para un sistema de multiprocesadores específico normalmente es menos portable.

En este trabajo se plantea el análisis de dos clases de algoritmos para codificación MPEG. Primero se procesa una imagen en paralelo (paralelización espacial), mientras que el segundo procesa grupos de imágenes en paralelo (paralelización temporal).

2. Generalidades del Codificador MPEG

Una imagen es descripta por tres componentes, un plano de imagen de luminancia (Y) y dos de crominancia (U y V). El MPEG-1 y la mayoría de los modos en MPEG-2 usan el espacio de color denominado 4:2:0, que significa que la cantidad de muestras de U y V son la mitad de la cantidad de muestras de Y tanto en la dirección horizontal como vertical.

Un frame está dividido en áreas que no se superponen llamadas macrobloques (MB). Un macrobloque contiene un bloque de 16x16 píxeles en el plano Y y un bloque de 8x8 píxeles tanto en el plano U como en el V. El contenido de Y está dividido en 4 bloques de 8x8 píxeles.

El video MPEG es aplicado a secuencias de imágenes naturales que contienen altas redundancias espaciales y temporales. Redundancia espacial significa que los elementos de la imagen (píxeles) son similares dentro de una cierta área en un frame. Esto es utilizado por una clase de transformada de frecuencia llamada *DCT bidimensional*, que se aplica a los bloques de 8x8 de cada macrobloque, resultando en 64 coeficientes por bloque. Una pequeña cantidad de ellos es significativamente más grande que las otras, la energía de un bloque normalmente se concentra en el dominio de baja frecuencia. Después de la cuantización, un bloque puede ser representado por unos pocos coeficientes (por ejemplo, 6...9), logrando de esta forma una buena compresión. Esta compresión mejora más al asignar Códigos de Longitud Variable (Variable Length Codes, VLC) a símbolos especiales creados a partir de los coeficientes. Un macrobloque codificado de esta manera se llama *intra*, ya que usa información sólo de él mismo. Puede reconstruirse por el procedimiento inverso.

La redundancia temporal se refiere a la similitud entre una cierta cantidad de frames consecutivos. Hace posible que las imágenes puedan predecirse unas a partir de otras. Una imagen a partir de la cual se hace una predicción se llama frame *de referencia* o sólo *imagen de referencia*. La predicción es mejorada por la *compensación de movimiento*, que es el uso de la información de movimiento en la codificación diferencial. En video MPEG, la información de movimiento se encuentra por medio de una *estimación de movimiento basada en bloques* usando el componente de luminancia de la imagen. En todos los macrobloques funciona de la siguiente manera: El contenido de Y de un macrobloque en el marco actual es cambiado (shifted) sobre una ventana de búsqueda en el marco de referencia, y se busca un área similar de 16x16 píxeles. La similitud se evalúa por medio de una función de costo en ciertas posiciones. El lugar donde la función tiene un mínimo se considera como el mejor "match". La traslación relativa del macrobloque es descripta por un vector de movimiento. Un conjunto de vectores de movimiento para una imagen se llama *campo de vectores de movimiento*.

Si el mejor match es lo suficientemente bueno, los píxeles correspondientes se restan unos de otros, o sea, el macrobloque real se compensa en cuanto al movimiento. En MPEG también se llaman *predictivos*. Como tal macrobloque todavía exhibe algo de redundancia espacial, la codificación DCT antes mencionada se aplica a los bloques de 8x8 en él.

Un decodificador reconstruye un macrobloque predictivo usando el área de referencia correspondiente que fue accesada por el vector de movimiento. Sin embargo, en el caso que no haya un buen match el macrobloque se codifica como intra porque es probable que produzca menos bits.

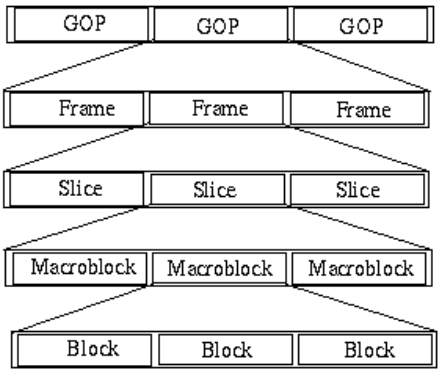
En base a los métodos de arriba, el video MPEG define tres tipos de imágenes, las imágenes I, P y B. Los frames I son *intra coded frames* (frames con codificación intra), se codifican sin referencia a otros frames utilizando sólo la redundancia espacial. En consecuencia, consisten sólo de macrobloques intra codificados. Los frames P son *predicted frames* (frames predichos), se predicen a partir del frame I o P anterior usando compensación de movimiento. Los B o *bi-directionally coded frames* (frames codificados bidireccionalmente) se predicen a partir del frame I o P anterior y posterior. En este último

caso se buscan dos vectores de movimiento para cada macrobloque, uno en la referencia previa (predicción pasada) y otro en la referencia siguiente (predicción futura). La calidad del "matching" determina si la predicción se hace a partir de la pasada o de la futura o de las dos. En este último caso, se saca un promedio de las áreas de referencia correspondientes. Si no se consigue un buen match, el macrobloque se codifica como intra. Como consecuencia, las imágenes P y B pueden contener macrobloques codificados de manera predictiva y de manera intra. La idea básica detrás del uso de las imágenes B es que algunas áreas pueden no encontrarse correctamente en ninguna de las dos referencias porque los objetos pueden taparse unos a otros, o también puede ocurrir un cambio de escena. Como un frame B usa la predicción futura, su referencia futura tiene que ser transmitida antes que ese frame. Las imágenes B nunca se usan como referencia.

El algoritmo de codificación mencionado más arriba se usa en MPEG-1. El MPEG-2 es mucho más complicado en lo que hace a la predicción, pero las ideas básicas son las mismas. Un patrón de imagen típico en MPEG podria ser I BBPBBP. En realidad, puede haber cualquier cantidad de imágenes P y B entre las de referencia. Más aun, su uso es opcional, una secuencia puede consistir sólo de imágenes I. Las imágenes I proporcionan puntos de acceso donde puede empezar la decodificación. Se puede facilitar un acceso al azar rápido por el uso de muchas imágenes I. Sin embargo, su radio de compresión es el menor, típicamente entre 6 y 9. Las imágenes P pueden comprimirse mejor, normalmente con un factor de 12 a 20. Las imágenes B proporcionan la mayor compresión, típicamente de 25 a 50. El radio de compresión total es entre 20 y 30 en el caso de las aplicaciones usadas en la práctica. Todos los valores se refieren a codificación de alta calidad y dependen del contenido de la imagen. Cuanto más compleja sea una imagen, más bits se necesitarán para representarla.

Una corriente de video MPEG tiene una estructura en capas. En la parte superior está la *Capa de la Secuencia* (Sequence Layer), cuyo encabezado contiene los parámetros generales para la secuencia (por ejemplo, las dimensiones del frame, la tasa del frame, la tasa de bits, etc.). La secuencia consiste de algunos *Grupos de Imágenes* (GOP) que indican la segunda *capa*. Un GOP siempre empieza con un frame I o B y termina con un frame I o P. Por ejemplo, la secuencia IBBPBBP puede considerarse un GOP. Ofrecen un acceso al azar en la secuencia y facilitan la capacidad de edición.

La siguiente capa es la *Capa de la Imagen* (Picture Layer). Una imagen está dividida en porciones (*Capa de las Porciones*, Slice Layer) que consisten de macrobloques sucesivos. Una porción es exactamente una fila horizontal de macrobloques en MPEG-2, mientras que el MPEG-1 permite mayor flexibilidad para su posición y tamaño. Las últimas dos capas son la *Capa de los Macrobloques* (Macroblock Layer) y la *Capa de los Bloques* (Block Layer).



3. Paralelización del proceso de codificación

Operaciones computacionalmente intensivas

Como ya se ha mencionado, algunas partes del proceso de codificación requieren una cantidad de calculo computacional. Por lo tanto, es esencial escribir un código secuencial optimizado rápido, sobre el cual se pueda basar la implementación paralela.

La DCT es una de las operaciones computacionalmente más intensivas.La cantidad de multiplicaciones está por encima de las 450.000 y la cantidad de sumas es mayor que 1.200.000 si se aplica a una imagen SIF (de 352x288 o 352x240). La estimación de movimiento requiere aun más cálculos, la complejidad computacional está determinada por el tamaño de la ventana de búsqueda, o sea, el rango del vector de movimiento. La evaluación de todas las posiciones posibles (búsqueda completa) es una pérdida de tiempo, aunque siempre encuentra el mejor match, es probable que la cantidad de bits producidos para ese macrobloque sea la menor. Sin embargo, la cantidad de operaciones está por encima de los 200 millones en un frame SIF si el rango del vector es 16.

La compensación de movimiento no contribuye tanto al tiempo de codificación, pero debe considerarse como importante porque la decisión del MB (cómo codificar un macrobloque dado) normalmente se lleva a cabo calculando varianzas de MB.

Dependencias en el algoritmo de codificación

Debido a la gran cantidad de cálculos, es recomendable hacer la codificación en paralelo. La sintaxis en capas ofrece varias posibilidades, pero también hay ciertas dificultades. El paralelismo más fino se puede lograr a nivel de los bloques, ya que todos los bloques se transforman y cuantizan de manera independiente.

El nivel de macrobloques sugiere una posibilidad de encontrar los vectores de movimiento para un frame de manera independiente. Sin embargo, ambos niveles de paralelismo serían demasiado finos para un sistema paralelo y/o distribuido, ambos implicarían una comunicación extremadamente pesada.

El siguiente nivel de procesamiento paralelo es ofrecido por la capa porciones (slice layer). Las porciones siempre se codifican de manera independiente unas de otras. El algoritmo se basará en este hecho. Una idea auto-explicatoria sería codificar los frames en paralelo. El codificador paralelo de Berkeley esencialmente se basa en este método, las máquinas codifican una cierta cantidad de imágenes de acuerdo a su poder. Asegura un excelente balance de carga, pero la codificación puede hacerse de manera incorrecta.

La siguiente seciencia IBBPBBPBBPBBP muestra el problema, suponiendo que las imágenes 0-6 se asignan al procesador P1, mientras que las imágenes 7-12 se asignan al procesador P2. P1 puede comenzar a codificar el primer frame porque es un intra. Sin embargo, P2 no puede codificar el frame 7 (B) porque usa sus referencias (el frame 6 y el frame 9) en la compensación de movimiento. El decodificador reconstruye las imágenes P y B por medio de su/s campo/s vectorial/es de movimiento y su/s referencia/s. Sin embargo, del lado de la decodificación sólo hay referencias decodificadas disponibles. Como consecuencia, la predicción debería hacerse también a partir de frames decodificados, caso contrario ocurre una desigualdad entre el codificador y el decodificador. Esto resulta en un frame P o B decodificado de peor calidad. Además, este error se propaga hasta la siguiente imagen I debido a que las imágenes P consecutivas se predicen unas a partir de otras. Así, para codificar el frame 7, debemos tener los frames 6 y 9 en su forma decodificada. Por lo

tanto, P2 debe esperar hasta que P1 termine con el frame 6, lo que destruye el paralelismo casi por completo.
La versión actual del codificador paralelo de Berkeley también usa frames de referencia originales para la compensación de movimiento. El autor dice que la degradación es despreciable, y que el uso de una cuantización más fina puede esconder las diferencias.

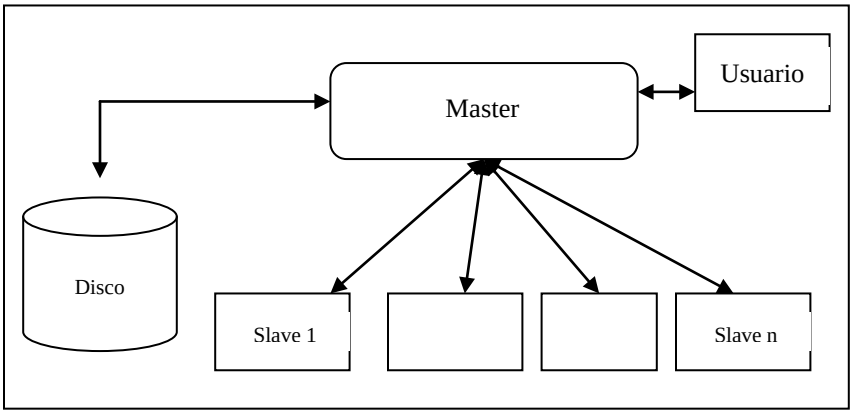
De hecho, la estimación de movimiento puede hacerse de manera independiente de las otras tareas de codificación, así, P2 podría encontrar campos vectoriales para los frames P y B, y continuar el procesamiento cuando es posible. Desafortunadamente, el "scheduling" se haría bastante complicado, y no hay garantía de una sincronización adecuada. Otro problema es que el frame 6 es leído por P1 y por P2, lo que aumenta la comunicación. Sin embargo, ninguno de estos problemas emerge si cada corrida de frames comienza con un frame I y termina con un frame P. Esto hace que un grupo de frames sea independiente de los otros. De esta manera, los requerimientos para un Grupo de Imágenes (GOP) se satisfacen, por lo tanto, podemos llamarlo paralelización a nivel de GOP.

El segundo algoritmo se basa en este método. Desafortunadamente, también hay problemas con esta clase de paralelismo. Para asegurar una buena calidad a una codificación con una tasa de bits baja la distancia entre dos imágenes I debe ser lo suficientemente grande, típicamente entre 10 y 15. Esto significa un gran tamaño de datos, lo que resulta en un peor balance de carga que la paralelización por frames. Demostraremos que al usar GOPs más chicos como el IBBP, no sólo se obtiene un buen balance de carga, sino que también se obtiene una buena performance paralela.

4. Descripción de los algoritmos

Modelo paralelo propuesto

En ambos casos la configuración es un modelo master-slave como se muestra a continuación:



Configuración paralela

Los esclavos residen en diferentes máquinas. El master lee las imágenes crudas de un archivo, y escribe los datos comprimidos (corriente de bits) en otro archivo. El master controla a los esclavos, proporcionando así el balance de carga. Distribuye datos crudos de

imágenes con algo de información de control entre ellos. Los esclavos hacen la codificación y reenvían los resultados al master sin comunicarse entre ellos.

El usuario puede contactar sólo al master ingresando varios parámetros de codificación, como ser el patrón GOP, el rango del vector de movimiento, la cantidad de esclavos a inicializar, la calidad del video, etc. Al final del procesamiento los esclavos le proporcionan al master sus tiempos de computación + comunicación y de inactividad (idle) para evaluar la performance del sistema.

El algoritmo del nivel de las porciones (slice level algorithm)

En este caso, la paralelización se hace dentro de un frame (paralelización espacial). El balance de carga puede ser dinámico sobre el nivel GOP, pero siempre es estático dentro de un GOP. Aplicamos las siguientes definiciones: una porción o slice consiste de una cantidad cualquiera de filas horizontales de macrobloques. Un segmento consiste de una porción y de las áreas extras asociadas por encima y por debajo de la porción o slice si esta porción viene de una imagen P o I. Estas áreas extras se necesitan para la compensación/estimación del movimiento. Como las imágenes B nunca se usan como referencia, en aquellos casos no se requieren áreas extras.

Un frame se divide en porciones o slice como la cantidad de esclavos se tenga. Esto proporciona la menor cantidad de inicios por frame. Se asignan segmentos a los esclavos de acuerdo a su poder instantáneo. Cuanto más rápido sea un esclavo dado, más grande deberá ser la porción que procea. El poder del esclavo se puede determinar previamente con un procedimiento de prueba anterior. Los resultados se envían de vuelta acompañados de los tiempos de codificación. Usando estos tiempos el master puede estimar la performance de los esclavos. Debe notarse que las pruebas frecuentes mejoran el balance de carga, pero también aumentan la carga computacional. En el modo normal de operación cada esclavo envía de vuelta su porción codificada, que pueden no llegar en el orden correcto. En esta situación, el master las almacena temporariamente, pero las pone en el archivo MPEG tan pronto como sea posible. Una consecuencia de este método es que algunas partes de las imágenes predichas pueden no codificarse correctamente. Las áreas extras en los frames de referencia también deben codificarse y decodificarse para hacer correctamente la predicción. Pueden procesarse si el segmento real viene de un frame I, pero no pueden si deriva de un frame P. Sólo el primer frame P que sigue a un frame I puede codificarse correctamente con cierta seguridad. Todos los otros frames predichos y referenciados por esta imagen P contendrán porciones codificadas incorrectamente si hay vectores de movimiento apuntando a las áreas extras no procesadas. Por ejemplo, supongamos la siguiente secuencia: IBBPBBPIBBP... Excepto por el primer frame P, una porción en cualquier otro frame predicho hasta la siguiente imagen I no puede usar áreas extras decodificadas porque están presentes en otros esclavos. Por supuesto, un esclavo dado podría adquirir estas áreas en la forma decodificada, pero introduciría una comunicación pesada entre los esclavos. La codificación incorrecta ocurre sólo si hay un movimiento vertical cerca de una línea de segmentación. Además, cuantos más esclavos hay en la configuración, más áreas extras tenemos. Afortunadamente, las secuencias de video en vivo siempre contienen mucho más desplazamiento horizontal que vertical, lo que evita que el material de video tenga una degradación de calidad en este caso.

El algoritmo del nivel GOP

Este método procesa Grupos de Imágenes en paralelo (paralelización temporal). El scheduling es mucho más simple que en el caso previo, hasta el balance de carga es dinámico en toda la secuencia. Funciona de una forma circular de la siguiente manera: Teniendo N esclavos, los primeros N GOPs se les asignan a su vez. Antes de codificar el último frame en un GOP, el esclavo envía una señal de pedido hacia el master para pedir un nuevo trabajo. Si no quedan más GOPs, el master cierra al esclavo que se acaba de liberar. Después de servir al esclavo, el master prepara el siguiente GOP para enviar. Mientras que los GOPs crudos se distribuyen, los codificados son puestos en archivos separados por los esclavos. Al final del procesamiento el master trata de concatenar los grupos codificados. Un pedido de un esclavo puede llegar cuando el master trabaja, así, el que llama no puede ser servido inmediatamente. Como el esclavo todavía está trabajando, es probable que se quede en estado inactivo por un período más corto como si estuviera señalado después de terminar.

5. Conclusión

En este trabajo se presentan dos clases de algoritmos paralelos para la compresión de video MPEG. Es de esperar que las porciones de codificación en paralelo introduzcan demasiada comunicación en comparación con la computación. Se espera un mejor resultado con la paralelización a nivel de los GOPs debido a la baja comunicación entre los slave con el master. El sistema se encuentra en desarrollo para una corriente de video MPEG-1 usando librerías de PVM.

6. Bibliografía

[Akl97] Akl S, “Parallel Computation. Models and Methods”, Prentice-Hall, Inc., 1997.
[Bri95] Brinch Hansen, P., “Studies in computational science: Parallel Programming Paradigms”, Prentice-Hall, Inc., 1995.
[Cas95] K. Castleman, "Digital Image Processing", Prentice Hall, 1995
[Cla95] R. J. Clake, "Digital Compression of Still Images and Video", Academic Press 1995.
[Gon92] Rafael C. Gonzáles, Richard E. Woods, "Digital Image Processing", Addison-Wesley Publishing Comp., 1992.
[Hus91] Zahid Hussain, "Digital Image Processing", Ellis Horwood Limited, 1991.
[Hwa93] Hwang K., “Advanced Computer Architecture: Paralelism, Scalability, Programability”, McGraw-Hill, 1993.
[IEEE] Colección de "Computer Graphics and Applications", IEEE.
[IEEE] Colección de "IEEE Transactions on Computers", IEEE.
[IEEE] Colección de "Signal Processing", IEEE.
[IEEE] Colección de Transactions on Parallel and Distributed Processing.
[Jah97] Bernd Jahne, "Digital Image Processing", Springer 1997.
[Jai89] Anil K. Jain, "Fundamentals of Digital Image Processing", Prentice Hall Inc., 1989.
[Lei92] Leighton F. T., “Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes”, Morgan Kaufmann Publishers, 1992.

[Mit96] Joan L. Mitchell, William B. Pennebaker, Chad E. Fogg and Didier J. LeGall, "MPEG Video Compression Standard" Chapman and Hall 1996.
[Mor94] Morse F., “Practical Parallel Computing”, AP Professional, 1994.
[Mur95] A. Murat Tekalp, "Digital Video Processing", Prentice Hall 1995.
[Ort98] Antonio Ortega and Kannan Ramchandran, "Image and Video Compression", IEEE Signal Processing Nov. 1998.
[Pit95] Loannis Pitas, "Digital Image Processing Algorithms", Prentice Hall 1995.
[Red99] Andre Redert, Emile Hendrinks and Jan Biemond,"Correspondence Estimation in Image Pairs", IEEE Signal Processing, May 1999.
[Say96] Khalid Sayood, "Introduction to Data Compression". Morgan Kaufmann 1996.
[Sim97] Sima D, Fountain T, Kacsuk P, “Advanced Computer Architectures. A Design Space Approach”, Addison Wesley Longman Limited, 1997.