JPEG Extendido con Particionamieto y Cuantificación Adaptiva.

Lic. Claudia C. Russo[1]
Lic. Hugo D. Ramón[2]
Ing. Armando De Giusti[3]

LIDI
Laboratorio de Investigación y Desarrollo en Informática
Departamento de Informática - Facultad de Ciencias Exactas
Universidad Nacional de La Plata[4]

Resumen

En este trabajo se presenta la técnica de compresión de imágenes con pérdida basadas en el método usado por el Standard de compresión Joint Photographic Experts Groups (JPEG) extendiendo el método de particionamiento Quadtree y se plantea la utilización de la adaptación en el proceso de cuantificación.

Se analiza el algoritmo JPEG baseline y se proponen optimizaciones que ponen énfasis en el particionamiento y la cuantificación adaptiva.

Los diferentes particionamientos adaptivos son utilizados para superar las dificultades presentadas por el particionamiento fijo del JPEG estándar, la adaptatividad de la cuantificación tiende a mejorar el radio de compresión y la calidad que se obtiene al utilizar cuantificación fija para todos los bloques de la imagen.

[1] Profesor Adjunto Dedicación Exclusiva, Dpto. de Informática, Facultad de Ciencias Exactas, UNLP. Email crusso@lidi.info.unlp.edu.ar
[2] JTP Dedicación Exclusiva, Dpto. de Informática, Facultad de Ciencias Exactas, UNLP. Email hramon@lidi.info.unlp.edu.ar
[3] Director LIDI. Investigador Principal del CONICET. Profesor Titular Dedicación Exclusiva, Dpto. de Informática, Facultad de Ciencias Exactas, UNLP. Email degiusti@lidi.info.unlp.edu.ar
[4] Calle 50 y 115 Primer Piso, La Plata (1900), Argentina, Pcia. de Bs. As. Teléfono 54-21-227707 e-mail lidi@ada.info.unlp.edu.ar

Introducción

Las siguientes son tres características con las cuales se puede juzgar los algoritmos de compresión de imágenes: radio de compresión, velocidad de compresión y calidad de la imagen recuperada. Se pueden determinar cuales de estas tres características predominaran en el algoritmo de compresión que se utilizara en una aplicación.

Radio de compresión: El radio de compresión es la relación entre el tamaño en bytes de la imagen original respecto de la comprimida.

El radio de compresión es un índice de la calidad de la imagen recuperada (descomprimida). Generalmente un alto radio de compresión da como resultado que la calidad de la imagen sea pobre. El *trade-off* entre radio de compresión y calidad de la imagen es importante cuando se piensa en comprimir imágenes.

Algunos esquemas de compresión producen radios que dependen del contenido de la imagen, este aspecto de la compresión se llama dependencias de datos.

Velocidad de compresión: El tiempo de compresión y descompresión se define como la cantidad de tiempo para comprimir y descomprimir imágenes, el valor depende de las siguientes consideraciones:

- La complejidad del algoritmo de compresión.
- La eficiencia del software o hardware que implementa el algoritmo.
- La velocidad del procesador a utilizar.

Calidad de la Imagen: La calidad de la imagen describe la fidelidad con la cual un esquema de compresión de imágenes recrea la imagen original. Se pueden caracterizar estos esquemas como sin perdida o con perdida, dependiendo si se preservan todos los datos o existe perdida de información en ellos. Igualmente los esquemas con pérdida tratan de no perder fidelidad con relación a la calidad de la visión humana.

El objetivo de este trabajo es poner énfasis en el radio de compresión tratando de no alterar la calidad de la imagen; es por ello que se pone hincapié en la adaptatividad del particionamiento y cuantificación del método JPEG utilizado para comprimir.

Joint Photographic Experts Groups (JPEG) es uno de los standards más conocidos para compresión de imágenes con pérdida. Se obtiene buen radio de compresión al aplicarlo sobre fotografías, trabajos de arte y material similar, aunque no ocurre así con textos, dibujos simples o líneas. JPEG explota las limitaciones del sistema visual humano [Cla95] [Nels91].

JPEG define tres sistemas de codificación diferentes:

- Un sistema de codificación baseline con pérdida, que utiliza como base la Transformada del Coseno Discreta (DCT)

- Un sistema de codificación extendido para aplicaciones con requerimientos de más precisión, de reconstrucción progresiva, etc.
- Un sistema de codificación independiente sin pérdida para compresión reversible.

El standard propone una sintaxis que debería cumplir cualquier secuencia de bits para ser JPEG, dejando *amplias libertades* en las etapas de cuantificación y codificación, de manera de poder efectuar mejoras y optimizaciones. Una propiedad útil del método es que el grado de pérdida puede ser variado, ajustando parámetros de compresión, que se explicarán mas adelante. Además existen varias variantes a la transformada DCT:

- *DCT en bloques:* este esquema es el que utiliza el método JPEG standard la imagen de longitud NxM es dividida en sub-bloques de longitud de nxm. Por ejemplo en JPEG los bloques son de 8x8.
- *Imagen completa:* El DCT en bloques introduce efectos de blocking en la imagen restaurada, con lo cual se puede utilizar el método de imagen completa para evitar este problema. Toma toda la imagen como un bloque. Esta forma mantiene mayor fidelidad con la imagen original.
- *Adaptiva a las formas:* Una vez identificado el objeto de interés de la imagen, este algoritmo codifica el primer plano y el fondo en forma independiente, empleando el SADCT en bloques de 8x8 pixels. Para cada bloque una máscara indica cuáles pixels pertenecen a que plano y solo los del primer plano se codifica, obviamente esta máscara es conocida por el codificador y decodificador.

Descripción del algoritmo JPEG baseline

El JPEG standard, está diseñado para comprimir tanto imágenes color como en escala de grises de escenas naturales del mundo real. Funciona bien sobre fotografías, trabajos artísticos naturalistas, y material similar. El JPEG maneja sólo imágenes quietas, pero hay un estándar relacionado llamado MPEG para imágenes en movimiento.

En el estándar del JPEG se incluye un método de compresión sin pérdida, pero virtualmente nadie lo ha adoptado por los radios de compresión obtenidos. Sin embargo, la compresión JPEG con pérdida ha acumulado un amplio apoyo, y la mayoría de los browsers de Web gráficos ahora la utilizan.

Con la compresión JPEG con pérdida hay un *trade-off* entre el tamaño del archivo y la calidad de la imagen. Se pueden lograr radios de compresión muy altos - del orden de 100:1 o más - pero a expensas de una notoria pérdida de calidad (efecto de bloques en la imagen). A radios de compresión más bajos, la degradación de la calidad de la imagen es mucho menos notable, y la mayoría de los observadores están de acuerdo en que las radios de alrededor 10:1 producen un resultado que es visualmente indistinguible del original.

La mayoría de los compresores JPEG permiten seleccionar el nivel de *trade-off* de tamaño del archivo versus calidad de la imagen seleccionando un seteo de la calidad.

Parece haber una confusión generalizada acerca del significado de estos settings. *Calidad 95* NO significa *mantener el 95% de la información*, como algunos han dicho. La escala de calidad es exclusivamente arbitraria; no es un porcentaje ni nada por el estilo.

El sistema recomendado por JPEG es una modificación al propuesto por Chen y Pratt y se basa en la técnica de codificación utilizando DCT. El proceso de compresión se realiza en tres pasos secuenciales [Say96]:

- **Transformación**: La transformada que se utiliza es la DCT.
- **Cuantificación**: Utiliza cuantificación MIDTREAD uniforme para cuantificar los coeficientes de un bloque transformado.
- **Codificación**: Los coeficientes DC y AC resultantes de la cuantificación son codificados utilizando códigos de longitud variable.

Generalización a particionamiento fijo

La primer implementación es una generalización para particionar la imagen en bloques de tamaños fijos (4x4, 8x8, 16x16, 32x32), Se utilizan diferentes tablas de cuantificación y codificación: para particionamiento fijo con bloques de 8x8, se utilizó la matriz de cuantificación default y los distintos niveles de pérdida se lograron multiplicando esta matriz por un escalar o factor; las tablas de codificación Huffman para bloques de 8x8 son las provistas por el standard. Para los demás tamaños de bloques se construyó la siguiente matriz de cuantificación: =1+(1+i+j)*factor; 0<= i, j < N.

Si bien existen técnicas para construir tablas de cuantificación adaptivas, el criterio utilizado es que generalmente los coeficientes decrecen en importancia desde del vértice superior izquierdo hacia el inferior derecho. Por esta razón la matriz se construye de tal forma que los coeficientes mas cercanos a la posición (0,0) conserven mayor precisión a la hora de ser decuantificados. Las tablas de códigos Huffman se construyeron realizando un recorrido sobre diversas imágenes sobre las que se aplicaron distintos grados de pérdida. Estos recorridos permitieron extraer las probabilidades para los símbolos, que se eligieron de acuerdo al tamaño del bloque.

Particionamiento Quadtree

Este particionamiento es utilizado en el procesamiento de imágenes, tratando de superar las dificultades presentadas en el particionamiento fijo. El algoritmo básico consiste en tomar un bloque cuadrado de imagen el cual es dividido en cuatro subloques cuadrados de igual tamaño. Esto se repite recursivamente desde la imagen original completa hasta que los cuadrados alcancen un tamaño deseado de acuerdo a un test de decisión, que dependerá de la aplicación. Se puede realizar el mismo particionamiento pero en forma *button up* comenzando el proceso con los subloques más chicos y uniéndolos para formar otros más grandes.

Generalización utilizando particionamiento Quadtree

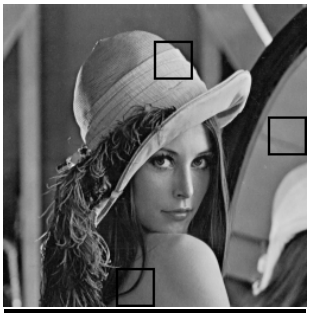
Con Quadtree se trata de adaptar las diversas zonas de la imagen a un determinado tamaño de bloque, de acuerdo a las variaciones de los niveles de grises dentro del mismo. Zonas grandes con cambios lentos en los niveles de detalle, son tratadas con bloques mas grandes y zonas con muchos detalles sean tratadas con bloques mas pequeños. Esta implementación se basa en un esquema propuesto por Chen (1989) en el cual la imagen es dividida en bloques iniciales de 32x32, los que a su vez son particionados completamente hasta obtener bloques de 4x4. Se aplica un test sobre 4 subloques hermanos de 4x4 para determinar si estos se procesan individualmente o bien son unidos para formar un posible bloque de 8x8. El mismo proceso se realiza para cada 4 bloques hermanos de 8x8, y 16x16.

El test es de la forma:

```
if abs(Mk(i)-Mk(j))>Tk    para cualquier i=j
        Procesar los 4 subloques individualmente;
Else
        Unir los subloques y marcarlos como un posible más grande;
```

Los términos Mk(i) y Mk(j) (i,j=1..4) representan promedios de niveles de grises de bloques hermanos en el k-ésimo nivel del árbol y Tk, el límite de decisión. La idea es que para determinar cuan activa es una zona de NxN la dividimos en 4 bloques iguales, hallamos sus promedios y vemos que tan similares son. Si no existe ningún par de promedios cuya diferencia supere un límite establecido previamente entonces consideramos a la zona de NxN como poco activa (un límite sugerido es 10 para imágenes de 8 bits) y la procesamos como un solo bloque. En caso contrario la dividimos. Para representar la información del header solamente necesitamos transmitir un bit por cada subrogue indicando si este se particiona o no, con lo cual en el peor de los casos codificaremos 21 bits por cada bloque de 32x32 equivalente a un costo de 0.02 bits por pixel. De acuerdo al tamaño de cada subbloque del particionamiento final (4x4, 8x8, 16x16 o 32x32) el procesamiento es similar al aplicado por JPEG a cada bloque de 8x8 con la diferencia de que el coeficiente DC solo se codifica con respecto al mismo elemento del bloque anterior si es que ambos bloques tienen el mismo tamaño. Si el bloque anterior es de tamaño diferente se codifica el mismo valor del coeficiente cuantificado. Se pueden variar los factores de pérdida para los distintos tamaños de bloques de acuerdo al nivel de calidad deseado para las distintas zonas. Por ejemplo, si se desea aumentar la pérdida en las zonas con mucha actividad lo mas conveniente es aumentar el factor de pérdida para los bloques mas chicos, debido a que es de esperarse que estos tamaños de bloques caigan en esas zonas. De la misma manera que si se quiere disminuir la calidad de la imagen en zonas grandes con poca actividad se debe ajustar los factores para los tamaños de bloques mas grandes. En general las zonas con poca actividad forman parte del fondo de la imagen y en consecuencia serán procesadas con bloques mas grandes (32x32, 16x16).

Tres ejemplos de secuencias de pasos que se siguen para particionar un bloque de 32x32



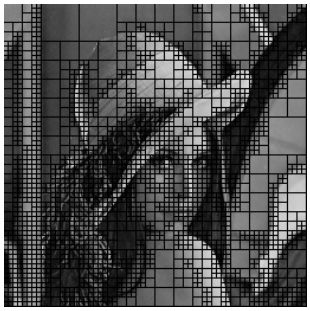
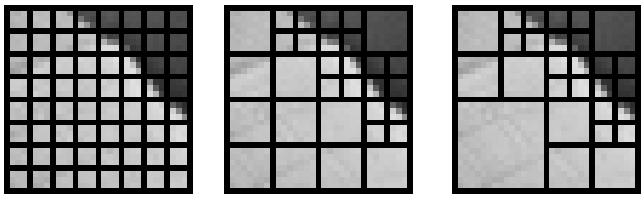
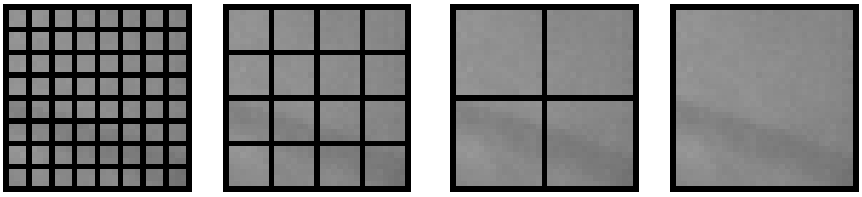


Imagen particionada con limites de 20,20,20

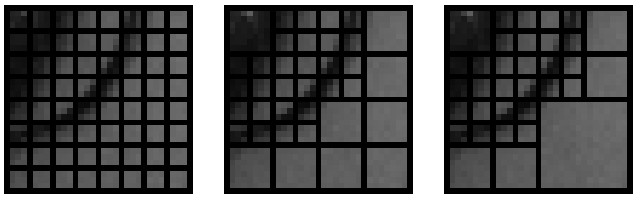
Bloque del sombrero



Bloque de espejo



Bloque del omoplato



En síntesis esta técnica de compresión cuenta con los siguientes parámetros que pueden ajustarse: Límite de decisión para decidir si particionar o no un bloque de 8x8, 16x16, 32x32; factor de pérdida para bloques de 4x4, 8x8, 16x16 y 32x32.

Observaciones y algunos resultados del método adaptivo

Por razones de espacio, en este artículo se analizan los resultados con 2 imágenes de características muy diferentes. Los resultados obtenidos de la imagen VLSI.BMP y LENA.BMP se lograron con un particionamiento de 10 y 20 respectivamente para cada uno de los tres limites. Con el particionamiento de la primer imagen se logró que las zonas de la fórmula fueran tratadas con bloques mas chicos (4x4 y 8x8), mientras la mayor parte de la

zona oscura fuera procesada con bloques de 32x32 y en menor cantidad con los de 16x16. Con esta distribución de bloques solo se produjeron distorsiones en los lugares muy cercanos a la fórmula y solo al aplicar factores muy altos. En las siguientes tablas se muestran algunos resultados obtenidos para las imágenes.

Si bien se puede aumentar el valor de los factores de pérdida para los bloques de 8x8, 16x16 y 32x32, sin distorsión visible para las zonas donde estos caen, estos incrementos no se ven reflejados en los tamaños de las imágenes comprimidas.

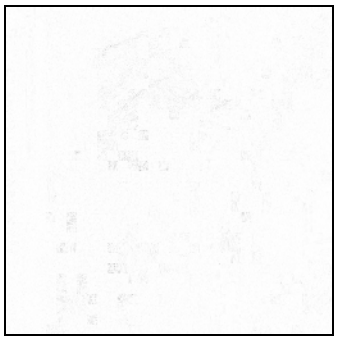
Tabla para VLSI.BMP

Descomprimida	Radio	Radio Factor 4x4	Radio Factor 8x8	Radio Factor 16x16	Radio Factor 32x32
pv1_1.bmp	6.4	2	2	2	2
pv1_2.bmp	9.9	9	2	2	2
pv1_3.bmp	9.98	9	9	2	2
pv1_4.bmp	11	15	9	2	2
pv1_5.bmp	13	20	9	2	2
pv1_6.bmp	14.4	25	9	2	2
pv1_7.bmp	14.5	25	9	9	9

Tabla para LENA.BMP (con valores de 20 para los límites de decisión en el particionamiento)

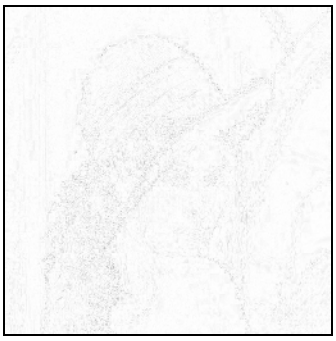
Descomprimida	Radio	Radio Factor 4x4	Radio Factor 8x8	Radio Factor 16x16	Radio Factor 32x32
plena4_1.bmp	4	2	2	2	2
plena4_2.bmp	8.3	9	2	9	9
plena4_3.bmp	6.4	5	3	5	5
plena4_4.bmp	9	10	3	10	10
plena4_5.bmp	10.8	15	2	15	15
plena4_6.bmp	11.18	15	3	15	15
plena4_7.bmp	11.4	15	4	15	15
plena4_7.bmp	13.16	20	4	15	15
plena4_7.bmp	13.3	20	5	15	15

A continuación se muestran la perdida producida como la diferencia entre la imagen original y la comprimida:



Error de plena4_1.bmp

Error de plena4_5.bmp



Error de plena4_9.bmp

Esquema para la cuantificación adaptiva

Explota el hecho de que las imágenes a tratar no ocupan la totalidad de la imagen, la idea es dada una imagen reconocer sus formas más importantes [Fáb96].

Se utilizará una variante de la transformada del coseno [Sik95] [Sik96] llamada DCT adaptiva a la forma (SADCT).

SADCT es un esquema basado en bloques diseñado especialmente para codificar regiones de formas arbitrarias y solamente se utilizan los pixels que pertenecen a la región de interés.

Una vez identificado el objeto de interés de la imagen, este algoritmo codifica el primer plano y el fondo en forma independiente, empleando el SADCT en bloques de 8x8 pixels. Para cada bloque una máscara indica cuáles pixels pertenecen a que plano y solo los del primer plano se codifica, obviamente esta máscara es conocida por el codificador y decodificador.

La DCT para el primer plano se calcula en dos pasos, ambas involucra transformaciones en una dimensión. Primero se debe calcular la DCT vertical, transformando cada columna del pixels del primer plano. Luego se calcula la DCT horizontal, la cual transforma cada fila de coeficientes obtenidos en el primer cálculo.
Para calcular la DCT vertical, cada columna del primer plano es subida hasta justificarse al borde superior del bloque. Para cada columna i, 0<i<9, que contiene m', 0<m'<9 pixels del primer plano, la DCT vertical se calcula como:

$$DCT_{m'}(i,v) = C_0 \cos\left[i\left(v + \left(\frac{1}{2}\right)\right)\left(\frac{\pi}{m'}\right)\right]$$

donde v = 0,....,m'-1

Aquí $C_0 = (1/2)^{½}$ si p = o ó C_0 =1 en otro caso. Los coeficientes c_j de cada vector columna con $X = [x_j]^t$ se calcula como:

$$c_j = \left(\frac{2}{m'}\right) DCT_{m'} * x_j$$

se debe notar que m’ puede ser diferente para cada columna.

$$DCT_{n'}(u,v) = C_0 \cos\left[u\left(v + \left(\frac{1}{2}\right)\right)\left(\frac{\pi}{n'}\right)\right]$$

La DCT horizontal se calcula de cada fila de coeficientes obtenidos desde el paso anterior, primero cada fila la shifteamos a izquierda, para que queden justificados a izquierda, el calculo para cada fila es:

donde v = 0,....,n’-1

y los coeficientes para la fila se calcula como:

$$c_i = \left(\frac{2}{n'}\right) DCT_{n'} * x_i$$

n' puede ser diferente para cada fila.

Luego de la transformación, el número de coeficientes de DCT es la misma que el número de pixels que forma el primer plano. Los coeficientes están localizados en el córner superior izquierdo del bloque, como ocurre en cualquier DCT basados en bloques.

Conclusiones y Trabajos Futuros

En el JPEG adaptivo, se observó que en imágenes con grandes zonas con poca actividad en niveles de grises permitían radios de compresión mayores si eran procesadas con bloques más grandes produciendo distorsiones poco visibles en las zonas poco activas, pero con el costo de pérdidas notables de calidad en zonas con poca actividad adyacentes a otras con mucho nivel de detalle. Con la reducción de los tamaños de bloques esas zonas conflictivas se fueron restringiendo pero con la desventaja de la disminución en los radios de compresión. Los excelentes resultados comparativos del esquema adaptivo respecto del fijo, nos han llevado a encarar la paralelización del algoritmo adaptivo, a fin de tener tiempos razonables para procesamiento en tiempo real.

En JPEG el umbral de pérdida visible es a menudo alrededor de una compresión de 5:1 para una escala de grises. El umbral exacto en el que los errores se vuelven visibles depende de sus condiciones de vista. Cuanto más pequeño sea un pixel individual, más difícil será ver un error; por lo tanto los errores son más visibles en una pantalla de computadora (a unos 70 puntos por pulgada) que en una impresora de alta calidad (300 o más puntos por pulgada). De esta manera, una imagen de resolución más alta puede tolerar más compresión, lo que es afortunado considerando que es mucho más grande desde el

comienzo. Las relaciones de compresión citadas más arriba son típicas para una visualización por pantalla. También debe notarse que el umbral de error visible varía considerablemente de una imagen a otra.

En ambos modelos, sería bueno si, habiendo comprimido una imagen, se pudiera descomprimir, manipular (cortar un borde, por ejemplo), y recomprimirla sin otra degradación de la imagen más que lo que se perdió inicialmente. Desdichadamente *este no es el caso*. En general, la recompresión de una imagen alterada pierde más información. Por lo tanto, es importante minimizar la cantidad de generaciones JPEG (cualquiera de los dos modelos) entre la versión inicial y la versión final de una imagen. Si se descomprime y recomprime una imagen con el mismo seteo de calidad que se usó primero, ocurre muy poca o nula degradación adicional. Esto significa que se le pueden hacer modificaciones locales a una imagen JPEG sin una degradación material de las otras áreas de la imagen. (Sin embargo, las áreas que cambian sí se degradarán). Contrariamente a lo que indicaría la intuición, esto funciona mejor cuanto más bajo sea el seteo de calidad.

Actualmente se están investigando implementaciones adaptivas sobre arquitecturas multi-DSP orientadas a procesamiento de imágenes.

Bibliografía

[Cla95] Roger Clarke, *Digital Compression of Still Images and Video*, Academic Press, 1995.

[Dou87] E. Dougherty, C. Giardina, *Matrix Structured Image Processing*, Prentice-Hall Inc., 1987.

[Fol90] Foley, van Dam, Feiner, Huges, *Computer Graphics*, Addison-Wesley,1990.

[Glas95-1] Andrew Glassner, *Principles of Digital Image Synthesis. Vol. 1*, Morgan Kaufmann Publishers, 1996.

[Glas95-2] Andrew Glassner, *Principles of Digital Image Synthesis. Vol. 2*, Morgan Kaufmann Publishers, 1996.

[Gon92] R. Gonzales, R. Woods, *Digital Image Processing*, Addison-Wesley, 1992.

[Jai89] Anil K. Jain, *Fundamentals of Digital Image Processing*, Prentice-Hall Inc., 1989.

[Nel91] Mark Nelson, *The Data Compresión Book*. Prentice Hall, 1991.

[Say96] Khalid Sayood, *Introduction to Data Compression*, Morgan Kaufmann Publishers, 1996.

[Fáb96] C. Solá Fábregas, N. P. Tri, *Ultrasound Image Coding using Shape-Adaptive DCT and Adaptive Quantization*, Signal Processing Laboratory, Swiss Federal Institute of Technology.

[Sik95] T. Sikora, B. Makai, *Shape Adaptive DCT for Generic Coding Video*, IEEE transaction on Circuits and Systems for Video Technology, Vol. 5, Nro. 1.

[Sik96] T. Sikora, *Low Complexity Shape-Adaptive DCT for Coding of Arbitrarily Shaped Image Segments*, Signal Processing: Image Communication, 1995.