Universitat Autònoma de Barcelona

Escola Tècnica Superior d'Enginyeria

Enginyeria Informàtica

Estudio para la extracción automatizada de contextos de significancia en JPEG2000

Memoria del proyecto de final de carrera correspondiente a los estudios de Ingeniería Informática presentado por Alberto Arcos Hurtado y dirigido por Francesc Aulí Llinàs.

Bellaterra, Junio del 2007

El firmante, Francesc Aulí Llinàs, profesor de la Universidad Autónoma de Barcelona

CERTIFICA:

Que la presente memoria ha sido realizada bajo su dirección por Alberto Arcos Hurtado

Bellaterra, Junio del 2007

Firmado: Francesc Aullí Llinàs

Agradecimientos

Quiero agradecer la paciencia y gran trabajo que ha realizado Cesc, guiándome para llevar a buen puerto esta investigación. A mis compañeros Sera y Mayte por amenizar las horas de trabajo a golpe de cafés, a Jordi por su compañía en todo momento, espero que aunque cada uno siga su camino en la vida, sigamos cuidando la amistad. A todos los compañeros que me han acompañado durante este ciclo universitario.

Y sin duda alguna, agradecimientos a mi niña, no podría aprender de nuevo a vivir sin ella sin morir en el intento, un largo camino hemos recorrido juntos pero más nos queda por descubrir. Agradecimientos a mi familia por su apoyo siempre incondicional.

Índice general

Capítulo 1

Introducción

1.1. Planteamiento general

La compresión de imágenes está directamente relacionado con la Informática, a medida que esta ha ido evolucionando, a la par ha evolucionado la compresión. Inicialmente la codificación era de texto (ASCII), era sencilla y con buenos resultados. Pero las imágenes han ido ganando terreno a los textos, de ahí viene la necesidad de comprimir las imágenes, buscando reducir la cantidad de información a transmitir o almacenar para representar una imagen.

En la compresión, juega un papel importante la redundancia de datos. Una imagen digital se compone matemáticamente de una distribución bidimensional de pixels y se quiere transformar en un conjunto de datos estadísticos sin correlacionar, este proceso se llama compresión.

Hace unos 25 años que se inicio la compresión de imágenes, inicialmente mediante métodos analógicos que dejaron paso a métodos digitales, la creación de estándares internacionales ayudó el avance en este campo. Un detalle que no pasa desapercibido a ojos de cualquiera, es la importancia que está tomando día tras día la compresión multimedia, englobada en esta queda la compresión de imágenes.

La utilización de imágenes comprimidas está muy extendida, algunas de sus aplicaciones son para uso de consumo (cámaras digitales, ...), profesional (arte gráfico, publicaciones,...), médico, trabajo remoto, Internet, escaneo e impresión, bases de datos, móviles, ... Atendiendo esta necesidad surgieron varios algoritmos de compresión, cada unos de los

cuales con características concretas para cubrir ciertas necesidades, de los cuales a día de hoy cabe destacar el estándar JPEG y como sucesor el JPEG2000.

El trabajo realizado se centra en la compresión de imágenes, mediante el estándar JPEG2000. Es un trabajo de investigación donde se quiere indagar en el estándar de codificación, buscando posibles mejoras de este. Se realiza un estudio del algoritmo de compresión y se centra en la utilización de contextos para realizar la compresión.

El proceso de codificación del estándar JPEG2000 se puede dividir en varias fases, durante las cuales se realizan transformaciones de los valores representativos de una imagen, con la finalidad de preparar estos valores para la fase de compresión, donde se realice una codificación adecuada a las opciones deseadas.

Unas de las transformaciones que se realiza justo antes de la compresión, está directamente relacionada con la representación de los contextos. Los contextos son una representación que aporta información al compresor sobre cada uno los valores que ha de codificar. La utilización de estos contextos estipulados por el estándar, son totalmente estáticos y se espera que recalculando estos contextos obtendremos buenos resultados para casi cualquier tipo de imagen que se procese. Todo este trabajo de investigación realizado gira alrededor de esta idea.

Es fácil pensar que si una imagen puede diferir en gran medida de otra, los contextos que permiten una buena codificación de la primera imagen, pueden diferir también de los que mejor codifican la segunda. Por lo tanto, ¿Cabe la posibilidad de extraer los contextos personalizados para cada tipo de imagen procesada? ¿Es posible automatizar este proceso? Si bien a priori no son preguntas que se puedan responder, las conclusiones a las que se llegan tras finalizar este trabajo de investigación nos serán de gran ayuda.

1.2. Estructura de la memoria

A continuación se especifica de que manera ha sido estructurada la memoria.

En el Capítulo 2 se explica como funciona el estándar de compresión JPEG2000, con una descripción general seguida de la introducción al motor de compresión y cada una de las etapas que lo componen.

En el Capítulo 3 se centra en la etapa de codificación, entrando en detalle en la relación

que hay entre los contextos utilizados en los planos de bits durante el proceso de codificación y el nivel de compresión que obtenemos utilizando esos contextos en el codificador aritmético. Para poder trabajar con los contextos, hay que clarificar como actualmente se codifican en el estándar JPEG2000. Se realizan varias propuestas para la configuración de contextos, valorando cual de ellas es la más acertada.

En el Capítulo 4 se podrá entrar en detalle en la investigación realizada, con la finalidad de automatizar los contextos, se explicará la configuración de contextos más adecuada acompañada con una implementación práctica. Se especifican diferentes algoritmos que se utilizan para extraer los mejores contextos y formar un mapa de contextos para la codificación. Se realiza una valoración de los mapas de contextos para realizar comparaciones entre ellos. Todo la explicación irá acompañada de ejemplos del funcionamiento y estudiando los resultados obtenidos.

El último capítulo se realiza una valoración de todo el trabajo realizado, aportando conclusiones y futuras líneas de investigación que quedan abiertas tras recorrer este camino

Capítulo 2

El estándar de compresión JPEG2000

JPEG2000 es un sistema de codificación de imágenes que consigue un nivel de compresión superior y gran cantidad de posibilidades a la hora de escalabilizar y flexibilizar la codificación, dotando de nuevas funcionalidades que se ajustan a las necesidades de diferentes aplicaciones o usos que sean necesarios. A continuación se muestra una descripción general del funcionamiento del estándar JPEG2000 y la codificación que utiliza. Tras esto se entra en detalle sobre el motor de compresión, parámetros influyen en la etapa de codificación y se encamina hacia el marco donde se ha centrado el trabajo de investigación.

2.1. Descripción general

La motivación sobre la creación del JPEG2000 viene derivada de intentar suplir las carencias que tiene el estándar JPEG, extendido y reconocido estándar que permitía una compresión simple con un nivel de compresión elevado. Para el desarrollo del JPEG2000 se dividió este en varias partes:

- *Parte 1:* Corazón del sistema de compresión.
- *Parte 2:* Extensiones.
- *Parte 3:* Movimiento en JPEG2000.
- *Parte 4:* Test de verificación.

- *Parte 5:* Software de referencia, engloba diferentes implementaciones (en C y en Java) que permiten la codificación del JPEG2000.
- *Parte 6:* Formato de imagen compuesto, necesario para la creación de documentos de imagen, preimpresiones, aplicaciones de fax, etc...
- *Parte 7:* Guía de las mínimas funciones de soporte.
- *Parte 8:* JPSEC, seguridad en JPEG2000.
- *Parte 9:* JPIP, herramientas interactivas, API's y Protocolos.
- *Parte 10:* JP3D, imágenes volumétricas (3-D y punto flotante).
- *Parte 11:* JPWL, aplicaciones wireless.
- *Parte 12:* Formato de archivo multimedia.

Parte de estos puntos ya están investigados y desarrollados o en fases finales, cuando se complete todos los puntos especificados se dispondrá de un sistema de codificación, tratamiento, transmisión y almacenamiento de imágenes con un gran potencial apoyado por las últimas innovaciones del momento. En 1997 se inició, por parte del Joint Photographic Experts Group (JPEG), el proceso de desarrollo de la primera parte del JPEG2000, este proceso finalizó durante el año 2000. Ya entonces, aparecieron algunas de las características del JPEG2000:

- Mayor índice de compresión: 20 % por encima del estándar JPEG.
- Representación con resoluciones múltiples: Fruto del uso de la transformada wavelet surgen las subbandas, que entre otras características, permiten operar con la imagen original en diferentes resoluciones.
- Transmisión progresiva: JPEG2000 permite la priorización de regiones de la imagen mediante diferentes técnicas.
- Lossless y lossly compresión: Permite compresión con o sin pérdidas (reversible o no reversible).

- Acceso aleatorio al codestream y procesamiento: La codificación que se realiza nos permite obtener una tira de bits que podrá reorganizarse según las necesidades.
- Robusto frente a errores de bit.
- Codificación secuencial.
- Formato de archivo flexible: El formato de archivo JP2 y JPX permiten guardar información, metadata, ...

2.2. El motor de compresión

A continuación se detalla el funcionamiento del motor de compresión del estándar JPEG2000. Tener una visión sobre el funcionamiento de la metodología en la codificación y compresión nos permitirá situar en el contexto concreto esta investigación. La explicación se acompaña con un ejemplo para que sea más comprensible.

Durante el proceso de codificación se realizan una serie de transformaciones a las muestras de la imagen, estas transformaciones son las que nos permitirán conseguir un índice alto de compresión, traduciendo las muestras al codificador aritmético, dando más importancia a la hora de transmitir a ciertas regiones de la imagen sobre otras. En la tabla 2.1 se puede ver un ejemplo de coeficientes de una imagen.

99	94	81	63	90	112	110	117	127	129	131	134	133	133	117	123
101	94	90	81	78	79	79	91	123	130	129	134	133	132	114	122
91	80	103	132	78	50	58	59	111	129	128	131	131	133	118	123
86	80	114	161	87	49	50	48	108	135	124	133	133	131	118	123
80	76	102	145	78	47	56	43	109	132	124	132	133	133	116	128
79	69	80	115	64	42	53	44	106	132	121	129	132	131	116	125
77	63	69	97	61	40	51	42	100	132	117	129	132	128	119	125
77	61	60	80	49	37	54	38	95	130	114	129	131	128	120	124
77	62	54	46	39	34	45	43	92	129	111	126	130	128	118	125
80	67	47	44	39	31	45	39	88	127	105	123	131	127	118	125
77	64	55	43	39	40	42	38	84	123	108	121	128	127	119	121
77	69	52	44	35	33	48	39	81	120	104	121	124	131	118	121
79	72	55	47	41	40	47	41	90	116	111	124	122	130	113	122
73	68	57	43	45	38	50	41	93	115	126	143	117	131	115	122
73	64	53	48	43	39	51	38	104	114	146	152	114	133	114	123
64	63	50	39	47	34	53	40	100	120	150	161	111	136	115	123

Cuadro 2.1: Ejemplo de coeficientes de una imagen

2.2.1. Desplazamiento de niveles

Partiendo de las muestras que se reciben, si su valor tiene representación sin signo, se realiza una resta de 2^{B-1} a cada uno de los valores, donde B es el número de bits

que utilizamos para codificar las muestras. Realizando esta transformación se obtiene un conjunto de muestras que estarán comprendidas entre $[-2^{B-1}, 2^{B-1})$.

Aplicando el desplazamiento de niveles a la tabla 2.1 y utilizando muestras con 8 bits (con 8 bits podemos representar valores entre $(0, 2^8 - 1]$) si restamos $2^7 = 128$ a todos los valores obtendremos que todos los valores esté distribuidos simétricamente alrededor del 0. Se obtienen los valores de la tabla 2.2.

-29	-34	-47	-65	-38	-16	-18	-11	-1	1	3	6	5	5	-11	-5
-27	-34	-38	-47	-50	-49	-49	-37	-5	2	1	6	5	4	-14	-6
-37	-48	-25	4	-50	-78	-70	-69	-17	1	0	3	3	5	-10	-5
-42	-48	-14	33	-41	-79	-78	-80	-20	7	-4	5	5	3	-10	-5
-48	-52	-26	17	-50	-81	-72	-85	-19	4	-4	4	5	5	-12	0
-49	-59	-48	-13	-64	-86	-75	-84	-22	4	-7	1	4	3	-12	-3
-51	-65	-59	-31	-67	-88	-77	-86	-28	4	-11	1	4	0	-9	-3
-51	-67	-68	-48	-79	-91	-74	-90	-33	2	-14	1	3	0	-8	-4
-51	-66	-74	-82	-89	-94	-83	-85	-36	1	-17	-2	2	0	-10	-3
-48	-61	-81	-84	-89	-97	-83	-89	-40	-1	-23	-5	3	-1	-10	-3
-51	-64	-73	-85	-89	-88	-86	-90	-44	-5	-20	-7	0	-1	-9	-7
-51	-59	-76	-84	-93	-95	-80	-89	-47	-8	-24	-7	-4	3	-10	-7
-49	-56	-73	-81	-87	-88	-81	-87	-38	-12	-17	-4	-6	2	-15	-6
-55	-60	-71	-85	-83	-90	-78	-87	-35	-13	-2	15	-11	3	-13	-6
-55	-64	-75	-80	-85	-89	-77	-90	-24	-14	18	24	-14	5	-14	-5
-64	-65	-78	-89	-81	-94	-75	-88	-28	-8	22	33	-17	8	-13	-5

Cuadro 2.2: Ejemplo de desplazamiento de niveles

2.2.2. Transformada de color

En el caso de que los tres primeros valores de la imagen que se procesa, tienen la misma medida, se les asigna a estas muestras la representación de canal rojo, verde y azul respectivamente (x_R, x_G, x_B). La transformada de color, nos permite realizar una conversión de estos tres canales a otros tres canales que nos aportan diferente información, el primero la luminosidad de la imagen (x_Y), el segundo y tercero la información del color, la crominancia respecto al rojo o azul.

Realizar esta transformación nos puede proporcionar un 20 % de compresión. Según el tipo de compresión que queramos realizar, reversible o no reversible podemos utilizar diferentes tipos de transformada de color. En el caso de que los valores que se reciben sean de un solo componente, esta transformada no es necesaria.

2.2.3. Transformada wavelet

La transformada wavelet es un proceso por el cual, podemos separar de la imagen original, las muestras que tengan altas y bajas frecuencias, agrupándolas en subbandas.

El funcionamiento es el siguiente: Seleccionamos un tamaño de muestras a procesar (según el tipo de transformada utilizado), para cada una de las líneas de valores que componen la imagen y para cada grupo de muestras según el tamaño, aplicamos un filtro pasaaltas y pasabajas.

La imagen resultante se divide en dos zonas, una primera zona que va de la zona izquierda al centro de la imagen y una segunda zona que va de la zona central a la derecha. El resultado del filtro pasabajas se almacena en la parte izquierda-centro de la imagen y el resultado del filtro pasaaltas se almacena en la parte centro-derecha de la imagen.

Este proceso permite que en la zona izquierda de la imagen resultante tenga los valores de frecuencias bajas, mientras que en la zona derecha están los valores con las frecuencias altas. Concretamente estas 2 zonas se dividen en 4 zonas o subbandas, subbanda LL, subbanda LH, subbanda HL y subbanda HH, se puede ver en la imagen 2.1 la posición que les corresponde a cada una de estas subbandas.

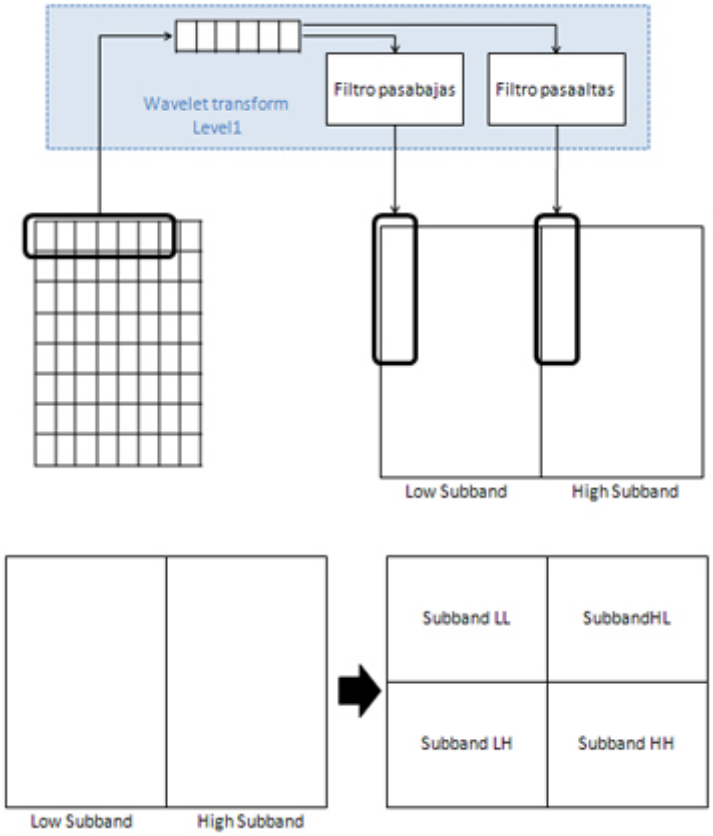

Figura 2.1: Funcionamiento de la transformada wavelet de nivel 1.

Esta primera imagen resultante que se obtiene es la transformada wavelet de nivel 1.

Se puede reiterar el proceso concentrándonos de nuevo en la zona superior izquierda y se obtendría la transformada wavelet de nivel 2 (imagen 2.2), pasando de tener 4 zonas a tener 8 zonas. Y sucesivamente repetir este proceso la cantidad que veces que se necesite según el nivel que se quiera obtener.

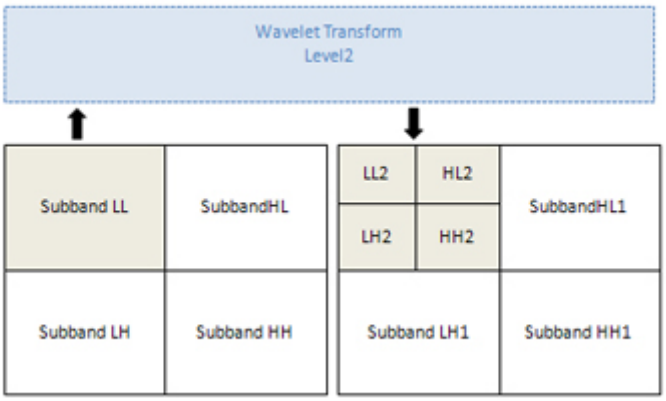

Figura 2.2: Funcionamiento de la transformada wavelet de nivel 2.

La información que se obtiene de cada subbanda es la siguiente, en la subbanda HL1 los detalles horizontales, en la subbanda LH1 los detalles verticales y finalmente en la subbanda HH1 los detalles diagonales. En la subbanda LL1 (o LL) se encuentra la imagen original con una resolución más pequeña, concretamente 1/4 de la resolución original, esta subbanda se denomina subbanda residual. Si se procesa LL1 con una transformada wavelet para obtener un segundo nivel, la subbanda LL1 pasa a dividirse en 4 zonas más, LL2, HL2, LH2 y HH2. Que contiene la información correspondientemente detallada con anterioridad.

2.2.4. Cuantificación

La cuantificación es una etapa que es irreversible, porque se producen pérdidas de información, así que en el caso de que se quiera realizar una compresión sin pérdidas, se prescindirá de esta etapa. La cuantificación permite, que los valores que se reciben de la transformación wavelet estén dentro de unos rangos que nos interesa, para luego ser utilizados en el codificador.

El tipo de cuantificación que se utiliza es un escalar con zona muerta. El rango de valores posibles se divide en partes iguales, exceptuando en la de alrededor del cero cuyo rango tiene el doble de tamaño que el resto. Con esto conseguimos que los valores cercanos a 0, se redondeen como 0 y no se transmitan. La medida del paso ($\triangle$) es un parámetro que

se tiene que especificar en esta etapa y depende de la subbanda en la que se encuentra el coeficiente. En la imagen 2.3 se puede ver los rangos de cuantificación según la medida del paso.

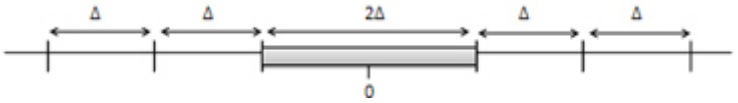

Figura 2.3: Cuantificación

2.2.5. Regiones de interés

JPEG2000 es capaz de priorizar ciertas regiones de una imagen. Para ello dispone de dos mecanismos, el primero encarado al proceso de codificación y reorganización posterior y el segundo mediante modificaciones de los coeficientes. Este primer mecanismo hace uso de la división de bloques, priorizando el procesamiento de los bloques que más importancia se le quieran dar. Cuando estos bloques se procesen y se obtengan una tira de bits, estos se transmitirán antes que los bits codificados del resto de bloques. La escalabilidad depende del tamaño de los bloques.

El segundo método que se puede utilizar para priorizar regiones de la imagen es mediante la modificación de los coeficientes tranformados. Este método no tiene limitación alguna por tamaño de bloque, pero si tiene limitación en que una vez realizada las modificaciones para priorizar una región de la imagen, no puede cambiarse la prioridad con solo reorganizar la tira de bits que se transmite, cosa que el primer mecanismo mencionado si que permite.

La priorización por modificación de coeficientes, se realiza multiplicando por un factor 2^U los coeficientes de la imagen que se desean priorizar, como veremos a continuación la codificación se realiza por planos de bits, como la codificación se inicia desde los planos de bits superiores a los inferiores y al multiplicar esos coeficientes conseguimos que estén en planos de bits superiores, indirectamente priorizamos su codificación.

Una técnica que te permite este tipo de priorización es el escalado, permite escoger el factor escalaje U (2^U) según el coeficiente y la zona de la imagen, por lo que permite otorgar diferentes prioridades a múltiples regiones de la imagen, claro que tanto la información de la zona priorizada como el factor de escalaje U utilizado, deben añadirse en la codificación para que el descompresor sepa interpretar correctamente los bits que recibe.

Una técnica que evita tener que codificar el factor U y las regiones de prioridad, es la técnica llamada desplazamiento máximo (max shift) donde el factor U se fija al plano de bits más significativo de todos los bloques a codificar. Por lo tanto, las regiones de interés (priorizadas) se codificarán antes que el fondo de la imagen, no es necesario especificar al descompresor cual ha sido el factor utilizado ni las regiones, ya que se deduce que son superiores a 2^U.

2.2.6. División de bloques

Una de las características principales del JPEG2000 es la capacidad de escalabilidad. Para proporcionar esta capacidad se utiliza la división de la imagen en bloques, que serán codificados y comprimidos independientemente.

Al dividir la imagen en bloques de medidas variables, se obtienen diferentes bloques de bits, que ordenados convenientemente permiten transmitir o comprimir la imagen dando más importancia a los valores que se deseen, como resolución, calidad, componente o región espacial.

Cuanto mayor sea la medida de los bloques en que se divide la imagen, mayor es el índice de compresión, esto se debe a que se aprovecha considerablemente la redundancia espacial en los bloques, aunque cuanto mayor sea la medida de los bloques menor será el índice de escalabilidad.

El tamaño de los bloques se define por parte del usuario o según la aplicación final, se debe especificar la amplitud y la altura del bloque con valores que sean potencia de 2 y dentro de un rango $[2^2,2^{10}]$. Esta división en bloques se realiza en cada una de la subbandas sin que se sobrepasen los límites de cada subbanda. En la figura 2.4 se muestra un ejemplo.

Una vez dividido las subbandas en bloques, estos se recorren en cierto orden y para cada uno de los planos de bits que contiene. Un plano de bits son todos aquellos bits de los coeficientes, que corresponden a la misma magnitud 2^p (figura 2.6).

Tras este proceso finalmente se obtiene una tira de bits. A partir de ahí, pasaremos a trabajar con esa tira de bits en vez de los valores de la imagen. Para procesar cada uno de los bloques, en cada uno de sus mapas de bits (bitplanes), se dividen los bloques en stripes con cierto tamaño, el tamaño máximo de un stripe será la altura de ese bloque. El stripe

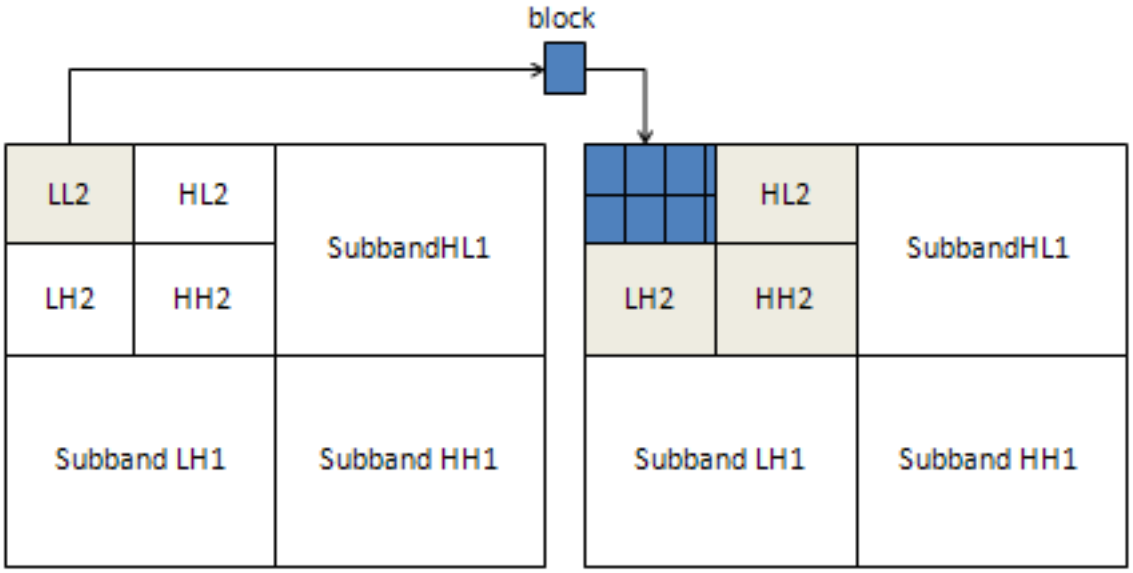

Figura 2.4: División de la subbanda LL2 en bloques.

corresponde al ancho de una columna, que se recorre de arriba a abajo y que cuando se llega a la parte inferior pasa a la siguiente columna, así sucesivamente, hasta que llegue al límite, que corresponde al ancho de bloque, al llegar a este límite pasa al siguiente stripe que esté seguidamente inferior (figura 2.5).

2.2.7. Codificación por planos de bits

El estándar JPEG2000 permite realizar una codificación por planos de bits, pasamos de un mapa de coeficientes a varios planos de bits (bitplanes), en la imagen 2.6 se muestra un ejemplo de bitplane.

En el capítulo 3 se entra en detalle sobre como se realiza la codificación contextual por planos de bits. La idea principal es que tras realizar la codificación por planos de bits de los coeficientes, pasamos de tener un bloque con coeficientes a una tira de bits que irán a parar al Codificador Aritmético MQ.

2.2.8. Codificador Aritmético MQ

El codificador aritmético MQ permite que un conjunto de símbolos asociado a una probabilidad sea codificado como un solo número, reduciendo así la cantidad de información que debe de transmitirse para almacenar o transmitir un solo número en vez de una cadena de símbolos.

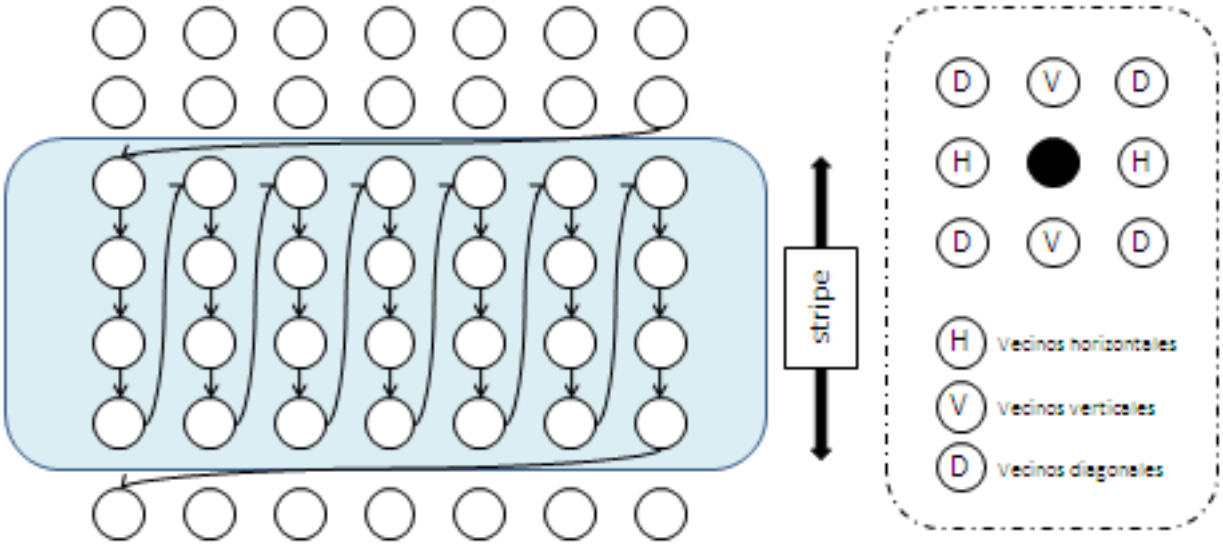

Figura 2.5: Procesamiento de un bloque

El funcionamiento del codificador aritmético binario es: disponemos de un intervalo del tipo $[C_n, C_n + A_n)$ que está entre [0,1) donde C_n es el valor mínimo de ese intervalo y A_n la longitud del intervalo (el valor máximo del intervalo viene dado por $C_n + A_n$). Según van llegando bits a codificar se actualizan los valores de C y A según la probabilidad asociada al símbolo, consiguiendo así subintervalos. El número que nos da el compresor aritmético como resultado de codificar la lista de símbolos, se encuentra en el último intervalo calculado, este es único y es el resultado de ir procesando cada uno de los símbolos.

Cada vez que llega un símbolo a codificar, se actualiza C y A, esta actualización consiste en dividir el intervalo en el que se encuentra en dos intervalos con una medida determinada que depende de la probabilidad que viene asociada al símbolo, esta es la probabilidad de que nos llegue un 1 o un 0 (en el caso de 1 será uno de las subdivisiones del intervalo y un 0 la restante). Según el símbolo que nos llega, escogeremos como intervalo actual, uno de los dos intervalos en que se ha dividido.

2.2.9. Reorganización EBCOT

Tras la etapa de la codificación mediante el codificador aritmético MQ, se obtiene una tira de bits, que las siguientes etapas no comprimirán más, simplemente, la reorganizarán y ordenarán según las necesidades. Esta organización posterior a la compresión, permite que el estándar JPEG2000 tenga flexibilidad a la hora de adaptarse a diferentes entornos

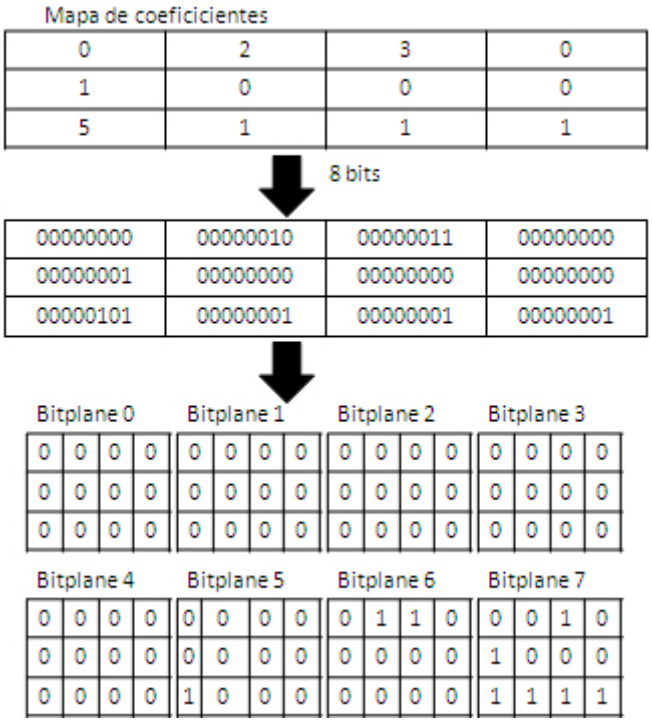

Figura 2.6: Mapa de bits (bitplane).

de operación.

Algoritmo de control de distorsión

La reorganización de la tira de bits, nos permite una transmisión progresiva, dando la capacidad de poder transmitir progresivamente la imagen por niveles de calidad, es decir, que los primeros bits en trasmitir y en recibir serán aquellos que tendrán más peso en la reconstrucción de la imagen original.

Para conseguir esto, se realiza un cálculo para todos los bloques codificados para saber en que puntos puede romper el flujo de la tira de bits, sin perder eficiencia en la ordenación posterior, este proceso se denomina como búsqueda de puntos de rotura (corte) óptimos. Tras esta búsqueda se dispone de los puntos de rotura óptimos y para cada uno una medida que nos relaciona su aportación en la calidad (Distorsión, ganancia en calidad) en la imagen reconstruida y la medida de la tira (Longitud).

Construcción final de la tira de bits

Hasta este punto ya podemos disponer un conjunto de puntos óptimos para el corte de cada una de las tiras de bits comprimidas de los bloques. Si quisiéramos transmitir progresivamente por calidad, tendríamos que ir buscando los mejores puntos de corte según la calidad y transmitir por este orden. Pero el problema deriva al añadir una cabecera para

identificar en todo momento el bloque que se transmite, cosa que perjudica la eficiencia en el almacenamiento. Si bien se podría fijar el orden de llegada de los bloques, utilizando el método entrelazado (interleaved), claro que perderíamos la capacidad de elegir el índice de la progresión.

El algoritmo EBCOT se situa entre estos dos métodos. La idea principal reside en definir unas capas de calidad (quality layers) que agrupan los bloques comprimidos según niveles de calidad. Y dentro de estas capas hay un orden definido de llegada de los bloques. Dentro de cada capa de calidad se identifican los paquetes, que son un conjunto de bits que definen una zona espacial de la imagen original a cierta resolución. Si queremos transmitir a cierta calidad, tan solo debemos ir transmitiendo las tiras de bits asociadas a esa capa de calidad (dentro de esa capa por el orden definido) y luego el resto de capas.

El uso de esta arquitectura, conjuntamente con la división de bloques y capas de bits de las etapas anteriores, permiten 4 tipos de progresividad: por calidad, resolución, localización espacial y componente.

Capítulo 3

Codificación contextual por planos de bits

Durante la etapa de codificación se procesan los valores que se reciben de las transformaciones realizadas, se divide la imagen transformada en bloques y planos de bits y se codifica eficientemente para que la información que se transmita sea mínima y se consiga mayor compresión.

3.1. Etapas de codificación

Ya sabiendo como se recorren los bloques teniendo en cuenta el tamaño de stripe, se realiza una codificación de los bits como resultado del paso por tres etapas. Estas etapas reciben el nombre de propagación de la significancia (*significance propagation*), refinamiento de la magnitud (*magnitude refinement*) y barrido (*cleanup*). Cada una de ellas tiene una funcionalidad.

Hay que clarificar, que lo que se busca durante esta codificación es ver cuando un coeficiente es significante. Se considera un coeficiente significante cuando su bit más significativo está en el plano que se está procesando actualmente o en planos superiores.

Propagación de la significancia: Esta etapa se basa en que es probable que un coeficiente sea significante cuando alguno de coeficientes ha sido significante en el plano superior o actual. Por lo tanto la primera pasada que se realiza por el bloque, siguiendo el orden

de los stripes, se tiene en cuenta esta premisa, se irá recorriendo el bloque y se marcan los vecinos de un coeficiente significante en ese plano de bits, para que ser visitados en esa etapa y así ver si son o no significantes.

Algorithm 1: Propagación de significancia

```
Data: Coeficientes a procesar
Result: Tira de bits para pasar al compresor
1 begin
2     foreach coeficiente con vecinos significantes en un plano de bits superior o actual do
3         codificar(coeficiente);
4         if coeficiente es significante then
5             Actualizar los coeficientes a visitar en este plano de bits;
6         end
7     end
8 end
```

Este algoritmo utiliza una función codificar, que se especifica a continuación:

Algorithm 2: Codificar coeficiente

```
Data: Coeficiente
Result: Tira de bits
1  begin
2      contexto = buscarContexto(coeficiente);
3      if coeficiente es significante en este plano de bits then
4          emitir(1);
5          emitirContexto(contexto);
6          contextoSigno = buscarContextoSigno(coeficiente);
7          emitirSigno(coeficiente);
8          emitirContexto(contextoSigno);
9      else
10         emitir(0);
11         emitirContexto(contexto);
12     end
13 end
```

Refinamiento de magnitud: Durante esta etapa, los bits de refinamiento de los coeficientes que han sido significantes en planos superiores se transmiten. Los bits de refinamiento de un coeficiente son aquellos que corresponden a los planos de bits inferiores a su bit más significativo.

Algorithm 3: Refinamiento de magnitud

```
Data: Coeficiente a procesar
Result: Tira de bits
1  begin
2      foreach coeficiente significante en los planos de bits superiores do
3          if es el primer bit de refinamiento del coeficiente then
4              if tiene algún vecino significante then
5                  contexto = 16;
6              else
7                  contexto = 15;
8              end
9          else
10             contexto = 17;
11         end
12         if coeficiente en este plano de bits == 1 then
13             emitir(1);
14             emitirContexto(contexto);
15         else
16             emitir(0);
17             emitirContexto(contexto);
18         end
19     end
20 end
```

Barrido: Todos los bits que no se han procesado en este plano de bits, durante las etapas anteriores, se procesan en esta etapa, con el fin de que todos los bits de este plano de bits sean visitados. Hay que destacar que durante esta etapa y con el fin de optimizar la codificación (ahorrando bits transmitidos) se pasa a un modo de funcionamiento especial, denominado *run mode*.

Se entra en este modo cuando al recorrer 4 coeficientes de una columna de un stripe ninguno tiene un vecino que sea significante. Al pasar a este modo, cada columna del stripe se codifica con un solo 0, siempre y cuando no haya ningún coeficiente significante en esa columna, en caso de que haya algún significante en esa columna, se sale del modo run y se sigue con la codificación normal.

Algorithm 4: Barrido

Data: Coeficiente a procesar

Result: Tira de bits

```
1  begin
2      foreach cada columna o stripe do
3          if Si los 4 coeficientes de la columna no han sido significantes en planos de bits
           superiores and ninguno de los 4 coeficientes tiene vecinos significantes then
4              if ninguno de los coeficientes es significante then
                   // PASAMOS A RUN MODE
5                  emitir(0);
6                  emitirContexto(9);
7              else
                   // SALIMOS DE RUN MODE
8                  emitir(1);
9                  emitirContexto(9);
10                 p = posicionDelCoeficienteSignificanteEnLaColumna; // posición
                       representada en 2 bits
11                 emitirBitMasSiginificativo(p);
12                 emitirContexto(18);
13                 emitirBitMenosSignificativo(p);
14                 emitirContexto(18);
15                 contextoSigno = buscarContextoSigno(coeficiente);
16                 emitirSigno(coeficiente);
17                 emitirContexto(contextoSigno);
18                 foreach coeficiente restante en la columna o stripe do
                       // Se codifica con normalidad
19                     codificar(coeficiente);
20                 end
21             end
22         else
23             foreach coeficiente de la columna do
24                 codificar(coeficiente);
25             end
26         end
27     end
28 end
```

3.2. Representación de contextos y su utilización

Se puede observar que para cada bit emitido en cada una de las etapas, se emite también un contexto. El contexto son unos valores que a posteriori se utilizan por el codificador aritmético para poder interpretar correctamente la tira de bits que irá recibiendo, de modo que pueda procesarlos de la manera más eficiente.

El codificador dispone de 19 contextos diferentes que se utilizan durante las etapas de codificación Tabla (3.1).

CONTEXTO	UTILIZACIÓN
0 - 8	Identificación del contexto
9	Run mode
10 - 14	Bits de signo
15 - 17	Refinamiento de magnitud
18	Probabilidad uniforme

Cuadro 3.1: Tipos de contexto del codificador

La asignación de contextos comprendidos desde el 0 al 8, depende de la subbanda que se esté procesando y de los vecinos que tenga ese coeficiente; a continuación se especifican en la tabla 3.2.

subbanda	LL y LH			HL			HH	
contexto	h	v	d	h	v	d	d	h+v
0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0	1
2	0	0	≥ 2	0	0	≥ 2	0	≥ 2
3	0	1	x	1	0	x	1	0
4	0	2	x	2	0	x	1	1
5	1	0	0	0	1	0	1	≥ 2
6	1	0	≥ 1	0	1	≥ 1	2	0
7	1	≥ 1	x	≥ 1	1	x	2	≥ 2
8	2	x	x	x	2	x	≥ 3	x

Cuadro 3.2: Tipos de contexto del codificador

La asignación de contextos comprendidos desde del 10 al 14, corresponden a los bits de signo y dependen de los vecinos horizontales y verticales del coeficiente. En la tabla 3.3 se especifican la asignación de estos contextos de signo.

El algoritmo que realiza la asignación de signo se especifica a continuación:

X^h	X^v	contexto	inversión
1	1	14	no
1	0	13	no
1	-1	12	no
0	1	11	no
0	0	10	no
0	-1	11	si
-1	1	12	si
-1	0	13	si
-1	-1	14	si

Cuadro 3.3: Asignación contextos de signo

Algorithm 5: Asignación contexto de signo

```
   Data: Coeficiente a procesar
   Result: Contexto a codificar
 1 begin
 2     foreach vecino del coeficiente do
 3         valoresHorizontales = 0;
 4         valoresVerticales = 0;
 5         if vecino es significante then
 6             valorVecino = 1;
 7         else
 8             valorVecino = -1;
 9         end
10         if vecino es horizontal then
11             valoresHorizontales += valorVecino;
12         end
13         if vecino es vertical then
14             valoresVertical += valorVecino;
15         end
16         if valoresHorizontales<0 then
17             valoresHorizontales = -1;
18         else
19             if valoresHorizontales==0 then
20                 valoresHorizontales = 0;
21             else
22                 valoresHorizontales = 1;
23             end
24         end
25         if valoresVerticales <0 then
26             valoresVerticales = -1;
27         else
28             if valoresVerticales==0 then
29                 valoresVerticales = 0;
30             else
31                 valoresVerticales = 1;
32             end
33         end
34         contexto = buscarTablaContextosSigno(valoresVerticales, valoresHorizontales);
35     end
36 end
```

El signo, se codifica como un 1 en caso de ser signo negativo y un 0 en caso de ser signo positivo, excepcionalmente, cuando se realiza una inversión en el signo (tabla 3.2) para mejorar la codificación, se codifica de modo inverso, el signo negativo con un 0 y el signo positivo con un 1. En la tabla 3.4 se muestra un ejemplo de una codificación del bloque 3.5 por mapa de bits.

Plano	Etapa	CODIFICACIÓN - bit(contexto)	$\sharp bits$
7	Barrido	(9)1(9)0(18)1(18)0(10)1(5)0(11)0(1)0(3)0(3)0(3)1(9)1(18)1(19)1(6)0(11)	19
6	Significancia	0(1)1(3)1(13)1(7)1(12)0(7)1(6)1(11)0(3)0(3)1(4)1(10)0(7)1(6)0(11)	15
6	Refinamiento	0(16)1(16)0(16)0(16)0(16)	5
6	Barrido	0(5)	1
5	Significancia	0(7)1(7)1(12)1(3)1(13)1(8)1(10)1(7)0(11)0(7)	10
5	Refinamiento	0(16)0(16)0(16)1(17)1(17)0(17)0(16)1(16)0(17)0(17)	10
5	Barrido		0
4	Significancia	1(7)1(14)1(7)1(12)	4
4	Refinamiento	(17)0(17)1(16)1(17)0(17)0(17)1(17)1(16)1(16)0(17)1(16)0(17)1(17)1(17)	14
4	Barrido		0
3	Significancia		0
3	Refinamiento	0(16)1(17)1(17)0(17)0(17)1(17)1(17)1(17)0(17)0(17)0(17)0(17)0(16)0(17)1(17)0(17)	16
3	Barrido		0
2	Significancia		0
2	Refinamiento	0(17)1(17)1(17)1(17)1(17)1(17)1(17)0(17)1(17)1(17)0(17)1(17)1(17)0(17)0(17)1(17)	16
2	Barrido		0
1	Significancia		0
1	Refinamiento	1(17)1(17)0(17)0(17)1(17)0(17)0(17)1(17)1(17)1(17)1(17)1(17)0(17)0(17)1(17)1(17)	16
1	Barrido		0

Cuadro 3.4: Codificación en planos de bits

9	-43	27	-10
-39	86	-27	-48
-38	118	-33	-77
-26	77	-27	-75

Cuadro 3.5: Bloque escogido para la codificación

3.3. Propuestas para la configuración de contextos

La codificación de los contextos está estrechamente ligada a si los vecinos del coeficiente son significantes.

Se ha visto con anterioridad, que el identificador de un contexto está comprendido entre 0 y 8, es decir, se dispone de un total de 9 contextos identificables. La identificación de contexto depende de la subbanda que se esté procesando y los vecinos de los coeficientes a codificar (tabla 3.2).

A la hora de identificar el contexto que le pertenece, se agrupa los vecinos en tres grupos, vecinos horizontales, vecinos verticales y vecinos diagonales. Esto sin duda tiene una relación con la subbanda que se esté procesando, otorgando más o menos importancia a la significancia de los vecinos dependiendo de la subbanda en la que se encuentre.

En vez de relacionar cada contexto con vecinos horizontales, verticales y diagonales y dependiendo de la subbanda procesada, se asigna un contexto concreto a cada posible estado del conjunto de vecinos. Se define como estado, el conjunto bits que representan si son significante o no cada uno de los vecinos de un coeficiente.

Hay un total de 8 posibles vecinos por coeficiente (exceptuando los coeficientes que están en el límite del stripe o del bloque procesado). El orden por el que se procesan los vecinos es el mismo por el cual se recorre un stripe:

Vecino 1	Vecino 4	Vecino 6
Vecino 2	Coeficiente	Vecino 7
Vecino 3	Vecino 5	Vecino 8

Cuadro 3.6: Vecinos de un coeficiente

La codificación del estado de los vecinos, viene dada por la significancia o no del vecino del coeficiente que se está procesando, por lo tanto, como se dispone de 8 vecinos hay un total de 2^8 posibles estados. La representación binaria del estado permite ver que vecinos son significantes o no.

1	0	0
0		1
0	0	0

Cuadro 3.7: Estado o Contexto = 10000010

Se puede observar que el estado, que está directamente relacionado con el contexto, tiene una representación de 8 bits (1 byte), esto de cara a la implementación práctica, será útil.

Con esta configuración de contextos, se dispone de un número muy superior de contextos

al que actualmente se utiliza. Sin duda no se quiere utilizar todos estos contextos para realizar la codificación que pasará al codificador aritmético, solo interesan los contextos que más se utilicen o más información aporten. De modo se necesita una reducción de estos contextos iniciales para obtener una tabla de contextos dinámica según la imagen procesada.

Con el fin de reducir los contextos que no aportan información importante, se introducen una serie de etapas para el cálculo de estadísticas de los contextos más utilizados, cuales son significativos a la hora de la codificación y cuales no. De este modo, se aplicaran diferentes técnicas o políticas de reducción para finalmente disponer tan solo de los contextos cuya información sea más relevante para la codificación de la imagen.

Pero antes de mostrar las técnicas de reducción, primero se debe especificar la manera en que se ha representado el conjunto de contextos, con el fin de operar con ellos. Se muestran dos propuestas para la configuración de contextos:

- Representación de contextos por grafos.
- Representación de contextos por listas.

A continuación se explica y analiza cada uno de las representaciones, con el fin de clarificar cual de las representaciones es la mejor para realizar este estudio.

3.3.1. Grafos de contextos

La primera representación viene dada por un grafo de contextos. Donde cada uno de los nodos es un contexto con la representación de su estado, además de información acerca el número de veces que ha sido significante (Sc, significant Context) o no significante (Uc, unsignificant context). Se considera un contexto significante cuando el coeficiente que se está procesando es significante en ese contexto, en caso contrario se considera el contexto no significante. Por lo tanto el número de elementos que representarán un contexto serán cuatro, el identificador, la definición (representación del estado), el número de veces que el contexto ha sido significante y el número de veces que el contexto ha sido no significante.

Cabe clarificar, que la definición de un contexto, se representa de la siguiente manera: para cada uno de los bits que componen el contexto, se codifica un 0 en caso de no ser

# Contexto	Definición	
	Significant Coefficient	Unsignificant Coefficient

Cuadro 3.8: Elementos de un contexo

significativo, un 1 en caso de ser significativo y una X en caso de que pueda o no ser significativo. Añadir la posibilidad de que un bit, pueda o no ser significativo, viene derivado de la necesidad de que un contexto pueda representar un conjunto de contextos, necesario para la reducción de contextos y la unificación de los contextos eliminados a contextos más generales. Este tipo de contextos son los contextos unificados.

Se puede ver un ejemplo sencillo de una representación de un contexto, concretamente este ejemplo hace referencia al contexto 0001000X, que tiene como identificador 13, ha sido 120 significativo y 54 veces no significativo. Este contexto representa tanto al contexto 00010001 como al 00010000 (X puede ser 1 o 0).

13	0001000X	
	120	54

Cuadro 3.9: Ejemplo de elementos de un contexo

La estructura de grafo, permite representar todas las relaciones que hay entre los contextos con las transiciones entre estos, a parte de la información que se guarda de cada nodo del grafo, cada transición almacena el número de veces que se utiliza. El grafo de contextos permite una visión rápida y completa de los contextos y transiciones importantes. Pero la estructura de grafo, no solo permite está visión global de todos los contextos, además nos indica un coeficiente pasará de un contexto a otro mediante una transición, siendo el contexto final uno de los contextos que sea nodos hijos del contexto en el que se encuentra, esta características es crucial para entender la reducción de los contextos. A continuación se muestra un ejemplo de un grafo de contextos, simplificándolo a contextos de 3 bits.

En la figura 3.1 se observa como de un contexto inicial, mediante transiciones, se pasan a otros contextos. Por ejemplo si partimos de un contexto 000, hay tres posibles contextos a los que derivaremos mediante una transición, 001, 010 y 100, con las transiciones $T_{0,3}$, $T_{0,2}$, $T_{0,1}$ respectivamente. En caso de partir de un contexto como 010, solo se tienen dos posibles transiciones 110 o 011.

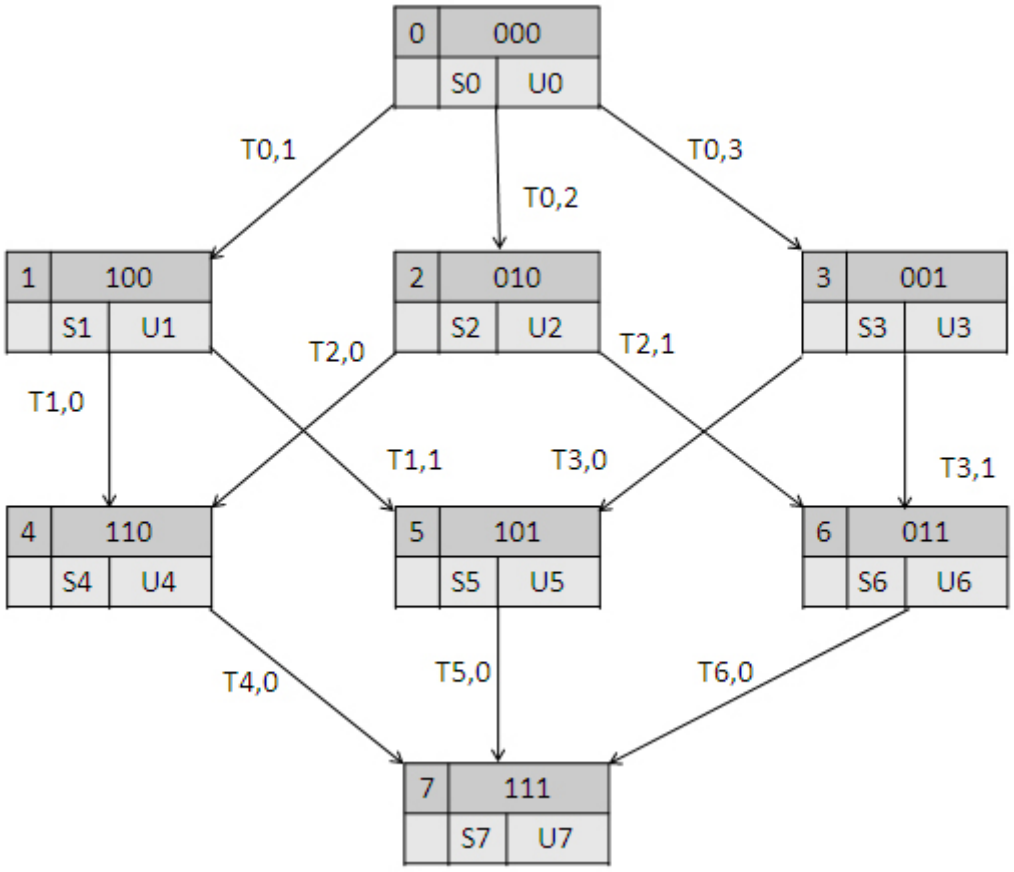

Figura 3.1: Grafo de contextos

Rellenar el grafo de contextos:

Con el fin de rellenar el grafo de contextos, se necesitan una estructura que se denomina mapa de contextos de los coeficientes (context coefficient map), esta nos indica para cada coeficiente que se procesa en que contexto está, hace de referencia entre los coeficientes y el grafo de contextos. Inicialmente todos los coeficientes están en el contexto inicial, haciendo referencia al ejemplo de la figura 3.1 es el contexto 000.

Utilizamos el scanning order definido por JPEG2000 para cada bit plane, pasando por todos los coeficientes actualizando las estadísticas del grafo, tanto de los contextos que se utilizan como de las transiciones. Debido a que el contexto de un coeficiente depende de la significancia de sus vecinos, si el coeficiente pasa a ser significante en un mapa de bits, todos sus vecinos deben actualizar su contexto. A continuación se especifica el algoritmo que se utiliza para rellenar el grafo de contextos:

Algorithm 6: Rellenar grafos de contextos

```
1  begin
       // Inicializamos el mapa de contextos de los coeficientes
2      InicializarContextCoefficientMap();
3      foreach mapa de bits do
           // Scanning Order JPEG2000
4          coeficiente = scanningOrder();
           // Buscamos el contexto que le corresponde a ese coeficiente en el
              mapa de contextos de los coeficientes.
5          contexto = contextCoefficientMap(coeficiente);
6          if coeficiente es significante en mapa bits actual then
               // Incrementamos la estadística Sc al contexto
7              contexto.Sc = contexto.Sc + 1;
               // Procesamos los 8 vecinos del coeficiente
8              foreach vecino de coeficiente do
                   // Calculamos el nuevo contexto
9                  nuevoContexto = calcularNuevoContexto(vecino);
                   // Incrementamos la estadística de la Transición
10                 contexto.Tvx = contexto.Tvx + 1;
                   // Actualizamos el mapa de contexto de los coeficientes, para
                      que el vecino señale al nuevo contexto
11                 contextCoefficientMap.coeficiente = nuevoContexto;
12             end
13         else
               // Incrementamos la estadística Uc al contexto
14             contexto.Uc = contexto.Uc + 1;
15         end
16     end
17 end
```

Tras recorrer todos los coeficientes de todos los mapas de bits, se dispone de una estructura de grafo que almacena información de cuales son los contextos usados que han sido significativos, o no significativos, de las transiciones más realizadas, de los contextos poco usados, etc...

Esta estructura que contiene 256 contextos posibles y sus estadísticas nos permite tener información importante para realizar la reducción de contextos.

Reducción de contextos:

Si bien para el cálculo de las estadísticas partimos de todos los contextos posibles, se reducirán para tener un mapa simplificado de contextos acorde con la imagen que se procesa. Para este cometido entra en juego la fase de reducción de contextos. A continuación se

detalla el algoritmo de reducción del grafo de contextos:

Algorithm 7: Reducción del grafo de contextos

```
1  begin
       // Mediante una politica de reducción seleccionamos el contexto a
          eliminar
2      contextoEliminar = politicaReducción(grafoContextos);
       // El contexto padre corresponde al nodo padre en el grafo
3      foreach contextoPadre del contextoEliminar do
           // Unificar nodo padre con el contextoEliminar.
4          contextoPadre = unificarContextos(contextoPadre, contextoEliminar);
           // Comprobamos los lazos.
5          if existeTransicion(contextoPadre,contextoPadre)==falso then
6              agregarTransicion(contextoPadre,contextoPadre);
7          end
           // Enlazamos el contexto padre del contexto a eliminar con los
              contexto hijos.
8          foreach contextoHijo del contextoEliminar do
9              if existeTransicion(contextoPadre,contextoHijo)==falso then
10                 agregarTransicion(contextoPadre,contextoHijo);
11             end
12         end
           // Eliminamos los hijos.
13         foreach contextoHijo del contextoEliminar do
14             if existeTransicion(contextoPadre,contextoHijo)==verdadero then
15                 eliminarHijoRecursivamente(contextoHijo);
16             end
17         end
18     end
19 end
```

En el algoritmo de reducción cabe aclarar:

- politicaReducción(grafoContextos): Las políticas de reducción basándose en la información que almacena el grafo de contextos, permiten adoptar diferentes decisiones para elegir el contexto a reducir entre todos los contextos que hay en el grafo de contextos.

- unificarContextos(contextoPadre, contextoEliminar): Permite que un contexto se unifique con un segundo contexto, de modo que el primer contexto represente a los dos. Un ejemplo de esto, sería unificar el contexto 000 con el 001, se obtiene el 00X. Si bien también el contexto 0XX o XXX pueden representar los dos contextos a unificar, siempre se busca el que mejor lo represente considerando como mejor representación la más específica.

Como 0XX a parte de representar 000 y 001 es capaz de representar otros contextos como 011 y 010, es una representación más general, por lo tanto la representación más especifica que unifique 000 y 001, es 00X.

A continuación se muestra un ejemplo simplificado de como se procedería a la reducción de un contexto en un grafo de contextos.

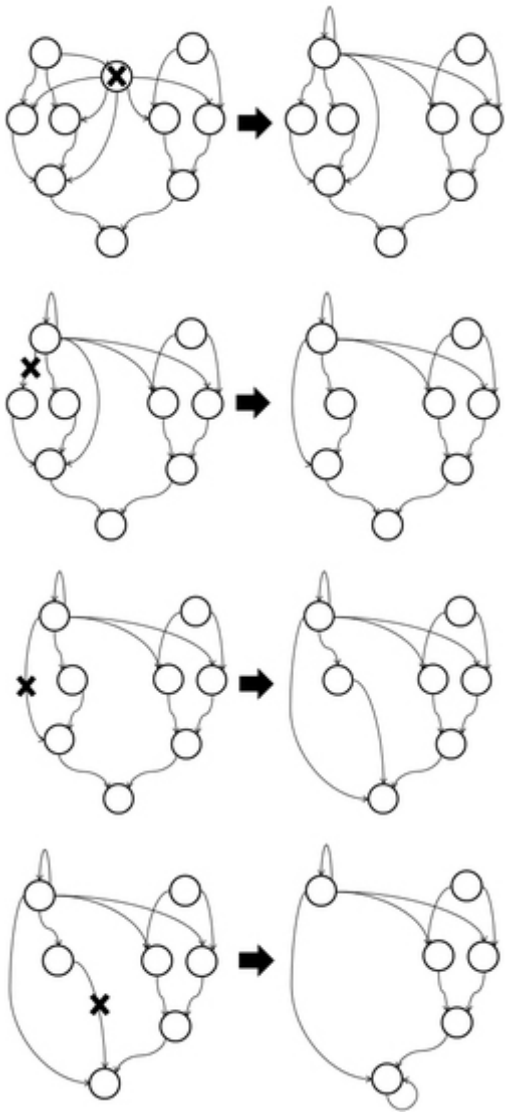

Figura 3.2: Reducción grafo de contextos

En la imagen 3.2 muestra un grafo simplificado, sin la información contenida en los contextos ni en las transiciones, muestra como al eliminar un contexto se va realizando varias reducciones que derivan de esta, eliminando hijos y transiciones del nodo eliminado y creando nuevas transiciones.

Criterios para la reducción de contextos:

A continuación se muestra algunas de las políticas para la reducción del grafo de contextos:

- **Contextos nulos**: Aquellos contextos cuyas estadísticas sean 0 se pueden prescindir de ellos ($Sc + Uc = 0$).
- **Contextos en desuso**: Se pretende detectar los contextos que han sido utilizados pocas veces para prescindir de ellos. Estos casos son los que $Sc + Uc$ tiene un valor muy pequeño (comparado con el $Sc + Uc$ máximo del grafo).
- **Contextos irrelevantes**: Los contextos irrelevantes se consideran aquellos que $Sc = Uc$, ya que desde el punto de vista de la información que aportan al mapa de contextos es reducida. A este tipo de reducción se le puede aplicar un threshold D, de modo que si cumple $Sc + D >= Uc >= Sc - D$ se consideraría como contexto irrelevante.
- **Mejora de contextos**: Si bien es interesante prescindir de los contextos que no son buenos, la idea de poder unificar contextos que si lo son, para dar mejores contextos es otra de las posibles políticas de reducción. Si prestamos atención a que cuando eliminamos un contexto que consideramos malo, los factores que lo caracterizan como contexto malo, se disgregan por los contextos que se unifiquen y lo engloben, empeorando estos contextos, nos seduce la idea de que podamos unificar varios contextos que se consideren buenos, para englobar las características que los hacen buenos contextos.

3.3.2. Lista de contextos

La lista de contextos es otro tipo de representación de todos los contextos posibles.

La representación de un contexto es la misma que se ha especificado con anterioridad en el cuadro 3.8. La diferencia radica en la estructura utilizada para la organización de estos. Se crea una tabla con N niveles, donde N = (número de bits que definen un contexto) + 1 (en nuestro caso son 8 bits y 9 niveles). En cada nivel se almacena una lista con los contextos que corresponden a ese nivel. El nivel de un contexto se calcula sumando el número de bits que son 1 en su definición.

A continuación mostramos una ejemplo de una lista de contextos, simplificada a contextos de 4 bits y por lo tanto con 5 niveles:

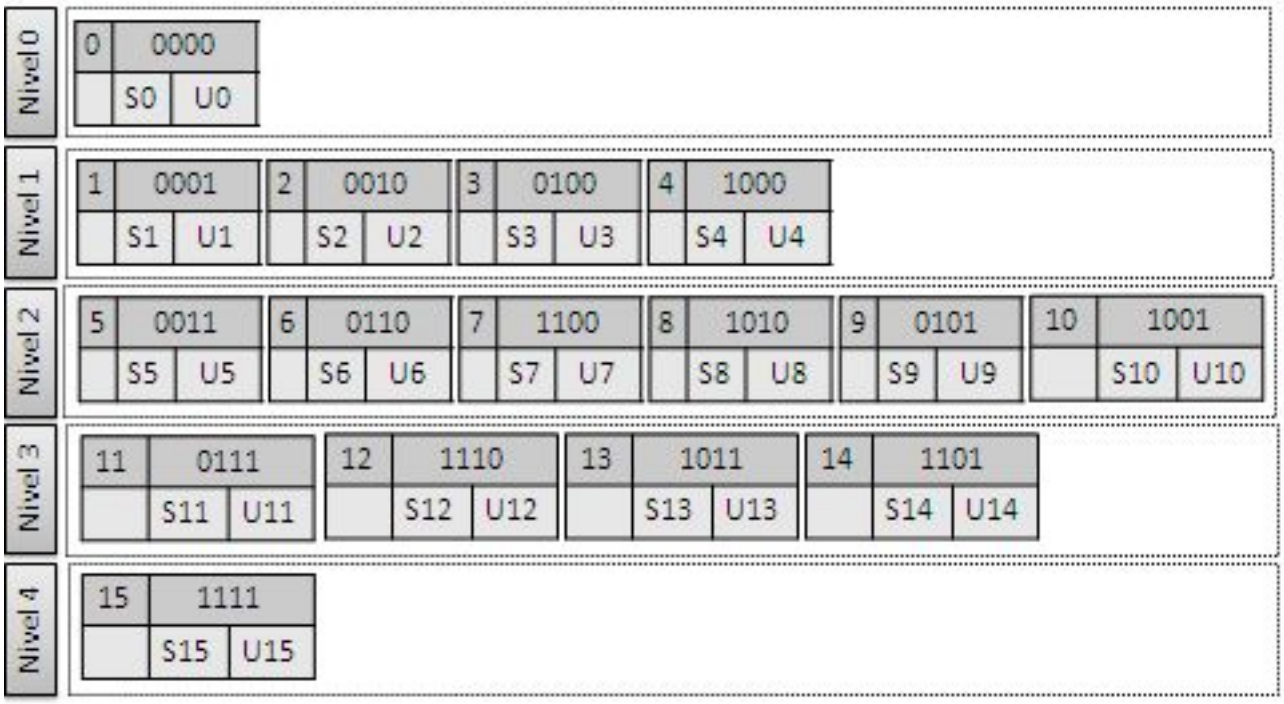

Figura 3.3: Lista de contextos

La agrupación de los contextos por niveles permite ordenar los contextos de modo que siempre que se pase a un nuevo contexto, se pasará al mismo nivel o a un nivel inferior. Por ejemplo, desde el contexto 0001 que está en el nivel 1, se pueden pasar al contexto 0011, 1001 o 0101 que todos ellos están en el nivel 2. En esta representación no se tiene en cuenta las transiciones, por lo tanto se prescinde de que contexto se viene y concretamente a cuales se van.

Rellenar la lista de contextos:

El proceso de rellenar la lista de contextos con la información mientras se va recorriendo los coeficientes se realiza de la siguiente manera:

- Recorremos los coeficientes con el scaning order del JPEG2000.
- Si el coeficiente es significativo, incrementamos el Sc de su contexto, sino incrementamos el Uc de su contexto.
- Si ha sido significativo, para cada vecino se actualiza su contexto, es decir para cada vecino se recoge el contexto, se calcula el nuevo contexto que le corresponde y lo buscamos en los niveles inferiores o actual en la lista de contextos y finalmente se asigna este nuevo contexto a el vecino.

Reducción de la lista de contextos:

La reducción de contextos se realiza de la siguiente manera:

- Elegimos el contexto a eliminar con una política de reducción.
- Buscamos en el nivel que le corresponde el contexto y lo eliminamos.
- En el nivel superior, unificamos solo los contextos que mediante una transición pudieran derivar al contexto eliminado.

3.4. Valoración

A continuación se explica cuales son las ventajas e inconvenientes de cada uno de las representaciones que se han especificado con anterioridad, para finalmente elegir la mejor de las dos representaciones.

- **Complejidad en la estructura de datos**: Sin duda alguna, la representación de un grafo de contextos es mucho más compleja que la representación de lista de

contextos. En la estructura de datos hay que reflejar las transición, realizando enlaces entre contextos y siempre manteniendo la integridad del grafo de contextos.

- **Información alberga la estructura**: Desde este punto de vista, la estructura de grafos de contexto es capaz de almacenar más información. Se almacenan estadísticas referentes a los contextos y a sus transiciones. Hay que decir, que el grafo visualiza de forma más fidedigna el esquema de todos los contextos posibles y los pasos entre ellos, en el caso de las listas se simplifica.

- **Políticas de reducción**: Ya que la lista de contextos, no dispone de transiciones, hay un conjunto de políticas relacionadas con las transiciones que no podrían aplicarse a este tipo de representación.

- **Complejidad algorítmica**: A la hora de operar con un grafo de contextos implica una complejidad algorítmica mucho más elevada, a la hora de añadir, buscar o eliminar contextos de una lista. En todo momento hay que mantener la integridad del grafo y de su contenido, este debe construirse correctamente y reflejar todos los contextos posibles sin ambigüedades, este es uno de los puntos más complejos.

Teniendo en cuenta cada uno de estos aspectos, en la investigación se ha decantado por la utilización de la representación de contextos con listas de contextos. Algunas de las razones es que permite agrupar los contextos de forma adecuada para realizar operaciones con simplicidad, reduciendo considerablemente la complejidad desde el punto de vista algorítmico y de implementación, como estás operaciones se reiterarán un número de veces muy elevado es imprescindible tener en cuenta el factor tiempo de cómputo.

Capítulo 4

Configuración automatizada de contextos

En este capítulo se entra en detalle en el trabajo de investigación realizado. Bajo la idea de conseguir adaptar los contextos preestablecidos al tipo de imagen que queremos procesar y con el fin de optimizar al máximo la transmisión y compresión de la imagen, se ha mostrado varias posibles representaciones de los contextos y se ha analizado la complejidad de cada una. Ahora se centra en la representación elegida y se entra en detalle en los algoritmo utilizados para conseguir la automatización utilizando la representación correspondiente.

4.1. Visión general

Se parte de la idea de que una pieza clave de la codificación viene dada por la configuración de los contextos. Esta configuración estática, que se ha especificado en el capítulo anterior, induce a pensar en una posible optimización en la codificación, realizando un cálculo de los contextos dinámicamente acorde la imagen que se esté procesando.

La configuración automática de contextos nos permite relacionar los contextos que se codificarán con el tipo de imagen original. Esta configuración parte de 256 contextos posibles, que se reducirán mediante políticas de reducción para obtener los más importantes de cara a la optimización codificación.

Figura 4.1: Esquema de codificación con contextos

En la figura 4.1 se muestra un esquema simplificado del uso de los contextos en la codificación del JPEG2000.

En la figura 4.2 se muestra una ampliación del esquema de la figura 4.1 y se muestran los pasos que se realizan para automatizar la extracción de contextos significantes. Se parte de una imagen concreta a codificar, se procesa la imagen y se obtiene una estructura de contextos y coeficientes las estadísticas correspondientes. Estos coeficientes y sus contextos se procesan para reducir la representación. Se reitera este paso hasta que se obtiene una representación de los contextos que no varíe (es el caso de que no se pueda reducir más el número de contextos) o que se haya reducido lo suficiente para la utilización de esta configuración. Finalmente con la configuración de contextos obtenidos se codifica la imagen. El algoritmo para la extracción de contextos se detalla en el algoritmo 8.

El mapa de contextos es una estructura donde se encuentran todos los contextos y que se utilizará para calcular la configuración de los contextos, la estructura puede tener diferentes representaciones, se han visto dos en el capítulo anterior, los grafos de contexto

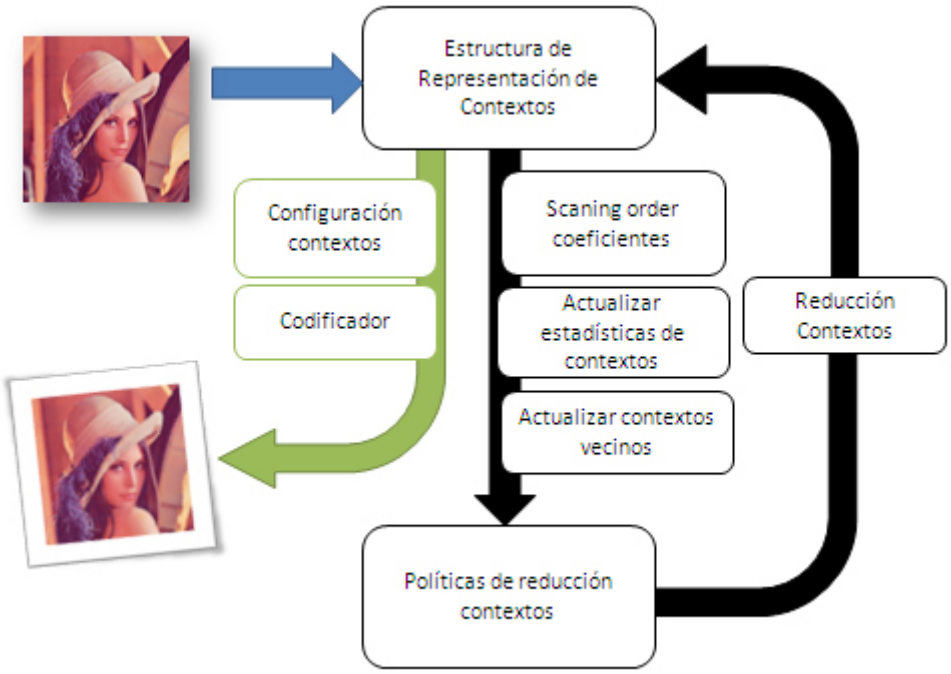

Figura 4.2: Esquema extracción automatizada contextos

y la lista de contextos, finalmente se ha optado por esta última. Se ha de disponer de dos estructuras más, una estructura donde están los coeficientes que se tienen que procesar que se denomina mapa de coeficientes y una tercera estructura que se relaciona los coeficientes con el mapa de contextos, esta se denomina mapa de contextos de los coeficientes.

Como el cálculo de contextos se realiza por bitplane, las estadísticas se calculan por cada bitplane independientemente, por lo tanto serán necesario que en cada bitplane se disponga de las siguientes estructuras:

- **Mapa de coeficientes**: Estructura donde se almacena los coeficientes que se irán procesando y recorriendo en el orden normal de codificación.

- **Mapa de contextos**: Estructura que almacena todos los contextos disponibles y las estadísticas de cada uno. Ya se ha explicado con detalle en el capítulo anterior la representación de los contextos. Es muy importante que se mantenga en todo momento la integridad del mapa de contextos sin ambigüedades. Un contexto, debe estar representado en la lista de contextos por un solo contexto (teniendo en cuenta que hay contextos unificados que pueden representar a varios contextos).

- **Mapa de contextos de los coeficientes**: Estructura que indica, para cada coe-

Algorithm 8: Extracción automatizada de contextos

```
Data: Bloque
Result: Lista de contextos
1  begin
2      acabar = no;
3      while acabar==no do
4          inicializarListaContextos;
5          foreach mapa de bits do
6              foreach coeficiente en Scanning Order do
7                  contexto = calcularContexto(coeficiente);
8                  listaContextos = actualizarEstadísticasListaContextos(contexto);
9              end
10         end
11         contextoReducir = politicaEliminacionContextos(listaContextos);
12         numContextos = reducirContexto(contextoReducir);
13         if no se han eliminado contextos then
14             acabar = si;
15         end
16     end
17 end
```

ficiente, el contexto que le corresponde dentro del mapa de contextos. De cara a la implementación, cada uno de los elementos de este mapa es un puntero a un contexto del mapa de contextos, concretamente la implementación de los punteros realizada en los ejemplos se hace almacenando una tupla con el nivel e índice, donde se encuentra el contexto en el mapa de contextos.

La imagen 4.3 es un esquema visual de la relación entre las estructuras necesarias para el cálculo de estadísticas.

4.2. Algoritmos

Para el proceso de la extracción automatizada de contextos, se utilizan tres algoritmos principales, el procesamiento de los bitplanes, actualización de las estadísticas de los contextos y eliminación de contextos del mapa de contextos.

Procesamiento de bitplanes para la extracción automatizada de contextos:

Es el algoritmo principal que nos permite recorrer los bitplanes en orden, procesando los

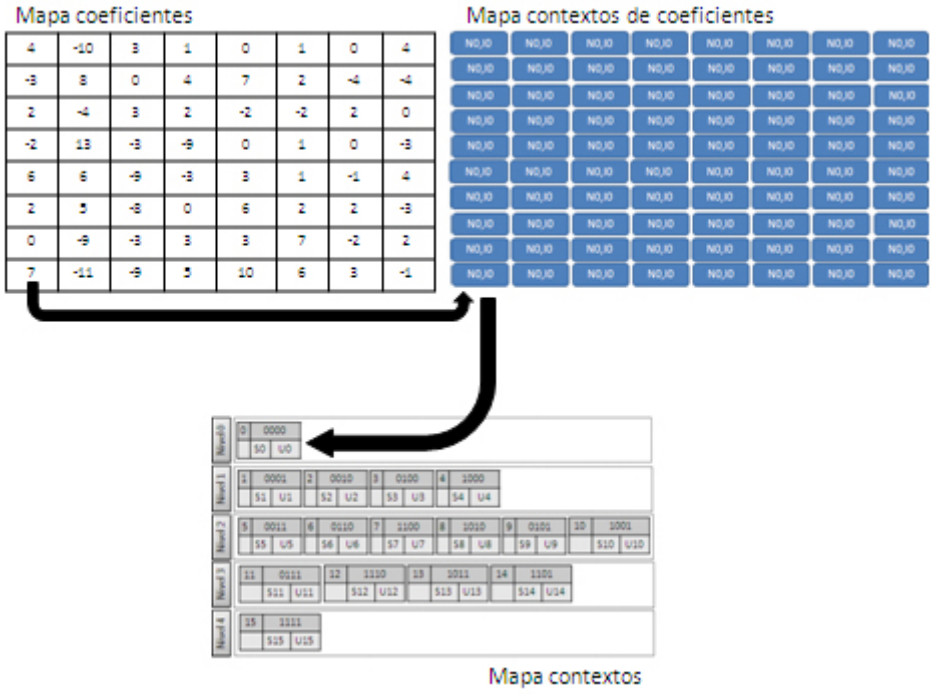

Figura 4.3: Esquema de relación entre los mapas

coeficientes y actualizando los mapas de contextos. En el algoritmo 9 se puede ver los pasos a seguir.

Actualización de vecinos del coeficiente significante:

El algoritmo de procesamiento de los bitplanes (algoritmo 9) hay una función importante que es actualizarContextoVecino. Como se ha explicado con anterioridad, que un coeficiente sea significante, implica que todos sus vecinos tengan que cambiar de contexto, por lo tanto, para cada uno de los vecinos debe calcularse el nuevo contexto en el que está y actualizar el mapa de contextos de coeficientes para que apunte al nuevo contexto.

Los pasos que sigue el algoritmo (vienen especificados en el algoritmo 10), son:

1. Calculamos el nivel del contexto actual.

2. Calcular el nuevo contexto del coeficiente vecino.

3. Desde el nivel actual calculado hacia los niveles inferiores, buscamos un contexto equivalente. Si no se encuentra, se queda en el contexto en el que está, si se encuentra se actualiza el mapa de contextos de coeficientes para que el coeficiente de esa posición haga referencia al nuevo contexto.

Algorithm 9: Procesamiento de los bitplanes

```
Data: Mapa de coeficientes
Result: Lista de contextos
1  begin
2      foreach mapa de bits do
           // Procesamos los coeficientes en orden normal de codificación
3          if coeficiente es significante then
               // Obtenemos el contexto del mapa de contextos de ese coeficiente
4              contexto = obtenerContexto(coeficiente);
               // Incrementamos la estadistica del contexto
5              mapaContextos[contexto].significante++;
               // Actualizamos el contexto de todos los vecinos de este bitplane e
                  inferiores
6              foreach mapa de bits <= mapa de bits actual do
7                  foreach vecino del coeficiente do
8                      actualizarContextoVecino(vecino);
9                  end
10             end
11         else
               // Obtenemos el contexto del mapa de contextos de ese coeficiente
12             contexto = obtenerContexto(coeficiente);
               // Incrementamos la estadistica del contexto
13             mapaContextos[contexto].noSignificante++;
14         end
15     end
16 end
```

En el algoritmo 10 se hace referencia a buscar contextos equivalentes, hay que aclarar que aunque inicialmente se parte de 256 contextos, cada vez que se itera el proceso de reducción de contextos aparecen contextos unificados que representan varios contextos a la vez. Por lo tanto cuando se realice una búsqueda de un contexto concreto, se tiene que tener en cuenta que puede estar representado por un contexto unificado. Por lo tanto no se realiza la búsqueda por contextos iguales sino equivalentes.

Reducción de contextos del mapa de contextos:

Tras haber calculado las estadísticas de cada bitplane se analiza y se aplica una política de eliminación, sabiendo el contexto a reducir se aplica la reducción del contexto (algoritmo 11).

Algorithm 10: Actualizar contexto de un coeficiente

```
Data: Coeficiente
1  begin
       // Calculamos el nuevo contexto de ese coeficiente.
2      nuevoContexto = calcularContexto(coeficiente);
       // Obtenemos el nivel de la estructura en que se encuentra.
3      nivelContexto = calcularNivelContexto(nuevoContexto);
4      nivel = nivelContexto + 1;
5      encontrado = no;
       // Recorremos los niveles inferiores.
6      while nivel <= nivelMaximo do
           // Buscamos el nuevo contexto o uno equivalente.
7          punteroContexto = buscarContextoEquivalente(nuevoContexto);
8          if punteroContexto existe then
               // Si lo encontramos se lo asignamos en el mapa de contextos de los
                  coeficientes.
9              mapaContextosCoeficientes[coeficiente] = punteroContexto;
10             encontrado = si;
11         else
               // Si no lo encontramos buscamos en el nivel inferior.
12             nivel=nivel+1;
13             encontrado = no;
14         end
15     end
       // En caso de que no se haya encontrado el nuevo contexto en los niveles
          inferiores y debido a la manera que está construido la estructura del
          mapa de contextos implica que debe quedarse en el contexto en el que
          está.
16 end
```

4.3. Ejemplo de funcionamiento

A continuación mostramos un ejemplo simplificado paso a paso para comprender mejor el funcionamiento de proceso a extracción automatizado de contextos y los algoritmos que lo componen.

Cálculo de estadísticas:

Se tienen 7 bitplanes (del bitplane 6 al bitplane 0) y en cada bitplane un mapa de coeficientes, un mapa de contextos y un mapa de contextos de coeficiente, el contexto se ha representado con un conjunto compuesto por definición del contexto, número de veces que ha sido significante, número de veces que ha sido no significante. En la tabla 4.1 se puede ver un ejemplo de un mapa de contextos de 8 bits.

Algorithm 11: Reducción de contextos

```
Data: Contexto a eliminar
Result: Lista de contextos
1 begin
    // Eliminamos el contexto en el nivel actual.
2   eliminarContextoNivel(contextoEliminar);
    // En el nivel superior unificamos los contextos.
3   foreach contexto en el nivel superior do
4     if contexto se puede acceder con una transición al contextoEliminar then
5       unificamosContexto(contexto);
6     end
7   end
8 end
```

Nivel 0	(00000000,0,0)			
Nivel 1	(00000001,0,0)	(00001000,0,0)	(01000000,0,0)	...
Nivel 2	(00000011,0,0)	(00101000,0,0)	(01000100,0,0)	...
Nivel 3	(00000111,0,0)	(00111000,0,0)	(01000110,0,0)	...
Nivel 4	(00001111,0,0)	(01111000,0,0)	(01000111,0,0)	...
Nivel 5	(00011111,0,0)	(11111000,0,0)	(11000111,0,0)	...
Nivel 6	(00111111,0,0)	(11111100,0,0)	(11100111,0,0)	...
Nivel 7	(01111111,0,0)	(10111111,0,0)	(11011111,0,0)	...
Nivel 8	(11111111,0,0)			

Cuadro 4.1: Mapa contextos 8 bits

En la tabla 4.2 se muestran el mapa de coeficientes del bitplane 7, por el que se empezará a codificar.

0	0	0	0
0	0	1	0
0	0	0	0

Cuadro 4.2: Mapa de coeficientes del bitplane 7

El mapa de contextos de coeficientes se muestra en la tabla 4.3, donde cada elemento del mapa es una tupla, que referencia (nivel,índice) de un contexto en el mapa de contextos.

Se realiza el procesamiento de los coeficientes con el orden scanning order de JPEG2000.

El primer coeficiente no es significante, por lo que se mira el contexto que le corresponde en el mapa de contextos de coeficientes, y se accede al mapa de contextos al nivel 0 posición 0 para incrementar el número de veces que ese contexto no ha sido significante. Segundo coeficiente no es significante, incrementamos el Us (unsignificant context) del contexto.

Así succesivamente hasta llegar al octavo coeficiente. El octavo coeficiente es significante, incrementamos el Sc (significant context) del contexto. Y como ha sido significante,

(0,0)	(0,0)	(0,0)	(0,0)
(0,0)	(0,0)	(0,0)	(0,0)
(0,0)	(0,0)	(0,0)	(0,0)

Cuadro 4.3: Mapa contexto de coeficientes

se ha de actualizar el contexto de los vecinos de alrededor.

Los valores que tiene el mapa de contextos en ese momento se pueden ver en la tabla 4.4.

Nivel 0	(00000000,0,**7**)			
Nivel 1	(00000001,0,0)	(00000010,0,0)	(00000100,0,0)	...
Nivel 2	(00000011,0,0)	(00101000,0,0)	(01000100,0,0)	...
Nivel 3	(00000111,0,0)	(00111000,0,0)	(01000110,0,0)	...
Nivel 4	(00001111,0,0)	(01111000,0,0)	(01000111,0,0)	...
Nivel 5	(00011111,0,0)	(11111000,0,0)	(11000111,0,0)	...
Nivel 6	(00111111,0,0)	(11111100,0,0)	(11100111,0,0)	...
Nivel 7	(01111111,0,0)	(10111111,0,0)	(11011111,0,0)	...
Nivel 8	(11111111,0,0)			

Cuadro 4.4: Mapa contextos 8 bits, tras procesar 7 contextos no significantes.

Para cada uno de los vecinos, calculamos el nuevo contexto que les corresponde. El vecino 1, situado en la casilla superior izquierda al coeficiente significante, inicialmente estaba en el contexto 00000000, pero ahora estaría en el 00000001, por lo tanto desde el nivel 1 hasta el 8 por se busca un contexto equivalente a 00000001 y se encuentra uno solo, justo en el nivel inferior (nivel 1). Se actualiza el mapa de contextos de coeficientes para que apunte al nuevo contexto. El vecino 2 pasa del contexto 00000000 al nuevo contexto que es 00000010, el vecino 3 pasa del contexto 00000000 al nuevo contexto que es 00000100. Sucesivamente se realiza los mismos pasos para los 5 vecinos restantes.

En la tabla 4.5 se muestra el mapa de contextos de coeficientes actualizado.

(0,0)	(1,1)	(1,4)	(1,7)
(0,0)	(1,2)	(0,0)	(1,8)
(0,0)	(1,3)	(1,6)	(1,9)

Cuadro 4.5: Mapa contexto de coeficientes

Cabe resaltar (como se ha especificado en el algoritmo con anterioridad) que la actualización del mapa de contexto de coeficientes se realiza en todos los bitplanes. Este proceso se sigue realizando para cada coeficiente del bitplane 6.

A continuación se realiza el mismo proceso con los coeficientes del bitplane 5 y se reiterará hasta llegar al bitplane 0. Tras el cual, obtendremos varios mapas de contexto (concretamente uno para cada bitplane), que nos indicarán las estadísticas de los contextos para cada bitplane. Se puede apreciar un ejemplo en la tabla 4.6 y 4.7 para los bitplanes 6 y 5 respectivamente, tras procesar la imagen Lena con un tamaño de 512x512.

	bitplane 6
nivel 0	(00000000,1299,174893)
nivel 1	(10000000,185,1918) (00000001,0,0) (00000010,0,0) (00000100,160,862) (00001000,0,0) (00010000,2134,1284) (00100000,638,1519) (01000000,382,767)
nivel 2	(10000001,0,0) (10000010,0,0) (10000100,30,55) (10001000,0,0) (10010000,1039,813) (10100000,85,168) (11000000,530,1422) (00000011,0,0) (00000101,0,0) (00000110,0,0) (00001001,0,0) (00001010,0,0) (00001100,0,0) (00010001,0,0) (00010010,0,0) (00010100,541,238) (00011000,0,0) (00100001,0,0) (00100010,0,0) (00100100,161,76) (00101000,0,0) (00110000,415,132) (01000001,0,0) (01000010,0,0) (01000100,14,23) (01001000,0,0) (01010000,714,156) (01100000,610,810)
nivel 3	(10000011,0,0) (10000101,0,0) (10000110,0,0) (10001001,0,0) (10001010,0,0) (10001100,0,0) (10010001,0,0) (10010010,0,0) (10010100,226,97) (10011000,0,0) (10100001,0,0) (10100010,0,0) (10100100,21,15) (10101000,0,0) (10110000,286,81) (11000001,0,0) (11000010,0,0) (11000100,19,33) (11001000,0,0) (11010000,16180,1184) (11100000,950,1632) (00000111,0,0) (00001011,0,0) (00001101,0,0) (00001110,0,0) (00010011,0,0) (00010101,0,0) (00010110,0,0) (00011001,0,0) (00011010,0,0) (00011100,0,0) (00100011,0,0) (00100101,0,0) (00100110,0,0) (00101001,0,0) (00101010,0,0) (00101100,0,0) (00110001,0,0) (00110010,0,0) (00110100,176,42) (00111000,0,0) (01000011,0,0) (01000101,0,0) (01000110,0,0) (01001001,0,0) (01001010,0,0) (01001100,0,0) (01010001,0,0) (01010010,0,0) (01010100,65,17) (01011000,0,0) (01100001,0,0) (01100010,0,0) (01100100,109,44) (01101000,0,0) (01110000,1030,96)
nivel 4	(10000111,0,0) (10001011,0,0) (10001101,0,0) (10001110,0,0) (10010011,0,0) (10010101,0,0) (10010110,0,0) (10011001,0,0) (10011010,0,0) (10011100,0,0) (10100011,0,0) (10100101,0,0) (10100110,0,0) (10101001,0,0) (10101010,0,0) (10101100,0,0) (10110001,0,0) (10110010,0,0) (10110100,132,23) (10111000,0,0) (11000011,0,0) (11000101,0,0) (11000110,0,0) (11001001,0,0) (11001010,0,0) (11001100,0,0) (11010001,0,0) (11010010,0,0) (11010100,439,101) (11011000,0,0) (11100001,0,0) (11100010,0,0) (11100100,115,107) (11101000,0,0) (11110000,29973,769) (00001111,0,0) (00010111,0,0) (00011011,0,0) (00011101,0,0) (00011110,0,0) (00100111,0,0) (00101011,0,0) (00101101,0,0) (00101110,0,0) (00110011,0,0) (00110101,0,0) (00110110,0,0) (00111001,0,0) (00111010,0,0) (00111100,0,0) (01000111,0,0) (01001011,0,0) (01001101,0,0) (01001110,0,0) (01010011,0,0) (01010101,0,0) (01010110,0,0) (01011001,0,0) (01011010,0,0) (01011100,0,0) (01100011,0,0) (01100101,0,0) (01100110,0,0) (01101001,0,0) (01101010,0,0) (01101100,0,0) (01110001,0,0) (01110010,0,0) (01110100,442,23) (01111000,0,0)
nivel 5	(10001111,0,0) (10010111,0,0) (10011011,0,0) (10011101,0,0) (10011110,0,0) (10100111,0,0) (10101011,0,0) (10101101,0,0) (10101110,0,0) (10110011,0,0) (10110101,0,0) (10110110,0,0) (10111001,0,0) (10111010,0,0) (10111100,0,0) (11000111,0,0) (11001011,0,0) (11001101,0,0) (11001110,0,0) (11010011,0,0) (11010101,0,0) (11010110,0,0) (11011001,0,0) (11011010,0,0) (11011100,0,0) (11100011,0,0) (11100101,0,0) (11100110,0,0) (11101001,0,0) (11101010,0,0) (11101100,0,0) (11110001,0,0) (11110010,0,0) (11110100,13520,124) (11111000,0,0) (00011111,0,0) (00101111,0,0) (00110111,0,0) (00111011,0,0) (00111101,0,0) (00111110,0,0) (01001111,0,0) (01010111,0,0) (01011011,0,0) (01011101,0,0) (01011110,0,0) (01100111,0,0) (01101011,0,0) (01101101,0,0) (01101110,0,0) (01110011,0,0) (01110101,0,0) (01110110,0,0) (01111001,0,0) (01111010,0,0) (01111100,0,0)
nivel 6	(10011111,0,0) (10101111,0,0) (10110111,0,0) (10111011,0,0) (10111101,0,0) (10111110,0,0) (11001111,0,0) (11010111,0,0) (11011011,0,0) (11011101,0,0) (11011110,0,0) (11100111,0,0) (11101011,0,0) (11101101,0,0) (11101110,0,0) (11110011,0,0) (11110101,0,0) (11110110,0,0) (11111001,0,0) (11111010,0,0) (11111100,0,0) (00111111,0,0) (01011111,0,0) (01101111,0,0) (01110111,0,0) (01111011,0,0) (01111101,0,0) (01111110,0,0)
nivel 7	(10111111,0,0) (11011111,0,0) (11101111,0,0) (11110111,0,0) (11111011,0,0) (11111101,0,0) (11111110,0,0) (01111111,0,0)
nivel 8	(11111111,0,0)

Cuadro 4.6: Mapa contextos del bitplane 6 tras procesar la imagen Lena 512x512

Reducción de contextos:

Tras haber procesado todos los bitplanes, se realiza un reducción del mapa de contextos, inicialmente se aplica la política de reducción de contextos nulos (sección 3.3.1 página 39). Aplicando esta reducción se puede observar un descenso considerable del número de contextos, en la tabla 4.9 se puede observar como de $256 * 7 = 1792$ contextos totales que contiene el mapa de contextos (sumando todos los bitplanes) tras aplicar esta reducción se obtienen 1489 contextos, reduciendo hasta un 13.88 %.

En imágenes con un tamaño menor (el mismo ejemplo con Lena 16x16), la política de eliminación de nulos puede alcanzar hasta una reducción inicial del 80 % de los contextos,

	bitplane 5
nivel 0	(00000000,1988,98864)
nivel 1	(10000000,435,2733) (00000001,70,324) (00000010,14,26) (00000100,212,1117) (00001000,20,9) (00010000,2045,2019) (00100000,545,2132) (01000000,581,1616)
nivel 2	(10000001,6,71) (10000010,2,6) (10000100,48,102) (10001000,6,1) (10010000,1023,1172) (10100000,175,446) (11000000,702,1845) (00000011,52,87) (00000101,16,24) (00000110,16,41) (00001001,26,14) (00001010,10,1) (00001100,2,1) (00010001,46,17) (00010010,15,2) (00010100,508,376) (00011000,12,2) (00100001,41,51) (00100010,8,1) (00100100,103,144) (00101000,14,16) (00110000,429,282) (01000001,14,12) (01000010,1,1) (01000100,39,55) (01001000,13,1) (01010000,732,400) (01100000,689,1074)
nivel 3	(10000011,10,15) (10000101,0,3) (10000110,2,3) (10001001,2,8) (10001010,0,0) (10001100,2,0) (10010001,19,24) (10010010,9,2) (10010100,270,169) (10011000,1,1) (10100001,4,9) (10100010,3,0) (10100100,44,31) (10101000,6,3) (10110000,468,230) (11000001,15,30) (11000010,1,1) (11000100,47,65) (11001000,9,0) (11010000,10838,1428) (11100000,1127,2043) (00000111,42,116) (00001011,43,8) (00001101,2,0) (00001110,8,1) (00010011,43,2) (00010101,19,6) (00010110,97,74) (00011001,10,3) (00011010,0,0) (00011100,5,0) (00100011,13,8) (00100101,13,7) (00100110,3,6) (00101001,56,34) (00101010,6,1) (00101100,6,0) (00110001,12,2) (00110010,3,1) (00110100,150,86) (00111000,5,1) (01000011,2,3) (01000101,0,0) (01000110,0,2) (01001001,9,1) (01001010,0,0) (01001100,0,0) (01010001,15,1) (01010010,4,0) (01010100,88,46) (01011000,5,1) (01100001,21,14) (01100010,6,0) (01100100,112,95) (01101000,90,31) (01110000,873,240)
nivel 4	(10000111,4,13) (10001011,13,3) (10001101,0,0) (10001110,1,0) (10010011,15,2) (10010101,5,3) (10010110,92,9) (10011001,1,1) (10011010,0,0) (10011100,0,0) (10100011,1,1) (10100101,0,2) (10100110,2,3) (10101001,6,14) (10101010,6,0) (10101100,0,0) (10110001,9,8) (10110010,2,1) (10110100,198,68) (10111000,3,1) (11000011,5,12) (11000101,0,2) (11000110,2,5) (11001001,9,1) (11001010,0,0) (11001100,3,0) (11010001,210,62) (11010010,83,0) (11010100,754,148) (11011000,76,0) (11100001,18,19) (11100010,5,4) (11100100,195,197) (11101000,165,15) (11110000,21012,1149) (00001111,49,5) (00010111,355,36) (00011011,16,0) (00011101,4,0) (00011110,0,0) (00100111,12,9) (00101011,73,4) (00101101,13,1) (00101110,9,0) (00110011,7,0) (00110101,5,3) (00110110,26,23) (00111001,5,2) (00111010,1,0) (00111100,3,1) (01000111,1,7) (01001011,5,0) (01001101,0,0) (01001110,0,0) (01010011,9,2) (01010101,0,0) (01010110,12,6) (01011001,5,0) (01011010,2,0) (01011100,0,0) (01100011,3,1) (01100101,0,2) (01100110,2,7) (01101001,91,8) (01101010,3,0) (01101100,6,6) (01110001,34,4) (01110010,4,2) (01110100,362,78) (01111000,11,8)
nivel 5	(10001111,9,1) (10010111,89,3) (10011011,9,2) (10011101,3,0) (10011110,0,0) (10100111,1,2) (10101011,19,3) (10101101,0,0) (10101110,4,0) (10110011,5,0) (10110101,5,0) (10110110,26,3) (10111001,5,5) (10111010,0,0) (10111100,1,0) (11000111,8,20) (11001011,4,5) (11001101,0,0) (11001110,0,0) (11010011,106,16) (11010101,60,8) (11010110,183,5) (11011001,71,7) (11011010,8,0) (11011100,9,0) (11100011,2,9) (11100101,5,3) (11100110,3,35) (11101001,70,1) (11101010,0,0) (11101100,41,6) (11110001,476,26) (11110010,164,0) (11110100,9657,229) (11111000,696,4) (00011111,51,0) (00101111,41,2) (00110111,55,6) (00111011,18,0) (00111101,2,0) (00111110,5,0) (01001111,4,0) (01010111,21,2) (01011011,15,0) (01011101,0,0) (01011110,1,0) (01100111,3,6) (01101011,60,0) (01101101,10,1) (01101110,1,0) (01110011,12,0) (01110101,17,0) (01110110,27,19) (01111001,20,2) (01111010,2,0) (01111100,8,12)
nivel 6	(10011111,10,0) (10101111,17,1) (10110111,23,5) (10111011,12,2) (10111101,3,1) (10111110,0,0) (11001111,17,6) (11010111,296,19) (11011011,64,31) (11011101,14,2) (11011110,13,1) (11100111,17,66) (11101011,22,1) (11101101,3,0) (11101110,4,1) (11110011,230,16) (11110101,359,2) (11110110,594,16) (11111001,300,5) (11111010,58,0) (11111100,310,1) (00111111,39,0) (01011111,20,0) (01101111,27,0) (01110111,71,8) (01111011,44,1) (01111101,7,1) (01111110,7,1)
nivel 7	(10111111,10,1) (11011111,130,5) (11101111,17,2) (11110111,1085,47) (11111011,322,13) (11111101,238,2) (11111110,136,1) (01111111,58,0)
nivel 8	(11111111,910,7)

Cuadro 4.7: Mapa contextos del bitplane 5 tras procesar la imagen Lena 512x512

esto se debe a que se procesan pocos coeficientes en comparación con la cantidad de contextos que hay en total, así que hay gran parte de contextos que no se dan uso y quedan como nulos ($Sc + Uc = 0$), pudiendo reducirlos.

Es importante observar que aunque se reducen el número de contextos, los valores correspondientes al número de significantes y no significantes permanece estático, lo cual nos indica que no se ha roto la integridad del mapa de contextos, que aunque se ha reducido en número de contextos, los contextos restantes son capaces de representar cada uno de los contextos que se han eliminado. El mapa de contextos resultantes (de los bitplanes 6 y 5) se puede observar en la tabla 4.8.

4.4. Valoración de un mapa de contextos

Hasta este punto se ha explicado como extraer de modo automatizado una posible configuración de contextos que se ajuste a la imagen que estamos procesando. Ahora bien,

	bitplane 6
nivel 0	(0000x0xx,1299,174893)
nivel 1	(1000x0xx,185,1918) (0000x1xx,160,862) (0001x0xx,2134,1284) (0010x0xx,638,1519) (0100x0xx,382,767)
nivel 2	(1000x1xx,30,55) (1001x0xx,1039,813) (1010x0xx,85,168) (1100x0xx,530,1422) (0001x1xx,541,238) (0010x1xx,161,76) (0011x0xx,415,132) (0100x1xx,14,23) (0101x0xx,714,156) (0110x0xx,610,810)
nivel 3	(1001x1xx,226,97) (1010x1xx,21,15) (1011x0xx,286,81) (1100x1xx,19,33) (1101x0xx,16180,1184) (1110x0xx,950,1632) (0011x1xx,176,42) (0101x1xx,65,17) (0110x1xx,109,44) (0111x0xx,1030,96)
nivel 4	(1011x1xx,132,23) (1101x1xx,439,101) (1110x1xx,115,107) (1111x0xx,29973,769) (0111x1xx,442,23)
nivel 5	(1111x1xx,13520,124)
nivel 6	
nivel 7	
nivel 8	
	bitplane 5
nivel 0	(00000000,1988,98864)
nivel 1	(10000000,435,2733) (00000001,70,324) (00000010,14,26) (00000100,212,1117) (00001000,20,9) (00010000,2045,2019) (00100000,545,2132) (01000000,581,1616)
nivel 2	(10000001,6,71) (1000x010,2,6) (10000100,48,102) (10010x0,6,1) (10010000,1023,1172) (10100000,175,446) (11000000,702,1845) (00000011,52,87) (0x000101,16,24) (00000110,16,41) (00001001,26,14) (xx0x1010,10,1) (0x001100,2,1) (00010001,46,17) (0001x010,15,2) (00010100,508,376) (000110x0,12,2) (00100001,41,51) (00100010,8,1) (00100100,103,144) (00101000,14,16) (00110000,429,282) (01000x01,14,12) (0100x010,1,1) (0100x10x,39,55) (01001xx0,13,1) (01010000,732,400) (01100000,689,1074)
nivel 3	(10000011,10,15) (1000x101,0,3) (10000110,2,3) (10001x01,2,8) (10xx110x,2,0) (10010001,19,24) (1001x010,9,2) (1001x100,270,169) (10011xx0,1,1) (10100001,4,9) (10100010,3,0) (1010x100,44,31) (10101x00,6,3) (10110000,468,230) (11000001,15,30) (1100x010,1,1) (11000100,47,65) (110010x0,9,0) (11010000,10838,1428) (11100000,1127,2043) (00000111,42,116) (00001011,43,8) (xx001101,2,0) (0x0x1110,8,1) (00010011,43,2) (0x010101,19,6) (0001x110,97,74) (00011001,10,3) (xx0111x0,5,0) (00100011,13,8) (00100101,13,7) (00100110,3,6) (00101001,56,34) (00101010,6,1) (x0101100,6,0) (00110001,12,2) (00110010,3,1) (00110100,150,86) (00111000,5,1) (01000011,2,3) (0100x110,0,2) (01001x01,9,1) (01010x01,15,1) (01010010,4,0) (0101x10x,88,46) (01011x00,5,1) (01100001,21,14) (01100010,6,0) (01100100,112,95) (01101000,90,31) (01110000,873,240)
nivel 4	(10000111,4,13) (10001011,13,3) (1x0x1110,1,0) (10010011,15,2) (10010101,5,3) (1001x110,92,9) (10011001,1,1) (10100011,1,1) (1010x101,0,2) (10100110,2,3) (10101x01,6,14) (1x1x1010,6,0) (10110001,9,8) (1011x010,2,1) (10110100,198,68) (101110x0,3,1) (11000011,5,12) (1100x101,0,2) (1100x110,2,5) (11001x01,9,1) (110011xx,3,0) (11010001,210,62) (11010010,83,0) (11010100,754,148) (11011000,76,0) (11100001,18,19) (1110x010,5,4) (11100100,195,197) (111010x0,165,15) (11110000,21012,1149) (00001111,49,5) (00010111,355,36) (00011011,16,0) (0x011101,4,0) (00100111,12,9) (00101011,73,4) (x0101101,13,1) (00101110,9,0) (00110011,7,0) (00110101,5,3) (00110110,26,23) (00111001,5,2) (x0111010,1,0) (00111100,3,1) (01000111,1,7) (01001011,5,0) (01010011,9,2) (01010110,12,6) (01011x01,5,0) (01011010,2,0) (01100011,3,1) (01100101,0,2) (01100110,2,7) (01101001,91,8) (x1101010,3,0) (01101100,6,6) (01110001,34,4) (01110010,4,2) (01110100,362,78) (01111000,11,8)
nivel 5	(10001111,9,1) (10010111,89,3) (10011011,9,2) (10011101,3,0) (10100111,1,2) (10101011,19,3) (101x1110,4,0) (10110011,5,0) (10110101,5,0) (1011x110,26,3) (10111001,5,5) (101111x0,1,0) (11000111,8,20) (11001011,4,5) (11010011,106,16) (11010101,60,8) (11010110,183,5) (11011001,71,7) (11011010,8,0) (11011100,9,0) (11100011,2,9) (11100101,5,3) (11100110,3,35) (11101001,70,1) (11101100,41,6) (11110001,476,26) (11110010,164,0) (11110100,9657,229) (11111000,696,4) (00011111,51,0) (00101111,41,2) (00110111,55,6) (00111011,18,0) (00111101,2,0) (x0111110,5,0) (01001111,4,0) (01010111,21,2) (01011011,15,0) (01011110,1,0) (01100111,3,6) (01101011,60,0) (01101101,10,1) (01101110,1,0) (01110011,12,0) (01110101,17,0) (01110110,27,19) (01111001,20,2) (01111010,2,0) (01111100,8,12)
nivel 6	(10011111,10,0) (10101111,17,1) (10110111,23,5) (10111011,12,2) (10111101,3,1) (11001111,17,6) (11010111,296,19) (11011011,64,31) (11011101,14,2) (11011110,13,1) (11100111,17,66) (11101011,22,1) (11101101,3,0) (11101110,4,1) (11110011,230,16) (11110101,359,2) (11110110,594,16) (11111001,300,5) (11111010,58,0) (11111100,310,1) (00111111,39,0) (01011111,20,0) (01101111,27,0) (01110111,71,8) (01111011,44,1) (01111101,7,1) (01111110,7,1)
nivel 7	(10111111,10,1) (11011111,130,5) (11101111,17,2) (11110111,1085,47) (11111011,322,13) (11111101,238,2) (11111110,136,1) (01111111,58,0)
nivel 8	(11111111,910,7)

Cuadro 4.8: Mapa contextos del bitplane 6 y 5, con contextos de 8 bits tras procesar la imagen Lena 512x512 y aplicar la reducción con la política de eliminación de contextos nulos.

se debe tener una medida para saber cuando una configuración de contextos o mapa de contextos es mejor que otra. Con este fin, introducimos la entropía de un mapa de contextos.

Entropía de un mapa de contextos:

Con la entropía de un mapa de contextos se busca una medida representativa de la información que nos aporta un mapa de contextos, con el fin de poder compararlo con otros mapas de contextos y ver si realmente se consigue una mejora.

La entropía aplicada al ámbito de la información, se define como el grado de incertidumbre que existe sobre un conjunto de datos o una medida de la información contenida

			Antes de la reducción
Bitplane	Número contextos	Significantes	NoSignificantes
0	256	4841	2371
1	256	9050	7212
2	256	15660	16262
3	256	29082	31922
4	256	61602	61004
5	256	66918	122606
6	256	72620	189524
Significantes: 259773 NoSignificantes: 430901 Número contextos: 1792			
			Tras la reducción
Bitplane	Número contextos	Significantes	NoSignificantes
0	212	4841	2371
1	255	9050	7212
2	254	15660	16262
3	255	29082	31922
4	248	61602	61004
5	233	66918	122606
6	32	72620	189524
Significantes: 259773 NoSignificantes: 430901 Número contextos: 1489			

Cuadro 4.9: Estadísticas antes y tras la reducción

en un mensaje, en este caso concreto, contenida en el mapa de contexto. La fórmula 4.1 se utiliza para el cálculo de la entropía.

$$H(x) = \sum_{i=1}^{n} *P(i) * \log(\frac{1}{P(i)}) \tag{4.1}$$

Para realizar el cálculo de la entropía de un mapa de contextos, primero hay que saber calcular la probabilidad de cada elemento que lo componen y la entropía de estos, para luego calcular la entropía de todo el mapa. La probabilidad de un contexto viene dada por la fórmulas 4.2 y 4.3.

$$P(significante) = P(1) = \frac{\sharp significantes}{\sharp significante + \sharp nosignificante} \tag{4.2}$$

$$P(nosignificante) = P(0) = \frac{\sharp nosignificantes}{\sharp significante + \sharp nosignificante} \tag{4.3}$$

La fórmula 4.4 especifica como calcular la entropía de un mapa de contextos. En esta fórmula se hace referencia a X_i que es la entropía de un contexto y Y_i que es la probabilidad

de un contexto, las cuales se calculan con la fórmula 4.5 y 4.6 respectivamente.

$$EntropiaMapaContexto = \sum_i X_i * Y_i \tag{4.4}$$

$$X_i = EntropiaContexto = P(1) * \log_2(\frac{1}{P(1)}) + P(0) * \log_2(\frac{1}{P(0)}) \tag{4.5}$$

$$Y_i = ProbabilidadContexto = \frac{\sharp significante + \sharp nosignificante}{\sharp CS_{mc} + \sharp CU_{mc}} \tag{4.6}$$

En la ecuación 4.6 CS_{mc} corresponde a Coeficientes significantes del mapa de contextos y CU_{mc} corresponde a Coeficientes no significantes del mapa de contextos.

4.5. Resultados experimentales

Ya se ha visto como extraer los mapas de contextos de una imagen en concreto y como valorar este mapa de contextos según la información que nos aporta, a continuación se realizan varios experimentos valorando los resultados que se obtienen y comparando con los resultados que nos aporta los contextos del estándar JPEG2000.

Se han realizado un total de 4 experimentos, con las siguientes características:

- Se han escogido imágenes en gris del Corpus CCITT, con un tamaño de 512x512, codificadas en PGM, 8 bits por pixel (bpp).
- Se ha procesado la imagen mediante la codificación JPEG2000, realizando la Transformada wavelet de nivel 1.
- Cada subbanda se ha cogido como un solo bloque.
- Las políticas utilizadas para la reducción de contextos, son inicialmente la reducción de contextos nulos y luego se aplica la reducción de contextos irrelevantes con un threshold de un 10 % hasta que no se reduzcan más contextos (sección 3.3.1 página 39).

Se han realizado los ejemplos con 4 imágenes que se muestran en la figura 4.5. Las imágenes corresponden a Lena, Portrait, Cafeteria y Bicyle respectivamente. Los resultados

de procesar la primera imagen de este grupo escogido se muestran en las tablas 4.10 y 4.11. Los resultados de la segunda se muestran en las tablas 4.12 y 4.13, los resultados de la tercera imagen se muestran en las tablas 4.14 y 4.15, finalmente los resultados de la última imagen se muestran en las tablas 4.16 y 4.17.

Figura 4.4: Imagenes de los experimentos

4.5.1. Análisis de los resultados

En las tablas 4.11, 4.13, 4.15, 4.17 se muestra la información de la entropía en cada una de las subbandas. Para cada subbanda tras realizar la extracción automatizada de contextos, obteniendo un mapa de contextos correspondiente para esa subbanda, se realiza inicialmente una reducción de contextos nulos. Para todos los ejemplos cuando se eliminan contextos nulos la entropía se mantiene, esto se debe a que estos contextos no aportan información alguna al mapa de contextos, así que prescindiendo de ellos el mapa de contextos sigue aportando la misma información, aunque no disminuya la entropía, si que se reduce el número de contextos.

Tras realizar la primera eliminación de contextos nulos, se reitera con la reducción de contextos irrelevantes con un threshold del 10 %, donde se observa que se reducen el número de contextos a la par que la entropía del mapa de contextos aumenta, es decir, al reducir el número de contextos del mapa de contextos los contextos restantes nos aportan

más información. Tras esto se puede comparar la entropía inicial que tiene el mapa de contextos (extraído automáticamente) y de el mapa de contextos final, en todos los casos se consigue un aumento en la entropía del mapa.

La medida de entropía además nos aporta información acerca cuantos bits teóricos son necesarios para la codificación final de la imagen utilizando ese mapa de contextos: $entropia * numSimbols = \sharp bits$.

Para analizar los resultados desde el punto de vista de toda la imagen, solo hay que hacer la media de la entropías de las subbandas, para obtener la entropía de de la codificación de la imagen. Con el fin de poder comparar el uso de los contextos del estándar JPEG2000 (JP2) con el uso de la extracción automatizada de contextos (CTX), calculamos para cada una de las imágenes la entropía de la codificación con el mapa de contextos que nos aporta el JP2 y el CTX, a parte de la cantidad de bits teóricos necesario para codificar con JP2 y con CTX. Los resultados para cada una de las imágenes se muestran en las tablas 4.11, 4.13, 4.15 y 4.17.

Puede observarse que en todos los casos, se consigue una reducción en la entropía de mapa de contextos de la imagen está entre el 5 % el 10 % de la entropía JP2. Esto quiere decir que se reduce el número de bits necesarios para representar la codificación del mapa de contextos. Estos resultados inducen a pensar que se puede conseguir cierta mejora en la codificación utilizando la extracción automatizada de contextos. Aunque ese porcentaje de mejora se ve reducido por la necesidad de codificar y transmitir el mapa de contextos elegido al codificador aritmético MQ.

		Imagen Lena
Subbanda LL - Entropía inicial = 0.37885457		
Política reducción	Contextos reducidos	Entropía
Contextos nulos	387	0.37885457
Contextos irrelevantes, th 10 %	57	0.37937087
Contextos irrelevantes, th 10 %	9	0.37945312
Contextos irrelevantes, th 10 %	1	0.3794735
Contextos irrelevantes, th 10 %	2	0.37947774
Contextos irrelevantes, th 10 %	2	0.37948203
Contextos irrelevantes, th 10 %	0	0.37948203
Subbanda HL - Entropía inicial = 0.35650977		
Política reducción	Contextos reducidos	Entropía
Contextos nulos	546	0.35650977
Contextos irrelevantes, th 10 %	277	0.36165738
Contextos irrelevantes, th 10 %	41	0.36181995
Contextos irrelevantes, th 10 %	4	0.36183938
Contextos irrelevantes, th 10 %	2	0.36184216
Contextos irrelevantes, th 10 %	1	0.3618424
Contextos irrelevantes, th 10 %	0	0.3618424
Subbanda LH - Entropía inicial = 0.38995394		
Política reducción	Contextos reducidos	Entropía
Contextos nulos	495	0.38995394
Contextos irrelevantes, th 10 %	251	0.39552894
Contextos irrelevantes, th 10 %	24	0.39575422
Contextos irrelevantes, th 10 %	5	0.39579687
Contextos irrelevantes, th 10 %	1	0.39583802
Contextos irrelevantes, th 10 %	1	0.39584807
Contextos irrelevantes, th 10 %	0	0.39584807
Subbanda HH - Entropía inicial = 0.36056277		
Política reducción	Contextos reducidos	Entropía
Contextos nulos	669	0.36056277
Contextos irrelevantes, th 10 %	286	0.36630696
Contextos irrelevantes, th 10 %	33	0.36663604
Contextos irrelevantes, th 10 %	7	0.36667833
Contextos irrelevantes, th 10 %	1	0.3666786
Contextos irrelevantes, th 10 %	0	0.3666786

Cuadro 4.10: Entropía de las subbandas tras procesar la imagen Lena con la extracción de automatizada de contextos

	Imagen Lena
Entropía con JP2	0.41
Entropía con CTX	0.38
Bytes codificación para JP2	64081.0
Bytes codificación para CTX	59851.0

Cuadro 4.11: Comparativa entre JP2 y CTX de la imagen Lena

Imagen Portrait n1		
Subbanda LL - Entropía inicial = 0.27292708		
Política reducción	Contextos reducidos	Entropía
Contextos nulos	600	0.27292708
Contextos irrelevantes, th 10 %	32	0.27392077
Contextos irrelevantes, th 10 %	1	0.2739226
Contextos irrelevantes, th 10 %	1	0.27392414
Contextos irrelevantes, th 10 %	0	0.27392414
Subbanda HL - Entropía inicial = 0.30288327		
Política reducción	Contextos reducidos	Entropía
Contextos nulos	504	0.30288327
Contextos irrelevantes, th 10 %	171	0.30464396
Contextos irrelevantes, th 10 %	13	0.3047674
Contextos irrelevantes, th 10 %	1	0.30476946
Contextos irrelevantes, th 10 %	0	0.30476946
Subbanda LH - Entropía inicial = 0.29315946		
Política reducción	Contextos reducidos	Entropía
Contextos nulos	561	0.29315946
Contextos irrelevantes, th 10 %	176	0.2956092
Contextos irrelevantes, th 10 %	19	0.29572603
Contextos irrelevantes, th 10 %	0	0.29572603
Subbanda HH - Entropía inicial = 0.28226855		
Política reducción	Contextos reducidos	Entropía
Contextos nulos	588	0.28226855
Contextos irrelevantes, th 10 %	161	0.28389186
Contextos irrelevantes, th 10 %	23	0.28399858
Contextos irrelevantes, th 10 %	4	0.28405207
Contextos irrelevantes, th 10 %	1	0.28405678
Contextos irrelevantes, th 10 %	0	0.28405678

Cuadro 4.12: Entropía de las subbandas tras procesar la imagen Portrait n1 con la extracción de automatizada de contextos

Imagen Portrait n1	
Entropía con JP2	0.31
Entropía con CTX	0.29
Bytes codificación para JP2	43390.0
Bytes codificación para CTX	40959.0

Cuadro 4.13: Comparativa entre JP2 y CTX de la imagen Portrait n1

Imagen Cafeteria n2		
Subbanda LL - Entropía inicial = 0.4511905		
Política reducción	Contextos reducidos	Entropía
Contextos nulos	858	0.4511905
Contextos irrelevantes, th 10 %	49	0.45198587
Contextos irrelevantes, th 10 %	6	0.452159
Contextos irrelevantes, th 10 %	2	0.45217904
Contextos irrelevantes, th 10 %	1	0.4521798
Contextos irrelevantes, th 10 %	0	0.4521798
Subbanda HL - Entropía inicial = 0.39717326		
Política reducción	Contextos reducidos	Entropía
Contextos nulos	470	0.39717326
Contextos irrelevantes, th 10 %	199	0.4035298
Contextos irrelevantes, th 10 %	17	0.40442047
Contextos irrelevantes, th 10 %	2	0.40444258
Contextos irrelevantes, th 10 %	0	0.40444258
Subbanda LH - Entropía inicial = 0.4087062		
Política reducción	Contextos reducidos	Entropía
Contextos nulos	487	0.4087062
Contextos irrelevantes, th 10 %	236	0.41788602
Contextos irrelevantes, th 10 %	26	0.41813457
Contextos irrelevantes, th 10 %	4	0.41815525
Contextos irrelevantes, th 10 %	0	0.41815525
Subbanda HH - Entropía inicial = 0.4129597		
Política reducción	Contextos reducidos	Entropía
Contextos nulos	465	0.4129597
Contextos irrelevantes, th 10 %	234	0.42403036
Contextos irrelevantes, th 10 %	19	0.42413634
Contextos irrelevantes, th 10 %	4	0.42420322
Contextos irrelevantes, th 10 %	0	0.42420322

Cuadro 4.14: Entropía de las subbandas tras procesar la imagen Cafeteria n2 con la extracción de automatizada de contextos

Imagen Cafeteria n2	
Entropía con JP2	0.44
Entropía con CTX	0.42
Bytes codificación para JP2	58341.0
Bytes codificación para CTX	56776.0

Cuadro 4.15: Comparativa entre JP2 y CTX de la imagen Cafeteria n2

Imagen Bicycle n5		
Subbanda LL - Entropía inicial = 0.37071365		
Política reducción	Contextos reducidos	Entropía
Contextos nulos	1106	0.37071365
Contextos irrelevantes, th 10 %	41	0.37094015
Contextos irrelevantes, th 10 %	5	0.3710111
Contextos irrelevantes, th 10 %	0	0.3710111
Subbanda HL - Entropía inicial = 0.30976343		
Política reducción	Contextos reducidos	Entropía
Contextos nulos	434	0.30976343
Contextos irrelevantes, th 10 %	166	0.31681982
Contextos irrelevantes, th 10 %	12	0.31710848
Contextos irrelevantes, th 10 %	6	0.31714323
Contextos irrelevantes, th 10 %	0	0.31714323
Subbanda LH - Entropía inicial = 0.30542585		
Política reducción	Contextos reducidos	Entropía
Contextos nulos	463	0.30542585
Contextos irrelevantes, th 10 %	173	0.31330472
Contextos irrelevantes, th 10 %	24	0.31479377
Contextos irrelevantes, th 10 %	4	0.31481782
Contextos irrelevantes, th 10 %	0	0.31481782
Subbanda HH - Entropía inicial = 0.28360537		
Política reducción	Contextos reducidos	Entropía
Contextos nulos	570	0.28360537
Contextos irrelevantes, th 10 %	206	0.29027653
Contextos irrelevantes, th 10 %	17	0.29076335
Contextos irrelevantes, th 10 %	5	0.29083863
Contextos irrelevantes, th 10 %	0	0.29083863

Cuadro 4.16: Entropía de las subbandas tras procesar la imagen Bicycle n5 con la extracción de automatizada de contextos

	Imagen Bicycle n5
Entropía con JP2	0.35
Entropía con CTX	0.32
Bytes codificación para JP2	50086.0
Bytes codificación para CTX	47928.0

Cuadro 4.17: Comparativa entre JP2 y CTX de la imagen Bicycle n5

Capítulo 5

Conclusiones

En este capítulo se realiza una mirada atrás al trabajo desarrollado para analizar los resultados alcanzados y responder a las preguntas que se planteaban en el primer capítulo, ¿Cabe la posibilidad de extraer los contextos personalizados para cada tipo de imagen procesada? ¿Es posible automatizar este proceso?

5.1. Valoración

Se parte de la representación de contextos que aporta el estándar JPEG2000, una configuración estática que da buenos resultados para todas las imágenes, pero que no se adapta a el tipo de imagen que se procesa, por lo que cabe pensar se que podría mejorar la codificación adaptando esos contextos a la imagen procesada. Partiendo de esa idea, se ha investigado como dar la capacidad de que la configuración de contextos fuera dinámica según la imagen procesada, a este proceso se ha llamado extracción automatizada de contextos significativos.

Para ello se ha propuesto varias representaciones de configuración de contextos y se ha estudiado cual sería mejor para llevar a cabo la extracción automática de contextos. Se han desarrollado varios algoritmos para calcular estadísticas sobre la significancia y no significancia de los contextos al procesar los coeficientes de una imagen. Con tal de reducir este conjunto de contextos, se han diseñado una serie de algoritmos para unificar contextos y se han mostrado algunas políticas de reducción que nos permitían elegir el tipo

de reducción.

Hasta ese punto ya disponíamos la posibilidad de obtener un mapa de contextos relacionado con la imagen que se procesada, pero para poder comparar de modo teórico con los mapas que nos aporta el estándar JPEG2000, se ha introducido la medida de la entropía de un mapa de contextos, de modo que nos permita comparar la entropía utilizando los contextos del JPEG2000 y la entropía utilizando los contextos automáticos calculados. Y se han interpretado los resultados obtenidos con un conjunto de pruebas.

La valoración es que finalmente es posible realizar la extracción automatizada de contextos personalizados de un imagen y se perciben mejoras en la codificación. La extracción de estos, es un proceso de compleja implementación, ya que durante el proceso de reducción de contextos y unificación de estos se debe mantener la integridad del mapa de contextos.

Pero sin embargo, cuando se planeta la cuestión de si este proceso de extracción automatizada es viable para comprimir una imagen directamente, no se considera viable a día de hoy. Esto debe a que la extracción automatizada de contextos implementada requiere un elevado tiempo de cómputo, por lo tanto no es posible realizar un uso fluido de la extracción de contextos. Además el mapa de contextos debe transmitirse al codificador aritmético MQ y debe modificarse para poder aceptar este tipo de contextos, por lo tanto este trabajo deberá acompañarse con un segundo trabajo de investigación que permita adaptar el codificador aritmético MQ a la extracción automatizada.

5.2. Futuras líneas de investigación

Todo el trabajo desarrollado abre un camino interesante desde el cual se puede seguir investigando. A continuación se muestran algunos puntos de interés desde los cuales se podría trazar líneas de investigación.

- **Algoritmos implementados**: En todo momento se ha fijado como requisito optimizar los algoritmos implementados en el desarrollo práctico (ver apéndice A de la página 67), una de las razones era que la extracción automatizada de contextos es un proceso costoso en tiempo por la cantidad de coeficientes a procesar y el número de iteraciones que se realiza, por lo tanto para poder obtener resultados experimentales se necesitaba optimizar las operaciones. Aún así, se puede realizar un estudio con el

fin de optimizar más aún estas implementaciones para reducir el coste computacional en tiempo, que requiere el proceso de extracción y reducción de contextos.

- **Política de reducción** Utilizando la medida de la entropía como valoración de un mapa de contextos, se han escogido dos políticas de reducción de contextos. Si bien se pueden indagar en posibles políticas de contextos para obtener mejores resultados, ya sea necesitando menos iteraciones para obtener el mapa de contextos final, como para reducir gran cantidad de contextos y que el procesamiento del resto sea más rápido.

- **Adaptación codificador aritmético MQ** Este trabajo deja un camino claro a investigar, como transmitir estos mapas de contextos al codificador aritmético MQ y modificar este para que los pueda utilizar en la codificación.

Apéndice A

Optimización de la implementación

Debido al la cantidad de coeficientes que deben procesar y el número de iteraciones que se utilizan para el cálculo de las estadísticas y reducción de contextos, a la hora de la implementación se tiene que optimizar al máximo.

Algunas de las optimizaciones realizadas en la implementación son:

Representación de contextos:

Para la representación de los contextos se descartó utilizar strings, cuya manipulación podía ser más cómoda a la hora de realizar las busquedas de contextos utilizando expresiones regulares, para ver si un contexto era equivalente a otro, teniendo en cuenta que un contexto puede estar unificado. La decisión de descartar la representación de los contextos mediante strings fue por el coste en tiempo que acarrea trabajar con strings. Por esa razón se planteó la representación de contextos de manera binaria, utilizando bytes y operaciones lógicas. Como un contexto tenía una representación con 8 bits, inicialmente podría representarse con un byte.

El contexto 00000001 correspondría al byte 1. El contexto 00000010 correspondría al byte 2....

La principal dificultad era representar contextos unificados (por ejemplo, 00000X0) utilizando una representación en bytes. Para contemplar estos casos se tubo que ampliar la representación a 2 bytes.

El primer byte se denomina RTXObligatorio y el segundo RTXCompleto. El primero

hace referencia al contexto más reducido que puede representar el contexto unificado, es decir, un contexto como 000000XX, es capaz de representar al 00000000, 00000001, 00000010, 00000011. Luego el RTXObligatorio será 00000000.

El RTXCompleto, hace referencia al contexto más elevado que puede representar. En el ejemplo anterior será 00000011. Por lo tanto el contexto 000000XX será representado por (00000000,00000011) siendo (RTXObligatorio,RTXCompleto). De modo inverso (00100010,00100111) equivale al contexto 00100X1X.

Cada contexto será representado con 2 bytes (sin tener en cuenta la información que debe guardar del contexto SC y UC). Lo más interesante de esta representación es poder realizar operaciones lógicas entre bytes, que son de coste muy reducido en tiempo.

Operaciones con contextos:

Hay dos operaciones importantes a realizar entre contextos. Cuando se elimina un contexto, hay que buscar el contexto en el nivel superior para unificar este con el contexto eliminado. Para saber que contexto corresponde unificar se realiza las operaciones especificadas en el algoritmo 12, donde *elimContexto* es el contexto que se elimina y *actContext* es el contexto que están en los niveles superiores y que pueden unificarse, en el algoritmo también se especifica como se realiza la unificación.

Algorithm 12: Contextos a unificar

```
1 begin
2     unificar = false;
3     if (elimContext.RTXobligatorio AND actContext.RTXobligatorio)==
      actContext.RTXobligatorio then
4         unificar = true;
          // Método para la unificación
5         actContext.RTXcompleto = (byte) (actContext.RTXcompleto OR
          elimContext.RTXobligatorio);
6     end
7 end
```

La segunda operación importante que es necesaria implementar y realizarla con operaciones entre bytes, es la búsqueda de un contexto equivalente en el mapa de contextos. Por ejemplo el contexto 0001010 es equivalente a 00010XX, esto es necesario durante el proceso que se calculan las estadísticas. En el algoritmo 13 se muestra las operaciones a realizar,

se hace referencia a *realContext* que es el contexto que se va comparando con los contextos que hay en el mapa de contexto, en busca de su contexto equivalente.

Algorithm 13: Contextos equivalentes

```
1 begin
2 |   equivalente = false;
3 |   if (realContext AND actContext.RTXobligatorio) == actContext.RTXobligatorio then
4 |   |   if (realContext AND actContext.RTXcompleto) == realContext then
  |   |   |   // realContext es equivalente a actContext
5 |   |   |   equivalente = true;
6 |   |   end
7 |   end
8 end
```

Bibliografía

[1] F. Aulí, *"L'estàndard de compressió d'imatges JPEG2000"*, tesis para master, Universitat Autònoma de Barcelona,2004.

[2] JPEG2000 Home page. [Online] Disponible en http://www.jpeg.org/jpeg2000/index.html.

[3] Michael W. Marcellin, Michael J. Gormish, Ali Bilgin, Martin P. Boliek, *.An Overview of JPEG-2000"*, appeared in Proc. of IEEE Data Compression Conference, pp. 523-541, 2000.

[4] David S. Taubman and Micheal W. Marcellin, Fellow, IEEE, *"JPEG2000: Standard for Interactive Imaging"*.

[5] Majid Rabbani*, Rajan Joshi, *.An overview of the JPEG2000 still image compression standard "*, Signal Processing: Image Communication 17 (2002) 3Ű48.

Firmado: Alberto Arcos Hurtado

Bellaterra, Junio de 2007

Resumen

Se ha desarrollado un trabajo de investigación para arrojar un haz de luz en ciertas regiones de interés en la codificación del estándar JPEG2000. La finalidad de esta investigación, es estudiar la posibilidad de realizar una extracción automatizada de contextos durante el proceso de codificación, de modo que se ajusten a la imagen procesada y observar que ventajas nos aporta esto frente a los contextos establecidos por el estándar JPEG2000.

Resum

S'ha desenvolupat un treball d'investigació per obrir un camí cap a certes parts d'interés en la codificació de l'estàndard JPEG2000. La finalitat d'aquesta investigació, és estudiar la possibilitat de realitzar una extracció automatitzada de contextes durant el procés de codificació, amb relació a la imatge codificada i observar les millores que ens pot aportar això en contraposició als contextes establerts per l'estàndard JPEG2000.

Summary

A work of investigation has been developed to throw a beam of light in certain regions of interest in the codification of standard JPEG2000. The purpose of this investigation, is to study the possibility of making an extraction automated of contexts during the codification process, so that they adjust to the processed image and to observe that contribute this front to us to the contexts established by standard JPEG2000.