Desenvolupament de programari per la compressió d'imatges utilitzant els wavelets com alternativa a la transformada del cosinus.

Josep Mª Sanz Subirana.

Enginyeria Tècnica en Informàtica de Sistemes

Consultor: Veronica Vilaplana Besler

31/05/2013

Agraïments i dedicatòria

Un cop arribat el moment de concloure els meus estudis a la UOC, vull deixar l'empremta del meu agraïment a tots aquells que durant el llarg període de temps m'han acompanyat, animat i ajudat a continuar.

Per l'Emi.

Resum

L'objectiu que es persegueix en aquest treball és el d'aprofundir en el món de les tecnologies emprades en la compressió d'imatges.

S'introdueixen els conceptes més elementals per així donar un repàs als diferents mètodes tecnològics que s'utilitzen per la compressió, mirant en detall un dels estàndards més utilitzats en l'actualitat, el JPEG. Aquest és un bon punt de partida perquè permet apreciar amb claredat l'estat de l'art de la compressió.

També es tracta l'estàndard JPEG2000, per ser un referent important d'aquest treball en el seu desenvolupament. En la part de programari es fa ús de les transformades wavelet que utilitza el JPEG2000, en particular la que es coneix com a irreversible CDF 9/7. La part de codificació de l'entropia es la que es presenta com a disseny original.

Per al seu desenvolupament s'utilitza el compilador de ***Windows Visual C++*** perquè permet un entorn més usable, on s'utilitzen metàfores que suggereixen les diferents funcionalitats de cada funció. L'objectiu és disposar d'un programari on les seves opcions siguin per si mateixes auto-explicatives.

Les prestacions que permet el programari són: llegir un fitxer BMP o JPEG, i presentar-ho en pantalla, fer la conversió a wavelet amb la possibilitat de poder veure-ho a partir de les seves sub-bandes de freqüència també anomenades rajoles, o com a imatge restaurada. salvar i recuperar el fitxer wavelet.

Hi ha una part addicional orientada a l'anàlisi de la incidència de la quantificació en el resultat final, i la incidència en la imatge en relació a la quantitat de dades generades. Per dur a terme aquesta anàlisi s'utilitza un recurs que mostra un histograma de la presència dels diferents valors que formen la imatge transformada. Dins d'aquest apartat es fa una valoració de com l'entropia incideix en la compressió de la imatge.

Com a conclusió es fa un resum comparatiu entre els resultats obtinguts amb aquest codificador wavelet envers JPEG, contrastant amb un fitxer BMP. A partir d'una mateixa imatge de referència amb el contingut de dos fitxers de dimensions similars, un JPEG i l'altre wavelet. Per aquest comparatiu s'utilitza un programari dissenyat específicament per dur aquesta tasca.

També introdueix com a línies de futur, una primera valoració orientada a la captura d'imatges en flux continu i les cerques de canvis en una seqüència gran d'imatges.

Índex de continguts

1. Introducció

1.1. Justificació del TFC i context

A dia d'avui existeix una forta dependència dels entorns virtuals i multimèdia en la nostra vida quotidiana. Poder gaudir d'un accés àgil amb percepció d'immediatesa dels continguts, comporta la utilització de tecnologies eficients que permeten comprimir grans quantitats d'informació a formats prou petits com per ser emmagatzemats i transportats amb eficiència.

L'abaratiment continuat dels dispositius multimèdia obre les portes a l'ús d'imatges cada cop més grans. Aquest fet té repercussions a diferents sectors de l'àmbit social i professional. Els televisors cada vegada amb majors formats, també tenen més punts d'imatge per polzada, les empreses de seguretat van migrant cap a càmeres megapixels, els continguts WEB son molt més elaborats i vistosos, etc...

L'augment de contingut multimèdia d'alta resolució planteja diversos reptes importants, l'emmagatzemat, el transport i velocitat de processat. Una correcta solució als desafiaments plantejats, aporta bona sensació d'usabilitat a l'usuari que acaba tenint una falsa percepció d'immediatesa com a resposta a les seves accions.

En l'actualitat, la compressió d'una part molt important dels continguts multimèdia es basa en la transformada del cosinus, el que es coneix com a DCT. Aquesta tècnica és utilitzada per la majoria d'algorismes de compressió amb múltiples variants.

Una alternativa interessant a explorar es la transformada wavelet (DWT), per al fet de presentar unes propietats prou atraients com per valorar-ne el seu estudi. L'objectiu que es persegueix per poder comprimir una imatge es descompondre la informació que conté de forma tal que sigui fàcil d'eliminar-ne aquella que es redundant, i també la que per la seva naturalesa l'ull no la pot percebre.

En el cas del DCT els punts d'imatge es disposen dins de blocs de 4x4 o 8x8 elements amb l'objectiu de fer un filtrat bi-direccional, fent el traspàs del domini temporal al freqüencial, i així poder separar els components de freqüència de forma controlada. Amb la DWT també es persegueix la fita de separar els diferents components de freqüència però sense canviar del domini temporal al freqüencial.

Amb independència de la tecnologia emprada per eliminar redundància i components de freqüència negligibles, en tots els cassos hi ha una part de codificació del resultat del filtratge. Aquesta és potser la que pren rellevància en el resultat obtingut. Per al que fa a la codificació de l'entropia, existeixen diferents tècniques per fer-ho, les més conegudes son: la codificació aritmètica i la basada amb les taules d'en Huffman.

La codificació és la part del treball on pot assolir-se una millora en el resultat final de la compressió d'imatge, prenent com a referència tecnologies emprades en altres estàndards i aplicant solucions basades en l'observació les propietats de la DWT.

1.2. Objectius del TFC

Per aquest treball es defineixen els següents objectius:

1- Disseny de l'algorisme de compressió d'imatge.
   L'algorisme ha de contemplar els mecanismes de transformació, quantificació i codificació que han de funcionar de forma independent a altres parts del processat de la imatge.
2- Disseny de l'algorisme de control de quantificació.
   A partir de la primera transformació, i prenent un valor extern, ha de poder calcular el percentatge de quantificació a realitzar a cada sub-banda.
3- Disseny de l'algorisme de descompressió d'imatge.
   Amb la informació continguda dins d'un fitxer, ha de poder produir-se el procés invers de codificació, de-quantificació i síntesi de la imatge. En aquest procés pot determinar-se el resultat final en termes de qualitat, aquests conjunts de funcions inverses no milloren ni degraden el contingut de les dades que processen.
4- Mínim temps d'execució.
   Minimitzar el temps d'execució. És potser una de les parts més importants del disseny. Cal valorar els tipus de variables a utilitzar fent una previsió de possible migració cap a dispositius de lògica programable.
5- Contemplar una evolució cap a vídeos o seqüència d'imatges amb redundància temporal.
   En el disseny de les capçaleres de la imatge ja codificada, és convenient preveure els mecanismes que permeten distingir entre els diferents tipus de continguts, imatge sencera, imatge predictiva o altres.
6- Preveure una evolució a un sistema de cerca de diferències entre imatges d'una mateixa font.

Si en un futur vol contemplar-se la possibilitat de poder fer cerques intensives de regions d'imatge on s'han produït canvis, és convenient que des del disseny inicial es disposi la informació de forma convenient.

1.3. Enfocament i mètode a seguir

Per la naturalesa del projecta, el mètode a seguir és el disseny iteratiu i incremental, tal com es mostra en el diagrama 1.

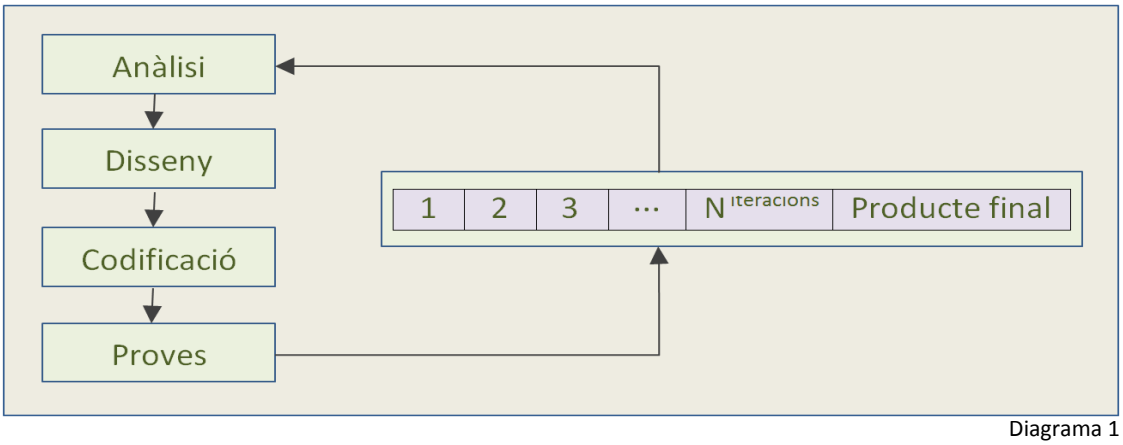


Diagrama 1

1.4. Planificació del projecte

El projecte es descompost en les tasques següents:

1. Anàlisi prèvia del projecte, recerca i documentació sobre algorismes de compressió. Fer una valoració dels diferents treballs existents amb l'objectiu de dur a terme una tria que s'ajusti als propòsits del treball.
2. Documentar el projecte. Aquesta tasca ha de realitzar-se durant tota la seva durada, en forma de recull de fragments parcials que han d'integrar-se en el tram final.
3. Disseny del diagrama de blocs. A partir de la informació recopilada, crear el diagrama de blocs on es defineixen totes les classes que en formen part.
4. Creació d'un esquelet de l'aplicació. Ha de distingir tres parts: iteració amb l'usuari, funcions auxiliars per donar el suport necessària per poder accedir a les funcions del compressor i des-compressor i la part de compressió i descompressió objecte del treball.
5. Construir els mòduls a partir de la documentació generada en el diagrama de blocs.
6. Proves i refinaments sobre els resultats obtinguts.
7. Redacció de la memòria del projecte.
8. Confecció del PowerPoint per dur a terme la presentació.

La seva distribució temporal es la que presenta el diagrama 2:

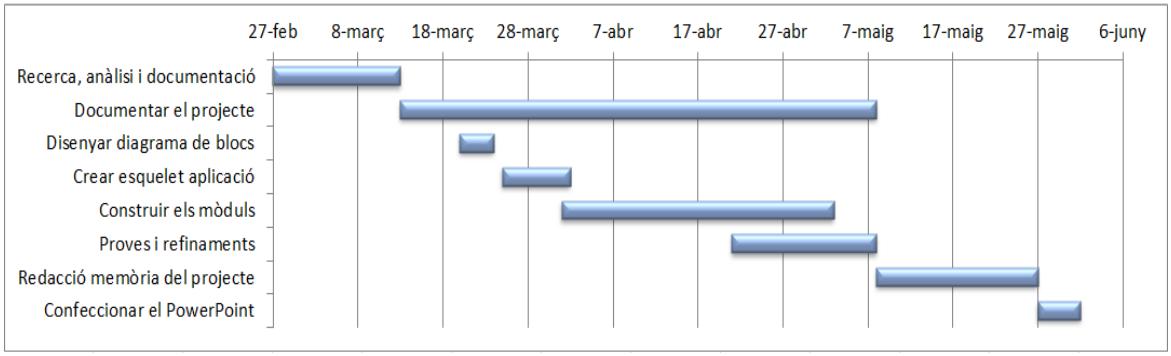


Diagrama 2

1.5. Producte obtingut

Com a elements distingibles que donen singularitat al TFC cal destacar els següents:

- Selecció d'una imatge original en format BMP o JPEG.

- Codificació d'una imatge emprant les transformades wavelets i un codificador inspirat en les taules d'en Huffman fen la incorporació de símbols especials per al tractament dels zeros.

- Editor dels valors de quantificació per als diferents blocs resultants del procés de filtrat per medi de la transformada DWT.

- Representació gràfica de l'entropia de la imatge sencera o d'un bloc en particular de la imatge ja transformada.

- Indicació numèrica de l'entropia, l'error quadràtic mig i la relació senyal soroll de la imatge respecte l'original.

- Estimador de valors de quantificació per l'assoliment del volum d'informació comprimida requerida.

- Descodificador per la síntesi d'una imatge a partir d'informació comprimida.

- Utilitat independent per la mesura de la relació senyal/soroll, PSNR.

1.6. Recursos per dur a terme el projecte

Tot el treball s'ha realitzat amb un ordinador personal que disposa d'un sistema operatiu Windows XP i un compilador de ***C++ Visual C6.0***.

També han estat necessàries per aquest treball imatges en format BMP i JPEG.

2. Antecedents

2.1. Tècniques per a compressió d'imatges

Per poder abordar amb solidesa els conceptes que es presenten a continuació cal primer fer un repàs al component més elemental d'una imatge, que a partir d'ara anomenarem el pixel.

Una imatge es formada per la reunió d'un conjunt de píxels, tal com pot apreciar-se en la figura 1.

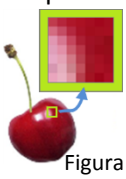


Figura 1

El punt d'imatge o pixel pot ser de dos tipus, monocrom o color. Per la naturalesa del treball que es presenta, únicament es tenen en consideració les propietats del pixel de color.

Cada pixel es composa de tres colors primaris, vermell, verd i blau, conegut com (RGB). A partir d'aquest tres colors es formen tots els possibles, des de negre al blanc més intens. L'anàlisi de la composició de la llum queda fora de l'abast d'aquest treball, per això es centra únicament en les propietats que interessen.

Tal com ha quedat esmentat, un pixel és la representació dels tres color primaris però també pot representar-se per medi de combinacions lineals que els relacioni. Aquesta propietat és per si mateixa una primera aproximació en el que pot considerar-se la compressió d'imatge.

L'estàndard CCIR 601, incorpora una conversió dels components R, G, B a Y, Cb i Cr. Aquesta conversió extrau la luminància Y, i dues components de crominància Cb i Cr. Inicialment contemplava dues possibles conversions (4:4:4 ó 4:2:2), però ha estat estesa a altres. El (4:2:0) és un d'ells, potser el més utilitzat. Aquesta descripció vol dir, (4:4:4) hi ha quatre components Y, quatre de Cb i quatre de Cr. (4:2:2) son 4Y, 2Cb i 2Cr. (4:2:0) aquest cas involucra a dues línies de imatge i són 4Y, 1Cb i 1Cr on els components queden alternats entre files i columnes.

A la figura 2, pot apreciar-se com es distribueix la luminància i la crominància per als diferents tipus de conversions.

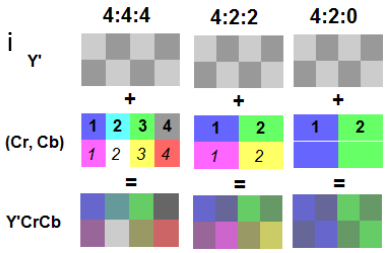


Figura 2

Imatge extreta i adaptada de:
http://en.wikipedia.org/wiki/Chroma_subsampling

La transformació es realitza per medi de les funcions:

$$Y = 0.257R + 0.505G + 0.098B + 16$$

$$Cb = -0.148R - 0.291G + 0.439B + 128$$

$$Cr = 0.439R - 0.368G - 0.071B + 128$$

Per il·lustrar la transformació esmentada, en una imatge de 512x512 hi ha 262.144 pixels o punts d'imatge. Tenint en compte que cada pixel és format per tres components de color, això vol dir que es requereixen 786.432 valors independents.

Amb una transformació de R,G,B a Y,Cb,Cr (4:2:0) es redueix a 327.680 valors que són menys de la meitat de l'original. Tal com pot apreciar-se sense aplicar cap algorisme de compressió d'imatge ja existeix un primer estadi de compressió respecte al R,G,B.

Aquesta transformació és possible per al fet que l'ull humà disposa d'elements sensors diferents per captar la imatge i els colors. També es molt més sensible als canvis ràpids en la imatge que en el color, es a dir la resposta en freqüència del color pot ser menor.

Compressió d'imatge. Aquest procés pot ser amb pèrdua o sense, les tecnologies de compressió sense pèrdua, són aquelles que permeten una reconstrucció de la imatge sense cap diferència amb l'original. Aquest tipus de compressió és poc agressiva i té utilitat en entorns professionals on la qualitat de la imatge reconstruïda prima per sobre del volum d'informació requerida per al seu transport o emmagatzemat.

Un dels tipus de compressió sense pèrdua més popular és el PNG que utilitza l'algorisme LZ77 i taules d'Huffman; un altre és la predictiva, es tracta de fer la predicció que el píxel del costat té el mateix valor que la mostra i es codifica l'error. També utilitza taules d'Huffman per codificar els valors.

Hi ha altres mètodes de compressió sense pèrdua, un exemple pot ser el JPEG2000 sempre que es configuri per funcionar d'aquest mode. Aquest estàndard ofereix la possibilitat de treballar de diferents maneres i dins del seu ventall d'opcions hi ha la de sense pèrdua.

El tipus de compressió que es presenta en aquest treball es amb pèrdua. El criteri general és eliminar tot allò que no ha de ser perceptible per l'ull humà. Com es lògic suposar, un cop es destrueix un fragment d'informació, en el procés de reconstrucció pot fer-se una aproximació al valor original, però mai serà idèntic.

Un exemple de compressió amb pèrdua és la compressió fractal. Aquest tipus de compressió es basa amb la premissa que dins d'una imatge hi ha parts que es repeteixen i per tant, re-aprofitables. El problema principal que presenta és l'esforç que requereix per la compressió, la descompressió però és molt ràpida. Per relacions 1:50 el resultat és equivalent al JPEG, per relacions més altes la millora és considerable.

2.2. Transformada del cosinus DCT

La transformada del cosinus, coneguda com a DCT, recull la informació de 64 components d'imatge ordenats en forma de matriu quadrada de vuit per vuit.

Al realitzar la transformació es produeix un canvi de domini, passant del temporal al freqüencial. Aquesta transformació ordena els diferents components en funció de la freqüència a la que pertanyen, on les més baixes queden a prop de l'origen i, en conseqüència, les més altes se n'allunyen.

La funció de càlcul de la transformada bidireccional :

$$\boldsymbol{G_{u,v}} = \sum_{\boldsymbol{x=0}}^{\boldsymbol{7}} \sum_{\boldsymbol{y=0}}^{\boldsymbol{7}} \propto (\boldsymbol{u}) \propto (\boldsymbol{v}) \boldsymbol{g_{x,y}} \boldsymbol{cos}\left[\frac{\boldsymbol{\pi}}{\boldsymbol{8}}\left(\boldsymbol{x}+\frac{\boldsymbol{1}}{\boldsymbol{2}}\right)\boldsymbol{u}\right] \boldsymbol{cos}\left[\frac{\boldsymbol{\pi}}{\boldsymbol{8}}\left(\boldsymbol{y}+\frac{\boldsymbol{1}}{\boldsymbol{2}}\right)\boldsymbol{v}\right]$$

On:

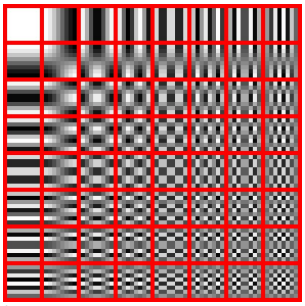
$\boldsymbol{u}$ es la freqüència espacial horitzontal per valors enters $0 \leq \boldsymbol{u} < 8$.

$\boldsymbol{v}$ es la freqüència espacial horitzontal per valors enters $0 \leq \boldsymbol{v} < 8$.

$$\propto (\boldsymbol{u}) = \begin{cases} \sqrt{\frac{1}{8}}, si\ \boldsymbol{u} = 0 \\ \sqrt{\frac{2}{8}}, si\ \boldsymbol{u} \neq 0 \end{cases}$$

$\boldsymbol{g_{x,y}}$ es el valor del pixel a les coordenades (x, y).

$\boldsymbol{G_{u,v}}$ es el coeficient DCT a les coordenades (u, v).



Imatge extreta de: http://en.wikipedia.org/wiki/File:Dctjpeg.png Figura 3

En la figura 3, pot apreciar-se la forma creixent en la que es distribueixen els components de freqüència.

Com la resposta de l'ull humà és poc sensible a les freqüències altes, es possible eliminar aquelles que són més febles per medi que un procés de quantificació.

Aquesta distribució afavoreix el poder fer un recorregut des de l'origen, i aturar-se en el punt on deixen d'existir informació, col·locant un símbol de final de bloc.

2.2.1. DCT en el JPEG, com s'utilitza

Un dels usos més populars de la DCT és en les imatges JPEG. Aquest sistema de compressió es basa en els tres passos representats en la figura 4.

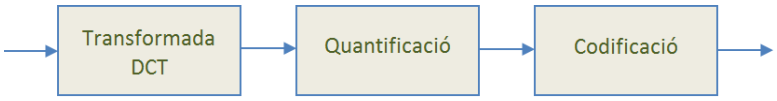


Figura 4

El concepte de transformada ja ha estat vist en l'apartat anterior, la quantificació és la part del procés que introdueix la compressió pròpiament dita. En funció del resultat esperat, es quantifiquen tots els elements del bloc amb un valor independent per cada cel·la. La disposició dels components de quantificació es distribueixen de forma tal, que els components de freqüències altes són quantificats de forma més agressiva que els corresponents a les baixes.

En la figura 5 pot apreciar-se la distribució dels valors.

Per cada nivell de compressió d'una imatge, ha de calcular-se una nova taula

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Imatge extreta i adaptada de: http://en.wikipedia.org/wiki/File:JPEG_ZigZag.svg Figura 5

La darrera part del procés correspon a la codificació. L'estàndard JPEG admet dos tipus de codificació l'aritmètic i la codificació de longitud variable VLC.

La codificació aritmètica pràcticament no s'utilitza en el JPEG, per això la més emprada és la de codi de longitud variable. Aquest tipus de codificació té l'avantatge de que per la descodificació és molt ràpid, perquè permet l'ús de taules tipus Huffman.

Per dur a terme aquesta codificació s'utilitza una col·lecció de símbols que són assignats a la longitud de la dada a transmetre, existeix un símbol especial per indicar el final del bloc EOB, i d'altres per al control de la seqüència de bits.

Existeix un tractament especial per la cel·la 0,0 que s'anomena DC, i que acostuma a tenir un valor molt alt. Per aquesta cel·la el que es codifica és la diferència amb la del bloc anterior.

Per la descodificació es realitza el procés invers, tal com es presenta a la figura 6.

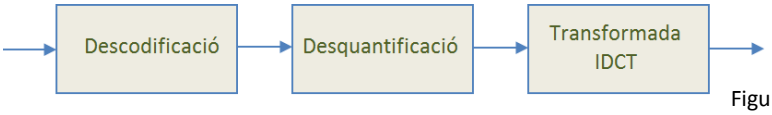


Figura 6

2.3. Transformada Wavelet DWT

Existeixen diferents mètodes per la compressió d'imatges per medi de transformades wavelet, la més coneguda és la JPEG2000. Aquest mètode de compressió és molt ampli i ofereix diferents possibilitats, el que permet ajustar-se al màxim a les necessitats de cada aplicació.

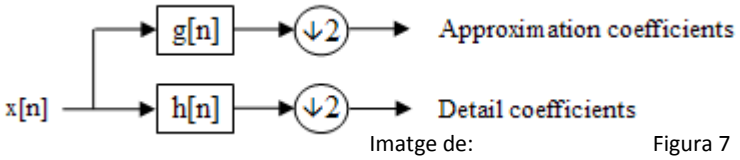
El JPEG2000 permet tant una transformació reversible, es a dir sense pèrdua la CDF 5/3, com també una compressió irreversible amb pèrdua, la CDF 9/7. La diferencia que hi ha entre aquestes dues transformades rau en que el CDF 9/7 treballa amb números de coma flotant, això introdueix soroll quantitatiu. La magnitud del soroll queda determinat per la precisió del descodificador.

L'equació de càlcul de la transformada discreta wavelet SWT es:

$$y_{low} = \sum_{k=-\infty}^{\infty} x[k]g[2n-k]$$

$$y_{high} = \sum_{k=-\infty}^{\infty} x[k]h[2n-k]$$

On g correspon al filtratge passa baix i h al filtratge passa alt.



Imatge de: http://en.wikipedia.org/wiki/Discrete_wavelet_transform Figura 7

El procés de filtratge produeix dos grups de valors on cada grup té la meitat de resolució de les dades originals. El resultat dels dos filtratges junts passa a contenir el mateix nombre de mostres que l'original, disposades en la part baixa de les freqüències i en la part alta. A la figura 7 pot apreciar-se com queden organitzades les mostres.

La naturalesa bidimensional d'una imatge fa que aquesta operació hagi de fer-se tant en sentit horitzontal com en el vertical. Un cop s'ha realitzat una transformada bidireccional s'obté un nivell compost de quatre regions de filtratge que alguns autors anomenen rajoles.
La figura 8 mostra com queden distribuïts aquestes regions. Aquesta operació pot anar-se repetint dins del bloc LL i establir diferents nivells de filtrat.

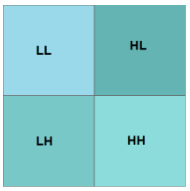


Figura 8

Cada nivell així tindrà els blocs HL, LH, i HH i l'últim disposarà del quart LL.
En les figures 9 i 10 queda il·lustrat aquest raonament.

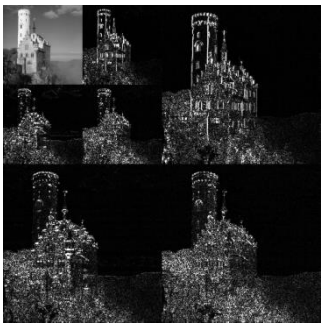


Figura 10

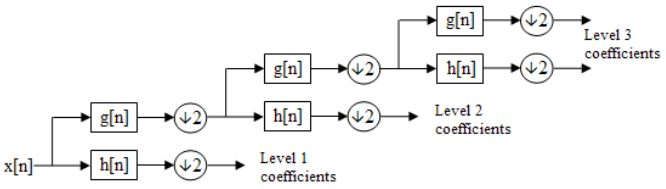


Figura 9

Imatges figura 8 i 9 extretes de:
http://en.wikipedia.org/wiki/Discrete_wavelet_transfor

2.3.1. DWT al JPEG200, com s'utilitza

L'ús més conegut de les transformades wavelets és el de JPEG2000, aquest estàndard inicialment va estar orientat a substituir a la compressió per DCT. Això no va tenir l'èxit que esperaven, la qual cosa es fa evident en el fet que gairebé no hi ha cap navegador que el suporti. Tal com pot apreciar-se en la figura 11, el seu diagrama de blocs és idèntic al de compressió per DCT.

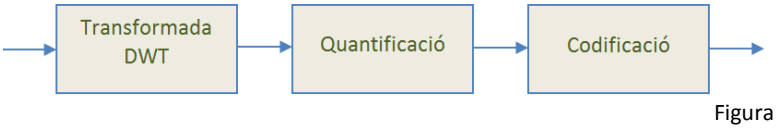


Figura 11

Un cop obtinguts els blocs transformats el JPEG2000 permet quantificar de forma independent regions determinades de la imatge de tal manera que el que no presenta un interès important perd resolució afavorint una zona destacada. En qualsevol cas cada component d'un bloc determinat pot tenir un tractament diferent dels altres components del mateix bloc i dels blocs veïns.

Per la codificació de la informació s'utilitza un sistema anomenat EBCOT (*Embedded Block Coding with Optimal Truncation*) que pot traduir-se com a bloc incrustat amb truncament òptim. Es tracta de disposar els bits en capes, començant pel signe i seguint per al de més pes fins el de menys pes, a aquestes capes se les anomena capes de mode. En aquest procés es generen una col·lecció de símbols que posteriorment es codificaran amb un codificador aritmètic.

Aquest mètode permet fer l'enviament de la imatge de forma progressiva cap el receptor. Des del moment que rep un primer grup de capes ja por generar la imatge i a mida que van arribant les altres, fer-la tornar més nítida.

El procés de restauració és repetir en forma inversa l'anterior tal com es mostra a la figura 12.

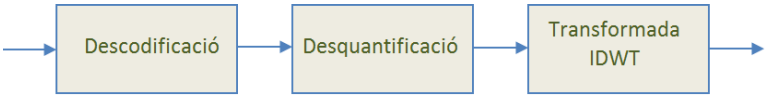


Figura 12

3. Descripció funcional

El programari està dividit en tres parts diferenciades, l'esquelet de l'aplicació, les utilitats que actuen com a recursos per l'anàlisi i manipulació de la imatge transformada i la classe especialitzada en la transformació directa i inversa de la imatge.

3.1. Diagrama de blocs

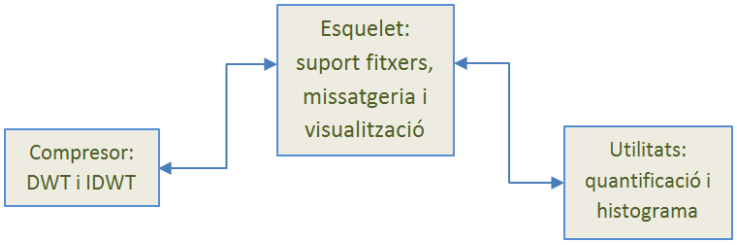


Figura 13

En al figura 13, es distingeixen perfectament diferenciades les tres parts que composen el projecte. L'esquelet aporta el lligam entre el mon exterior i la funcionalitat operativa, es a dir les entrades en forma d'accés al sistema de fitxers, el sistema de missatgeria interna entre recursos i la sortida per medi del monitor de l'ordinador on s'executa el programari.

Una tasca important de l'esquelet és el traspàs de dades entre les diferents classes especialitzades en les tasques de: comprimir/descomprimir, generar els valors de quantificació per la compressió d'imatges i l'histograma que aporta informació sobre el resultat de la compressió.

Les utilitats són la base on s'ha recolzat una part molt important d'aquest treball. Han estat l'element clau per poder decidir el mètode idoni per la codificació de les dades un cop transformades. L'observació directa de com es distribuïen els diferents valors numèrics que formen la imatge un cop transformada en funció del grau de quantificació ha permès arribar al resultat esperat.

El compressor queda dividit en tres grans blocs: compressió, descompressió i recursos de suport. Dins de l'apartat de la compressió es defineixen tres parts diferenciades, el càlcul de la transformada directa, la quantificació de cada un dels blocs de filtrat i la codificació. Per la descompressió s'identifiquen els mateixos tres passos però fets en sentit invers. Els recursos de suport generen informació per les utilitats de quantificació i l'histograma.

Aquesta separació en els diferents grups de processos ha de permetre poder transportar el programari a diferents plataformes. Tot el codi és escrit en C++ i les úniques limitacions quedarien focalitzades en l'esquelet propi del programari, la classe sencera per la compressió pot compilar-se directament. Les classes de quantificació i l'histograma, pel fet d'utilitzar una finestra que pertany a l'esquelet caldria fer la reconversió de la part gràfica.

3.1. Diagrama de classes

El diagrama de classes (figura 14) il·lustra la forma en què es relacionen les diferents classes que formen el programari.

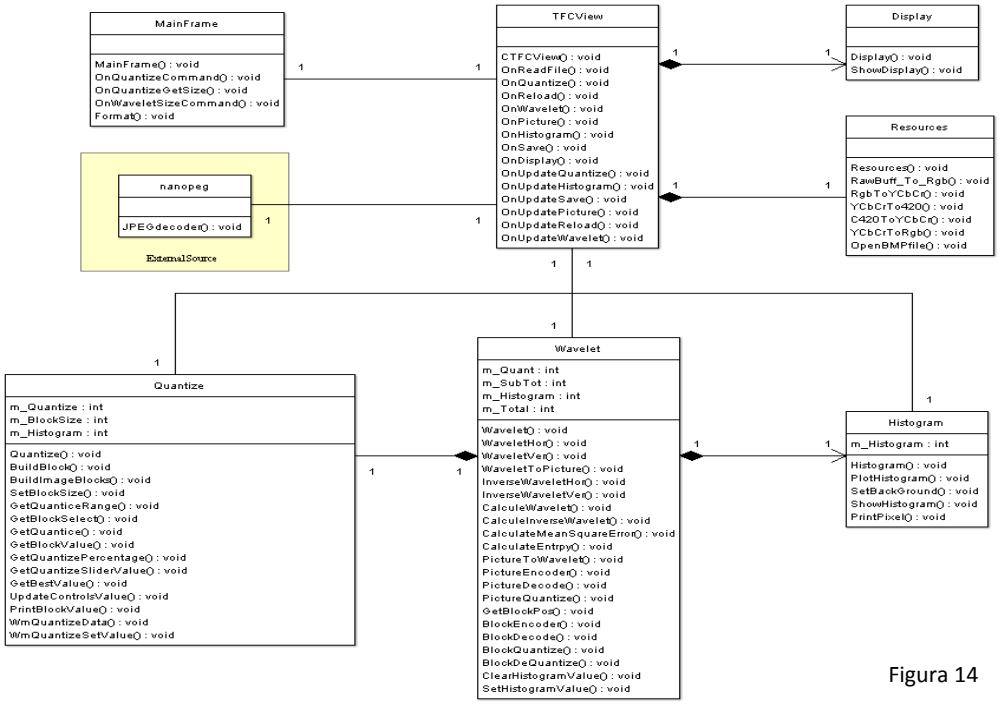


Figura 14

L'esquelet és format per les classes ***MainFrame***, ***TFCView***, ***Display***, ***Resources*** i una cinquena classe procedent d'una font externa, ***nanopeg***. Aquestes classes no s'analitzaran en detall perquè la seva finalitat és la de servir com a contenidor de les classes objecte del treball. Es presenta una descripció general de la part més rellevant.

La forma d'operar dins l'estructura de l'esquelet del programari és l'enviament de missatges per medi de la funció `PostMessage`. Aquest mecanisme inherent a la pròpia estructura, permet mantenir independència entre l'esquelet del programari, les classes de suport i la classe dels wavelets.

Per al que fa al sistema de fitxers també s'ha procurat deslligar-lo dels recursos propis del contenidor amb la finalitat de simplificar un possible transport a una altra plataforma, així per exemple la conversió dels fitxers BMP a les seves components R,G,B ha estat expressament dissenyada per aquest treball i per la lectura de fitxers JPEG s'ha utilitzat un codi lliure extern.

Classes de suport: ***Quantize***, eina orientada a generar de forma manual o automàtica els valors necessaris per poder quantificar la imatge. Aquesta classe aporta un sistema gràfic interactiu on es presenta el resultat del càlcul de les DWT per cada bloc de filtrat. Per medi de l'esquelet del programari facilita al compressor els valors de quantificació de cada regió de filtrat amb els components que la composen i recull els resultats parcials de comprimir cada bloc.

Histogram, eina gràfica especialitzada en la representació de la freqüència a la que es repeteix cada valor present dins el conjunt de la imatge o en un bloc determinat. També aporta informació de l'entropia, l'error quadràtic mitjà i la relació senyal/soroll. Aquesta classe pot funcionar per si sola o conjuntament amb ***Quantize***.

Les classes ***Quantize*** i ***Histogram***, no disposen de funcions públiques, són accessibles únicament per medi dels seus atributs que és per on recullen i dispensen la informació. En el cas d'***Histogram*** només recull informació per la representació del gràfic i les dades. La connexió amb l'exterior es du a terme per medi de la missatgeria inherent a l'esquelet del programari.

Classe ***Wavelet***, és la raó que justifica el treball, i és accessible per medi de les funcions publiques següents:

```
// transforma una imatge en format (4:2:0) a mosaics wavelets
// el contenidor d'imatge es dins l'estructura pYCbCr, el resultat de la transformació
// torna a quedar a dins la mateixa estructura. No calen buffers d'entrada i sortida
void PictureToWavelet(
                      pYCbCr Container,     // contenidor de components Y, Cb i Cr
                      WORD   Horizontal,    // mida de la imatge en horitzontal
                      WORD   Vertical       // mida de la imatge en vertical
                      );

// codifica una imatge formada per rajoles wavelet per poder ser posat en un fitxer
// retorna el nombre de bytes que ocupa la imatge codificada
DWORD PictureEncoder(
                      pYCbCr BuffIn,        // contenidor de components Y, Cb i Cr
                      BYTE  *BuffOut,       // contenidor de la imatge codificada
                      WORD   Horizontal,    // mida de la imatge en horitzontal
                      WORD   Vertical       // mida de la imatge en vertical
                      );

// converteix una imatge formada per rajoles wavelet (4:2:0) a imatge sencera
void WaveletToPicture(
                      pYCbCr Container,     // contenidor de components Y, Cb i Cr
                      WORD   Horizontal,    // mida de la imatge en horitzontal
                      WORD   Vertical       // mida de la imatge en vertical
                      );

// descodifica el contingut d'un fitxer i ho converteix a imatges en rajoles wavelet
// si el procés de descodificació ha tingut èxit, retorna 1
int PictureDecode(
                      pYCbCr Buffer,        // contenidor de components Y, Cb i Cr
                      BYTE  *MemIn,         // contenidor de punters de components
                      DWORD  size,          // contenidor de punters de components
                      WORD  *Horizontal,    // retorna la mida d'imatge en horitzontal
                      WORD  *Vertical       // retorna la mida d'imatge en vertical
                      );

// quantifica una imatge formada per rajoles wavelet.
// També genera l'efecte de bloc il·luminat.
// Aquesta funció auxiliar, té utilitat únicament com a suport de les classes Quantize
// i Histogram.
void PictureQuantize(
                      pYCbCr Buffer,        // contenidor de components Y, Cb i Cr
                      BYTE   BlockSel,      // selector del component del bloc Y:Cb:Cr
                      short  Index,         // selector del bloc que cal il·luminar
                      WORD   Horizontal,    // mida de la imatge en horitzontal
                      WORD   Vertical       // mida de la imatge en vertical
                      );

// retorna el càlcul de l'error quadràtic mig.
// Aquesta funció auxiliar, té utilitat únicament com a suport de la classe Histogram.
float CalculateMeanSquareError(
                      pYCbCr BuffIn,        // contenidor de la imatge original
                      pYCbCr BuffOut,       // contenidor de la imatge transformada
                      WORD   Horizontal,    // mida de la imatge en horitzontal
                      WORD   Vertical       // mida de la imatge en vertical
                      );

// esborra els atributs públics i privats de la classe Wavelet relacionats amb Histogram.
// Aquesta funció auxiliar, té utilitat únicament com a suport de la classe Histogram.
void ClearHistogramValue();

// retorna el càlcul de l'entropia de la imatge.
// Aquesta funció auxiliar, té utilitat únicament com a suport de la classe Histogram.
float CalculateEntrpy();
```

3.2. Mètode de codificació

La codificació ha estat realitzada en C++, la qual cosa permet transportar el programari a altres plataformes. La generació del prototip de les funcions en la primera fase de codificació de les classes s'ha realitzat de forma automàtica per medi del programari de codi obert **ArgoUML**. Un cop iniciada la codificació les declaracions s'han fet directament a cada classe.

Amb l'esquelet funcional, la classe següent ha estat la de ***Display***, necessària per poder presentar a la pantalla el contingut d'un buffer en R,G,B. Aquesta classe és l'únic recurs emprat per pintar tant les imatges com els elements gràfics. Això es així per tal de poder seguir el criteri de fer el codi transportable.

A partir d'un suport capaç d'accedir al contingut d'un fitxer, ha estat desenvolupada la classe ***Resources*** on hi ha implementades les funcions de conversió de BMP a R,G,B també de R,G,B a (4:2:0) i viceversa. Amb els recursos disponibles en aquesta classe és possible convertir un fitxer BMP a R,G,B i presentar-ho a la pantalla. Això inclou passar per totes les conversions intermèdies sense que el contingut de la imatge pateixi una degradació apreciable.

Un cop ha estat possible obrir un fitxer BMP i convertir els components R,G,B a components de luminància i crominància en format (4:2:0), la classe següent ha estat la ***Wavelet***. La forma que ha estat desenvolupada aquesta classe s'ha distribuït en sis parts:

1- Codificació de la transformada directa
2- Codificació de la transformada inversa
3- Funcions per donar suport a ***Quantize***
4- Funcions per donar suport a ***Histogram***
5- Codificació d'una imatge transformada per ser continguda dins d'un fitxer
6- Descodificació del contingut d'un fitxer i transformar-ho a rajoles wavelet

Al assolir els punts 1 i 2 en la codificació de la classe ***Wavelet*** ha estat dissenyada la classe ***Quantize***. Aquesta classe representa les rajoles del wavelet i en l'estadi inicial ja permet quantificar de forma manual els blocs resultant de la transformació wavelet. La part de càlcul automàtic dels valors de quantificació ha estat un d'els darrers en poder ser escomesa.

A partir d'els recursos que permeten fer la transformada directa i inversa a una imatge i disposant d'una utilitat per poder-la quantificar, es dissenya la classe ***Histogram***. Aquesta utilitat combinada amb ***Quantize***, permet apreciar de forma dinàmica com es distribueixen els diferents valors d'una imatge un cop transformada a wavelets, el contingut dels seus blocs, els components i la incidència de la quantificació sobre el seu conjunt o per separat.

3.3. Transformada discreta wavelet

Per aquest treball s'ha codificat la transformada discreta wavelet que ve documentada en l'estàndard JPEG200 i es coneix con a irreversible CDF 9/7. La funció codificada està descrita en el llibre: *JPEG2000 ISBN 0-7923-7519-X capítol 10 pagina 433*. Hi ha altres treballs on és presenten transformades amb coeficients per al filtrat equivalents.

L'objecte d'aquest treball no és el de fer noves aportacions en el càlcul de la transformada discreta wavelet encara que hi ha codificada una segona transformada amb el propòsit de fer un estudi comparatiu.

3.4. Codificació de la imatge transformada

3.4.1. Criteri d'avaluació

A l'hora de valorar el mètode per codificar la imatge transformada a rajoles wavelet, s'han explorat diferents alternatives. En primer lloc al observar la freqüència d'aparició dels valors en l'histograma, on pot apreciar-se que el valor zero es distingeix amb esqueix de qualsevol altre tal com pot apreciar-se en la figura 15.

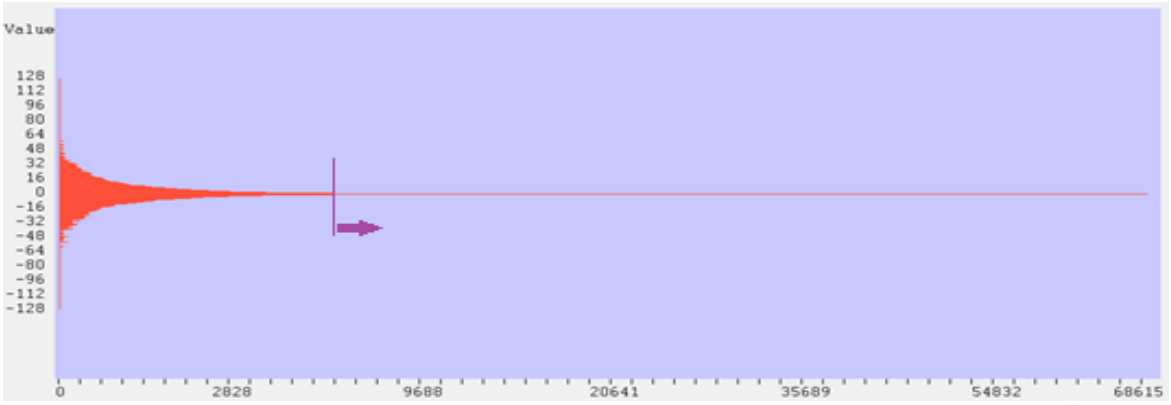


Figura 15

Com a primera valoració hi ha la codificació aritmètica, sistema basat en l'interval definit entre els símbols amb la probabilitat d'aparició més alta i el de probabilitat més baixa. En aquest cas queda clar que el símbol que representa el zero ha de ser el de probabilitat més alta i els que representen els valors 127 i -127 els correspondrà la més baixa.

La segona valoració són les taules d'Huffman, alguns autors ho consideren una extensió de la codificació aritmètica pel fet de tractar símbols ordenats en nombre de bits en funció de la seva probabilitat. Les taules de símbols també tenen l'inconvenient que el nombre de bits necessaris per poder-les codificar creix de forma desmesurada a mesura que s'allunyen dels valors més provables.

Amb això, segueix existint l'inconvenient que el símbol zero ha de codificar-se de forma independent per cada un que aparegui. Encara que es faci ús d'un sol bit per cada zero, segueixen sent molt abundants. El gràfic únicament presenta l'entropia pel que fa a la freqüència d'aparició dels valors però no en la forma que es troben distribuïts.

En la figura 16, hi ha una imatge transformada a rajoles wavelet, es la mateixa imatge que ha generat els valors de l'histograma de la figura 15.

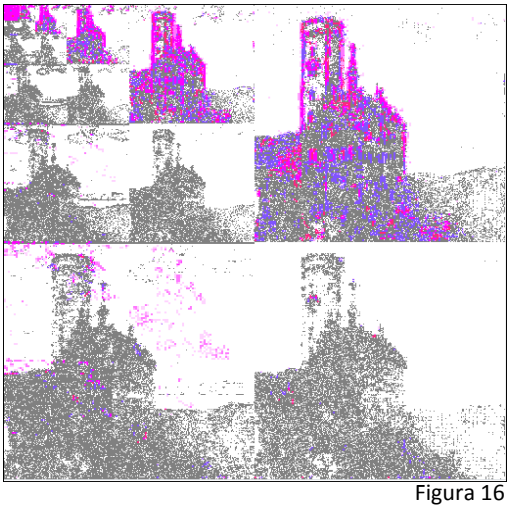


Figura 16

Aquesta imatge ha estat tractada amb el Pseudocodi:

```
binarize(data, size)
begin
    do i=0 to size
        if data(i) <1 and data(i) >-1 then
            data(i)=127
        else
           data(i)=0
        end
    end
end
```

Per als components de crominància, els valors de 0 i 127 han estat intercanviats.

El primer que es fa evident és que els zeros es troben localitzats en regions senceres. Observant l'histograma de la figura 15, pot apreciar-se com a mesura que en els valors que s'allunyen del zero la probabilitat d'aparèixer decreix ràpidament.

3.4.2. Model d'escombrat dels blocs d'imatge

Tal com pot apreciar-se en la figura 16, la distribució dels zeros comprèn zones compactes senceres, en un primer plantejament l'exploració es duia a terme de forma lineal, es a dir, fent escombrat horitzontal per tota la superfície del bloc. Aquesta solució per si mateixa ja és efectiva, però hi ha tot un grup d'imatges on per la seva naturalesa resulta poc eficient.

Si s'amplien els resultats d'imatges diferents és fa evident que una solució interessant es la de fer l'escombrat per blocs. En el treball s'ha implementat la solució de blocs de 4*4 components. En aquesta mida de bloc s'han realitzat diferents proves: escombrat horitzontal, vertical i zig-zag. Existeix una notable diferència en el rendiment entre els escombrats horitzontals i verticals respecte el de zig-zag que és el que finament ha quedat implementat.

La distribució de l'exploració es realitza en la forma que mostra la figura 17, on la presencia de valors dispersos no afecta de forma notable les seqüències de zeros consecutius.

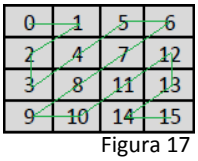


Figura 17

Un cop analitzat el comportament que tenen els valors transformats per medi de DWT, la solució que es planteja és el tractament dels zeros de forma especial per medi d'una taula de símbols on grups de zeros quedin codificats amb símbols de poc pes de bit. Entre els símbols especials hi ha comptadors de zeros contigus.

Una propietat que presenta l'escombrat en zig-zag és que un mateix comptador de zeros si les condicions són favorables, pot recórrer diferents blocs successius dins de la mateixa fila i fins i tot continuar per blocs de les files següents.

3.4.3. Disseny de la taula de símbols

La taula amb la que s'ha treballat es la següent:

Valor	símbol	bits	reducció respecta a un byte
1 zero	**00**	2	0,250
2 zeros	**000**	3	0,187
3 zeros	**0000**	4	0,166
4 zeros	**00000**	5	0,156
5 zeros	**000000**	6	0,150
6 zeros	**0000000**	7	0,145
7 a 35	xxxxx**010**	8	0,142 a 0,028
36 a 65571	xxxxxxxxxxxxxxx**0110**	20	0,065 a 36*10^-6

La part identificada amb **xx..xx** correspon a un comptador de zeros consecutius.

No existeixen símbols per indicar blocs sencers a zero, pel fet que un comptador pot recórrer diferents blocs i també cal observar que el contingut d'un bloc buit emprant el símbol '**xxxxx010**' ocupa únicament vuit bits amb la possibilitat de contenir fins a dos blocs.

Els valors diferents de zero, segueixen una distribució simètrica i la seva probabilitat d'aparèixer pren un patró quadràtic. En la figura 18 s'aprecia clarament com els valors diferents de zero predominants queden compresos en l'interval ±1 al ±7.

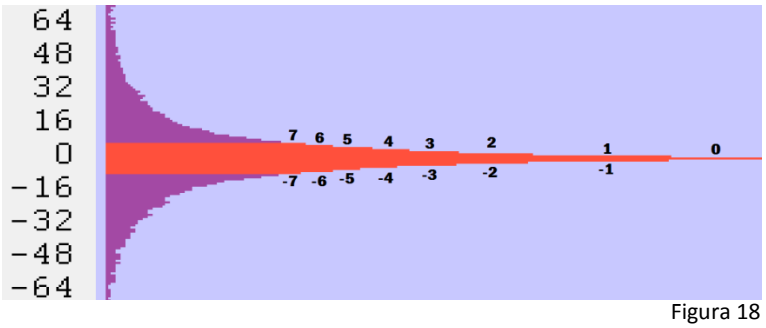


Figura 18

-1 a 1	s**01**	3	0,375
-3 a 3	sx**011**	5	0,625
-7 a 7	sxx**0111**	7	0,875
-15 a 15	sxxx**01111**	9	1,125
-31 a 31	sxxxx**011111**	11	1,375
-63 a 63	sxxxxx**0111111**	13	1,625
-127 a 127	sxxxxxx**01111111**	15	1,875

La taula de símbols comprèn tres parts: s, correspon al signe del valor, x, conte el/s bit/s de menys pes, i la resta de bits és el símbol de la taula pròpiament dita. En el cas dels valors ±1, únicament es codifica el signe. Els símbols representats en color vermell, es corresponen amb les regions morades de la figura 18 i representen aquells valors que en lloc de comprimir expandeixen.

Existeix un símbol especial de control, aquest símbol té una longitud de 64 bits i encapsula tota la informació pertanyent a un bloc d'un component determinat.

control hhhhhhhhhhvvvvvvvvvvvX**011111111**ycbbbbbbbbppppppppppppppppppppp

Descripció:
Amplada del bloc hhhhhhhhhhh 11 bits, 0 a 2047. És l'amplada màxima de bloc.
Alçada del bloc vvvvvvvvvvv 11 bits, 0 a 2047. És l'alçada màxima de bloc.
Futura utilitat X.
Símbol **011111111**.
Indicador de component de luminància y si És zero luminància, ú crominància.
Indicador de component de crominància c si És zero Cb, ú Cr.
Valor de quantificació bbbbbbbb 8 bits, permet valors entre 1 i 255.
Origen de la posició del bloc dins del buffer ppppppppppppppppppppp 14 bits 4194303.

4. Desenvolupament

El desenvolupament del programari s'ha distribuït per diferents fases. L'objectiu principal ha estat el de codificar l'entropia d'una imatge ja transformada en rajoles wavelet de forma que ocupi el mínim espai possible explorant noves alternatives. Per poder dissenyar l'algorisme de codificació primer ha calgut construir les parts corresponents a la transformació DWT, IDWT les utilitats ***Quantize*** i la de l'***Histogram***.

4.1. Recursos per la construcció de programari

Els recursos emprats per la construcció del programari, han sigut un ordinador personal provis d'un compilador de C++ per Windows, el **Microsoft Visual C++**. També s'ha disposat d'una col·lecció d'imatges tant en format BMP com format JPEG.

4.1.1. Interacció humana

El programari s'ha dissenyat de forma tal que pot funcionar directament en qualsevol sistema operatiu Windows, no cal fer instal·lació. Els programaris han estat compilats amb les crides a llibreries dinàmiques integrades. No fa crides a DLL's externes, tampoc fa ús de llibreries gràfiques especialitzades, amb la qual cosa no ha de presentar incompatibilitats a l'hora de funcionar en diferents versions de Windows.

Per a seu funcionament disposa d'un accés a menú en forma de barra on hi ha representades les diferents opcions que permet, tal com mostra la figura 19.

Figura 19

4.2. Operativa de funcionament

4.2.1. Obrir un fitxer

Pot fer-se de quatre formes diferents, arrossegant el fitxer d'imatge fins a sobre la icona del programari  , o un cop el programari ja estigui obert prement de forma simultània les tecles de control+O i també clicant a sobre les icones '*File*' o '*Open*', tal com és veu en la figura 20.

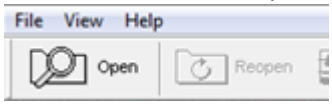


Figura 20

Al obrir el fitxer, la seva imatge substitueix el fons gris i es presenta en el seu format original. Les opcions que permet s'activen i la barra d'estats aporta la primera informació sobre la mida de la imatge original, la mida de la imatge un cop transformada a wavelets sense quantificar i el percentatge que ocupa respecte del fitxer original figura 21.

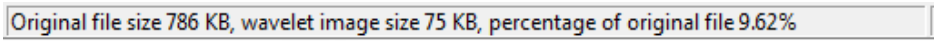


Figura 21

4.2.2. Re-obrir

Torna a obrir l'últim fitxer, al fer-ho també es re-inicialitzen totes les dades de quantificació. Es fa per medi de la icona 

La reobertura té el mateix comportament que una obertura de nou, es a dir es presenta la imatge tal com és en l'original, i a la barra d'estats s'actualitza la informació.

4.2.3. Wavelets

La imatge en forma de wavelets és accessible per medi de la icona 

, que mostra el resultat de la transformació en la forma de blocs dels corresponents filtrats tal com es veu a la figura 22.

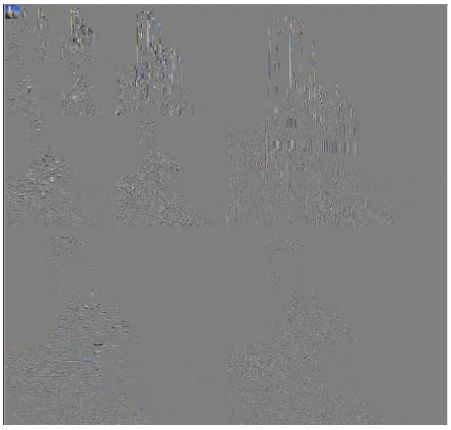
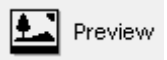


Figura 22

4.2.4. Visualització prèvia

La visualització prèvia és accessible des de la icona 

i és el resultat de transformar la imatge a wavelets, quantificar, de-quantificar i reconvertir a imatge altre cop.

4.2.5. Quantificació

Accessible per medi de la icona 

, al accedir-hi, s'obre una finestra flotant on hi ha representats els diferents blocs de filtrat de la imatge. Aquesta utilitat pot funcionar de dues formes diferents, mode manual o automàtic. En el mode manual és pot triar sobre quins components es vol actuar, la selecció dels components es fa per medi dels CheckBox Y,Cb,Cr. En mode automàtic s'assigna el percentatge respecte la mida de l'original a reduir.

Quantificació manual, permet assignar manualment el valor de quantificació a cada bloc i de forma independent a cada component. Al clicar a sobre d'un bloc, queda identificat i si hi ha oberta l'opció de wavelet, també s'il·lumina. Clicant fora de l'àrea de blocs de filtratge, es desconnecta l'opció. En la figura 23 pot apreciar-se la correspondència entre el bloc seleccionat a ***Quantize*** i la seva representació en la visualització wavelet.

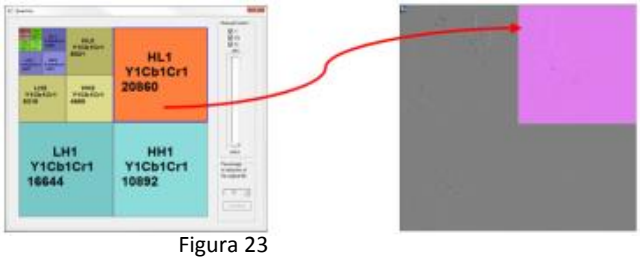


Figura 23

Figura 24


Figura 25

Un cop seleccionat el bloc, depenent del component triat figura 24, al desplaçar l'slider figura 25, cap amunt, augmenta el valor de quantificació i si hi ha oberta la presentació wavelet pot apreciar-se com afecta al bloc seleccionat. El valor generat per l'slider afecta directament i amb el mateix valor als components seleccionats.

Els blocs presenten la següent informació:

Filtrat i nivell HL, LH, HH
Valor de quantificació de cada component
Memòria en bytes que ocupa el bloc
En la figura 26 hi ha descrit el detall de cada valor.

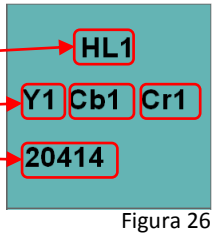


Figura 26

Quantificació automàtica, també és possible quantificar a partir d'un percentatge de reducció respecte la mida de la imatge original.

Aquesta opció es el mes simple, tal com mostra la figura 27, es tracta d'assignar en quin percentatge respecta la mida del fitxer el qual es vol fer la reducció.

Si el valor es 90 vol dir que esperem una imatge del 10% de l'original.

Figura 27

Si la reducció de la imatge en el moment d'obrir-la queda per sota de la reducció que es requereix, no fa res. La quantificació es fa per medi de números enters, en conseqüència i degut a l'efecte de truncat dels valors en coma flotant a enters, pot comportar petites diferències entre el resultat esperat i l'obtingut.

En el cas de tenir oberta l'opció de visualització prèvia, les actuacions que es fan en aquesta finestra sobre els diferents blocs de filtrat, es reflecteixen directament en la imatge. Així es possible poder apreciar el nivell de degradació que pateix en funció de la quantificació.

4.2.6. Histograma

Accessible per medi de la icona 

presenta de forma gràfica l'entropia de la imatge un cop transformada.

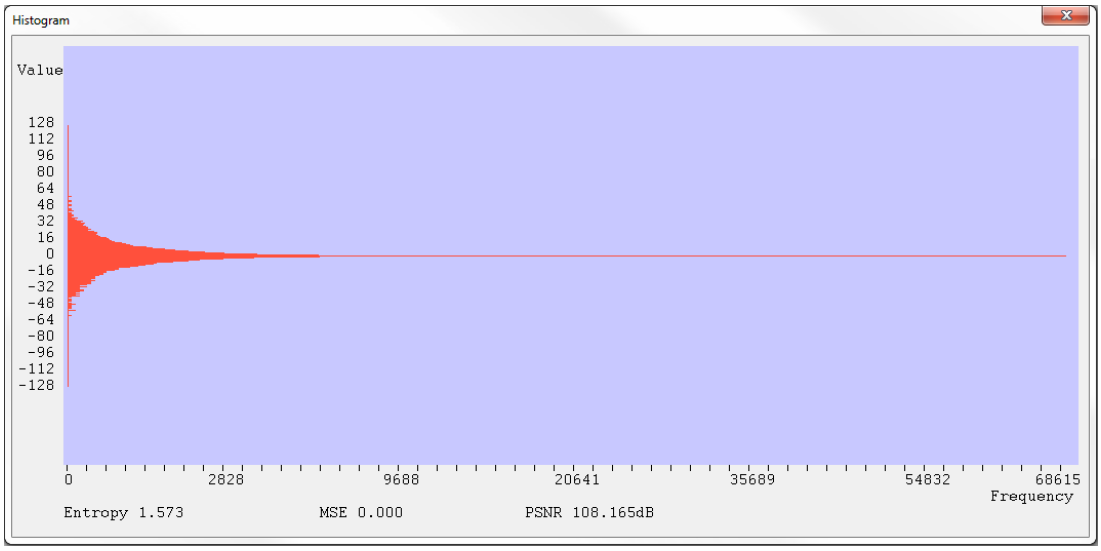


Figura 28

Dona informació gràfica de la forma en què es distribueixen els diferents valors en funció de la seva freqüència d'aparició. També aporta informació sobre l'entropia, l'error quadràtic mitjà i la relació de senyal/soroll, com pot apreciar-se a la figura 28.

Aquesta utilitat funciona en combinació amb la de '***Quantize***' que a l'hora, tant pot utilitzar-se amb la selecció de wavelets com amb visualització prèvia. En cas de tenir activat un bloc, la informació que presenta és únicament la del bloc, el MSE i PSNR segueix sent de tota la imatge.

4.2.7. Salvar

Salva la imatge convertida a wavelets i codificada a mapa de símbols, dins d'un fitxer amb el mateix nom que l'original i en el mateix directori però, substituint l'extensió per WLT..

És accessible per medi de la icona 

, i prement de forma simultània de les tecles de control+S,

4.3. Mètodes d'assaig

Per l'assaig del comportament del compressor es té en consideració l'MSE, error quadràtic mig, PSNR, relació senyal/soroll, i la percepció visual del resultat.

Per obtenir l'error quadràtic mitjà es fa ús de la següent funció:

$$MSE = \frac{1}{m\,n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\left[\hat{Y}(i,j) - Y(i,j)\right]^2$$

On $\hat{Y}$ és la imatge original i Y la imatge que ha sofert el procés de transformar directament, quantificar, des-quantificar, i transformat inversament. Els paràmetres m, n són les dimensions en vertical i horitzontal.

El càlcul de la relació senyal/soroll es realitza per medi de la funció:

$$PSNR = 10 \log_{10} \left(\frac{MAX_I^2}{MSE} \right)$$

On MAX_I^2 és el valor màxim que pot assolir un component d'imatge. Com el rang queda comprés entre -127 i 127 que són 8 bits passa a ser 255^2.

Original 786KB 70KB PSNR 39.09dB 50KB PSNR 37.78dB 30KB PSNR 35.06dB 12KB PSNR 31.48dB

Figura 29

Imatge extreta de:
http://commons.wikimedia.org/wiki/File:Lichtenstein_img_processing_test.png

En les figures 29 i 30 hi ha el resultat de comprimir i descomprimir dues imatges aplicant diferents valors de quantificació a la codificació wavelet. La percepció visual entre la imatge original i la més quantificada (figura 29) és que la diferència és poc apreciable. Cal tenir present que ocupa un 1,5% de l'original, el que es correspon amb una relació de 1:65.

Original 786KB 70KB PSNR 39.04dB 50KB PSNR 36.01dB 30KB PSNR 31.89dB 12KB PSNR 27.89dB

Figura 30

Imatge extreta de:
http://people.sc.fsu.edu/~jburkardt/data/pgma/barbara.png

En aquest cas es tracta d'una imatge complexa (figura 30) on hi un gran nombre de components d'alta freqüència i en conseqüència la relació senyal/soroll per un mateix nivell de compressió dóna un valor més baix. Cal tenir en compte que si la relació tendeix a 1 vol dir que hi ha tant senyal com soroll.

És important senyalar el fet que la representació de les mostres exposades en els exemples anteriors, al estar reduït, els defectes derivats del procés de compressió queden minimitzats per la qual cosa cal prendre com a referència les dimensions del fitxer final amb el seu PSNR.

4.3.1. Comparatiu amb DCT

Per dur a terme una comparativa correcta entre el resultat de comprimir imatges wavelet en aquest projecte respecte imatges JPEG, s'ha desenvolupat un programari específic: el **PSNR** TFC.

El programari **PSNR,** identificat amb TFC, permet realitzar la mesura de la relació senyal soroll entre un fitxer original BMP i un fitxer JPEG o WAVELET. Al obrir el programari presenta les següents opcions (figura 31):

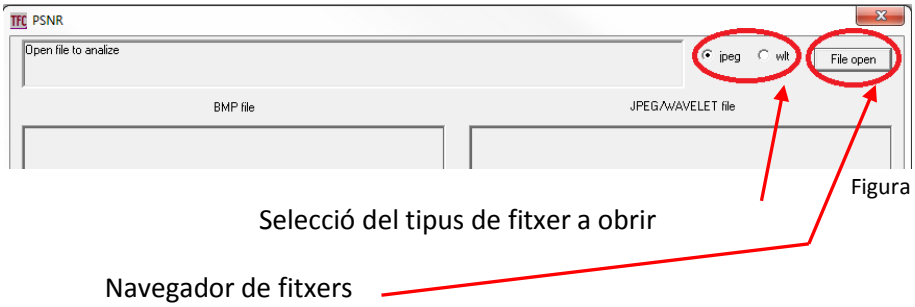


Figura 31

Per al seu ús, únicament cal triar quin tipus de fitxer es vol valorar i obrir el programari. L'única opció possible és un fitxer BMP, l'altre l'obre a partir de l'opció triada. Un cop obert, pot fer-se el canvi de selecció de fitxer de forma dinàmica (figura 32).

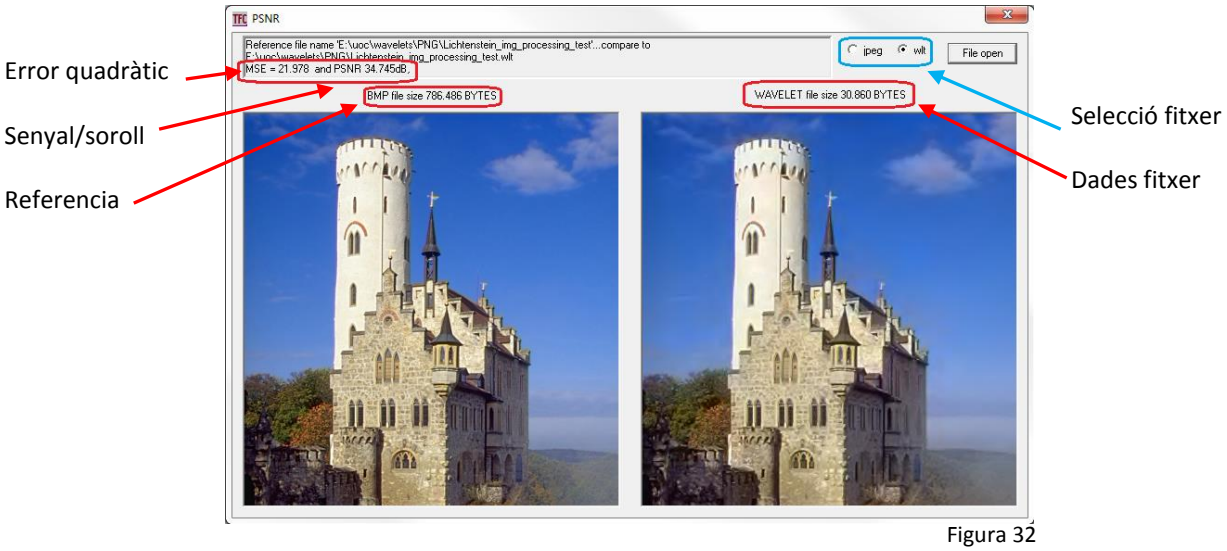


Figura 32

Per medi d'aquesta utilitat, es fa una anàlisi de dues imatges les quals han estat reduïdes a 1,5% del fitxer original tant en JPEG com en Wavelet.

La primera és el castell de Lichtenstein. L'assaig és amb tres fitxers que duen el mateix nom però diferents extensions. L'original Lichtenstein_12K.bmp de 786Kbytes, Lichtenstein_12K.jpg d'12Kbytes i Lichtenstein_12K.wlt de 12Kbytes. Amb la finalitat de poder apreciar un mínim de detall en el resultat les imatges es presenten sense reducció.

És important assenyalar que el nom de tots els fitxers ha de ser exactament el mateix però amb les extensions que els identifiquen, *.BMP, *.JPG, *.WLT.

La figura 33, presenta una imatge JPEG, que ocupa 12,9Kbytes amb una relació senyal soroll de 31,4 dB. Pot apreciar-se un fort efecte d'enrajolat però manté algunes regions amb força detall.

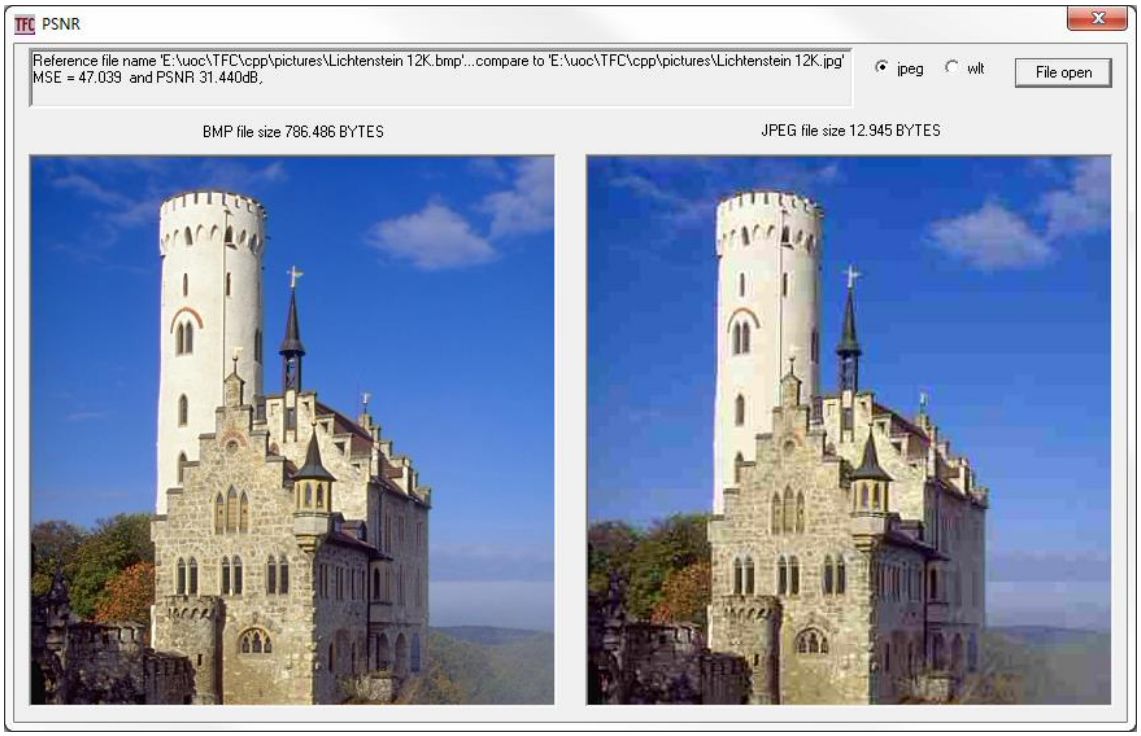


Figura 33

La figura 34 presenta la mateixa imatge que en el cas anterior amb un fitxer de dimensions similars, la diferència més notable és l'homogeneïtat de la imatge, no hi ha l'efecte d'enrajolat, per contra els detalls també queden difosos tot i que la sensació és més agradable a la vista.

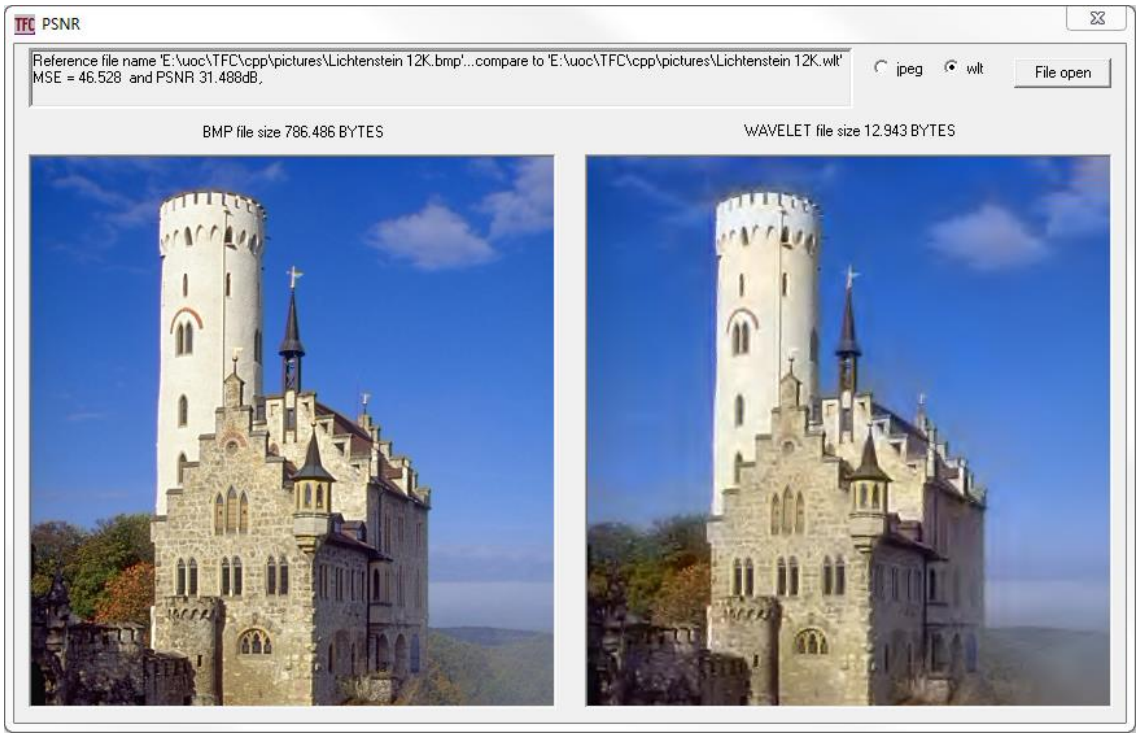


Figura 34

La imatge que es mostra tot seguit (figura 35) aporta informació interessant per tractar-se d'una imatge molt complexa a nivell de components d'alta freqüència. També són dos fitxers reduïts a un 1,5% de l'original, que equival a una reducció 1:65.

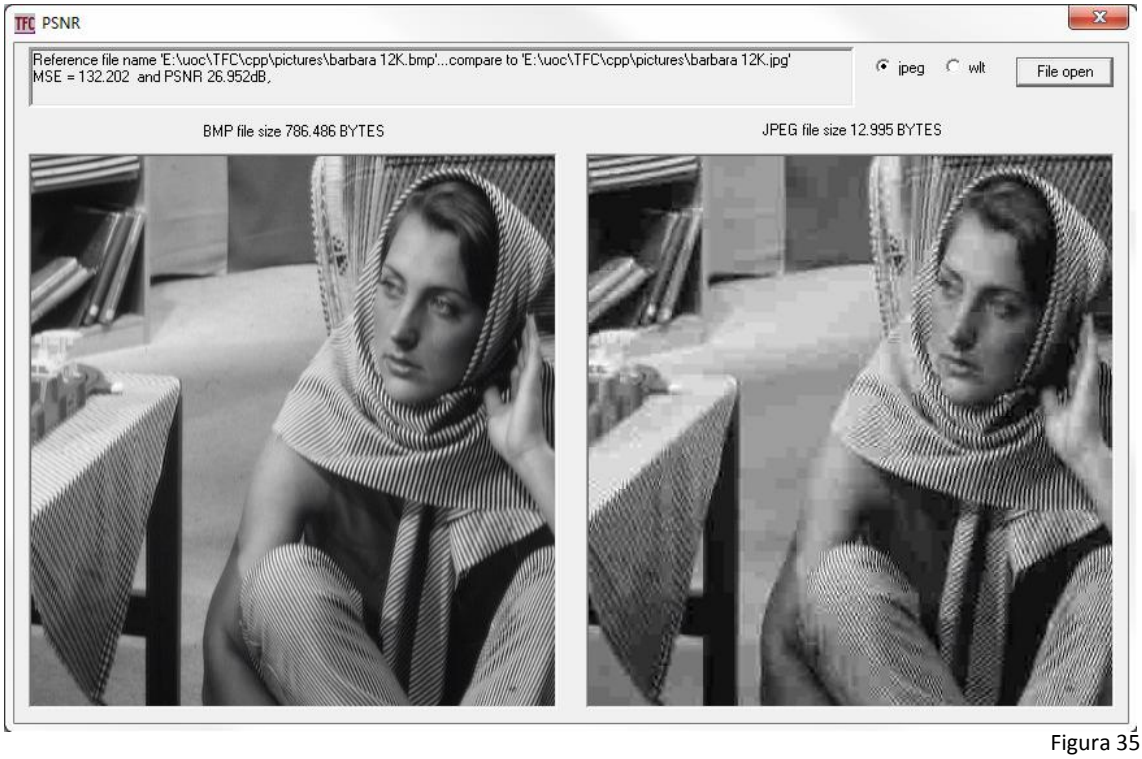


Figura 35

A l'igual que en la figura 33, la imatge presenta un aspecte de mosaic en aquest cas la relació de senyal/soroll ha baixat el que vol dir que la imatge és pitjor.

Figura 36

La figura 36 presenta una imatge transformada a wavelets que ha perdut detall, però la sensació visual és més agradable, la seva relació senyal/soroll és l'equivalent de la figura 34, semblant a la del JPEG.

4.3.1. Entropia

L'entropia es defineix com la mesura del desordre. En el cas de compressió d'imatges és una part molt important a tenir en compte. La finalitat principal de les diferents tècniques de transformació és la d'assolir una entropia el més baixa possible.

En aquest treball ha estat una constant que ha condicionat el disseny en tot el seu conjunt. En conseqüència, en el disseny s'han establert cinc nivells de filtrat amb DWT per tal d'assolir un valor reduït en el conjunt de la imatge, amb això s'aconsegueix que existeixi una única taula de símbols per codificar tots els valors, inclòs el bloc **LL5** que correspon als DC's que és el que presenta una entropia més alta, com pot observar-se a la figura 37.

Per 'l càlcul de l'entropia en aquest treball, s'ha utilitzat l'expressió:

$$H = \sum_j pj \log_2 \left(\frac{1}{p_j}\right)$$

On p_j es la probabilitat que aparegui un símbol $s_j \in S$, on S es conjunt de tots els símbols.

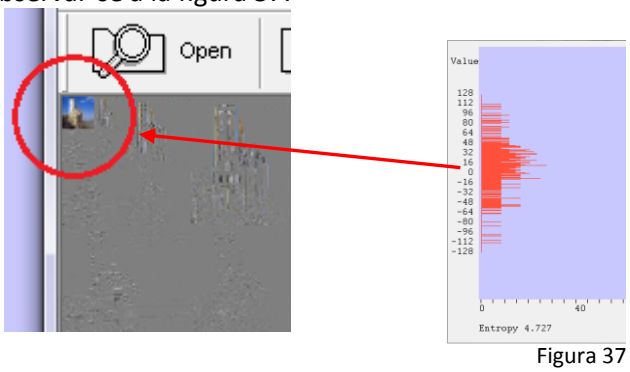


Figura 37

5. Conclusions i línies de futur

5.1.Objectius aconseguits

Construir un programari que permet comprimir imatges transformades per medi de DWT, on la part de codificació de l'entropia ha estat dissenyada íntegrament en el curs d'aquest treball. Es molt important per a mi poder haver arribat a les conclusions que han portat al disseny del codificador a partir de les eines que s'han anat construint a mida que el treball ha anat avançant.

També valoro positivament el comportament del compressor, fent un comparatiu amb un compressor JPEG es pot dir que, en general, el rendiment és molt igualat. Per tractar-se d'un projecte desenvolupat íntegrament en un curt termini de temps on la part de refinament sobre el producte acabat ha estat molt breu, el resultat és més que satisfactori.

5.2. Conclusions

La conclusió que s'extreu del treball presentat és que, en el seu conjunt, s'obté un rendiment alt a l'hora de comprimir una imatge. Revisant els resultats comparatius entre el JPEG i el compressor desenvolupat en aquest treball es fa evident que, en l'estat actual la relació senyal soroll per una mateixa imatge comprimida amb JPEG i wavelets tenen un resultat semblant al JPEG.

En aquesta primera versió hi ha dos punts que caldria revisat amb detall. Un es trobar la forma de minimitzar el soroll quantitatiu derivat del truncat de bits en el procés de quantificar els valors a codificar. El truncat és degut a la transformació de variables tipus `float` a `int` en el moment de quantificar i transformar a símbols.

L'altre aspecte a millorar és la forma que s'exploren els símbols en el procés de codificació. En el treball actual l'exploració es realitza de forma de zig-zag en caixes de 4*4 símbols dins del bloc corresponent. Si ens fixem amb atenció en com es distribueixen els diferents valors dins del bloc, tal com està descrit en el punt **3.4. Codificació de la imatge transformada**, els zeros es localitzen per àrees més o menys extenses depenent del tipus d'imatge i del grau de quantificació aplicada. Cal explorar noves formes d'escombrar la imatge.

5.3. Compressió de fluxos d'imatges continus

La generació de fluxos continus d'imatges pot fer-se de dues formes, una comprimint imatge a imatge. Aquesta solució és la que utilitza el MJPEG, és la més senzilla però la que ocupa més recursos. L'altre és la d'enviar una imatge comprimida i seguidament una seqüència d'imatges diferencia respecte la imatge original.

A partir del disseny que presenta aquest treball és possible que partint d'una imatge transformada original obtenir una imatge diferència directament des d'els mateixos blocs de filtrat. Aquesta imatge diferència per la seva naturalesa, on els valors en origen són propers a zero donaria com a resultat grans grups de zeros. Com el disseny del codificador d'entropia ja està orientat a comprimir zeros el resultat de les imatges diferències ha de ser molt petit.

5.4. Cerques de canvis en grans grups d'imatges

La transformació d'una imatge a wavelets, està feta per nivells de filtrat, aquests nivells estan identificats pels tipus de resposta que tenen i es distingeixen com a HL1, HH1, LH1, HL2, ... LL5. Bloc LL únicament existeix un de sol. Aquest bloc a diferència de tots els altres té la particularitat de ser un resum de tota la imatge.

Cada Valor del bloc LL5 representa el valor resum d'una àrea d'imatge de 16x16 components. Si dins de la seqüència d'imatges tota la informació corresponent al bloc LL5 queda disposada en l'inici, llegint únicament aquesta informació és possible detectar canvis entre imatges de forma eficient sense necessitat de fer cap mena de reconstrucció de la imatge sencera.

6. Glossari

(4:2:0)
: codificació de quatre valors de lluminària un de crominància Cb i un de crominància Cr

(4:2:2)
: codificació de quatre valors de lluminària dos de crominància Cb i dos de crominància Cr

(4:4:4)
: codificació de quatre valors de lluminària quatre de crominància Cb i quatre de crominància Cr

JPEG2000
: estàndard de compressió d'imatges basat amb transformades wavelet

JPEG
: estàndard de compressió d'imatges basat amb transformades del cosinus

ArgoUML
: eina gràfica orientada al disseny de classes per medi de diagrames de blocs

BMP
(Bit-map) arxiu per a gràfics basat en el mapa de bits

CalculateEntrpy
calcula l'entropia de la imatge

CalculateMeanSquareError
realitza el càlcul de l'error quadràtic mig

Cb
component de crominància diferencia entre el blau i el verd

CCIR 601
primera norma sobre la televisió digital

CDF 5/3
transformada wavelet reversible

CDF 9/7
transformada wavelet irreversible

classe
conjunts de peces de codi de programa que contè objectes instanciats

ClearHistogramValue
esborra els atributs públics i privats de la classe Wavelet relacionats amb ***Histogram***

Cr
component de crominància diferencia entre el blau i el vermell

crominància
components de color d'una imatge

DC
(Direct current)component continua

DCT
(Discrete Cosine Transform)transformada discreta del cosinus

Display
classe especialitzada en pintar a la pantalla

DWT
(Discrete Wavelet Transform)transformada wavelet discreta

EBCOT
(Embedded Block Coding with Optimal Truncation) pot traduir-se com a bloc incrustat amb truncament òptim

Entropia
mesura del desordre

HH
(high, high) bloc de filtrat de freqüències alta, alta

Histogram
eina gràfica especialitzada en la representació de la freqüència que es repeteix cada valor present dins el conjunt de la imatge

Histograma
representació gràfica del valor d'una o més variables

HL
(high, low) bloc de filtrat de freqüències alta, baixa

Huffman
mètode de codificació basat amb taules de símbols

LH
(low,high) bloc de filtrat de freqüències baixa, alta

LL
(low, low) bloc de filtrat de freqüències baixa, baixa

luminància
component de la intensitat de la imatge

MainFrame
Classe de la finestra principal

MAX
es el valor màxim en bits que pot assolir un component d'imatge

megapixel
definició d'una imatge formada per milions de pixels

MSE
(mean squared error)error quadràtic mig

PictureDecode
descodifica el contingut d'un fitxer i ho converteix a imatges en rajoles wavelet

PictureEncoder
codifica una imatge formada per rajoles wavelet per poder ser posat en un fitxer

PictureQuantize
quantifica una imatge formada per rajoles wavelet

PictureToWavelet
transforma una imatge en format (4:2:0) a mosaics wavelets

pixel
(picture element)element elemental per composar una imatge

Pseudocodi
forma genèrica d'escriure un algorisme

PSNR
(Peak Signal-to-Noise Ratio) relació senyal/soroll

quantificació
expressió numèrica d'una magnitud, en aquest treball el resultat de realitzar una divisió entre dos valors

Quantize
eina orientada a generar de forma manual o automàtica els valors necessaris per poder quantificar la imatge

Resources
classe d'instàncies de suport al programari

sub-banda
conjunt de freqüències definides en un interval determinat

TFCView
classe de vista, contè els recursos per interactuar amb l'usuari

VLC
(Variable Code Length) codificador de longitud variable

wavelet
terme original ondelette traduït l'angles com a wavelet

Wavelet
classe especialitzada en la compressió d'imatges

WaveletToPicture
converteix una imatge formada per rajoles wavelet (4:2:0) a imatge sencera

WEB
(World Wide Web)xarxa d'abast mundial

Windows Visual C++
compilador de C++ per sistemes operatius Windows

Y
luminància

7. Autoavaluació

Des del lliurament del programari de l'ultima entrega, anterior al lliurament de la memòria, s'ha detectat una deficiència en la classe ***TFCView***, en concret en la visualització prèvia. Afecta a algun tipus d'imatge quan produeix una quantificació forta. Aquesta anomalia ja ha estat corregida.

També ha quedat substituïda la imatge de soroll aleatori de la pantalla d'obertura per un color gris suau.

El model d'escombrat lineal de la imatge en la codificació de l'entropia, ha estat reemplaçat per un escombrat en forma de zig-zag en blocs de 4x4 components.

En el desenvolupament del programari **PSNR**, per mesurar la relació de senyal soroll de les imatges JPEG i wavelets s'ha detectat un defecte en la funció `OpenBMPfile` de la classe ***Resources***. Aquesta anomalia ja ha estat corregida.

La classe ***Quantize***, no té funcions publiques, les que originalment estaven declarades com a públiques han estat desplaçades a la part privada.

En el manual de funcionament on diu '*reversible CDF 9/7*' ha de dir '*irreversible CDF 9/7*'. També ha estat ampliat, s'ha inclòs el funcionament de la utilitat **PSNR**.

8. Bibliografia

8.1. Documentació en paper

S. Taubman David	*Jpeg2000*	ISBN 0-7923-7519-x	2002
Ceballos Fco. Javier	*Programación orientada a objetos*	ISBN 84-7897-268-4	1997
Watkinson John	*The art of digital video*	ISBN 0-240-51586-2	2000
Watkinson John	*The mpeg handbook*	ISBN 0-240-51656-7	2001
Keith Jack	*Video demystified*	ISBN 1-878707-36-1	1996

8.2. Documentació WEB

8.8.1. des-compressor JPEG

http://keyj.emphy.de/nanojpeg/

8.8.2. Tècniques per a compressió d'imatges

http://es.wikipedia.org/wiki/CCIR_601
http://es.wikipedia.org/wiki/Teorema_de_muestreo_de_Nyquist-Shannon
http://en.wikipedia.org/wiki/Chroma_subsampling
http://es.wikipedia.org/wiki/Entrop%C3%ADa
http://en.wikipedia.org/wiki/JPEG
http://en.wikipedia.org/wiki/Discrete_wavelet_transform
http://en.wikipedia.org/wiki/Cohen-Daubechies-Feauveau_wavelet

8.8.3. Articles i material didàctic

http://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio
http://www.dc.uba.ar/materias/cid/2012/descargas/Practicas/practica1CID.pdf
http://office.microsoft.com/es-es/excel-help/presentar-datos-en-un-diagrama-de-gantt-en-excel-HA010238253.aspx#BMfloatingcolumns
http://icrovett.wordpress.com/2010/02/08/uml-diseno-de-agregacion-vs-composicion/
http://www.olhovsky.com/content/wavelet/2dwavelet97lift.py
http://www.mathworks.com/matlabcentral/fileexchange/11846-cdf-97-wavelet-transform/content/wavecdf97.m

9. Annexos

9.1. Compilació programari

El programari és format per dues unitats separades: el **TFC,** que és la raó d'aquest treball, i el **PSNR**, programari complementari orientat a l'avaluació de la qualitat de la compressió i contrast amb l'estàndard JPEG.

Els fitxers de projecte del programari **TFC** es troben disposats dins de la carpeta *'.\TFC\cpp\TFC*' i els del projecte **PSNR**, són dins de la carpeta '.*TFC**cpp*\ *PSNR*'.

Ambdós programaris han estat compilats amb ***Windows Visual C++*** amb la configuració que integra les llibreries dins el propi codi, amb això s'evita haver de fer el procés d'instal·lació.

9.2. Execució

Els dos programaris d'aquest treball han de funcionar en qualsevol versió de sistema operatiu Windows, ha estat provat amb màquines virtuals de Windows 98, Windows XP i en un Windows 7 real. Els fitxers executables de **TFC** i **PSNR**, són dins de la carpeta '.*TFC**cpp'*.

Per al funcionament dels dos programaris hi ha un manual d'usuari que permet d'utilitzar de forma senzilla a la carpeta '.*TFC*\ *manual*'. Per als exemples hi ha un directori amb imatges a la carpeta '.*TFC*\ *cpp**pictures*' per al **TFC** i '.*TFC*\ *cpp**TetsPictures*' per al **PSNR**.