Implementació d'un sistema per al control de la qualitat d'imatges JPEG (III)

Treball

Final

Carrera

UNIVERSITAT OBERTA DE CATALUNYA
Consultor: Julià Minguillón Alfonso
Autor: Antoni Francés Conejero
Juny de 2002

Índex

1. Introducció

1.1. Les imatges a l'ordinador

Cada dia és més habitual incorporar elements multimèdia (gràfics, so, vídeo, ...) en qualsevol treball generat amb ordinador, però aquests elements fan crèixer molt la mida dels arxius, per això és important optimitzar aquests elements.

Si parlem de les imatges podem dir que hi ha dos mètodes generals per visualitzar imatges a l'ordinador:

- Vector

  En aquest format s'usa una sèrie de comandaments per representar una imatge. Normalment els monitors i les impressores làser tenen un software per convertir els comandaments dels vectors en pixels.

- Bitmap, també coneguda com imatge *raster*

  Es representa amb un array bi-dimensional per cada component de la imatge.

  Amb aquest tipus d'imatge cada vegada que volem canviar la seva mida tenim els següents problemes, pèrdua de resolució quan reduïm la imatge, obtenció de nous punts amb algun mètode per fer ampliacions,...

  Nosaltres tractarem el problema de comprimir una imatge, és a dir, donada una imatge amb una mida i resolució concreta, obtenir-ne una altra amb quasi les mateixes característiques que ocupi menys espai al disc.

L'estàndard JPEG és el format gràfic per a imatges fixes de to continu que s'utilitza com a mitjà de publicació en la web. La seva bona relació entre la raó de compressió obtinguda i la qualitat de la imatge comprimida, així com la seva àmplia difusió entre els usuaris de la xarxa Internet, han fet que hagi estat adoptat com a l'estàndard per a compressió d'imatges, desbancant altres formats gràfics.

La qualitat i la relació de compressió de les imatges JPEG estan determinades per les matrius de quantificació usades en el procés de compressió (cada component de la imatge pot estar quantificada amb una matriu diferent). Les matrius utilitzades s'emmagatzemen amb l'arxiu JPEG per a permetre la descodificació de la imatge amb els algoritmes adequats. Per defecte, l'estàndard JPEG proporciona una matriu de quantificació que, empíricament, s'ha demostrat que produeix imatges de qualitat mitja i/o alta i relacions de compressió bones per la majoria d'imatges i d'usuaris. Multiplicant la matriu de quantificació per un escalar s'obtenen diferents relacions de compressió i qualitats d'imatge.

Un dels problemes actuals de l'estàndard JPEG és la manca de relació entre el factor de qualitat especificat per la matriu de quantificació i la qualitat real que s'obté després del procés de compressió. A més, la percepció de l'observador és un terme completament subjectiu que habitualment és ignorat en el procés de compressió, quan realment hauria de tenir-se en compte en el moment de calcular la matriu de quantificació. El resultat és que l'única manera de saber com quedarà guardada la imatge després de tot el procés és tornant a visualitzar-la, és a dir: utilitzant el procediment d'assaig i error. Aquest procés té un cost computacional elevat ja que la imatge s'ha de comprimir, descomprimir i després calcular l'error quadràtic mitjà[1] per a saber la qualitat d'imatge obtinguda.

És necessari, doncs, disposar d'eines que permetin experimentar modificant paràmetres de la matriu de quantificació per a aconseguir una qualitat d'imatge satisfactòria segons les percepcions de l'usuari, sense penalitzar la raó de compressió i amb un cost computacional reduït. Aquestes eines han de permetre establir el valor de qualitat desitjat i veure els resultats abans de processar la imatge definitivament.

[1] En anglès *Mean Square Error (MSE)*

1.2. Objectius

Tenint en compte tot el descrit en el punt anterior, amb aquest projecte es vol aconseguir tenir una eina visual que ens permeti comprimir imatges modificant els paràmetres de quantització.

L'objectiu del TFC és construir una aplicació gràfica que permeti comprovar la qualitat de compressió d'una imatge amb l'estàndard JPEG i en permeti modificar els paràmetres de qualitat basant-se en el model de visualització humà[2]. Aquests paràmetres venen reflexats per una variable corresponent al valor de pic de la relació senyal–soroll[3] amb la qual es construeix una matriu de quantització que ens definirà els valors de qualitat de la imatge comprimida, així com la possibilitat d'escollir entre diferents funcions a l'hora d'obtenir la matriu de quantificació. També és un objectiu el fer l'aplicació multiplataforma i incorporant les 3 components de color.

L'aplicació està basada en una llibreria començada en el TFC de l'estudiant Jordi Pino durant el segon semestre del curs 2000/2001 i que implementa les eines bàsiques per obtenir la informació reflectida en el paràgraf anterior, així com en l'aplicació desenvolupada en el TFC de l'estudiant Daniel Fusté en el primer semestre del curs 2001/2002.

1.3. Convencions tipogràfiques

Les referències a codi de programació o pseudocodi s'escriuen en lletra d'amplada fixa tipus `Courier New`.

[2] En anglès *Human Visual System (HVS)*

[3] En anglès *Peak Signal Noise Ratio (PSNR)*

1.4. Estructura de la memòria

La memòria està organitzada en 6 capítols que contenen una explicació de les tècniques que s'han utilitzat en el treball amb imatges, l'explicació del disseny de l'aplicació i de la seva interfície, els detalls de la implementació i les conclusions obtingudes. També s'inclou la bibliografia utilitzada i uns annexos que contenen les estructures emprades i el codi font de l'aplicació.

Més concretament:

- Introducció: explica els objectius generals del TFC, les convencions tipogràfiques i l'estructura general del document.
- Descripció: explica una descripció genèrica de tota l'aplicació i el pla de treball seguit.
- Llibreria JAI: explica les llibreries utilitzades en la implementació de la part gràfica i en fa una anàlisi de les seves funcionalitat en front d'altres eines que també podien ser utilitzades.
- L'estàndard JPEG: introdueix al lector en la manera com es generen els arxius JPEG i en quines eines teòriques està basat.
- Implementació: explica els detalls de la implementació, tant en la part de l'aplicació gràfica com en la part de l'enllaç amb la llibreria.
- Conclusions: extreu unes conclusions sobre l'aplicació. Explica fins on s'ha arribat i quines són les previsions d'ampliació del projecte.

2. Descripció

2.1. Descripció del treball

L'objectiu del treball és construir una aplicació que permeti comprovar la qualitat d'un arxiu JPEG abans de guardar-lo definitivament.

Aquest treball és la continuació del treball *Implementació d'un sistema per al control de la qualitat d'imatges JPEG (I)*[4] realitzat per l'estudiant Jordi Pino Lacosta durant el segon semestre del curs 2000/2001 i del treball *Implementació d'un sistema per al control de la qualitat d'imatges JPEG (II)*[5] realitzat per l'estudiant Daniel Fusté durant el primer semestre del curs 2001/2002.

El TFC de l'estudiant Jordi Pino va consistir en construir una llibreria de rutines bàsiques que permetin obtenir els paràmetres de qualitat de la imatge.

El TFC de l'estudiant Daniel Fusté va consistir en construir una aplicació gràfica que permetés carregar una imatge, i mitjançant una llibreria dinàmica fer les cridres a les rutines per obtenir i representar els paràmetres que representen la imatge, per després comprimir-la.

En aquest treball el que s'ha fet és dissenyar una nova interfície gràfica per adaptar-la a la programació Java seguint el model d'events per delegació i les classes gràfiques *Swing*, encara que s'ha intentat mantenir un aspecte semblant a l'anterior. S'incorporen les 3 components de la imatge en cada matriu. També s'han adaptat les crides a la llibreria per poder passar-li com a paràmetre la funció *phi* desitjada per fer la quantificació, així com s'ha corregit el codi C d'algunes funcions que estaven malament.

[4] Veure bibliografia

[5] Veure bibliografia

2.2. Descripció de les funcionalitats del TFC de l'estudiant Jordi Pino

El TFC de l'estudiant Jordi Pino consisteix en unes rutines que permeten fer el següent procés:

1. Llegir una imatge original i realitzar la transformació de color quan sigui necessari.

2. Segmentar cada component de la imatge en blocs de 8x8 pixels i calcular la DCT de cada bloc.

3. Calcular els descriptors estadístics de cada component transformada segons un cert model probabilístic (mitjana, variança i factor de forma d'una variable aleatòria Laplaciana o Gaussiana generalitzada).

4. A partir de la qualitat desitjada del model del sistema visual humà utilitzat i dels descriptors estadístics calculats anteriorment, calcular la matriu de quantificació utilitzant l'algorisme descrit a l'article del professor Julià Minguillón "*JPEG standart uniform quantization error modeling with applications to sequential and progressive operations modes*".

5. Un cop calculada la matriu de quantificació, cada component de la imatge transformada és quantificada i desquantificada, i després es calcula l'antitransformada de cada bloc, de manera que es simula el procés que realitza l'estàndard JPEG (excepte l'etapa de codificació ja que, en no introduir cap pèrdua i ser completament reversible, només és necessària per generar la imatge comprimida) per a obtenir la imatge reconstruïda a partir de la imatge comprimida.

2.3. Descripció de les funcionalitats del TFC de l'estudiant Daniel Fuste

El TFC de l'estudiant Daniel Fusté consisteix en una aplicació gràfica que permet fer el següent procés:

1. Una part on es permet localitzar un fitxer de imatge dins el disc, mitjançant un quadre de diàleg estàndard de localització de fitxers. També es presenta la imatge original per pantalla i permet introduir el paràmetre PSNR, aquest paràmetre ha d'estar donat per l'usuari i cal que sigui major de 0.

2. En la segona part és on, utilitzant la llibreria dinàmica, es realitzen els càlculs, s'emmagatzemen els valors necessaris obtinguts i es presenten per pantalla els valors obtinguts, només en una component.

3. En la tercera part, utilitzant els valors anteriors, es comprimeix la imatge i es visualitza per pantalla.

2.4. Objectius de treball

Es vol aconseguir crear una eina gràfica capaç de comprimir imatges en format JPEG permetent modificar els paràmetres de qualitat i la funció *phi* en temps real i permetent així aconseguir el nivell desitjat de qualitat. Els passos bàsics que caldrà implementar són els següents:

- Corregir les rutines implementades en el TFC anterior de l'estudiant Daniel Fusté i modificar la llibreria d'enllaç dinàmic per poder fer les noves crides des de l'aplicació gràfica.

- Crear un programa que estarà compost de les següents parts:

  - Llegir una imatge original en diferents formats i presentar-la per pantalla.

  - Permet entrar un paràmetre anomenat PSNR (Peak Signal Noise Ratio) que servirà per poder introduir una correcció d'error de quantificació en

el moment de fer el càlcul dels coeficients de quantificació. Es permet tant l'entrada manual com mitjançant una barra de desplaçament.

- Permet seleccionar la funció *phi* desitjada, es dona la possibilitat d'escollir entre la *identitat* i una que te en compte el sistema de visió humà *HVS*.

- A partir de la imatge, el PSNR, la funció *phi*, utilitzant la llibreria dinàmica creada, obtenir la matriu de quantificació i presentar-la per pantalla.

- Crear la imatge comprimida a partir de la matriu trobada i presentar-la per pantalla juntament amb la mida de l'arxiu generat.

- Permet modificar els valors de la matriu de quantificació per tal de poder generar una nova imatge comprimida a partir d'ella.

La idea és que un usuari pugui carregar la imatge que vulgui comprimir, la visualitzi, vegi la matriu de quantització i visualitzi la imatge comprimida segons aquella matriu. A partir d'aquí es dona la funcionalitat afegida de poder fer els canvis que es vulguin a la matriu de quantificació per tal d'aconseguir millors factors de qualitat en la compressió.

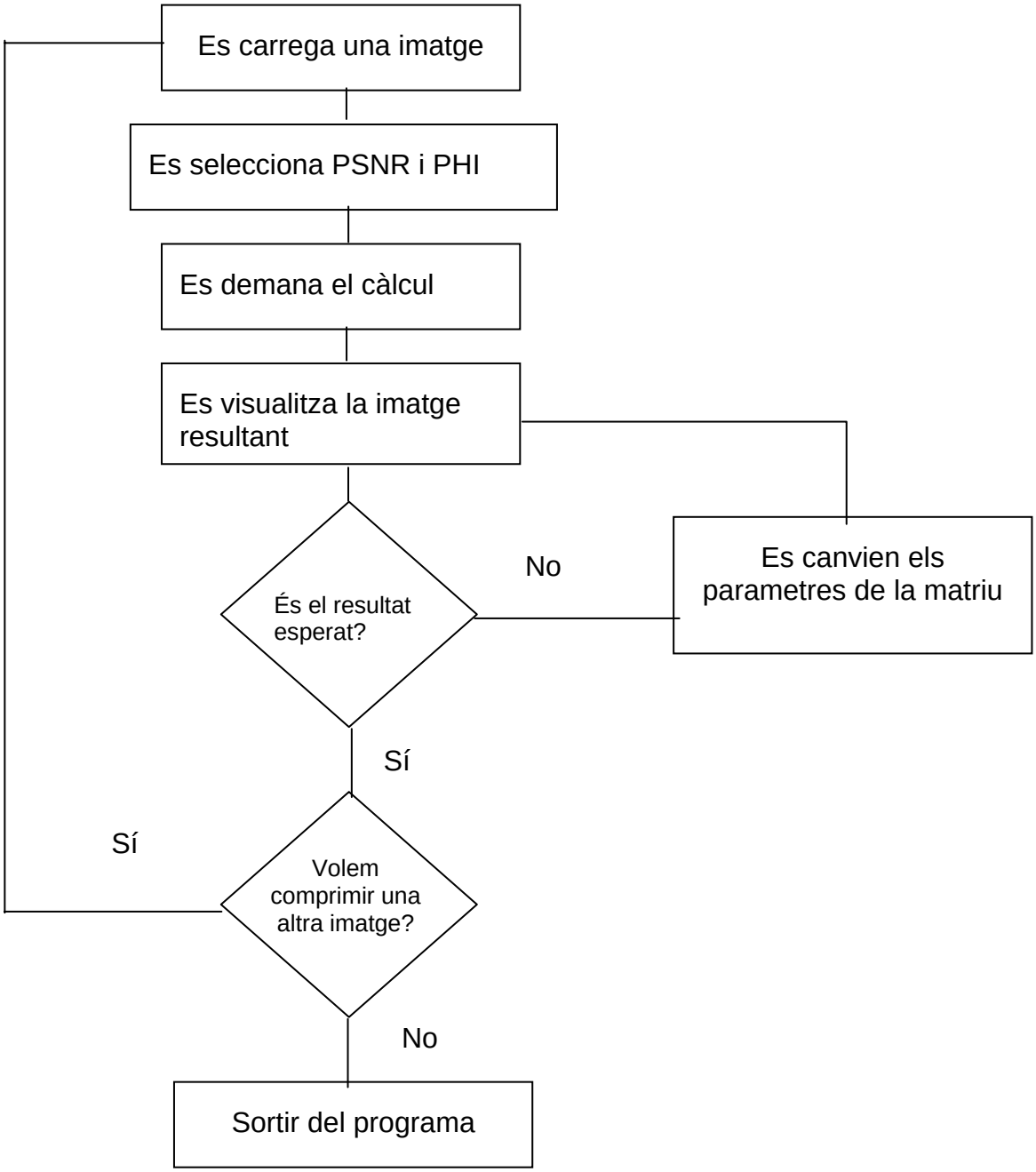
Per donar aquestes funcionalitats l'aplicació consta de tres parts, encara que integrades en una mateixa interfície i formant part d'un tot indivisible per l'usuari. Les parts són les següents:

1. Una part on es permet localitzar un fitxer de imatge dins el disc, mitjançant un quadre de diàleg estàndard de localització de fitxers, totalment integrat amb la interfície gràfica del sistema operatiu. També permet desar l'arxiu comprimit mitjançant un quadre de diàleg estàndard de localització de fitxers.

2. En la segona part és on, utilitzant la llibreria dinàmica, es realitzen els càlculs, s'emmagatzemen els valors necessaris obtinguts i es presenten per pantalla els valors obtinguts, amb aquests valors es comprimeix la imatge i es visualitza al costat de la imatge original.

3. En la tercera part, modificant els paràmetres desitjats, es torna a comprimir la imatge i es compara amb l'anterior.

La dinàmica de utilització del programari és la següent:



2.5. Eines utilitzades en la implementació

La llibreria s'ha desenvolupat (així ho va començar els meus companys que realitzaren les parts anteriors) en C.

S'ha utilitzat l'eina DevC++ v 4.0 que és un compilador i un IDE de programació en C i C++ de lliure distribució i també per comprovar compatibilitats amb les eines més utilitzades en PCs s'ha utilitzat Microsoft Visual C++ 6.0.

D'altra banda, per fer l'aplicació independent de la plataforma, s'ha compilat sota Linux, preparant els arxius Makefile necessaris per fer instal·lació i compilació de tota l'aplicació.

Per fer la interfície gràfica s'ha utilitzat el llenguatge JAVA per tal de fer una aplicació amb la màxima portabilitat possible i a l'hora de generar una aplicació amb una tecnologia que permeti la seva utilització compartida mitjançant la xarxa Internet o qualsevol altra xarxa de comunicacions.

S'ha utilitzat la versió JDK 1.4 per la compilació del codi i com a IDE de desenvolupament s'ha utilitzat JCreator LE 2.0 i FreeJava 3.0 que són entorns petits, però amb la potencia i parametrització necessaris per poder integrar totes les eines que s'han anat utilitzant i de lliure distribució amb el que aconseguim realitzar el projecte sense utilitzar programari amb copyright.

Així mateix, tot i que no és necessari, per fer el disseny gràfic s'ha utilitzat la versió limitada de IBM Visual Age for Java 4.0, amb la qual es generen les pantalles i ens construeix els mètodes *get* i *set* de cada classe automàticament, amb el corresponent estalvi de temps.

Per fer l'enllaç entre l'aplicació de Java i la llibreria DLL s'ha utilitzat la tecnologia desenvolupada per Sun Microsystems JNI (Java Native Interface) que permet crear un protocol de comunicació entre objectes Java i funcions en altres llenguatges, especialment C/C++;

Per tal d'aconseguir la màxima qualitat en l'aplicació s'ha fet una recerca de recursos per estandarditzar l'aplicació, com a resultat d'aquesta recerca s'ha decidit utilitzar un paquet de llibreries desenvolupades de Sun Microsystems anomenades JAI (Java Avanced Imaging) amb les quals s'ha pogut parametritzar la majoria dels aspectes de la presentació de les imatges per poder fer les modificacions posteriors que es vulguin.

En el capítol següent es descriuen els motius pels quals s'ha escollit la llibreria JAI i es fa una comparació amb altres llibreries de propòsit gràfic. En aquest capítol es descriuen les funcionalitats principals de la llibreria.

2.6. Pla de treball

Descripció del treball	Data prevista de finalització	Data real de finalització
Estudi del TFC de l'estudiant Daniel Fusté.	15/03/2002	15/03/2002
Estudi de la bibliografia complementaria sobre formats d'imatge.	05/04/2001	5/04/2002
Instal·lació del programari i llibreries. Jocs de proves amb el TFC de l'estudiant Daniel Fusté i solució de petites inconsistències trobades.	15/04/2002	14/04/2002
Millora del disseny de la interfície de l'aplicació.	10/05/2002	10/06/2002
Ampliació de la interfície pel tractament dels diferents components de colors.	20/05/2002	10/06/2002
Ampliació dels diferents tipus d'imatges suportats.	3/06/2002	3/06/2002
Ajustos finals en l'aplicació.	24/06/2002	24/06/2002
Memòria i documentació.	24/06/2002	30/06/2002

3. La llibreria JAI (Java Avanced Imaging)

3.1. Introducció

La programació gràfica introdueix una dificultat afegida a l'hora de crear una aplicació amb la màxima portabilitat possible. El problema és que cada plataforma de maquinari i/o programari implementen unes funcions gràfiques determinades i no independents de la plataforma, això fa que s'hagi d'optar per un paquet de llibreries estandarditzades i amb la màxima difusió possible, per tal d'aconseguir el propòsit de la portabilitat.

El treball amb imatges *raster* (imatges de mapa de bits) també introdueix una dificultat a l'hora de crear una aplicació portable, cada sistema operatiu treballa amb els seus formats de registres, el que fa que els camps de capçalera dels fitxers d'imatge s'hagin de llegir d'una forma o d'una altra depenen del model de memòria utilitzat pel sistema. També en aquesta ocasió utilitzar una llibreria estàndard ens soluciona aquest problema d'una manera còmoda.

3.2. Llibreries d'imatges per Java

Des de poc després de la creació d'aquest llenguatge al 1991, el llenguatge Java inclou en el seu paquet de desenvolupament una llibreria gràfica anomenada AWT (Avanced Windows Toolkit) que dóna funcionalitats per crear gràfics vectorials, crear interfícies gràfiques d'usuari i utilitzar imatges tipus *raster*, però només en format *gif* i *jpeg*.

La segona llibreria gràfica arriba amb Java 2 en la seva versió 1.2, s'anomena Java2D i deriva de l'antiga AWT. Afegeix funcionalitats geomètriques, de tractament del color, tractament de fonts de lletres i noves funcionalitats per treballar amb imatges *raster*. Les noves funcionalitats respecte a les imatges raster són:

- Provisió de funcions per fer transformacions geomètriques a les imatges.
- Provisió de funcions per controlar la brillantor i el contrasts de les imatges.

- Provisió de funcions per transformacions relacionades amb soroll en les imatges.

- Provisió de codecs per treballar amb els formats d'imatges més populars.

La tercera llibreria gràfica arriba per cobrir un dels elements pitjor resolts en Java, el tractament de les imatges *raster* i s'anomena JAI (Java Avanced Imaging).

En l'actualitat les aplicacions tenen la tendència a ser gràfiques, per no dir que la majoria del programari que es desenvolupa avui en dia està implementat en algun tipus d'interfície gràfica. Partint d'aquesta premissa i de l'evolució del maquinari, la inclusió de les imatges en l'utilització quotidiana dels ordinadors està cada vegada més estesa. Es per això que a Java li mancava una bona llibreria pel tractament de les imatges.

La llibreria JAI deriva de Java2D a la qual afegeix un gran nombre de noves funcionalitats orientades totalment al tractament de les imatges *raster*.

JAI és una llibreria que ha estat escrita pensant només en aquest tipus d'imatges, el que ha fet que el codi estigui optimitzat al màxim per tal d'aconseguir una llibreria molt robusta i amb una velocitat d'execució més que acceptable. A més a més, JAI encapsula les dades de la imatge de forma que permet fer-li modificacions amb una sèrie d'operacions molt senzilles possibilitant així el tractament de les imatges de forma molt còmoda.

3.3. La llibreria Java Avanced Imaging

La decisió d'agafar la llibreria JAI ha estat presa per diversos motius, a continuació enumerarem algunes de les funcionalitats que han fet prendre aquesta decisió, encara que això hagi fet que una part molt important del desenvolupament del treball s'hagi orientat en la recerca i documentació dels productes utilitzats, ja que com qualsevol producte nou, només està documentat pel fabricant, però no s'ha pogut trobar exemples que no fossin implementats pel propi fabricant el que ha fet que en moltes ocasions s'hagin

utilitzat funcions poc documentades i en cap cas exemplificades, el que suposa una pèrdua de temps molt més alta que en altres casos.

Característiques de la llibreria JAI que han afectat sobre la decisió:

- La majoria de llibreries gràfiques estan orientades a una plataforma concreta, JAI intenta oferir una plataforma independent del maquinari i del programari del sistema. JAI intenta que es puguin escriure aplicacions en una plataforma però poder-les distribuir en qualsevol plataforma.

- JAI implementa funcions per poder treballar amb arquitectures distribuïdes del tipus client-servidor mitjançant plataformes amb arquitectures de xarxa i tecnologies d'execució remota. Per implementar això utilitza Java RMI amb el qual es torna a garantir la compatibilitat.

- El fet d'utilitzar una llibreria totalment orientada a objectes es una de les premisses bàsiques per poder fer futures ampliacions.

- JAI suporta un gran nombre de formats d'imatges i permet estendre aquest nombre ja que implementa un sistema de codecs parametritzable.

- Permet, en el cas que es vulgui, crear codis optimitzats per plataformes concretes com l'MMX de Intel o VIS de Sun.

- Permet intercanviar dades amb la resta de API's de Java com Java3D i utilitzar els objectes de JAI totalment encapsulats.

4. L'estàndard JPEG

4.1. Què són els arxius JPEG

JPEG és l'acrònim "Join Photographic Experts Group", organització que ha desenvolupat aquest estàndard per a la compressió d'imatges. És un format propietari, però han aparegut d'altres organitzacions que han desenvolupat aplicacions que no utilitzen els mateixos algoritmes ni patents, però que són completament compatibles amb les especificacions estàndards, per exemple "Independent JPEG Group".

JPEG està dissenyat per treballar amb imatges fotogràfiques o artístiques, no donant tant bons resultats amb imatges de línies, de dibuixos senzills o bé de text.

Els arxius JPEG són els més utilitzats per a emmagatzemar imatges fotogràfiques degut a les seves característiques de relació entre la qualitat d'imatge i la compressió de l'arxiu.

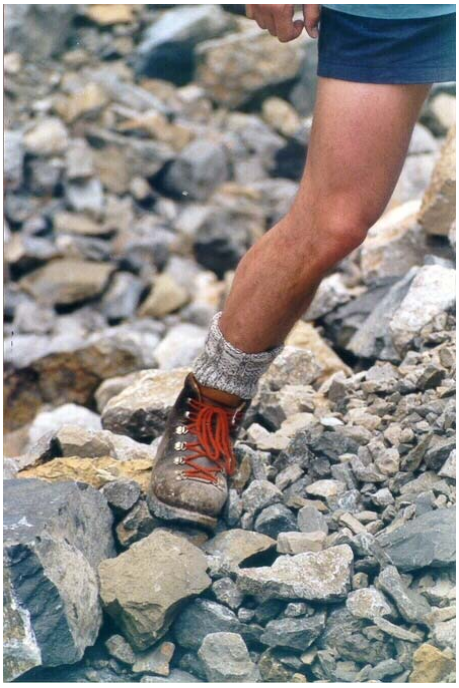


Figura 3

Figura 4

Les imatges anteriors s'han obtingut a partir d'una imatge BMP de resolució 447x 667 pixels i 24 bits de color. La mida original és de 909.942 bytes. La imatge de la figura 3 s'ha obtingut amb un factor de compressió de 95 i ocupa 167.253 bytes. Per la imatge de la figura 4 el factor de compressió és de 25 i la mida 29.169 bytes. En els dos casos la resolució continua sent de 447x 667 pixels i 24 bits de color.

4.2. Introducció a l'estàndard JPEG

L'estàndard JPEG per a imatges fixes de to continu (és tal i com està definit i tal com s'anomena) es fonamenta en les característiques d'aquest tipus d'imatges i del sistema visual humà. Concretament:

- Les imatges de to continu, (o naturals, en contra de les imatges sintètiques com ara plànols, text, etcètera) presenten degradacions suaus: hi ha una alta correlació entre el valor d'un pixel de la imatge i tots els que l'envolten, excepte quan hi ha un canvi brusc degut a dos objectes diferents, per exemple.
- La sensibilitat de l'ull humà a certes freqüències i contrastos no és uniforme, sinó que és variable.

L'estàndard JPEG combina aquests dos fets de manera que és possible reduir la quantitat d'informació necessària per a representar el senyal (la imatge) seleccionant tan sols aquella informació visual que és veritablement necessària sota un cert criteri de qualitat.

La qualitat de la imatge comprimida queda determinada per la quantitat de senyal eliminada en el procés de compressió. L'estàndard JPEG utilitza un procés de compressió amb pèrdua (*lossy*), de manera que la imatge comprimida no és idèntica a l'original sinó només una reconstrucció amb un cert grau de fidelitat. En general, un procés de compressió amb pèrdua està composat per les següents etapes:

- Preprocessament: la imatge és segmentada en blocs (8x8 en el cas de l'estàndard JPEG) i convertida a un model de color adient (YCbCr).

- Transformació: cada bloc és transformat de manera que l'energia de cada bloc (el senyal) es concentra en pocs coeficients (l'estàndard JPEG utilitza la transformada discreta del cosinus, o DCT).
- Quantificació (o quantització): els coeficients són discretitzats de manera que passen a tenir una representació finita (l'estàndard JPEG utilitza una quantificació escalar uniforme).
- Codificació: els valors generats en l'etapa anterior són agrupats de manera que s'intenta reduir el nombre de bits necessaris per a representar-los (l'estàndard JPEG permet tan la codificació aritmètica com la codificació Huffman).

A diferència del que succeeix en un procés de compressió sense pèrdua (*lossless*), tant la qualitat de la imatge com la raó de compressió (la raó entre la mida de la imatge original i la imatge comprimida) són variables, i queden determinades per les operacions realitzades en cadascuna de les etapes esmentades anteriorment. No obstant això, l'única etapa que admet ser fàcilment parametritzada és la de quantificació. En el cas concret de l'estàndard JPEG, cada coeficient dels 64 que genera la transformada discreta del cosinus d'un bloc de 8x8 és discretitzat mitjançant una quantificació uniforme, de manera independent de la resta de coeficients (d'aquí l'adjectiu escalar). Aquesta quantificació consisteix en dividir per un coeficient enter i quedar-se amb la part entera del resultat obtingut fent un arrodoniment.

Suposem per exemple que es quantifica utilitzant un valor Q=5. Aleshores, qualsevol valor entre –2. 5 i +2. 5 passa a valer 0, qualsevol valor entre +2. 5 i +7. 5 passa a valer +1, qualsevol valor entre –7. 5 i –2. 5 passa a valer –1, etcètera. La reconstrucció (o desquantificació) consisteix en multiplicar de nou pel valor Q, de manera que el valor +1 es reconstrueix per 5, per exemple. S'observa que tot un interval de valors dels coeficients originals com ara [2. 5, 7. 5) es converteix en un únic valor enter (+1, o 5 un cop desquantificat). Això provoca la pèrdua que de fet, és la que permet assolir una raó de compressió elevada.

Els sistemes de compressió sense pèrdua d'imatges fixes de to continu només permeten assolir raons de compressió properes de fins 5:1 (la imatge

comprimida ocupa 5 cops menys que la imatge original). En el cas concret de l'estàndard JPEG, és fàcil assolir raons de compressió al voltant de 15:1 sense una gran degradació de la qualitat de la imatge comprimida.com és possible? Doncs aprofitant que l'energia dels 64 pixels de cada bloc queda molt concentrada en uns pocs dels 64 coeficients, i aquests són tractats de manera que aquelles freqüències on la resposta del sistema visual humà és pitjor són més quantificades. La idea bàsica és no utilitzar cap bit per a representar informació present en la imatge original que l'ull humà no pot percebre.

Queda clar, per tant, que el procés de quantificació és crític, ja que determina tant la raó de compressió que s'aconsegueix com la qualitat de la imatge comprimida. En el cas de l'estàndard JPEG això s'aconsegueix mitjançant una matriu de quantificació predefinida per a cada component de la imatge i que conté els 64 valors Q utilitzats (un per cada coeficient). Avui dia encara no hi ha mecanismes que de manera senzilla permetin calcular matrius de quantificació adaptades a cada imatge (i de fet a cada observador). Manquen, doncs, eines que permetin comprimir una imatge donada amb una qualitat desitjada a partir de la generació de matrius de quantificació personalitzades.

Gràficament el procés que s'ha de seguir per a guardar una imatge en format JPEG és el que mostra la figura 5.

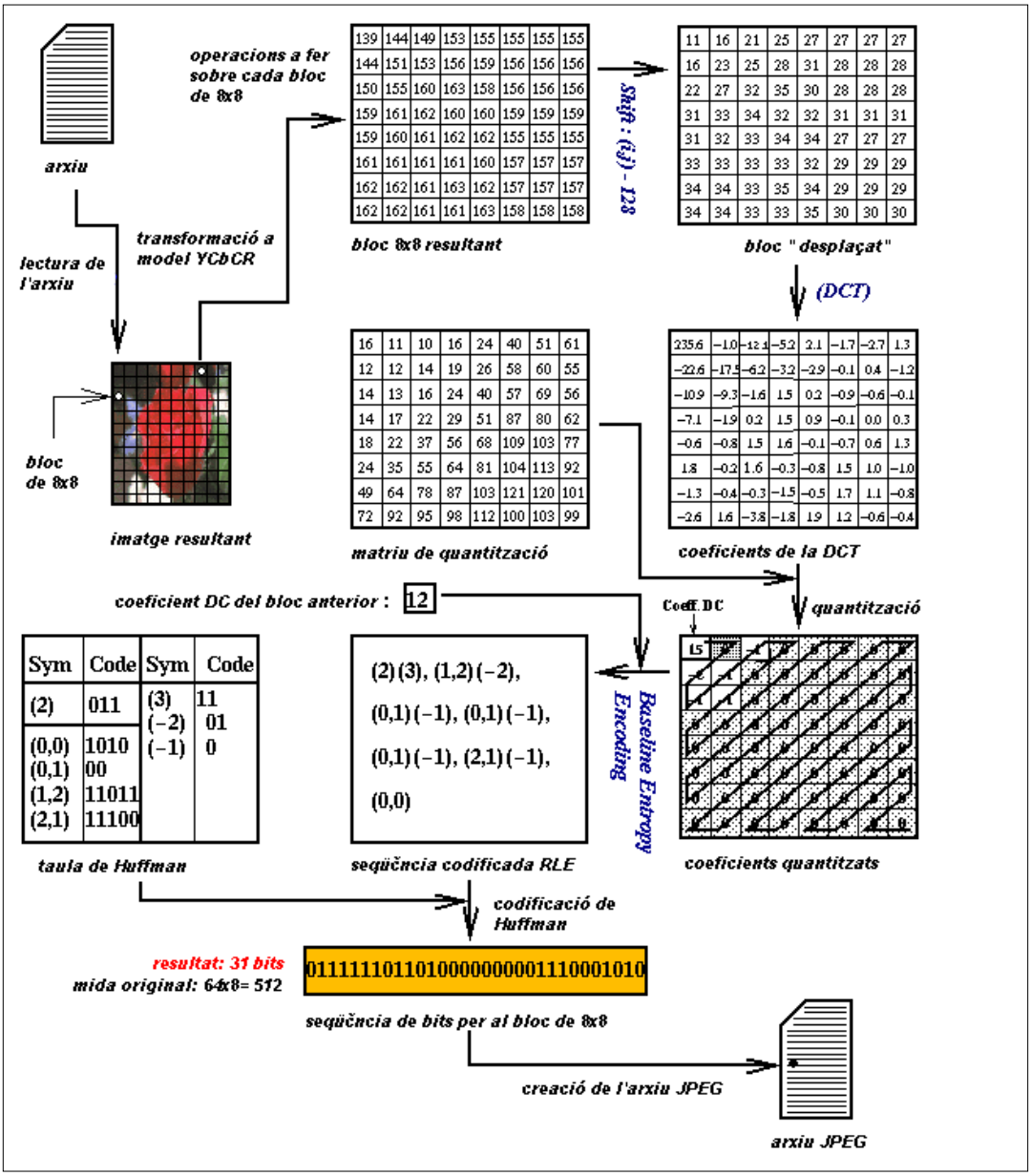


Figura 5. El procés de compressió dels arxius JPEG

4.3. Modes de compressió JPEG

L'estàndard JPEG defineix quatre modes de compressió d'arxius: compressió *hierarchical*, progressiva, seqüencial i sense pèrdues (*lossless*)

- La compressió seqüencial és la més utilitzada i codifica les imatges d'una sola passada i des del principi fins al final de l'arxiu.
- En el mode progressiu la imatge s'escaneja en successives passades, entre 2 i 896, refinant les dades a cada escanejada. A la descodificació es produeix l'efecte invers i es pot veure la millora de qualitat de la imatge a cada escaneig.
- El mode *hierarchical* és un mode super-progressiu on la imatge es descomposta en subimatges anomenades trames. La primera trama crea una versió de molt baixa qualitat i la resta refinen la imatge incrementant la resolució.
- El mode *lossless* simplement conserva la imatge original exacte i ha quedat obsolet.

4.4. La Transformada Discreta del Cosinus (DCT)

La Transformada Discreta del Cosinus dóna informació sobre la freqüència dels valors emmagatzemats als blocs de 8x8. Transforma un conjunt de valors d'entrada en un altre conjunt de valors de funcions cosinuidals amb freqüències creixents. Això permetrà eliminar aquelles components de freqüència per a les quals l'ull humà no té tanta sensibilitat, concretament les altes, sense modificar les components de baixa freqüència. La transformada inversa DCT no produeix una pèrdua significant d'informació, tret dels errors per arrodoniments efectuats durant la transformació directa.

Abans d'aplicar la DCT es fa una operació de desplaçament a l'esquerra de 128:

$I(i, j)=I(i, j) -128$

D'aquesta forma el rang de valors està comprés entre –128 i +128 en lloc d'entre 0 i 255.

La DCT s'aplicarà sobre aquesta matriu, que està subdividida en blocs de 8x8. En el cas que hi hagi files o columnes que no siguin múltiples de 8 s'hauran de completar amb zeros, ja que la DCT s'ha d'aplicar a blocs complerts de 8x8.

El primer coeficient obtingut amb la DCT es coneix amb el nom de coeficient DC, la resta són els coeficients AC. El coeficient DC mesura el valor mitjà dels 64 mostres de la imatge i els coeficients AC aporten informació sobre la freqüència. Ambdós coeficients es tractaran de diferent manera en la codificació JPEG.

4.5. Quantificació

L'objecte de la quantificació és aconseguir el màxim de compressió representant els components de la DCT amb el mínim de resolució possible tal que permeti obtenir la qualitat d'imatge desitjada.

Els valors de quantificació idealment han de ser triats de forma que s'adeqüin al llindar de percepció (el límit d'apreciació de la diferència entre imatges) ja que no té sentit fer diferenciacions de valors que després no puguin ésser apreciats. Aquest llindar depèn de les característiques de la imatge original, de les característiques dels dispositius de visualització i de la distància de visió.

Qualsevol bloc 8x8 al qual se li ha aplicat la DCT es quantificarà segons l'expressió:

$$F^{Q}(u, v)=round(F(u, v)/Q(u, v))$$

on $Q(u, v)$ és la matriu de quantificació i F(u, v) és la matriu dels coeficients de la DCT.

Escalant els valors de la matriu de quantificació es pot variar la quantitat d'informació eliminada de la imatge. La variació pot estar entre 1 (imatges de poca qualitat a causa de l'eliminació de molta informació) i 100 (imatges de gran qualitat amb poca pèrdua d'informació) Tot i això els valors es poden reduir al rang entre 25 i 95, ja que un factor de 25 ja dóna imatges de molt baixa qualitat i valors superiors a 95 produeixen imatges grans sense una millora apreciable de qualitat. Proporciona i actuarà de la següent forma sobre

la matriu de quantificació. L'escalat s'aplica a cada un dels coeficients de la matriu de quantificació *Q(u, v)*:

$Q^S(u, v)= (Q(u, v)* escalat+ 50) 100)$

$Q^S(u, v)$ és la nova matriu de quantificació 8x8 escalada.

A la fase de descodificació, la desquantificació els coeficients de la DCT es calculen segons:

$F^Q(u, v)=F^Q(u, v)\ Q(u, v)$

Aquest és el mètode que utilitzant la major part de les llibreries JPEG de l'IJG (Independent JPEG Group), que es poden trobar a Internet a l'adreça: http://www.ijg.org

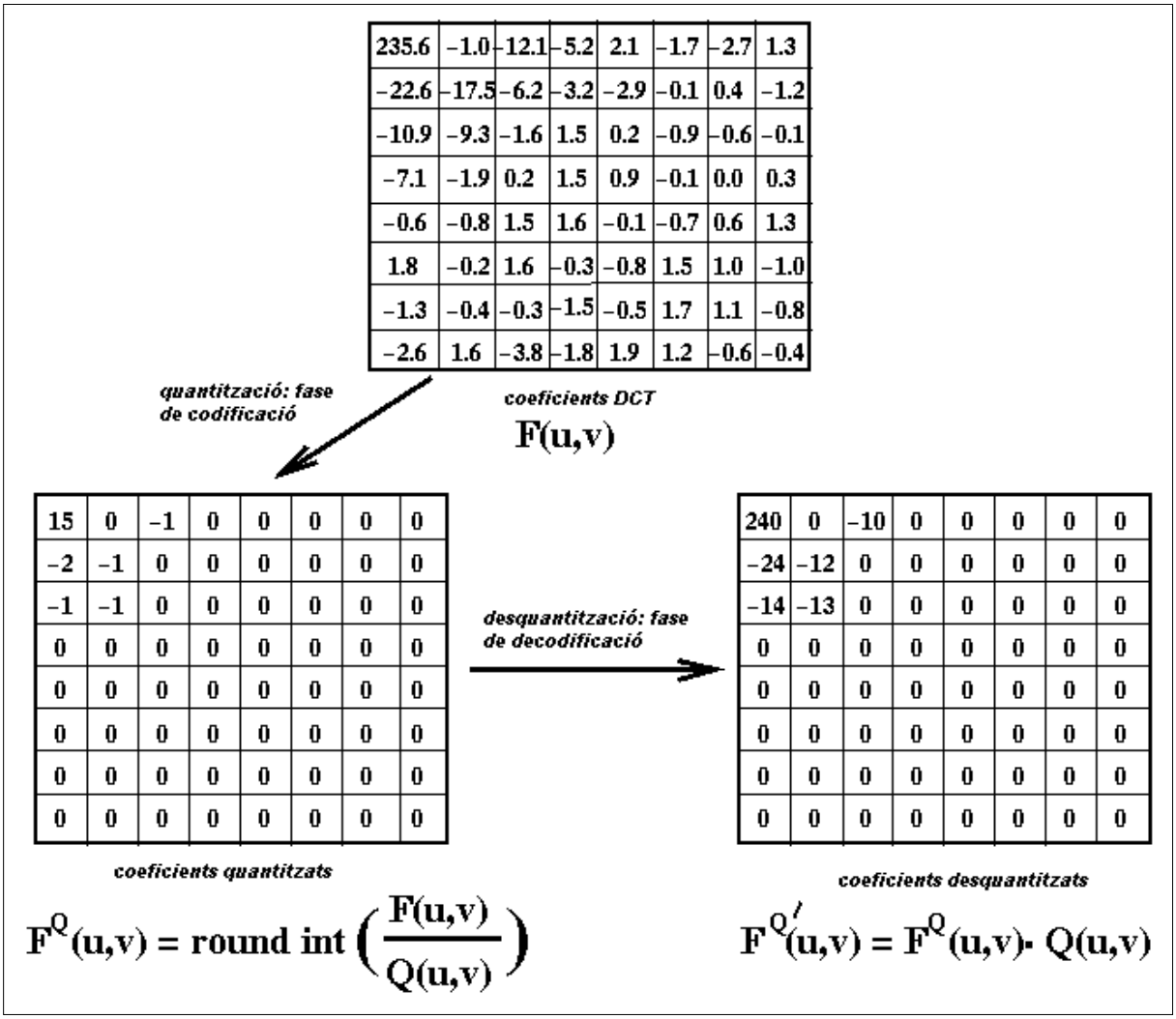


Figura 6. Quantificació dels coeficients DCT

4.6. Ordenació en Ziga-Zaga

Després de la quantificació apareixen molts coeficients AC amb valor igual a 0. Ens interessa tenir seqüències de zeros tant llargues com sigui possible per a aconseguir la màxima compressió en els processos posteriors. Per això s'ordenen seguint el recorregut en ziga-zaga (ZZ)

El coeficient DC és tractat de manera independent dels altres 63 coeficients.

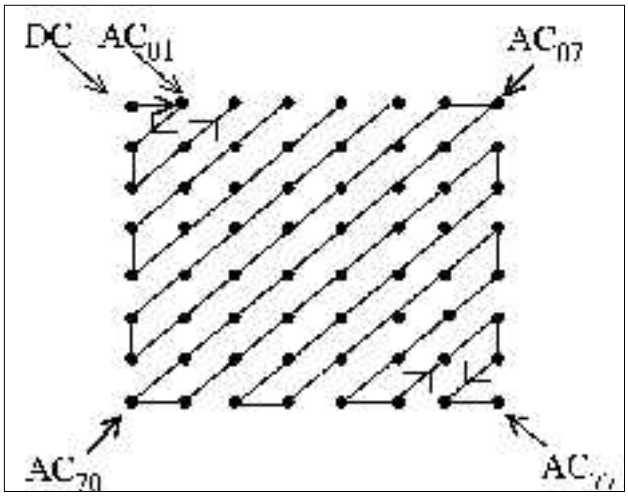


Figura 7. El recorregut de la seqüència ziga-zaga (ZZ)

Aquesta ordenació situa els coeficients que representen les baixes freqüències, i que tenen major probabilitat de tenir valors diferents de 0, abans dels coeficients d'alta freqüència. Aconseguim doncs tenir els valors propers a 0 junts, la qual cosa facilitarà l'eliminació de valors en el moment de la codificació Huffman.

4.7. L'algorisme PSNR2Q

Una part important d'aquest projecte es centrarà en el càlcul dels coeficients de les matrius de quantificació, tenint en compte la qualitat desitjada i el sistema visual humà. Per defecte s'utilitzaran matrius de quantificació que s'han elaborat empíricament i donen uns resultats acceptables en la majoria dels casos, tot i que no tenen en compte les peculiaritats i l'entorn d'un usuari en particular.

Per fer els càlculs dels coeficients de les matrius de quantificació se segueix l'algorisme PSNR2Q explicat a l'article *JPEG Standard Uniform Quantization Error Modeling with Applications to Sequential and Progressive Operation Modes*[6].

Bàsicament, aquest algorisme per a calcular la matriu de quantificació Q per a una imatge donada i un MSE desijat, nosaltres ho introduim mitjançant el PSNR donat la relació que hi ha entre ells

$$MSE = \frac{L^2}{10^{\frac{PSNR}{10}}} \qquad \Leftrightarrow \qquad PSNR = 10 * \log_{10}\left(\frac{L^2}{MSE}\right)$$

consisteix en tres etapes.

1. Es calculen els descriptors estadístics de la imatge, transformada utilitzant la DCT, que volem comprimir.

2. Es distribueix el MSE total desitjat entre cadascun dels coeficients MSEu,v del bloc de 8x8. Aquesta és l'etapa més important, per tal de distribuir el MSE total entre cada MSEu,v cal fixar-se que depén de la funció Φ[f(u,v)] i que a més a més s'ha de complir que $MSE_{u,v} \le \min\left\{{\sigma_{u,v}}^2, MSE(255, \sigma_{u,v})\right\}$. Intentarem saturar el màxim nombre de coeficients, es a dir, aquells que la desigualtat es converteix en

[6] Veure bibliografia

igualtat, i seguint la seqüència en zig-zag, començant pel valor 64 → u=8, v=8 aconseguirem obtenir el màxim nombre de zeros seguits possible, el que farà obtenir millor factor de compressió.

3. Es calcula cada valor Qu,v utilitzant el mode apropiat.

per al valor DC $\begin{cases} Q(MSE_{0,0}) = 1, & \text{si } MSE_{0,0} \leq 4.449257 \\ Q(MSE_{0,0}) = \dfrac{-a_1 + \sqrt{a_1^2 - 4a_2\sqrt{(a_0 - MSE_{0,0})}}}{2a_2}, & \text{en altre cas} \end{cases}$

per als valors AC es. $Q_{u,v}(MSE_{u,v}, \sigma_{u,v}) = \sigma\sqrt{2}F^{-1}\left(1 - \dfrac{MSE_{u,v}}{\sigma_{u,v}^{\ 2}}\right)$

4.8. Codificació Huffman pels arxius JPEG

Per a codificar els coeficients després de la quantificació s'usa primer una codificació del tipus Run Length Encoding, després de la qual es pot aplicar la codificació de Huffman. En aquest procés no es produeix pèrdua d'informació.

5. Implementació

5.1. Introducció

L'aplicació s'ha implementat en Java i la llibreria està escrita en C, en la implementació hi ha quatre mòduls en Java i dinou en C, dels quals en el TFC anterior de l'estudiant Daniel Fusté se'n van implementar els 17 de C i 2 en Java, ara s'han implementat dos més en Java i en C. Els mòduls de Java corresponen a les classes `JAIImageReader`, `tfc`, `tfcAjuda`, `tfcAboutBox`, alguns mòduls en C s'han rescrit per adapatar-los a les noves crides a la llibreria dinàmica i els 2 nous corresponen a les ampliacions per llegir altre tipus d'imatges.

❖ **Components Swing de Java**

Per fer l'aplicació en Java s'ha adaptat la interfície al nou model de control d'events, (model d'events per delegació) que s'utilitza des de la versió JDK 1.2, donat que els components Swing no soporten el model d'events de propagació (el de les primeres versions de Java) i s'han utilitzat aquest tipus de components (JLabel, JButton, JMenu,...) perquè proporcionen un conjunt complet de components escrits en Java que ja no utilitzen components "peer" dependents del sistema operatiu.

Entre els avantatges que ens proporciona l'ús de Swing podem destacar:

- La navegació amb el teclat és automàtica.
- Les etiquetes d'informació s'escriuen en una sola línia.
- És "plug look and feel ", es a dir, l'aparença de l'aplicació s'adapta dinàmicament al sistema operatiu i plataforma sobre la que estigui executant-se.

❖ Model d'events en Java

Els events, esdeveniments que afecten la interfície de l'usuari com ara pulsar una tecla, desplegar un menú, ..., en Java són informacions que es propaguen dins l'aplicació. Es representa amb tot un conjunt de classes *event*. Cada classe *event* és definida per les dades que representen un tipus d'event o un conjunt d'events relacionats, (per exemple, MouseEvent representa fer clic amb el ratolí, moure'l, ...).

Els *Listeners* o "escoltadors d'events" són objectes dins de l'aplicació que reben, tracten i propaguen events.

L'accés als atributs de les classes *event* es fa a través dels mètodes d'accés get<Atribut> i set<Atribut>(Valor)

❖ Descripció de les classes implementades amb Java:

- Classe `tfc`: És la classe principal en ella és presenta l'aplicació gràfica i es fan totes les crides necessàries pel funcionament de l'aplicació.

- Classe `JAIImageReader`: Llegeix una imatge del disc i la col·loca dins una estructura que permetrà fer operacions amb ella. Aquesta classe és una modificació d'una classe amb el mateix nom, implementada per Sun Microsystems. La classe original de Sun està lliure de royaltis tant per utilitzar-la com per fer-li modificacions.

- Classe `tfcAjuda`: És la classe que s'encarrega de gestionar l'ajuda de l'aplicació. Mostra una finestra amb les explicacions pertinents.

- Classe `tfcAboutBox`: És una altra classe auxiliar mostra un quadre de diàleg sobre l'aplicació.

❖ Descripció dels mòduls amb C:

- Mòdul `testjpegq.c`: implementa dues funcions que retornen els descriptors estadístics i la matriu de quantització respectivament. S'ha modificat per tal de treballar amb les 3 components.

- Mòdul `enllacJava.c`: implementa les funcions que permeten la exportació de les dades utilitzades per l'aplicació. Aquestes dades són: els descriptors estadístics[7], la matriu de quantització, el PSNR màxim[8], el PSNR mínim[9].

- Mòdul `llegirimatgetiff.c`: prepara les funcions que permetran la lectura d'imatges TIFF.

- Mòdul `llegirimatgetga.c`: prepara les funcions que permetran la lectura d'imatges TGA.

[7] La matriu de mitjanes i la matriu de desviacions obtingudes a partir de la imatge original.

[8] Valor màxim del PSNR de la imatge original.

[9] Valor màxim del PSNR de la imatge original.

5.2. Descripció de les classes en Java

❖ **Classe JAIImageReader**

La classe no conté camps tan sols conté un mètode estàtic.

Mètodes:

1. `ReadImage:` Mètode estàtic que llegeix una imatge a partir de nom amb que es guarda en el disc i la diposita en un objecte PlanarImage. El mètode busca el codec necessari per llegir la imatge i en cas que no el trobi dona un missatge d'error de codec no reconegut.

❖ **Classe tfc**

Mètodes:

1. `tfc`: És el constructor i inicialitza la situació en pantalla dels diferents elements que composen la part visual de l'aplicació. No té paràmetres.

2. `CarregaImatge`: Rep per paràmetre el nom d'un fitxer d'imatge. Obre aquest fitxer i el presenta per pantalla. També rep un paràmetre que indica si la imatge és original o comprimida i ubica la presentació d'ella en un lloc o un altre segons li correspongui.

3. `descriptors`: Crida a la funció nativa que calcula els descriptors estadísitics de la imatge. La matriu on es guardaran els descriptors MatriuMS és una matriu de 6x64 elements, 3 components per les mitjanes i 3 components per les desviacions.

4. `processarImatge`: Crida a la funció nativa que calcula la matriu Q a partir dels descriptors estadísitics, el PNSR entrat i la funció PHI. Després amb aquesta matriu Q generem la imatge comprimida i mostrem tant els valors de la matriu Q com la imatge comprimida.

5. `reprocessarImatge`: Llegeix els valors de Q a la pantalla, genera la imatge comprimida a partir d'aquests valors, i la mostra per pantalla

6. `omplirMatriuQ`: Omple els valors de la matriu Q que mostraré per pantalla amb els valor de la matriu Q calculada per la funció de llibreria calculamatriuQ.

7. `crearJPEG`: Calcula la nova imatge comprimida amb la matriu Q i la presenta per pantalla.

8. `crearOutput`: Crea una fitxer generic per poder crear un fitxer JPEG

9. `calculaPSNR`: Calcula el PSNR real amb els paràmetres actuals.

10. `calculaPsnrMax`: Calcula el PSNR maxim.

11. `calculaPsnrMin`: Calcula el PSNR minim.

12. `obtenirMSE`: Calcula el MSE d'un valor de la matriu Q amb una desviació (*sigma*) donada.

13. `gestorBotons`: Classe que s'encarrega de gestionar tots els events produits pels botons de l'aplicació. Determina quin és el botó que s'ha pulsat i fa els càlculs corresponents.

14. `gestorBarres`: Classe que s'encarrega de gestionar tots els events produits per la barra de desplaçament.

15. `ActionListener`: Tenim també mètodes que s'encarreguen de gestionar totes les accions produides pels menús, com són obrir el quadre de diàleg per carregar un fitxer, demanar ajuda, sortir de l'aplicació, etc.

16. Altres mètodes per construir els diferents panells gràfics, cadascun d'ells es crida per primera vegada quan inicialitzem el panell, creant-se i situant-se al lloc corresponent. Després mitjançant els mètodes `get` i `set` anirem modificant els valors.

5.3. Descripció dels mòduls en C

❖ **Mòdul testjpejq.c**

Funcions:

1. `retornaDes`: Funció que rep per paràmetre el nom d'un arxiu d'imatge, crida a les funcions necessàries per calcular els descriptors estadístics de la imatge i retorna una taula de dos dimensions on hi ha els descriptors.

❖ **Mòdul enllacJava.c**

Funcions:

1. `Java_tfc_calculamatriuD`: Funció que rep per paràmetre el nom d'un arxiu d'imatge, crida a la funció retornaDes i retorna l'array on hi ha els descriptors fent la conversió a estructures compatibles amb Java, prèviament.

2. `Java_tfc_calculamatriuQ`: Funció que rep una array d'objectes Java que contenen els descriptors estadístics, un valor corresponent al PSNR desitjat i un valor que indica quina funció *phi* es vol utilitzar, amb això calcula la matriu de quantització, converteix aquesta matriu a estructures compatibles amb Java i retorna la matriu.

3. `Java_tfc_MSEmaxim`: Funció que calcula el PSNR mínim (que correspon al MSE màxim). Rep per paràmetre la matriu de quantització i a partir d'aquí crida a les funcions necessàries per fer el càlcul del MSE màxim, després retorna el valor del PSNR corresponent a aquest MSE.

4. `Java_tfc_MSEmínim`: Funció que calcula el PSNR màxim (que correspon al MSE mínim). Rep per paràmetre la matriu de quantització i a partir d'aquí crida a les funcions necessàries per fer el càlcul del MSE mínim, després retorna el valor del PSNR corresponent a aquest MSE.

NOTA: Aquestes dues funcions han estat escrites també en Java i ja no són necessaries.

- **llegirimatgetiff.c**

Funcions:

1. `obtenirheaderTIFF:` Funció per llegir la capçalera dels arxius TIFF.
2. `obtenirdataTIFF:` Funció per llegir la imatge d'un arxiu TIFF.

- **llegirimatgetga.c**

Funcions:

1. `obtenirheaderTGA:` Funció per llegir la capçalera dels arxius TGA.
2. `obtenirdataTGA:` Funció per llegir la imatge d'un arxiu TGA.

6. Conclusions

L'aplicació implementada permet guardar una imatge en format JPEG exercint un control sobre diferents variables (com el PSNR, la funció *phi* i la matriu de quantització) i veient-ne els resultats obtinguts, modificar-ne uns altres per tal d'aconseguir el nivell de qualitat desitjat.

Per veure com afecta a la compressió d'una imatge el valor del PSNR, es pot veure l'annex E on es comparen una imatge original amb unes imatges comprimides amb diferent PSNR. En l'annex es pot comprovar com una variació petita en el PSNR pot suposar una millora de qualitat molt substancial, sense penalitzar el tamany de l'arxiu resultant.

També s'ha de concloure que, encara que, la matriu de quantització estigui composta de 64 valors la modificació d'un sol d'aquests valors pot aportar un canvi molt gran en el resultat final. Així doncs si comprimim la imatge lenna.bmp amb un PSNR 20 obtenim una imatge de qualitat molt baixa i que ocupa un tamany de 6.955 bytes. Si variem un sol coeficient de la matriu de quantització, concretament el (4,4), la imatge encara perd més qualitat però el tamany augmenta fins a 21.227 bytes. D'aquí se'n treuen algunes conclusions clares:

- La matriu de quantització determina la qualitat i el tamany de la imatge comprimida utilitzant cadascun dels seus elements.

- La tria d'una bona matriu de quantització influirà d'una manera determinant en el nostre resultat.

- La imatge més petita no sempre és la que es veu pitjor. Una bona relació entre els elements de la matriu de quantització, aconsegueixen imatges de poc tamany i amb una qualitat d'imatge acceptable.

6.1. Possibles ampliacions

- El sistema funciona ara amb totes les components, ja no cal modificar les crides fetes per l'aplicació Java i tampoc les crides a les rutines de C, però és necessari revisar el comportament d'aquestes rutines de la llibreria de C i estudiar els possibles errors.

- Lectura de més tipus d'arxiu d'imatge estandarditzat. El programa accepta imatges BMP[10] i algunes imatges PNM[11], concretament els formats PBM[12], PGM[13] i PPM[14]. S'ha prepara't el mòdul C per llegir imatges TIFF i TGA però no s'ha arribat a llegir l'arxiu del disc en cada cas. Cal parar atenció també a la lectura d'arxius que fa cada sistema operatiu, com a mínim s'hauria de poder llegir correctament els arxius des de Windows i Linux.

- Moltes operacions es podrien fer directament des de Java i així ens estalviariem la llibreria dinàmica. En aquest sentit ja s'ha escrit també en Java les operacions per calcular el PsnrReal, PsnrMaxim, PsnrMinim, i obtenirMSE. Però caldria reescriure els mòduls de C en Java que calculen la matriu Q i les matrius dels descriptors Mitjana i Desviacions si volem prescindir de la llibreria dinàmica.

- Cal investigar les noves possibilitats que ofereix la llibreria JAI pel que fa a l'obtenció dels descriptors, així com els mètodes per dibuixar la imatge per pantalla, donat que el mètode actual de pintar sobre un *ImageCanvas* serà eliminat en properes versions de Java.

[10] BitMap

[11] PortableNythingMap

[12] ProtableBitMap

[13] PortableGreyMap

[14] PortablePixMap

7. Bibliografia

- Agustín Froufe, "JAVA 2 Manual de usuario y tutorial", 2a edición, ed RA-MA, Madrid 2000

- Calvin Austin and Monica Pawlan, "Advanced Programming for the Java2™ Platform". http://developer.java.sun.com/developer/Books/j2ee/advancedprogramming/jni.pdf

- Daniel Fusté, "Implementació d'un sistema per al control de qualitat d'imatges JPEG (II)", memòria del Treball Final de Carrera, gener 2002.

- Francisco J. Menendez, "¿Como hacer una win32 DLL? http://www.fjmenendez.f2s.com/spa/htm/tut01/tut01.pdf

- Independent JPEG Group. *JPEG al W3C*: http://www.w3.org/Graphics/JPEG/jfif3. pdf

- Independent JPEG Group. *JPEG al W3C*: http://www.faqs.org/faqs/jpeg-faq/

- John Miano, "Compressed Image File Formats JPEG, PNG, GIF, XBM, BMP", Ed Addison- Wesley, January 2000.

- Joint Photographic Experts Group. http://www.jpeg.org/public/jpeghomepage.htm

- Jordi Pino, "Implementació d'un sistema per al control de qualitat d'imatges JPEG (I)", memòria del Treball Final de Carrera, juliol 2001.

- Julià Minguillón and Jaume Pujol, "JPEG standart uniform quantization error modeling with applications to sequential and progressive operation modes" Journal of Electrònic Imaging, april 2000, vol. 11.

- Sun Microsystems, "Programming in Java Advanced Imaging ": http://java.sun.com/products/java-media/jai/forDevelopers/jai1_0_1guide-unc/ index.html

- Tnt, "Image File Formats". http://www.tnt.uni-hannover.de/soft/imgproc/fileformats/

- "The Graphics File Formats Page" http://www.dcs.ed.ac.uk/home/mxr/gfx/2d-hi.html

Apendix A - Codi font dels mòduls escrits en C

testjpegq.c

```
/* TFC. Modificació de la implementació del sistema de control de qualitat
 * per a imatges JPEG implementat per l'estudiant Jordi Pino.
 * La modificació esmentada s'ha fet per tal d'enllaçar el codi anterior amb
 * una aplicació Java mitjançant Java Native Interface
 * Ampliació per Antoni Francés per tenir en compte totes les components.
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* totes les capçaleres juntes haurien d'anar en un unic .h
 * de fet en jpegq.h
 */
#include "jpegq.h"
#include "processarimatge.h"
#include "desprocessarimatge.h"
#include "imatgetransforma.h"
#include "blocsquantitza.h"
#include "desquantitzablocs.h"
#include "descriptorsestadistics.h"
#include "psnr2q.h"

void creaTaulaF(void);

float **retornarDes(char *nom)
{
  imatgeRAW *i;
  imatgeBLOCS *ib;
  descEstadistic *de;
  float **desc;
  int j;


  i = llegirimatge(nom);

  if (!i) {
    fprintf(stderr, "ERROR: couldn't open %s or invalid image\n", nom);
    exit(1);
  }

  ib = processarimatge(i);           /* convertir la imatge a una sequencia de blocs */
  ib = imatgetransforma(ib, TRUE);   /* aplicar la DCT */
  de = descriptorsestadistics(ib, FALSE);/* calcular els descriptors estadistics */
  desc =  (float **) malloc( 2*(de->C)*sizeof(float));

  for(j=0;j<de->C;j++){
      desc[2*j]=de->s[j];
      desc[2*j+1]=de->m[j];
  }

  return desc;
}                           /* fi testjpegq.c */
```

enllacJava.c

```
#include <jni.h>
#include <iostream.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

#include "tfc.h"
#include "jpegq.h"
#include "processarimatge.h"
#include "desprocessarimatge.h"
#include "imatgetransforma.h"
#include "blocsquantitza.h"
#include "desquantitzablocs.h"
#include "descriptorsestadistics.h"
#include "psnr2q.h"

float **retornarDes(char *);
float maximMSE(descEstadistic *, int, int);
float obtenirMSE(float , float);


JNIEXPORT jobjectArray
     JNICALL Java_tfc_calculamatriuD(
            JNIEnv *env, jobject jobj, jstring nom)
{

    float **desc;
    jboolean esCopia;
    jfloatArray columna;
    jobjectArray ret;
    int i;
    const char *nomC;

    nomC=(*env)->GetStringUTFChars(env, nom, &esCopia);

    //Calcul dels Descriptors
    desc=retornarDes( (char *)nomC );

    //Creacio d'un array intermig per retornar els valors
    columna = (jfloatArray)(*env)->NewFloatArray(env, DCTSIZE2);
    ret=(jobjectArray)(*env)->NewObjectArray(env, 2*3,
                                (*env)->GetObjectClass(env,columna),0);
                           // 2 variables (Mit i Des) x 3 components

    //Actualització dels valors l'array intermig
    for(i=0;i<2*3;i++)
    {
        columna= (jfloatArray)(*env)->NewFloatArray(env,DCTSIZE2);
        (*env)->SetFloatArrayRegion(env,(jfloatArray)columna,(jsize)0,
                                    DCTSIZE2,(jfloat *)desc[i]);
        (*env)->SetObjectArrayElement(env,ret,i,columna);
    }
    return(ret);

}


JNIEXPORT jobjectArray
     JNICALL Java_tfc_calculamatriuQ(
     JNIEnv *env, jobject jobj, jobjectArray matriuD, jfloat valpnsr, jint phi)
{
    float **Q;
    jfloatArray columna;
    jobjectArray ret;
    int i, j;
    descEstadistic *de;
    jfloatArray columna1;
    jfloatArray columna2;
    jfloat *element;

    // Reserva d'espai de memòria per l'estructura
    de = (descEstadistic *) malloc(sizeof(descEstadistic));
```

```
        // Ja està preparat per més d'una component.
        // Però repassar per si de cas.
    de->C=3;

    de->m = (float **) malloc(2*(de->C)*sizeof(float));
    de->s = (float **) malloc(2*(de->C)*sizeof(float));
    for(i=0; i< de->C; i++){
        de->m[i] = (float *) malloc(DCTSIZE2*sizeof(float));
        de->s[i] = (float *) malloc(DCTSIZE2*sizeof(float));
    }

    // Importació de les dades de Java
    for(j=0; j<de->C; j++){
        columna1 = (jfloatArray) (*env)->GetObjectArrayElement(env, matriuD, 2*j);
        element= (*env)->GetFloatArrayElements(env,columna1, 0);
        for(i=0; i<DCTSIZE2; i++)
                de->s[j][i]= element[i];
        (*env)->ReleaseFloatArrayElements(env, columna1, element,0);

        columna2 = (jfloatArray) (*env)->GetObjectArrayElement(env, matriuD, 2*j+1);
        element= (*env)->GetFloatArrayElements(env,columna2, 0);
        for(i=0; i<DCTSIZE2; i++)
               de->m[j][i]= element[i];
        (*env)->ReleaseFloatArrayElements(env, columna2, element,0);
    }

    //Calcul de la matriu Q
    Q=calculmatriuQ(de, valpnsr, phi);

    //Creacio d'un array intermig per retornar els valors
    columna = (jfloatArray)(*env)->NewFloatArray(env, DCTSIZE2);
    ret=(jobjectArray)(*env)->NewObjectArray(env, 3 ,(*env)-
>GetObjectClass(env,columna),0);
                        // Necessito 3 arrays, per retornar les 3 components

    //Actualització dels valors al array intermig
    for(i=0;i<de->C;i++)
    {
        columna= (jfloatArray)(*env)->NewFloatArray(env,DCTSIZE2);
        (*env)->SetFloatArrayRegion(env,(jfloatArray)columna,(jsize)0,DCTSIZE2,(jfloat
*)Q[i]);
        (*env)->SetObjectArrayElement(env,ret,i,columna);
    }

    return(ret);

}


JNIEXPORT jfloat
     JNICALL Java_tfc_MSEmaxim(
     JNIEnv *env, jobject jobj, jobjectArray matriuD)
{
    jfloat MSE=0.0;
    jfloatArray columna1;
    jfloatArray columna2;
    jfloat *element;
    int i,j;
    float aux;
    descEstadistic *de;


    // Reserva d'espai de memòria per l'estructura
    de = (descEstadistic *) malloc( sizeof(descEstadistic));

        // Ja està preparat per més d'una component.
        // Però repassar per si de cas.
    de->C=3;

    de->m = (float **) malloc(2*(de->C)*sizeof(float));
    de->s = (float **) malloc(2*(de->C)*sizeof(float));
    for(i=0; i< de->C; i++){
        de->m[i] = (float *) malloc(DCTSIZE2*sizeof(float));
        de->s[i] = (float *) malloc(DCTSIZE2*sizeof(float));
    }
```

```
    // Importació de les dades de Java
    for(j=0; j<de->C; j++){
        columna1 = (jfloatArray) (*env)->GetObjectArrayElement(env, matriuD, 2*j);
        element= (*env)->GetFloatArrayElements(env,columna1, 0);
        for(i=0; i<DCTSIZE2; i++)
                de->s[j][i]= element[i];
        (*env)->ReleaseFloatArrayElements(env, columna1, element,0);

        columna2 = (jfloatArray) (*env)->GetObjectArrayElement(env, matriuD, 2*j+1);
        element= (*env)->GetFloatArrayElements(env,columna2, 0);
        for(i=0; i<DCTSIZE2; i++)
                de->m[j][i]= element[i];
        (*env)->ReleaseFloatArrayElements(env, columna2, element,0);
    }
                // Calcul del màxim MSE de totes les components
    for( i=0, MSE=0.0; i< de->C ; i++)
        for( j=0; j< DCTSIZE2; j++){
                aux = maximMSE(de, i, j);
                if( MSE > aux)
                    MSE = aux;
        }
    // Retorna el mínim PSNR
    return (float) (10.0*log( (255.0*255.0 )/MSE) );
}

JNIEXPORT jfloat
     JNICALL Java_tfc_MSEminim(
     JNIEnv *env, jobject jobj, jobjectArray matriuD)
{

    jfloat MSE;
    jfloatArray columna1;
    jfloatArray columna2;
    jfloat *element;
    int i,j;
    float aux;
    descEstadistic *de;

    // Reserva d'espai de memòria per l'estructura
    de = (descEstadistic *) malloc( sizeof(descEstadistic));

        // Ja està preparat per més d'una component.
    de->C=3;

    de->m = (float **) malloc(2*(de->C)*sizeof(float));
    de->s = (float **) malloc(2*(de->C)*sizeof(float));
    for(i=0; i< de->C; i++){
        de->m[i] = (float *) malloc(DCTSIZE2*sizeof(float));
        de->s[i] = (float *) malloc(DCTSIZE2*sizeof(float));
    }

    // Importació de les dades de Java
    for(j=0; j<de->C; j++){
        columna1 = (jfloatArray) (*env)->GetObjectArrayElement(env, matriuD, 2*j);
        element= (*env)->GetFloatArrayElements(env,columna1, 0);
        for(i=0; i<DCTSIZE2; i++)
                de->s[j][i]= element[i];
        (*env)->ReleaseFloatArrayElements(env, columna1, element,0);

        columna2 = (jfloatArray) (*env)->GetObjectArrayElement(env, matriuD, 2*j+1);
        element= (*env)->GetFloatArrayElements(env,columna2, 0);
        for(i=0; i<DCTSIZE2; i++)
                de->m[j][i]= element[i];
        (*env)->ReleaseFloatArrayElements(env, columna2, element,0);
    }
                                            // Calcul del minim MSE
    for(i=0, MSE=1000.0; i< de->C; i++){
        aux = minimMSE(de, i);
        if ( MSE > aux )
            MSE = aux;
    }
                                            // Retorna el maxim PSNR
    return (float) (10.0*log( (255.0*255.0)/MSE));
}
```

llegirimatgetiff.c

```
/* TFC.
* Aquest arxiu i el corresponent de capçalera s'incorpora per tenir l'estructura
* preparada per llegir aquest tipus d'arxiu, però queda com una ampliació el
* llegir correctament aquest tipus d'imatge.
*

#include <stdlib.h>
#include <stdio.h>

#include "estructures.h"
#include "tiffheader.h"

int obtenirheaderTIFF(FILE *fp , imatgeRAW *i);
int obtenirdataTIFF(FILE *fp , imatgeRAW *i);
```

llegirimatgetga.c

```
/* TFC.
* Aquest arxiu i el corresponent de capçalera s'incorpora per tenir l'estructura
* preparada per llegir aquest tipus d'arxiu, però queda com una ampliació el
* llegir correctament aquest tipus d'imatge.
*

#include <stdlib.h>
#include <stdio.h>

#include "estructures.h"
#include "tgafheader.h"

int obtenirheaderTGA(FILE *fp , imatgeRAW *i);
int obtenirdataTGA(FILE *fp , imatgeRAW *i);
```

Annex B - Codi font dels mòduls escrits en Java

JAIImageReader.java

```
import java.awt.image.RenderedImage;
import javax.media.jai.JAI;
import javax.media.jai.PlanarImage;
import com.sun.media.jai.codec.ImageCodec;
import com.sun.media.jai.codec.ImageDecoder;
import com.sun.media.jai.codec.FileSeekableStream;
import com.sun.media.jai.codec.SeekableStream;

public class JAIImageReader
{
    public static PlanarImage readImage(String filename)
    {
        // Use the JAI API unless JAI_IMAGE_READER_USE_CODECS is set
        if (System.getProperty("JAI_IMAGE_READER_USE_CODECS") == null) {
            return JAI.create("fileload", filename);
        }
        else
        {
            try {
                // Use the ImageCodec APIs
                SeekableStream stream = new FileSeekableStream(filename);
                String[] names = ImageCodec.getDecoderNames(stream);
                ImageDecoder dec =
                ImageCodec.createImageDecoder(names[0], stream, null);
                RenderedImage im = dec.decodeAsRenderedImage();
                return PlanarImage.wrapRenderedImage(im);
            }
            catch (Exception e)
            {
                e.printStackTrace();
                System.out.println("Error al llegir el fitxer imatge!");
                return null;
            }
        }
    }
}
```

tfcAjuda.java

```
/*****************************************************
*
*      Classe per mostrar un quadre de Diàleg amb ajuda.
*
*****************************************************/

import javax.swing.*;
import java.awt.*;

class tfcAjuda extends JDialog {
   private JPanel ivjButtonPane = null;
   private JLabel ivjFila1 = null;
   private JLabel ivjFila2 = null;
   private JLabel ivjFila3 = null;
   private JLabel ivjFila4 = null;
   private JLabel ivjFila5 = null;
   private JLabel ivjFila6 = null;
   private JLabel ivjFila7 = null;
   private JLabel ivjFila8 = null;
   private JLabel ivjFila9 = null;
   private JLabel ivjFila10 = null;

   IvjEventHandler ivjEventHandler = new IvjEventHandler();
   private JLabel ivjIconLabel = null;
   private JPanel ivjIconPane = null;
   private JPanel ivjJDialogContentPane = null;
   private JButton ivjOkButton = null;
```

```
    private JPanel ivjTextPane = null;
    private GridLayout ivjTextPaneGridLayout = null;
class IvjEventHandler implements java.awt.event.ActionListener {
        public void actionPerformed(java.awt.event.ActionEvent e) {
            if (e.getSource() == tfcAjuda.this.getOkButton())
                sortirAjuda(e);
        };
    };
/**
 * tfcAjuda constructor comment.
 */
public tfcAjuda() {
    super();
    initialize();
}
/**
 * tfcAjuda constructor comment.
 * @param owner java.awt.Dialog
 */
public tfcAjuda(Dialog owner) {
    super(owner);
}
/**
 * tfcAjuda constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 */
public tfcAjuda(Dialog owner, String title) {
    super(owner, title);
}
/**
 * tfcAjuda constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 * @param modal boolean
 */
public tfcAjuda(Dialog owner, String title, boolean modal) {
    super(owner, title, modal);
}
/**
 * tfcAjuda constructor comment.
 * @param owner java.awt.Dialog
 * @param modal boolean
 */
public tfcAjuda(Dialog owner, boolean modal) {
    super(owner, modal);
}
/**
 * tfcAjuda constructor comment.
 * @param owner java.awt.Frame
 */
public tfcAjuda(Frame owner) {
    super(owner);
}
/**
 * tfcAjuda constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 */
public tfcAjuda(Frame owner, String title) {
    super(owner, title);
}
/**
 * tfcAjuda constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 * @param modal boolean
 */
public tfcAjuda(Frame owner, String title, boolean modal) {
    super(owner, title, modal);
}
/**
 * tfcAjuda constructor comment.
 * @param owner java.awt.Frame
 * @param modal boolean
 */
```

```
public tfcAjuda(Frame owner, boolean modal) {
    super(owner, modal);  }
/**
 * sortirAjuda:  (OkButton.action.actionPerformed(java.awt.event.ActionEvent) -->
tfcAjuda.dispose()V)
 * @param arg1 java.awt.event.ActionEvent
 */
private void sortirAjuda(java.awt.event.ActionEvent arg1) {
    try {
        this.dispose();
    } catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
    }
}
/**
 * Return the ButtonPane property value.
 * @return javax.swing.JPanel
 */
private javax.swing.JPanel getButtonPane() {
    if (ivjButtonPane == null) {
        try {
            ivjButtonPane = new javax.swing.JPanel();
            ivjButtonPane.setName("ButtonPane");
            ivjButtonPane.setLayout(new java.awt.FlowLayout());
            getButtonPane().add(getOkButton(), getOkButton().getName());
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjButtonPane;
}
/**
 * Return the Fila1 property value.
 * @return javax.swing.JLabel
 */
private javax.swing.JLabel getFila1() {
   if (ivjFila1 == null) {
      try {
         ivjFila1 = new javax.swing.JLabel();
         ivjFila1.setName("Fila1");
         ivjFila1.setText("   Menu ARXIU:  Opcions per llegir i escriure una imatge
                           a/des d'un fitxer.");
      } catch (java.lang.Throwable ivjExc) {
         handleException(ivjExc);
      }
   }
   return ivjFila1;
}
/**
 * Return the Fila2 property value.
 * @return javax.swing.JLabel
 */
private javax.swing.JLabel getFila2() {
   if (ivjFila2 == null) {
      try {
         ivjFila2 = new javax.swing.JLabel();
         ivjFila2.setName("Fila2");
         ivjFila2.setText("   Menu AJUDA: Presenta informació sobre l'aplicació i el seu
                           desenvolupador.");
      } catch (java.lang.Throwable ivjExc) {
         handleException(ivjExc);
      }
   }
   return ivjFila2;
}

/**
 * Return the Fila3 property value.
 * @return javax.swing.JLabel
 */
private javax.swing.JLabel getFila3() {
   if (ivjFila3 == null) {
      try {
         ivjFila3 = new javax.swing.JLabel();
         ivjFila3.setName("Fila3");
```

```
            ivjFila3.setText("   Matriu Q: Mostra la matriu de Quantificació utilitzada en
                                 cada component. Es poden modificar els seus valors. ");
         } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
         }
      }
      return ivjFila3;
}
/**
 * Return the Fila4 property value.
 * @return javax.swing.JLabel
 */
private javax.swing.JLabel getFila4() {
      if (ivjFila4 == null) {
         try {
            ivjFila4 = new javax.swing.JLabel();
            ivjFila4.setName("Fila4");
            ivjFila4.setText("   Boto REPROCESSAR: Torna a calcular la imatge comprimida a
                                 partir dels nous valors de la matriu Q.");
         } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
         }
      }
      return ivjFila4;
}
/**
 * Return the Fila5 property value.
 * @return javax.swing.JLabel
 */
private javax.swing.JLabel getFila5() {
      if (ivjFila5 == null) {
         try {
            ivjFila5 = new javax.swing.JLabel();
            ivjFila5.setName("Fila5");
            ivjFila5.setText("   Matriu M: Mostra la matriu de Mitjanes resultant en cada
                                 component.");
         } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
         }
      }
      return ivjFila5;
}
/**
 * Return the Fila6 property value.
 * @return javax.swing.JLabel
 */
private javax.swing.JLabel getFila6() {
      if (ivjFila6 == null) {
         try {
            ivjFila6 = new javax.swing.JLabel();
            ivjFila6.setName("Fila6");
            ivjFila6.setText("   Matriu D: Mostra la matriu de Desviacions resultant en
                                 cada component.");
         } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
         }
      }
      return ivjFila6;
}
/**
 * Return the Fila7 property value.
 * @return javax.swing.JLabel
 */
private javax.swing.JLabel getFila7() {
      if (ivjFila7 == null) {
         try {
            ivjFila7 = new javax.swing.JLabel();
            ivjFila7.setName("Fila7");
            ivjFila7.setText("   Funció PHI: Permet escollir la funció que s'utilitzarà per
                                 calcular la matriu Q.");
         } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
         }
      }
      return ivjFila7;
}
```

```
/**
 * Return the Fila8 property value.
 * @return javax.swing.JLabel
 */
private javax.swing.JLabel getFila8() {
   if (ivjFila8 == null) {
      try {
         ivjFila8 = new javax.swing.JLabel();
         ivjFila8.setName("Fila8");
         ivjFila8.setText("   PSNR: Paràmetre que modula la matriu de quantificació. Més
                           alt millor qualitat, més baix més compressió.");
      } catch (java.lang.Throwable ivjExc) {
         handleException(ivjExc);
      }
   }
   return ivjFila8;
}
/**
 * Return the Fila9 property value.
 * @return javax.swing.JLabel
 */
private javax.swing.JLabel getFila9() {
   if (ivjFila9 == null) {
      try {
         ivjFila9 = new javax.swing.JLabel();
         ivjFila9.setName("Fila9");
         ivjFila9.setText("   PSNR real: Valor real després de fer la Quantització.");
      } catch (java.lang.Throwable ivjExc) {
         handleException(ivjExc);
      }
   }
   return ivjFila9;
}
/**
 * Return the Fila10 property value.
 * @return javax.swing.JLabel
 */
private javax.swing.JLabel getFila10() {
   if (ivjFila10 == null) {
      try {
         ivjFila10 = new javax.swing.JLabel();
         ivjFila10.setName("Fila10");
         ivjFila10.setText("   Boto PROCESSAR: Recalcula la matriu Q, els descriptors i
                           la imatge comprimida amb els paràmetres actuals.");
      } catch (java.lang.Throwable ivjExc) {
         handleException(ivjExc);
      }
   }
   return ivjFila10;
}
/**
 * Return the IconLabel property value.
 * @return javax.swing.JLabel
 */
private javax.swing.JLabel getIconLabel() {
    if (ivjIconLabel == null) {
        try {
            ivjIconLabel = new javax.swing.JLabel();
            ivjIconLabel.setName("IconLabel");
            ivjIconLabel.setIcon(new
                javax.swing.ImageIcon(getClass().getResource("jpeg.gif")));
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjIconLabel;
}
/**
 * Return the IconPane property value.
 * @return javax.swing.JPanel
 */
private javax.swing.JPanel getIconPane() {
    if (ivjIconPane == null) {
        try {
            ivjIconPane = new javax.swing.JPanel();
            ivjIconPane.setName("IconPane");
```

```
            ivjIconPane.setLayout(new java.awt.FlowLayout());
            getIconPane().add(getIconLabel(), getIconLabel().getName());
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjIconPane;
}
/**
 * Return the JDialogContentPane property value.
 * @return javax.swing.JPanel
 */
private javax.swing.JPanel getJDialogContentPane() {
    if (ivjJDialogContentPane == null) {
        try {
            ivjJDialogContentPane = new javax.swing.JPanel();
            ivjJDialogContentPane.setName("JDialogContentPane");
            ivjJDialogContentPane.setLayout(new java.awt.BorderLayout());
            getJDialogContentPane().add(getButtonPane(), "South");
            getJDialogContentPane().add(getTextPane(), "Center");
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjJDialogContentPane;
}
/**
 * Return the OkButton property value.
 * @return javax.swing.JButton
 */
private javax.swing.JButton getOkButton() {
    if (ivjOkButton == null) {
        try {
            ivjOkButton = new javax.swing.JButton();
            ivjOkButton.setName("OkButton");
            ivjOkButton.setText("OK");
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjOkButton;
}

/**
 * Return the TextPane property value.
 * @return javax.swing.JPanel
 */
private javax.swing.JPanel getTextPane() {
    if (ivjTextPane == null) {
        try {
            ivjTextPane = new javax.swing.JPanel();
            ivjTextPane.setName("TextPane");
            ivjTextPane.setLayout(getTextPaneGridLayout());
            getTextPane().add(getFila1(), getFila1().getName());
            getTextPane().add(getFila2(), getFila2().getName());
            getTextPane().add(getFila3(), getFila3().getName());
            getTextPane().add(getFila4(), getFila4().getName());
            getTextPane().add(getFila5(), getFila5().getName());
            getTextPane().add(getFila6(), getFila6().getName());
            getTextPane().add(getFila7(), getFila7().getName());
            getTextPane().add(getFila8(), getFila8().getName());
            getTextPane().add(getFila9(), getFila9().getName());
            getTextPane().add(getFila10(), getFila10().getName());
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjTextPane;
}
/**
 * Return the TextPaneGridLayout property value.
 * @return java.awt.GridLayout
 */
private java.awt.GridLayout getTextPaneGridLayout() {
    java.awt.GridLayout ivjTextPaneGridLayout = null;
    try {
```

```
        /* Creació de les parts. Disposo de 10 files i 1 columnes */
        ivjTextPaneGridLayout = new java.awt.GridLayout(10,1);
    } catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
    };
    return ivjTextPaneGridLayout;
}
/**
 * Called whenever the part throws an exception.
 * @param exception java.lang.Throwable
 */
private void handleException(java.lang.Throwable exception) {

    /* Uncomment the following lines to print uncaught exceptions to stdout */
    // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
    // exception.printStackTrace(System.out);
}
/**
 * Initializes connections
 * @exception java.lang.Exception The exception description.
 */
private void initConnections() throws java.lang.Exception {
    getOkButton().addActionListener(ivjEventHandler);
}
/**
 * Initialize the class.
 */
private void initialize() {
    try {
        setName("IndexAjuda");
        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        setSize(550, 300);
        setTitle("Ajuda sobre aquesta aplicació.");
        setContentPane(getJDialogContentPane());
        initConnections();
    } catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
    }
}

}   /*---- Final de la Classe Ajuda ------*/
```

tfcAboutBox.java

```
/*****************************************************
 *
 *      Classe per mostrar un quadre de Diàleg.
 *
 *****************************************************/

 import javax.swing.*;
 import java.awt.*;

 class tfcAboutBox extends JDialog {
    private JLabel ivjAppName = null;
    private JPanel ivjButtonPane = null;
    private JLabel ivjCopyright = null;
    private IvjEventHandler ivjEventHandler = new IvjEventHandler();
    private JLabel ivjIconLabel = null;
    private JPanel ivjIconPane = null;
    private JPanel ivjJDialogContentPane = null;
    private JButton ivjOkButton = null;
    private JLabel ivjSpacer = null;
    private JPanel ivjTextPane = null;
    private GridLayout ivjTextPaneGridLayout = null;
    private JLabel ivjUserName = null;
    private JLabel ivjVersion = null;

class IvjEventHandler implements java.awt.event.ActionListener {
        public void actionPerformed(java.awt.event.ActionEvent e) {
```

```
            if (e.getSource() == tfcAboutBox.this.getOkButton())
                sortirAplicacio(e);
        };
    };
/**
 * tfcAboutBox constructor comment.
 */
public tfcAboutBox() {
    super();
    initialize();
}
/**
 * tfcAboutBox constructor comment.
 * @param owner java.awt.Dialog
 */
public tfcAboutBox(Dialog owner) {
    super(owner);
}
/**
 * tfcAboutBox constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 */
public tfcAboutBox(Dialog owner, String title) {
    super(owner, title);
}
/**
 * tfcAboutBox constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 * @param modal boolean
 */
public tfcAboutBox(Dialog owner, String title, boolean modal) {
    super(owner, title, modal);
}
/**
 * tfcAboutBox constructor comment.
 * @param owner java.awt.Dialog
 * @param modal boolean
 */
public tfcAboutBox(Dialog owner, boolean modal) {
    super(owner, modal);
}
/**
 * tfcAboutBox constructor comment.
 * @param owner java.awt.Frame
 */
public tfcAboutBox(Frame owner) {
    super(owner);
}
/**
 * tfcAboutBox constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 */
public tfcAboutBox(Frame owner, String title) {
    super(owner, title);
}
/**
 * tfcAboutBox constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 * @param modal boolean
 */
public tfcAboutBox(Frame owner, String title, boolean modal) {
    super(owner, title, modal);
}
/**
 * tfcAboutBox constructor comment.
 * @param owner java.awt.Frame
 * @param modal boolean
 */
public tfcAboutBox(Frame owner, boolean modal) {
    super(owner, modal);
}
/**
```

```
 * sortirAplicacio:  (OkButton.action.actionPerformed(java.awt.event.ActionEvent) -->
tfcAboutBox.dispose()V)
 * @param arg1 java.awt.event.ActionEvent
 */
private void sortirAplicacio(java.awt.event.ActionEvent arg1) {
    try {
        this.dispose();
    } catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
    }
}
/**
 * Return the AppName property value.
 * @return javax.swing.JLabel
 */
private javax.swing.JLabel getAppName() {
    if (ivjAppName == null) {
        try {
            ivjAppName = new javax.swing.JLabel();
            ivjAppName.setName("AppName");
            ivjAppName.setText("UOC - Treball Final de Carrera");
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjAppName;
}
/**
 * Return the ButtonPane property value.
 * @return javax.swing.JPanel
 */
private javax.swing.JPanel getButtonPane() {
    if (ivjButtonPane == null) {
        try {
            ivjButtonPane = new javax.swing.JPanel();
            ivjButtonPane.setName("ButtonPane");
            ivjButtonPane.setLayout(new java.awt.FlowLayout());
            getButtonPane().add(getOkButton(), getOkButton().getName());
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjButtonPane;
}
/**
 * Return the Copyright property value.
 * @return javax.swing.JLabel
 */
private javax.swing.JLabel getCopyright() {
    if (ivjCopyright == null) {
        try {
            ivjCopyright = new javax.swing.JLabel();
            ivjCopyright.setName("Copyright");
            ivjCopyright.setText("(c) Juny 2002");
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjCopyright;
}
/**
 * Return the IconLabel property value.
 * @return javax.swing.JLabel
 */
private javax.swing.JLabel getIconLabel() {
    if (ivjIconLabel == null) {
        try {
            ivjIconLabel = new javax.swing.JLabel();
            ivjIconLabel.setName("IconLabel");
            ivjIconLabel.setIcon(new
            javax.swing.ImageIcon(getClass().getResource("jpeg.gif")));
            ivjIconLabel.setText("");
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
```

```
    return ivjIconLabel;
}
/**
 * Return the IconPane property value.
 * @return javax.swing.JPanel
 */
private javax.swing.JPanel getIconPane() {
    if (ivjIconPane == null) {
        try {
            ivjIconPane = new javax.swing.JPanel();
            ivjIconPane.setName("IconPane");
            ivjIconPane.setLayout(new java.awt.FlowLayout());
            getIconPane().add(getIconLabel(), getIconLabel().getName());
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjIconPane;
}
/**
 * Return the JDialogContentPane property value.
 * @return javax.swing.JPanel
 */
private javax.swing.JPanel getJDialogContentPane() {
    if (ivjJDialogContentPane == null) {
        try {
            ivjJDialogContentPane = new javax.swing.JPanel();
            ivjJDialogContentPane.setName("JDialogContentPane");
            ivjJDialogContentPane.setLayout(new java.awt.BorderLayout());
            getJDialogContentPane().add(getButtonPane(), "South");
            getJDialogContentPane().add(getTextPane(), "Center");
            getJDialogContentPane().add(getIconPane(), "West");
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjJDialogContentPane;
}
/**
 * Return the OkButton property value.
 * @return javax.swing.JButton
 */
private javax.swing.JButton getOkButton() {
    if (ivjOkButton == null) {
        try {
            ivjOkButton = new javax.swing.JButton();
            ivjOkButton.setName("OkButton");
            ivjOkButton.setText("OK");
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjOkButton;
}
/**
 * Return the Spacer property value.
 * @return javax.swing.JLabel
 */
private javax.swing.JLabel getSpacer() {
    if (ivjSpacer == null) {
        try {
            ivjSpacer = new javax.swing.JLabel();
            ivjSpacer.setName("Spacer");
            ivjSpacer.setText("");
            ivjSpacer.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjSpacer;
}
/**
 * Return the TextPane property value.
 * @return javax.swing.JPanel
 */
private javax.swing.JPanel getTextPane() {
```

```
    if (ivjTextPane == null) {
        try {
            ivjTextPane = new javax.swing.JPanel();
            ivjTextPane.setName("TextPane");
            ivjTextPane.setLayout(getTextPaneGridLayout());
            getTextPane().add(getAppName(), getAppName().getName());
            getTextPane().add(getVersion(), getVersion().getName());
            getTextPane().add(getSpacer(), getSpacer().getName());
            getTextPane().add(getCopyright(), getCopyright().getName());
            getTextPane().add(getUserName(), getUserName().getName());
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjTextPane;
}
/**
 * Return the TextPaneGridLayout property value.
 * @return java.awt.GridLayout
 */
private java.awt.GridLayout getTextPaneGridLayout() {
    java.awt.GridLayout ivjTextPaneGridLayout = null;
    try {
        /* Create part */
        ivjTextPaneGridLayout = new java.awt.GridLayout(5, 1);
    } catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
    };
    return ivjTextPaneGridLayout;
}
/**
 * Return the UserName property value.
 * @return javax.swing.JLabel
 */
private javax.swing.JLabel getUserName() {
    if (ivjUserName == null) {
        try {
            ivjUserName = new javax.swing.JLabel();
            ivjUserName.setName("UserName");
            ivjUserName.setText("Antoni Francés Conejero");
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjUserName;
}
/**
 * Return the Version property value.
 * @return javax.swing.JLabel
 */
private javax.swing.JLabel getVersion() {
    if (ivjVersion == null) {
        try {
            ivjVersion = new javax.swing.JLabel();
            ivjVersion.setName("Version");
            ivjVersion.setText("Versió 1.0.        Resolució mínima recomanada 800x600");
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjVersion;
}
/**
 * Called whenever the part throws an exception.
 * @param exception java.lang.Throwable
 */
private void handleException(java.lang.Throwable exception) {

    /* Uncomment the following lines to print uncaught exceptions to stdout */
    // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
    // exception.printStackTrace(System.out);
}
/**
 * Initializes connections
 * @exception java.lang.Exception The exception description.
 */
```

```
private void initConnections() throws java.lang.Exception {
    getOkButton().addActionListener(ivjEventHandler);
}
/**
 * Initialize the class.
 */
private void initialize() {
    try {
        setName("QuantAtfc");
        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        setSize(330, 160);
        setTitle("Quant a TFC");
        setContentPane(getJDialogContentPane());
        initConnections();
    } catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
    }
}

}   /*---- Final de la Classe AboutBox ------*/
```

Tfc.java

```
/* Aplicació per controlar les matrius Q d'una imatge
 * al transformar-la en JPEG en funció d'un PNSR
 *
 *  Versió 1.0: Antoni Francés    --- Juny 2002 ---
 *
 * Utilitzo el model gràfic Swing en compte de l'antic Awt.
 */

//Clases necessaries per la interfície gràfica
import java.awt.*;
import java.awt.event.*;
import java.awt.image.*;
import java.awt.image.WritableRaster;
import java.awt.image.DataBuffer;
import java.awt.image.Raster;
import java.awt.image.renderable.RenderedImageFactory;
import java.awt.image.RenderedImage;
import java.awt.image.renderable.ParameterBlock;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.border.*;

//Classes necessaries per la lectura i escriptura d'arxius
import java.io.File;
import java.io.FilenameFilter;
import java.io.FileOutputStream;
import java.io.IOException;

//Classes necessaries per treballar amb les imatges
//(Classes de la llibreria JAI de Sun Microsystems)
import javax.media.jai.JAI;
import javax.media.jai.ParameterBlockJAI;
import javax.media.jai.OperationDescriptor;
import javax.media.jai.OperationRegistry;
import javax.media.jai.PlanarImage;
import javax.media.jai.InterpolationNearest;
import javax.media.jai.widget.ImageCanvas;
import javax.media.jai.*;
import com.sun.media.jai.codec.ImageCodec;
import com.sun.media.jai.codec.ImageEncoder;
import com.sun.media.jai.codec.JPEGEncodeParam;
/*********************************************************
 *
 *      TFC---> Classe principal de l'aplicació.
 *
/*********************************************************/
public class tfc extends JFrame {

    private IvjEventHandler ivjEventHandler = new IvjEventHandler();
    private JPanel ivjJFrameContentPane = null;
```

```
    private JPanel ivjtfcPane = null;

    private JMenuBar ivjBarraMenu = null;
    private JMenu ivjMenuArxiu = null;
    private JMenu ivjMenuAjuda = null;
    private JMenuItem ivjMenuItemObrir = null;
    private JSeparator ivjJSeparator1 = null;
    private JMenuItem ivjSaveMenuItem = null;
    private JMenuItem ivjSave_AsMenuItem = null;
    private JSeparator ivjJSeparator2 = null;
    private JMenuItem ivjExitMenuItem = null;
    private JMenuItem ivjMenuItemIndex = null;
    private JMenuItem ivjMenuItemQuantA = null;
    private JPanel ivjPanellImatge = null;
    private JPanel ivjPanellMatriu = null;
    private JPanel ivjPanellDesviacio = null;
    private JPanel ivjPanellMitjana = null;
    private JTabbedPane ivjPanells = null;
    private JProgressBar ivjBarraProg = null;
    private JSlider ivjBarraSlid = null;
    private JButton ivjBotoProcessar = null;
    private JButton ivjBotoReProcessar = null;
    private JLabel ivjEtiquetaPSNR = null;
    private JLabel JPsnrReal = null;
    private JTextField ivjEtiqValorPSNR = null;

    private JPanel ivjJPan1_1 = null;
    private JPanel ivjJPan1_2 = null;
    private JPanel ivjJPan1_3 = null;
    private JPanel ivjJPan1_4 = null;
    private JPanel ivjJPan1_5 = null;

    private JLabel ivjJLabOrig = null;
    private JLabel ivjJLabComp = null;
    private JLabel ivjJLabTamOrig = null;
    private JLabel ivjJLabTamComp = null;
    private JLabel ivjJLabelPsnrMin = null;
    private JLabel ivjJLabelPsnrMax = null;
    private JLabel ivjJLabelMSE = null;
    private JLabel JLabCompQ1 = null;
    private JLabel JLabCompQ2 = null;
    private JLabel JLabCompQ3 = null;
    private JLabel JLabCompM1 = null;
    private JLabel JLabCompM2 = null;
    private JLabel JLabCompM3 = null;
    private JLabel JLabCompD1 = null;
    private JLabel JLabCompD2 = null;
    private JLabel JLabCompD3 = null;

    private JTextField ivjJTextTamOrig = null;
    private JTextField ivjJTextTamComp = null;
    private JTextField ivjJTextPsnrMin = null;
    private JTextField ivjJTextPsnrMax = null;
    private JTextField ivjJTextMSE = null;

    private int PsnrInicial = 25;
    private float PsnrReal = 0;
    private int phi = 1;
    private int COLORS = 3;
    private static int DCTSIZE2 = 64;

    private JTextField [][]TMatriuQ;
    private JTextField [][]TMatriuM;
    private JTextField [][]TMatriuS;

    private String NomFitxer =" ";
    private String NomFitxerSortida =" ";
    private PlanarImage im0i;
    private PlanarImage im0o;
    private ImageCanvas imageCanvas;

    private ImageEncoder eim1;
    private JPEGEncodeParam pIm1;
    private int [][]matriuQ;
    private float [][]MatriuMS = new float[2*COLORS][DCTSIZE2];
```

```
    private static javax.swing.ButtonGroup ivjGrupBotons= null;
    private static javax.swing.JRadioButton BotoRadio1= null;
    private static javax.swing.JRadioButton BotoRadio2= null;
    private static String IdPhis[] = { "Identitat", "HVS" };

    //*------- Final declaracions d'Objectes ---------*/



    // Metodes escrits en C inclossos en la llibreria 'quantitza'

    //Mètode extern pel càlcul de la matriu Q
    private native  float [][]calculamatriuQ(float [][]x, float z, int p);
    static{
           System.loadLibrary("quantitza");
        }

    //Mètode extern pel càlcul dels descriptors estadístics
    private native  float [][]calculamatriuD(String y);
    static{
           System.loadLibrary("quantitza");
        }

    //Mètode extern pel càlcul de MSE màxim.
    private native float MSEmaxim(float [][]x);
    static{
           System.loadLibrary("quantitza");
        }

    //Mètode extern pel càlcul de MSE mínim.
    private native float MSEminim(float [][]x);
    static{
           System.loadLibrary("quantitza");
        }


/*********************************************************
 *
 * Aquesta classe em permet gestionar events generats per
 * la barra de desplaçament.
 *
 *********************************************************/
class gestorBarres implements ChangeListener{


    /*--- Quan s'ha mogut la barra de desplaçament ---*/
    public void stateChanged( ChangeEvent evt){

           ivjBarraProg.setValue( ivjBarraSlid.getValue() );
           ivjEtiqValorPSNR.setText( Integer.toString( ivjBarraSlid.getValue()) );

        // Si encara no s'ha carregat cap fitxer no fer res.
           if( NomFitxer.equals(" ") ) return;

           getJTextMSE().setText(new Double(Math.pow(255,2)/Math.pow(10,
               ((double)Double.valueOf(getEtiqValorPSNR().getText()
                   ).doubleValue()/10.0))).toString().substring(0,6));

           reprocessarImatge();
           calculaPSNR();
           calculaPsnrMaxim();
           calculaPsnrMinim();

           }
}   /*---- Final del GESTOR DE BARRES DE DESPLAÇAMENT. ------------*/



/*********************************************************
 *
 * Aquesta classe em permet gestionar events generats pels
 * botons de l'aplicació.
 *
 *********************************************************/
class gestorBotons extends JFrame implements ActionListener{
```

```
    public void actionPerformed( ActionEvent evt){

                // Cal anar comparant per veure quin és
                //  el boto que s'ha pulsat
            if( ((JButton)evt.getSource()).getText().equals("Processar imatge"))
                {
                        // Si encara no s'ha carregat cap fitxer.
                if( NomFitxer.equals(" ") ) return;

                getJTextMSE().setText(new Double(Math.pow(255,2)/Math.pow(10,
                    ((double)Double.valueOf(getEtiqValorPSNR().getText()
                        ).doubleValue()/10))).toString().substring(0,6));

                ivjBarraProg.setValue( Integer.valueOf(
                                        ivjEtiqValorPSNR.getText()).intValue() );
                ivjBarraSlid.setValue( Integer.valueOf(
                                        ivjEtiqValorPSNR.getText()).intValue() );
                processarImatge();
                calculaPSNR();
                calculaPsnrMaxim();
                calculaPsnrMinim();
                }
            if( ((JButton)evt.getSource()).getText().equals("ReProcessar imatge"))
                {
                        // Si encara no s'ha carregat cap fitxer.
                if( NomFitxer.equals(" ") ) return;

                        // Supossadament quan modifiquem algun element de les matrius Q.
                reprocessarImatge();
                }

            }

}   /*----------- Final del GESTOR DE BOTONS. -------------------*/



/*********************************************************
 * Aquesta classe em permet gestionar els events produits
 * per la finestra de l'aplicació, pels menús.
 *
 *********************************************************/
class IvjEventHandler implements java.awt.event.ActionListener {

        public void actionPerformed(java.awt.event.ActionEvent e) {

            if (e.getSource() == tfc.this.getExitMenuItem())
                sortirAplicacio(e);
            if (e.getSource() == tfc.this.getMenuItemQuantA())
                mostrarQuantA(e);
            if (e.getSource() == tfc.this.getMenuItemObrir())
                ObrirArxiu(e);
            if (e.getSource() == tfc.this.getSaveMenuItem())
                desarArxiu(e);
            if (e.getSource() == tfc.this.getMenuItemIndex())
                mostrarIndexAjuda(e);
            if (e.getSource() == tfc.this.getSave_AsMenuItem())
                desarAarxiu(e);
        };
    };

/*------------------------ Gestors dels mètodes ------------------*/
```

```
/**
 * sortirAplicacio:  (ExitMenuItem.action.actionPerformed(java.awt.event.ActionEvent) --
> tfc.dispose()V)
 * @param arg1 java.awt.event.ActionEvent
 */
private void sortirAplicacio(java.awt.event.ActionEvent arg1) {
    try {
            JOptionPane.showMessageDialog( this,
                  "Adéu... ;-)", "TFC                Sortida",
                  JOptionPane.WARNING_MESSAGE );
            this.dispose();
    } catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
    }
}
/**
 * desarAarxiu:  (Save_AsMenuItem.action.actionPerformed(java.awt.event.ActionEvent) -->
tfc.save_AsMenuItem_ActionPerformed(Ljava.awt.event.ActionEvent;)V)
 * @param arg1 java.awt.event.ActionEvent
 */
private void desarAarxiu(java.awt.event.ActionEvent arg1) {
    try {
        this.save_AsMenuItem_ActionPerformed(arg1);
    } catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
    }
}
/**
 * desarArxiu:  (SaveMenuItem.action.actionPerformed(java.awt.event.ActionEvent) -->
tfc.saveMenuItem_ActionPerformed(Ljava.awt.event.ActionEvent;)V)
 * @param arg1 java.awt.event.ActionEvent
 */
private void desarArxiu(java.awt.event.ActionEvent arg1) {
    try {
        this.saveMenuItem_ActionPerformed(arg1);
    } catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
    }
}
/**
 * Return the BarraProg property value.
 * @return javax.swing.JProgressBar
 */
private javax.swing.JProgressBar getBarraProg() {
    if (ivjBarraProg == null) {
        try {
            ivjBarraProg = new javax.swing.JProgressBar();
            ivjBarraProg.setName("BarraProg");
            ivjBarraProg.setStringPainted(true);
            ivjBarraProg.setBounds(23, 55, 190, 20);
            ivjBarraProg.setValue( PsnrInicial );
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjBarraProg;
}
/**
 * Return the BarraSlid property value.
 * @return javax.swing.JSlider
 */
private javax.swing.JSlider getBarraSlid() {
    if (ivjBarraSlid == null) {
        try {
            ivjBarraSlid = new javax.swing.JSlider();
            ivjBarraSlid.setName("BarraSlid");
            ivjBarraSlid.setPaintLabels(false);
            ivjBarraSlid.setInverted(false);
            ivjBarraSlid.setComponentOrientation(java.awt.ComponentOrientation.UNKNOWN);
            ivjBarraSlid.setPaintTicks(true);
            ivjBarraSlid.setPaintTrack(true);
            ivjBarraSlid.setValue( PsnrInicial );
            ivjBarraSlid.setMajorTickSpacing(20);
            ivjBarraSlid.setMinorTickSpacing(10);
            ivjBarraSlid.setSnapToTicks( false );
            ivjBarraSlid.setBounds(18, 30, 200, 20);
```

```
                                                // Gestor per a la barra.
            ChangeListener alm = new gestorBarres();
            ivjBarraSlid.addChangeListener( alm );
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjBarraSlid;
}
/**
 * Return the BotoProcessar property value.
 * @return javax.swing.JButton
 */
private javax.swing.JButton getBotoProcessar() {
    if (ivjBotoProcessar == null) {
        try {
            ivjBotoProcessar = new javax.swing.JButton();
            ivjBotoProcessar.setName("BotoProcessar");
            ivjBotoProcessar.setMnemonic('P');
            ivjBotoProcessar.setText("Processar imatge");
            ivjBotoProcessar.setContentAreaFilled(true);
            ivjBotoProcessar.setBounds(28, 80, 180, 30);
            ivjBotoProcessar.setForeground(java.awt.Color.blue);
            ActionListener alb = new gestorBotons();
            ivjBotoProcessar.addActionListener( alb );
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjBotoProcessar;
}
/**
 * Return the BotoReProcessar property value.
 * @return javax.swing.JButton
 */
private javax.swing.JButton getBotoReProcessar() {
    if (ivjBotoReProcessar == null) {
        try {
            ivjBotoReProcessar = new javax.swing.JButton();
            ivjBotoReProcessar.setName("BotoReProcessar");
            ivjBotoReProcessar.setMnemonic('R');
            ivjBotoReProcessar.setText("Reprocessar imatge");
            ivjBotoReProcessar.setContentAreaFilled(true);
            ivjBotoReProcessar.setBounds(420, 80, 140, 30);
            ivjBotoReProcessar.setForeground(java.awt.Color.blue);
            ActionListener alb = new gestorBotons();
            ivjBotoReProcessar.addActionListener( alb );
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjBotoReProcessar;
}
/**
 * Return the EtiquetaPSNR property value.
 * @return javax.swing.JLabel
 */
private javax.swing.JLabel getEtiquetaPSNR() {
    if (ivjEtiquetaPSNR == null) {
        try {
            ivjEtiquetaPSNR = new javax.swing.JLabel();
            ivjEtiquetaPSNR.setName("EtiquetaPSNR");
            ivjEtiquetaPSNR.setText("Valor PSNR ");
            ivjEtiquetaPSNR.setBounds(10, 5, 85, 20);
            ivjEtiquetaPSNR.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjEtiquetaPSNR;
}
/**
 * Return the EtiqValorPSNR property value.
 * @return javax.swing.JTextField
 */
private javax.swing.JTextField getEtiqValorPSNR() {
```

```
    if (ivjEtiqValorPSNR == null) {
        try {
            ivjEtiqValorPSNR = new javax.swing.JTextField();
            ivjEtiqValorPSNR.setName("EtiqValorPSNR");
            ivjEtiqValorPSNR.setText( String.valueOf(PsnrInicial) );
            ivjEtiqValorPSNR.setBounds(95, 5, 30, 20);
            ivjEtiqValorPSNR.setColumns(3);
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjEtiqValorPSNR;
}
/**
 * Return the ExitMenuItem property value.
 * @return javax.swing.JMenuItem
 */
private javax.swing.JMenuItem getExitMenuItem() {
    if (ivjExitMenuItem == null) {
        try {
            ivjExitMenuItem = new javax.swing.JMenuItem();
            ivjExitMenuItem.setName("ExitMenuItem");
            ivjExitMenuItem.setMnemonic('S');
            ivjExitMenuItem.setText("Sortir");
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjExitMenuItem;
}
/**
 * Return the BarraMenu property value.
 * @return javax.swing.JMenuBar
 */
private javax.swing.JMenuBar getBarraMenu() {
    if (ivjBarraMenu == null) {
        try {
            ivjBarraMenu = new javax.swing.JMenuBar();
            ivjBarraMenu.setName("BarraMenu");
            ivjBarraMenu.setEnabled(true);
            ivjBarraMenu.setVisible(true);
            ivjBarraMenu.add(getMenuArxiu());
            ivjBarraMenu.add(getMenuAjuda());
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjBarraMenu;
}
/**
 * Return the JFrameContentPane property value.
 * @return javax.swing.JPanel
 */
private javax.swing.JPanel getJFrameContentPane() {
    if (ivjJFrameContentPane == null) {
        try {
            ivjJFrameContentPane = new javax.swing.JPanel();
            ivjJFrameContentPane.setName("JFrameContentPane");
            ivjJFrameContentPane.setLayout(new java.awt.BorderLayout());
                // Per si vulgués mostrar la barra d'estats. Descartat de moment.
            //getJFrameContentPane().add(getStatusBarPane(), "South");
            getJFrameContentPane().add(gettfcPane(), "Center");
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjJFrameContentPane;
}
/**
 * Return the JLabTamOrig property value.
 * @return javax.swing.JLabel
 */
private javax.swing.JLabel getJLabTamOrig() {
    if (ivjJLabTamOrig == null) {
        try {
            ivjJLabTamOrig = new javax.swing.JLabel();
```

```
            ivjJLabTamOrig.setName("JLabTamOrig");
            ivjJLabTamOrig.setText("Mida arxiu :");
            ivjJLabTamOrig.setBounds(6, 5, 60, 20);
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjJLabTamOrig;
}
/**
 * Return the JLabelPsnrMin property value.
 * @return javax.swing.JLabel
 */
private javax.swing.JLabel getJLabelPsnrMin() {
    if (ivjJLabelPsnrMin == null) {
        try {
            ivjJLabelPsnrMin = new javax.swing.JLabel();
            ivjJLabelPsnrMin.setName("JLabelPsnrMin");
            ivjJLabelPsnrMin.setText("PSNR Mínim :");
            ivjJLabelPsnrMin.setBounds(6, 30, 80, 20);
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjJLabelPsnrMin;
}
/**
 * Return the JLabelPsnrMax property value.
 * @return javax.swing.JLabel
 */
private javax.swing.JLabel getJLabelPsnrMax() {
    if (ivjJLabelPsnrMax == null) {
        try {
            ivjJLabelPsnrMax = new javax.swing.JLabel();
            ivjJLabelPsnrMax.setName("JLabelPsnrMax");
            ivjJLabelPsnrMax.setText("PSNR Màxim :");
            ivjJLabelPsnrMax.setBounds(6, 55, 80, 20);
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjJLabelPsnrMax;
}
/**
 * Return the JLabelPsnrMax property value.
 * @return javax.swing.JLabel
 */
private javax.swing.JLabel getJPsnrReal() {
    if ( JPsnrReal == null) {
        try {
            JPsnrReal = new javax.swing.JLabel();
            JPsnrReal.setName("JPsnrReal");
            JPsnrReal.setText("PSNR real:  "+ PsnrReal );
            JPsnrReal.setBounds(6, 30, 150, 20);
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return JPsnrReal;
}



/**
 * Return the JLabelMSE property value.
 * @return javax.swing.JLabel
 */
private javax.swing.JLabel getJLabelMSE() {
    if (ivjJLabelMSE == null) {
        try {
            ivjJLabelMSE = new javax.swing.JLabel();
            ivjJLabelMSE.setName("JLabelMSE");
            ivjJLabelMSE.setText("MSE :");
            ivjJLabelMSE.setBounds(6, 80, 40, 20);
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
```

```
        }
    }
    return ivjJLabelMSE;
}
/**
 * Return the JLabTamComp property value.
 * @return javax.swing.JLabel
 */
private javax.swing.JLabel getJLabTamComp() {
    if (ivjJLabTamComp == null) {
        try {
            ivjJLabTamComp = new javax.swing.JLabel();
            ivjJLabTamComp.setName("JLabTamComp");
            ivjJLabTamComp.setText("Mida arxiu :");
            ivjJLabTamComp.setBounds(6, 5, 60, 20);
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjJLabTamComp;
}
/**
 * Return the JLabOrig property value.
 * @return javax.swing.JLabel
 */
private javax.swing.JLabel getJLabOrig() {
    if (ivjJLabOrig == null) {
        try {
            ivjJLabOrig = new javax.swing.JLabel();
            ivjJLabOrig.setName("JLabOrig");
            ivjJLabOrig.setText("ORIGINAL: "+ NomFitxer );
            ivjJLabOrig.setBounds(5, 5, 280, 25);
            ivjJLabOrig.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
            ivjJLabOrig.setHorizontalTextPosition(javax.swing.SwingConstants.LEFT);
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjJLabOrig;
}
/**
 * Return the JLabComp property value.
 * @return javax.swing.JLabel
 */
private javax.swing.JLabel getJLabComp() {
    if (ivjJLabComp == null) {
        try {
            ivjJLabComp = new javax.swing.JLabel();
            ivjJLabComp.setName("JLabComp");
            ivjJLabComp.setText("COMPRIMIDA: "+NomFitxerSortida);
            ivjJLabComp.setBounds(320, 5, 280, 25);
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjJLabComp;
}

/**
 * Return the JLabCompQ1 property value.
 * @return javax.swing.JLabel
 * Etiqueta per anomenar les matrius.
 */
private javax.swing.JLabel getJLabCompQ1() {
    if (JLabCompQ1 == null) {
        try {
            JLabCompQ1 = new javax.swing.JLabel();
            JLabCompQ1.setName("JLabCompQ1");
            JLabCompQ1.setText("COMPONENT 1");
            JLabCompQ1.setBounds( 60, 80, 100, 25);
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return JLabCompQ1;
}
```

```
/**
 * Return the JLabCompQ2 property value.
 * @return javax.swing.JLabel
 * Etiqueta per anomenar les matrius.
 */
private javax.swing.JLabel getJLabCompQ2() {
    if (JLabCompQ2 == null) {
        try {
            JLabCompQ2 = new javax.swing.JLabel();
            JLabCompQ2.setName("JLabCompQ2");
            JLabCompQ2.setText("COMPONENT 2");
            JLabCompQ2.setBounds( 100, 390, 100, 25);
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return JLabCompQ2;
}
/**
 * Return the JLabCompQ3 property value.
 * @return javax.swing.JLabel
 * Etiqueta per anomenar les matrius.
 */
private javax.swing.JLabel getJLabCompQ3() {
    if (JLabCompQ3 == null) {
        try {
            JLabCompQ3 = new javax.swing.JLabel();
            JLabCompQ3.setName("JLabCompQ3");
            JLabCompQ3.setText("COMPONENT 3");
            JLabCompQ3.setBounds( 400, 390, 100, 25);
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return JLabCompQ3;
}

/**
 * Return the JLabCompM1 property value.
 * @return javax.swing.JLabel
 * Etiqueta per anomenar les matrius.
 */
private javax.swing.JLabel getJLabCompM1() {
    if (JLabCompM1 == null) {
        try {
            JLabCompM1 = new javax.swing.JLabel();
            JLabCompM1.setName("JLabCompM1");
            JLabCompM1.setText("COMPONENT 1");
            JLabCompM1.setBounds( 5, 80, 100, 25);
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return JLabCompM1;
}

/**
 * Return the JLabCompM2 property value.
 * @return javax.swing.JLabel
 * Etiqueta per anomenar les matrius.
 */
private javax.swing.JLabel getJLabCompM2() {
    if (JLabCompM2 == null) {
        try {
            JLabCompM2 = new javax.swing.JLabel();
            JLabCompM2.setName("JLabCompM2");
            JLabCompM2.setText("COMPONENT 2");
            JLabCompM2.setBounds( 100, 390, 100, 25);
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return JLabCompM2;
}
```

```
/**
 * Return the JLabCompM3 property value.
 * @return javax.swing.JLabel
 * Etiqueta per anomenar les matrius.
 */
private javax.swing.JLabel getJLabCompM3() {
    if (JLabCompM3 == null) {
        try {
            JLabCompM3 = new javax.swing.JLabel();
            JLabCompM3.setName("JLabCompM3");
            JLabCompM3.setText("COMPONENT 3");
            JLabCompM3.setBounds( 400, 390, 100, 25);
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return JLabCompM3;
}

/**
 * Return the JLabCompD1 property value.
 * @return javax.swing.JLabel
 * Etiqueta per anomenar les matrius.
 */
private javax.swing.JLabel getJLabCompD1() {
    if (JLabCompD1 == null) {
        try {
            JLabCompD1 = new javax.swing.JLabel();
            JLabCompD1.setName("JLabCompD1");
            JLabCompD1.setText("COMPONENT 1");
            JLabCompD1.setBounds( 5, 80, 80, 25);
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return JLabCompD1;
}

/**
 * Return the JLabCompD2 property value.
 * @return javax.swing.JLabel
 * Etiqueta per anomenar les matrius.
 */
private javax.swing.JLabel getJLabCompD2() {
    if (JLabCompD2 == null) {
        try {
            JLabCompD2 = new javax.swing.JLabel();
            JLabCompD2.setName("JLabCompD2");
            JLabCompD2.setText("COMPONENT 2");
            JLabCompD2.setBounds( 100, 390, 100, 25);
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return JLabCompD2;
}
/**
 * Return the JLabCompD3 property value.
 * @return javax.swing.JLabel
 * Etiqueta per anomenar les matrius.
 */
private javax.swing.JLabel getJLabCompD3() {
    if (JLabCompD3 == null) {
        try {
            JLabCompD3 = new javax.swing.JLabel();
            JLabCompD3.setName("JLabCompD3");
            JLabCompD3.setText("COMPONENT 3");
            JLabCompD3.setBounds( 400, 390, 100, 25);
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return JLabCompD3;
}
```

```
/**
 * Return the JPan1_1 property value.
 * @return javax.swing.JPanel
 */
private javax.swing.JPanel getJPan1_1() {
    if (ivjJPan1_1 == null) {
        try {
            ivjJPan1_1 = new javax.swing.JPanel();
            ivjJPan1_1.setName("JPan1_1");
            ivjJPan1_1.setLayout(null);
            ivjJPan1_1.setBounds(2, 310, 175, 108);
            getJPan1_1().add(getJTextTamOrig(), getJTextTamOrig().getName());
            getJPan1_1().add(getJLabTamOrig(), getJLabTamOrig().getName());
            getJPan1_1().add(getJTextPsnrMin(), getJTextPsnrMin().getName());
            getJPan1_1().add(getJLabelPsnrMin(), getJLabelPsnrMin().getName());
            getJPan1_1().add(getJTextPsnrMax(), getJTextPsnrMax().getName());
            getJPan1_1().add(getJLabelPsnrMax(), getJLabelPsnrMax().getName());
            getJPan1_1().add(getJTextMSE(), getJTextMSE().getName());
            getJPan1_1().add(getJLabelMSE(), getJLabelMSE().getName());
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjJPan1_1;
}
/**
 * Return the JPan1_2 property value.
 * @return javax.swing.JPanel
 */
private javax.swing.JPanel getJPan1_2() {
    if (ivjJPan1_2 == null) {
        try {
            ivjJPan1_2 = new javax.swing.JPanel();
            ivjJPan1_2.setName("JPan1_2");
            ivjJPan1_2.setLayout(null);
            ivjJPan1_2.setBounds(180, 310, 235, 115);
            getJPan1_2().add(getBarraSlid(), getBarraSlid().getName());
            getJPan1_2().add(getBarraProg(), getBarraProg().getName());
            getJPan1_2().add(getBotoProcessar(), getBotoProcessar().getName());
            getJPan1_2().add(getEtiquetaPSNR(), getEtiquetaPSNR().getName());
            getJPan1_2().add(getEtiqValorPSNR(), getEtiqValorPSNR().getName());
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjJPan1_2;
}
/**
 * Return the JPan1_3 property value.
 * @return javax.swing.JPanel
 */
private javax.swing.JPanel getJPan1_3() {
    if (ivjJPan1_3 == null) {
        try {
            ivjJPan1_3 = new javax.swing.JPanel();
            ivjJPan1_3.setName("JPan1_3");
            ivjJPan1_3.setLayout(null);
            ivjJPan1_3.setBounds(417, 310, 175, 108);
            getJPan1_3().add(getJLabTamComp(), getJLabTamComp().getName());
            getJPan1_3().add(getJTextTamComp(), getJTextTamComp().getName());
            getJPan1_3().add(getJPsnrReal(), getJPsnrReal().getName());
            BotoRadio1 = new JRadioButton(IdPhis[0] );
            BotoRadio1.setSelected(true);
            BotoRadio2 = new JRadioButton(IdPhis[1] );
            BotoRadio2.setSelected(false);
            ivjGrupBotons = new ButtonGroup();
            ivjGrupBotons.add(BotoRadio1);
            ivjGrupBotons.add(BotoRadio2);
            JPanel panell = new JPanel();
            panell.setBorder( new TitledBorder( "Funció PHI") );
            panell.setBounds(2,50,170,60);
            panell.add( BotoRadio1);
            panell.add( BotoRadio2);
            getJPan1_3().add( panell );
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
```

```
        }
    }
    return ivjJPan1_3;
}
/**
 * Return the JPan1_4 property value.
 * @return javax.swing.JPanel
 */
private javax.swing.JPanel getJPan1_4() {
    if (ivjJPan1_4 == null) {
        try {
            ivjJPan1_4 = new javax.swing.JPanel();
            ivjJPan1_4.setName("JPan1_4");
            ivjJPan1_4.setLayout(null);
            ivjJPan1_4.setBounds(150, 30, 280, 280);
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjJPan1_4;
}
/**
 * Return the JPan1_5 property value.
 * @return javax.swing.JPanel
 */
private javax.swing.JPanel getJPan1_5() {
    if (ivjJPan1_5 == null) {
        try {
            ivjJPan1_5 = new javax.swing.JPanel();
            ivjJPan1_5.setName("JPan1_5");
            ivjJPan1_5.setLayout(null);
            ivjJPan1_5.setBounds(450, 30, 280, 280);
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjJPan1_5;
}
/**
 * Return the JSeparator1 property value.
 * @return javax.swing.JSeparator
 */
private javax.swing.JSeparator getJSeparator1() {
    if (ivjJSeparator1 == null) {
        try {
            ivjJSeparator1 = new javax.swing.JSeparator();
            ivjJSeparator1.setName("JSeparator1");
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjJSeparator1;
}
/**
 * Return the JSeparator2 property value.
 * @return javax.swing.JSeparator
 */
private javax.swing.JSeparator getJSeparator2() {
    if (ivjJSeparator2 == null) {
        try {
            ivjJSeparator2 = new javax.swing.JSeparator();
            ivjJSeparator2.setName("JSeparator2");
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjJSeparator2;
}
/**
 * Return the JTextTamOrig property value.
 * @return javax.swing.JTextField
 */
private javax.swing.JTextField getJTextTamOrig() {
    if (ivjJTextTamOrig == null) {
        try {
            ivjJTextTamOrig = new javax.swing.JTextField();
```

```
            ivjJTextTamOrig.setName("JTextTamOrig");
            ivjJTextTamOrig.setText("  0");
            ivjJTextTamOrig.setBounds(70, 5, 100, 20);
            ivjJTextTamOrig.setEditable(false);
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjJTextTamOrig;
}
/**
 * Return the JTextPsnrMin property value.
 * @return javax.swing.JTextField
 */
private javax.swing.JTextField getJTextPsnrMin() {
    if (ivjJTextPsnrMin == null) {
        try {
            ivjJTextPsnrMin = new javax.swing.JTextField();
            ivjJTextPsnrMin.setName("JTextPsnrMin");
            ivjJTextPsnrMin.setText(" ");
            ivjJTextPsnrMin.setBounds(91, 30, 80, 20);
            ivjJTextPsnrMin.setEditable(false);
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjJTextPsnrMin;
}
/**
 * Return the JTextPsnrMax property value.
 * @return javax.swing.JTextField
 */
private javax.swing.JTextField getJTextPsnrMax() {
    if (ivjJTextPsnrMax == null) {
        try {
            ivjJTextPsnrMax = new javax.swing.JTextField();
            ivjJTextPsnrMax.setName("JTextPsnrMax");
            ivjJTextPsnrMax.setText(" ");
            ivjJTextPsnrMax.setBounds(91, 55, 80, 20);
            ivjJTextPsnrMax.setEditable(false);
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjJTextPsnrMax;
}
/**
 * Return the JTextMSE property value.
 * @return javax.swing.JTextField
 */
private javax.swing.JTextField getJTextMSE() {
    if (ivjJTextMSE == null) {
        try {
            ivjJTextMSE = new javax.swing.JTextField();
            ivjJTextMSE.setName("JTextMSE");
            ivjJTextMSE.setText("0");
            ivjJTextMSE.setBounds(46, 80, 125, 20);
            ivjJTextMSE.setEditable(false);
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjJTextMSE;
}
/**
 * Return the JTextTamComp property value.
 * @return javax.swing.JTextField
 */
private javax.swing.JTextField getJTextTamComp() {
    if (ivjJTextTamComp == null) {
        try {
            ivjJTextTamComp = new javax.swing.JTextField();
            ivjJTextTamComp.setName("JTextTamComp");
            ivjJTextTamComp.setText("  0");
            ivjJTextTamComp.setBounds(70, 5, 100, 20);
            ivjJTextTamComp.setEditable(false);
```

```
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjJTextTamComp;
}
/**
 * Return the HelpMenu property value.
 * @return javax.swing.JMenu
 */
private javax.swing.JMenu getMenuAjuda() {
    if (ivjMenuAjuda == null) {
        try {
            ivjMenuAjuda = new javax.swing.JMenu();
            ivjMenuAjuda.setName("MenuAjuda");
            ivjMenuAjuda.setMnemonic('j');
            ivjMenuAjuda.setText("Ajuda");
            ivjMenuAjuda.add(getMenuItemIndex());
            ivjMenuAjuda.add(getMenuItemQuantA());
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjMenuAjuda;
}
/**
 * Return the FileMenu property value.
 * @return javax.swing.JMenu
 */
private javax.swing.JMenu getMenuArxiu() {
    if (ivjMenuArxiu == null) {
        try {
            ivjMenuArxiu = new javax.swing.JMenu();
            ivjMenuArxiu.setName("MenuArxiu");
            ivjMenuArxiu.setSelected(false);
            ivjMenuArxiu.setMnemonic('A');
            ivjMenuArxiu.setText("Arxiu");
            ivjMenuArxiu.add(getMenuItemObrir());
            ivjMenuArxiu.add(getJSeparator1());
            ivjMenuArxiu.add(getSaveMenuItem());
            ivjMenuArxiu.add(getSave_AsMenuItem());
            ivjMenuArxiu.add(getJSeparator2());
            ivjMenuArxiu.add(getExitMenuItem());
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjMenuArxiu;
}
/**
 * Return the Help_TopicsMenuItem property value.
 * @return javax.swing.JMenuItem
 */
private javax.swing.JMenuItem getMenuItemIndex() {
    if (ivjMenuItemIndex == null) {
        try {
            ivjMenuItemIndex = new javax.swing.JMenuItem();
            ivjMenuItemIndex.setName("MenuItemIndex");
            ivjMenuItemIndex.setMnemonic('I');
            ivjMenuItemIndex.setText("Index");
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjMenuItemIndex;
}
/**
 * Return the OpenMenuItem property value.
 * @return javax.swing.JMenuItem
 */
private javax.swing.JMenuItem getMenuItemObrir() {
    if (ivjMenuItemObrir == null) {
        try {
            ivjMenuItemObrir = new javax.swing.JMenuItem();
            ivjMenuItemObrir.setName("MenuItemObrir");
            ivjMenuItemObrir.setMnemonic('O');
```

```
            ivjMenuItemObrir.setText("Obrir");
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjMenuItemObrir;
}
/**
 * Return the About_BoxMenuItem property value.
 * @return javax.swing.JMenuItem
 */
private javax.swing.JMenuItem getMenuItemQuantA() {
    if (ivjMenuItemQuantA == null) {
        try {
            ivjMenuItemQuantA = new javax.swing.JMenuItem();
            ivjMenuItemQuantA.setName("MenuItemQuantA");
            ivjMenuItemQuantA.setMnemonic('Q');
            ivjMenuItemQuantA.setText("Quant a...");
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjMenuItemQuantA;
}

/**
 * Return the PanellDesviacio property value.
 * @return javax.swing.JPanel
 */
private javax.swing.JPanel getPanellDesviacio() {
    if (ivjPanellDesviacio == null) {
        try {
            ivjPanellDesviacio = new javax.swing.JPanel();
            ivjPanellDesviacio.setName("PanellDesviacio");
            ivjPanellDesviacio.setLayout(null);
            setPanellDesviacio();
            getPanellDesviacio().add(getJLabCompD1(), getJLabCompD1().getName());
            getPanellDesviacio().add(getJLabCompD2(), getJLabCompD2().getName());
            getPanellDesviacio().add(getJLabCompD3(), getJLabCompD3().getName());
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjPanellDesviacio;
}
/**
 * Return the PanellImatge property value.
 * @return javax.swing.JPanel
 */
private javax.swing.JPanel getPanellImatge() {
    if (ivjPanellImatge == null) {
        try {
            ivjPanellImatge = new javax.swing.JPanel();
            ivjPanellImatge.setName("PanellImatge");
            ivjPanellImatge.setLayout(null);
            getPanellImatge().add(getJPan1_4(), getJPan1_4().getName());
            getPanellImatge().add(getJLabOrig(), getJLabOrig().getName());
            getPanellImatge().add(getJPan1_5(), getJPan1_5().getName());
            getPanellImatge().add(getJLabComp(), getJLabComp().getName());
            getPanellImatge().add(getJPan1_1(), getJPan1_1().getName());
            getPanellImatge().add(getJPan1_2(), getJPan1_2().getName());
            getPanellImatge().add(getJPan1_3(), getJPan1_3().getName());
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjPanellImatge;
}
/**
 * Return the PanellMatriu property value.
 * @return javax.swing.JPanel
 */
private javax.swing.JPanel getPanellMatriu() {
    if (ivjPanellMatriu == null) {
        try {
            ivjPanellMatriu = new javax.swing.JPanel();
```

```
            ivjPanellMatriu.setName("PanellMatriu");
            ivjPanellMatriu.setLayout(null);
            setPanellMatriu();
            getPanellMatriu().add(getBotoReProcessar(), getBotoReProcessar().getName());
            getPanellMatriu().add(getJLabCompQ1(), getJLabCompQ1().getName());
            getPanellMatriu().add(getJLabCompQ2(), getJLabCompQ2().getName());
            getPanellMatriu().add(getJLabCompQ3(), getJLabCompQ3().getName());
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjPanellMatriu;
}
/**
 * Return the PanellMitjana property value.
 * @return javax.swing.JPanel
 */
private javax.swing.JPanel getPanellMitjana() {
    if (ivjPanellMitjana == null) {
        try {
            ivjPanellMitjana = new javax.swing.JPanel();
            ivjPanellMitjana.setName("PanellMitjana");
            ivjPanellMitjana.setLayout(null);
            setPanellMitjana();
            getPanellMitjana().add(getJLabCompM1(), getJLabCompM1().getName());
            getPanellMitjana().add(getJLabCompM2(), getJLabCompM2().getName());
            getPanellMitjana().add(getJLabCompM3(), getJLabCompM3().getName());
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjPanellMitjana;
}
/**
 * Return the Panells property value.
 * @return javax.swing.JTabbedPane
 */
private javax.swing.JTabbedPane getPanells() {
    if (ivjPanells == null) {
        try {
            ivjPanells = new javax.swing.JTabbedPane();
            ivjPanells.setName("Panells");
            ivjPanells.setBounds(5, 5, 600, 500);
            ivjPanells.insertTab("Imatges", null, getPanellImatge(), null, 0);
            ivjPanells.insertTab("Matriu Q", null, getPanellMatriu(), null, 1);
            ivjPanells.insertTab("Mitjana", null, getPanellMitjana(), null, 2);
            ivjPanells.insertTab("Desviació", null, getPanellDesviacio(), null, 3);
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjPanells;
}
/**
 * Return the Save_AsMenuItem property value.
 * @return javax.swing.JMenuItem
 */
private javax.swing.JMenuItem getSave_AsMenuItem() {
    if (ivjSave_AsMenuItem == null) {
        try {
            ivjSave_AsMenuItem = new javax.swing.JMenuItem();
            ivjSave_AsMenuItem.setName("Save_AsMenuItem");
            ivjSave_AsMenuItem.setMnemonic('a');
            ivjSave_AsMenuItem.setText("Desar a");
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
    return ivjSave_AsMenuItem;
}
/**
 * Return the SaveMenuItem property value.
 * @return javax.swing.JMenuItem
 */
private javax.swing.JMenuItem getSaveMenuItem() {
    if (ivjSaveMenuItem == null) {
```

```
            try {
                ivjSaveMenuItem = new javax.swing.JMenuItem();
                ivjSaveMenuItem.setName("SaveMenuItem");
                ivjSaveMenuItem.setMnemonic('D');
                ivjSaveMenuItem.setText("Desar");
            } catch (java.lang.Throwable ivjExc) {
                handleException(ivjExc);
            }
        }
        return ivjSaveMenuItem;
    }

    /**
     * Return the tfcPane property value.
     * @return javax.swing.JPanel
     */
    private javax.swing.JPanel gettfcPane() {
        if (ivjtfcPane == null) {
            try {
                ivjtfcPane = new javax.swing.JPanel();
                ivjtfcPane.setName("tfcPane");
                ivjtfcPane.setFont(new java.awt.Font("Arial", 1, 14));
                ivjtfcPane.setLayout(null);
                gettfcPane().add(getPanells(), getPanells().getName());
            } catch (java.lang.Throwable ivjExc) {
                handleException(ivjExc);
            }
        }
        return ivjtfcPane;
    }

    /**
     * Called whenever the part throws an exception.
     * @param exception java.lang.Throwable
     *
     *  Aquest mètode imprimirà tota la traça cas que es produeixi
     *   una excepcio no prevista.
     */
    private void handleException(java.lang.Throwable exception) {

        /* Uncomment the following lines to print uncaught exceptions to stdout */
         System.out.println("--------- UNCAUGHT EXCEPTION ---------");
         exception.printStackTrace(System.out);
    }
    /**
     * Initializes connections
     * @exception java.lang.Exception The exception description.
     */
    private void initConnections() throws java.lang.Exception {
        getExitMenuItem().addActionListener(ivjEventHandler);
        getMenuItemQuantA().addActionListener(ivjEventHandler);
        getMenuItemObrir().addActionListener(ivjEventHandler);
        getSaveMenuItem().addActionListener(ivjEventHandler);
        getMenuItemIndex().addActionListener(ivjEventHandler);
        getSave_AsMenuItem().addActionListener(ivjEventHandler);
    }
    /**
     * Initialize the class.
     */

    private void initialize() {
        try {
            setName("Finestratfc");
            setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
            setVisible(true);
            setJMenuBar(getBarraMenu());
            setState(0);
            setSize(610, 510);
            setResizable( false );
            setTitle("Tractament d\'imatges amb PSNR");
            setContentPane(getJFrameContentPane());
            initConnections();
        } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
        }
    }
```

```
/**
 * tfc constructor per defecte.
 */
public tfc() {
    super();
    initialize();
}
/**
 * tfc constructor comment.
 * @param title java.lang.String
 */
public tfc(String title) {
    super(title);
}


/*------------------------ Mètodes PROPIS de la classe -----------------*/


    /*****************************************************************************
     * Operacions per dur a terme el cos del TFC.                                *
     * Les operacions s'encarreguen de les tasques següents:                *
     *
     *  -   descriptors: Crida a la funció nativa que calcula els descriptors    *
     *                estadísitics de la imatge.                                 *
     *
     * -    processarImatge: Crida a la funció nativa que calcula la matriu Q a   *
     *     partir dels descriptors estadísitics, el PNSR entrat i la funció PHI.  *
     *      Després amb aquesta matriu Q generem la imatge comprimida i mostrem   *
     *      tant els valors de la matriu Q com la imatge comprimida.          *
     *
     * -    reprocessarImatge: Llegeix els valors de Q a la pantalla, genera la    *
     *      imatge comprimida a partir d'aquests valors, i la mostra per pantalla  *
     *
     * -    omplirMatriuQ: Mostra per pantalla els valor de la matriu Q            *
     *
     * -    crearJPEG: Calcula la nova imatge comprimida amb la matriu Q i la pre- *
     *      senta per pantalla.                                                    *
     *
     * -    crearOutput: Crea una fitxer generic per poder crear un fitxer JPEG    *
     *
     * -    calculaPSNR: Calcula el PSNR real amb els paràmetres actuals           *
     *
     * -    calculaPsnrMax: Calcula el PSNR maxim                                   *
     *
     * -    calculaPsnrMin: Calcula el PSNR minim                                   *
     *
     * -    obtenirMSE: Calcula el MSE d'un valor Q amb una sigma                   *
     *
     *****************************************************************************/
public void descriptors(){

    MatriuMS = calculamatriuD(NomFitxer);

    // Omplo la matriu de Desviacions (Components 0,2,4 de MS )
    for(int col=0;col<COLORS;col++){
        for(int i=0;i<DCTSIZE2;i++){
            Float Ftemp = new Float(MatriuMS[2*col][i]);
            String Stemp = new String(Ftemp.toString());
            TMatriuS[col][i].setText(Stemp.substring(0,(
                                (Stemp.length()<6)?Stemp.length():6)));
        }
    }
    // Omplo la matriu de Mitjanes (Components 1,3,5 de MS )
    for(int col=0;col<COLORS;col++){
        for(int i=0;i<DCTSIZE2;i++){
            Float Ftemp = new Float(MatriuMS[2*col+1][i]);
            String Stemp = new String(Ftemp.toString());
            TMatriuM[col][i].setText(Stemp.substring(0,(
                                (Stemp.length()<6)?Stemp.length():6)));
        }
    }
}
```

```
// Aquest mètode és per calcular el PSNR real
// en funcio de la formula 25 relació entre PSNR i MSE

private void calculaPSNR(){

    // El Logaritme en Java només és en BASE E.
    // Per això   log10( x ) = ln (x) / ln (10)
   double psnr = new Double( 10.0*Math.log( (255.0*255.0)/
               ((double) Double.valueOf(getJTextMSE().getText()).doubleValue()
               )) / Math.log( 10) ).doubleValue();

    PsnrReal = new Float(psnr).floatValue();
    getJTextPsnrReal().setText( " " + PsnrReal );
}

// Aquest metode és per calcular el PSNR Màxim,
// que serà quan el Mse sigui minim.
// És el mateix que el de la llibreria dinàmica,
private void calculaPsnrMaxim(){
    float mse, aux, sigma;
    float mseMin = 255.0F;

               // Calcul del màxim MSE de totes les components
    for( int i=0 ; i< COLORS ; i++){
        for(int j=0; j< DCTSIZE2; j++){
            sigma = MatriuMS[i][j];   // obtenim la variància
            mse = obtenirMSE( 1.0F, sigma);
            if( mse < mseMin  )
                    mseMin = mse;
        }
    }

            // El maxim PSNR pq el MSE és minim.
    if( mseMin == 0)
        getJTextPsnrMax().setText( "95");
    else
        getJTextPsnrMax().setText(
             new Float(10.0*Math.log( (255.0*255.0 )/mseMin)
                       /Math.log(10.0) ).toString());
}


// Aquest metode és per calcular el PSNR Mínim,
// que serà quan el Mse sigui màxim.
// És el mateix que el de la llibreria dinàmica,
private void calculaPsnrMinim(){
    float MSE = 0.0F;
    float mseMax, aux, sigma;

               // Calcul del màxim MSE de totes les components
    for( int i=0 ; i< COLORS ; i++){
        for(int j=0; j< DCTSIZE2; j++){
                sigma = MatriuMS[i][j];   // obtenim la variància
                if (sigma==0 )
                     aux=0;
                else{
                    mseMax = obtenirMSE( 255.0F, sigma);
                    if( mseMax > (sigma*sigma) )
                            mseMax = sigma*sigma;
                    aux=mseMax;
                }
                if( aux > MSE )    MSE = aux;
        }
    }

            // El mínim PSNR pq el MSE és màxim.
    if( MSE== 0)
        getJTextPsnrMin().setText( "1");
    else
        getJTextPsnrMin().setText(
             new Float(10.0*Math.log((255.0*255.0 )/MSE)
                       /Math.log(10.0) ).toString());

}
```

```
//
// funció per al càlcul de MSE(Q, sigma) (segons equació 13)
//      També forma part de la llibreria.
//
private float obtenirMSE(float Q, float sigma)
{
  float mse = 0.0F;

  mse = (float) ( (sigma*sigma) -
          ((Q * Math.sqrt(2.0) * sigma * Math.exp(-Q / (sigma * Math.sqrt(2.0)))) /
        (1.0 - Math.exp(-Q * Math.sqrt(2.0) / sigma))));
  return ( mse > (sigma*sigma) ? (sigma*sigma) : mse );

}


public void processarImatge(){

    float [][]MatriuQf = new float[COLORS][DCTSIZE2];

        // També li passo la funció phi triada.
    if ( BotoRadio1.isSelected() )
            phi=1;
    else if ( BotoRadio2.isSelected() )
            phi=2;

    MatriuQf =
calculamatriuQ(MatriuMS,Float.valueOf(getEtiqValorPSNR().getText()).floatValue(), phi);
    matriuQ = new int[COLORS][DCTSIZE2];

    for( int col=0; col< COLORS; col++){
        for(int i=0;i<MatriuQf[col].length;i++){
            Float Ftemp = new Float(MatriuQf[col][i]);
            matriuQ[col][i]=Ftemp.intValue();
        }
    }
    omplirMatriuQ();
    crearJPEG();
}

public void omplirMatriuQ(){

    for( int col=0; col< COLORS; col++){
        for(int i=0;i<DCTSIZE2;i++){
            Float Ftemp = new Float(matriuQ[col][i]);
            TMatriuQ[col][i].setText(Integer.toString(Ftemp.intValue()));
        }
    }

}

public void reprocessarImatge(){

    for( int col=0; col< COLORS; col++){
        for(int i =0;i<DCTSIZE2;i++)
            matriuQ[col][i]=Integer.valueOf(TMatriuQ[col][i].getText()).intValue();
        }
    crearJPEG();
}

public void crearJPEG(){
    FileOutputStream out1 = crearOutput( NomFitxerSortida );
    pIm1 = new JPEGEncodeParam();

    pIm1.setQTable(0,0,matriuQ[0]);
    pIm1.setQTable(1,0,matriuQ[1]);
    pIm1.setQTable(2,0,matriuQ[2]);
    eim1 = ImageCodec.createImageEncoder("JPEG", out1, pIm1);
    try{
        eim1.encode(im0i);
        out1.close();
        }
    catch(IOException e)
        {
        System.out.println("Error al tancar el fitxer de sortida");
        }
```

```
    carregaImatge( NomFitxerSortida , false);
    File sortida = new File( NomFitxerSortida );
    getJLabComp().setText("COMPRIMIDA: "+ NomFitxerSortida);
    getJTextTamComp().setText(Long.toString(sortida.length()));

}

public FileOutputStream crearOutput(String outFile){

    FileOutputStream out;

    try
        {
        out=new FileOutputStream(outFile);
        return out;
        }
    catch(IOException e)
        {
        System.out.println("Error al crear el fitxer");
        }
    return null;
}



/*****************************************************************************
 * Operació que carrega una imatge (original o comprimida) depenent del *
 * paràmetre entrada. La operació consisteix en llegir la imatge del disc,   *
 * escalar-la per que capigui a la pantalla.                                 *
 *****************************************************************************/
public void carregaImatge(String imageName, boolean entrada){

    getPanellImatge().setVisible(false);
            // Si entrada=TRUE llavors és la imatge original,
            // altrament és la comprimida.
    if(entrada)
        {
        // Llegim la imatge
        im0i = JAIImageReader.readImage(imageName);

        // Creem un bloc de parametres que servirà per anar
        // afegint parametres de la imatge.
        ParameterBlock pbi = new ParameterBlock();
        pbi.addSource(im0i);

                // Calculem la escala.
        float scale = 190.0F/Math.max(im0i.getWidth(),im0i.getHeight());

                // Afegim els parametres necessaris per
                // escalar i presentar la imatge
        pbi.add(scale);
        pbi.add(scale);
        pbi.add(0.0F);
        pbi.add(0.0F);
        pbi.add(new InterpolationNearest());
        RenderedImage im1 = JAI.create("scale", pbi);
                    // Creem una zona de dibuix amb la imatge
                    // i l'afegim al panel corresponent
        ImageCanvas imageCanvas = new ImageCanvas( im1, true );
        JPanel pan = new JPanel();
        pan.add( imageCanvas);
            getPanellImatge().setVisible(false);
            getJPan1_4().setVisible(false);
            getJPan1_4().removeAll();
            getJPan1_4().add(pan);
            getJPan1_4().setVisible(true);
        }
    else
        {
                    // Llegim la imatge
        im0o = JAIImageReader.readImage(imageName);

                // Creem un bloc de parametres que servirà
                // per anar afegint paràmetres de la imatge.
        ParameterBlock pbo = new ParameterBlock();
        pbo.addSource(im0o);
```

```
                        // Calculem l'escala.
        float scale = 190.0F/Math.max(im0o.getWidth(), im0o.getHeight());

                        // Afegim els paràmetres necessaris per
                        // escalar i presentar la imatge
        pbo.add(scale);
        pbo.add(scale);
        pbo.add(0.0F);
        pbo.add(0.0F);
        pbo.add(new InterpolationNearest());
        RenderedImage im1 = JAI.create("scale", pbo);

                        // Creem una zona de dibuix amb la imatge
                        // i l'afegim al panel corresponent
        ImageCanvas imageCanvas = new ImageCanvas( im1, true);
        JPanel pan = new JPanel();
        pan.add( imageCanvas);
            getJPan1_5().setVisible(false);
            getJPan1_5().removeAll();
            getJPan1_5().add(pan);
            getJPan1_5().setVisible(true);
        }
                    // Finalment fem visible el Panell Principal
    getPanellImatge().setVisible(true);

}  /*-------- Final del mètode  CARREGA IMATGE -----------*/


//
//      SEGON PANELL
// Panell per mostrar la matriu Q
private void setPanellMatriu(){

    int factor=0;
    int col=0;

    TMatriuQ = new JTextField[COLORS][DCTSIZE2];

    // 1a primera component una mica més gran
    for(int i=0;i<8;i++){
        for(int j=0;j<8;j++){
            TMatriuQ[col][j+factor] = new JTextField("",4);
            TMatriuQ[col][j+factor].setBounds(150+i*32,25+j*22,30,20);
            TMatriuQ[col][j+factor].setEditable(true);
            TMatriuQ[col][j+factor].setFont(new java.awt.Font("Arial", 1, 10));
            getPanellMatriu().add(TMatriuQ[col][j+factor]);
            }
        factor+=8;
        }

    // 2a i 3a component una mica més petites.
    for( col=1; col< COLORS; col++){
        factor=0;
        for(int i=0;i<8;i++){
            for(int j=0;j<8;j++){
                TMatriuQ[col][j+factor] = new JTextField("",4);
                TMatriuQ[col][j+factor].setBounds(2+(col-1)*300+i*32,210+j*22,30,20);
                TMatriuQ[col][j+factor].setEditable(true);
                TMatriuQ[col][j+factor].setFont(new java.awt.Font("Arial", 1, 10));
                getPanellMatriu().add(TMatriuQ[col][j+factor]);
            }
            factor+=8;
        }
    }
}


// Panell per mostrar la matriu de Desviacions.
//
private void setPanellDesviacio(){

    int factor=0;
    int col=0;

    TMatriuS = new JTextField[COLORS][DCTSIZE2];
```

```
    // 1a primera component una mica més gran
    for(int i=0;i<8;i++){
        for(int j=0;j<8;j++){
            TMatriuS[col][j+factor] = new JTextField("",6);
            TMatriuS[col][j+factor].setBounds(100+i*42,25+j*22,40,20);
            TMatriuS[col][j+factor].setEditable(false);
            TMatriuS[col][j+factor].setHorizontalAlignment(JTextField.LEFT);
            TMatriuS[col][j+factor].setFont(new java.awt.Font("Arial", 1, 10));
            getPanellDesviacio().add(TMatriuS[col][j+factor]);
        }
        factor+=8;
    }

    // 2a i 3a component una mica més petites.
    for( col=1; col< COLORS; col++){
        factor=0;
        for(int i=0;i<8;i++){
            for(int j=0;j<8;j++){
                TMatriuS[col][j+factor] = new JTextField("",6);
                TMatriuS[col][j+factor].setBounds(2+(col-1)*300+i*32,210+j*22,30,20);
                TMatriuS[col][j+factor].setEditable(false);
                TMatriuS[col][j+factor].setHorizontalAlignment(JTextField.LEFT);
                TMatriuS[col][j+factor].setFont(new java.awt.Font("Arial", 1, 9));
                getPanellDesviacio().add(TMatriuS[col][j+factor]);
            }
            factor+=8;
        }
    }

}
// Panell per mostrar la matriu de Mitjanes
//
private void setPanellMitjana(){

    int factor=0;
    int col=0;
    TMatriuM = new JTextField[COLORS][DCTSIZE2];

        // 1a primera component una mica més gran
    for(int i=0;i<8;i++){
        for(int j=0;j<8;j++){
            TMatriuM[col][j+factor] = new JTextField("",6);
            TMatriuM[col][j+factor].setBounds(100+i*42,25+j*22,40,20);
            TMatriuM[col][j+factor].setEditable(false);
            TMatriuM[col][j+factor].setFont(new java.awt.Font("Arial", 1, 10));
            getPanellMitjana().add(TMatriuM[col][j+factor]);
        }
        factor+=8;
    }

    // 2a i 3a component una mica més petites.
    for( col=1; col< COLORS; col++){
        factor=0;
        for(int i=0;i<8;i++){
            for(int j=0;j<8;j++){
                TMatriuM[col][j+factor] = new JTextField("",6);
                TMatriuM[col][j+factor].setBounds(2+(col-1)*300+i*32,210+j*22,30,20);
                TMatriuM[col][j+factor].setEditable(false);
                TMatriuM[col][j+factor].setFont(new java.awt.Font("Arial", 1, 9));
                getPanellMitjana().add(TMatriuM[col][j+factor]);
            }
            factor+=8;
        }
    }

}
/*-------- Final dels mètodes de la classe ------------*/
```

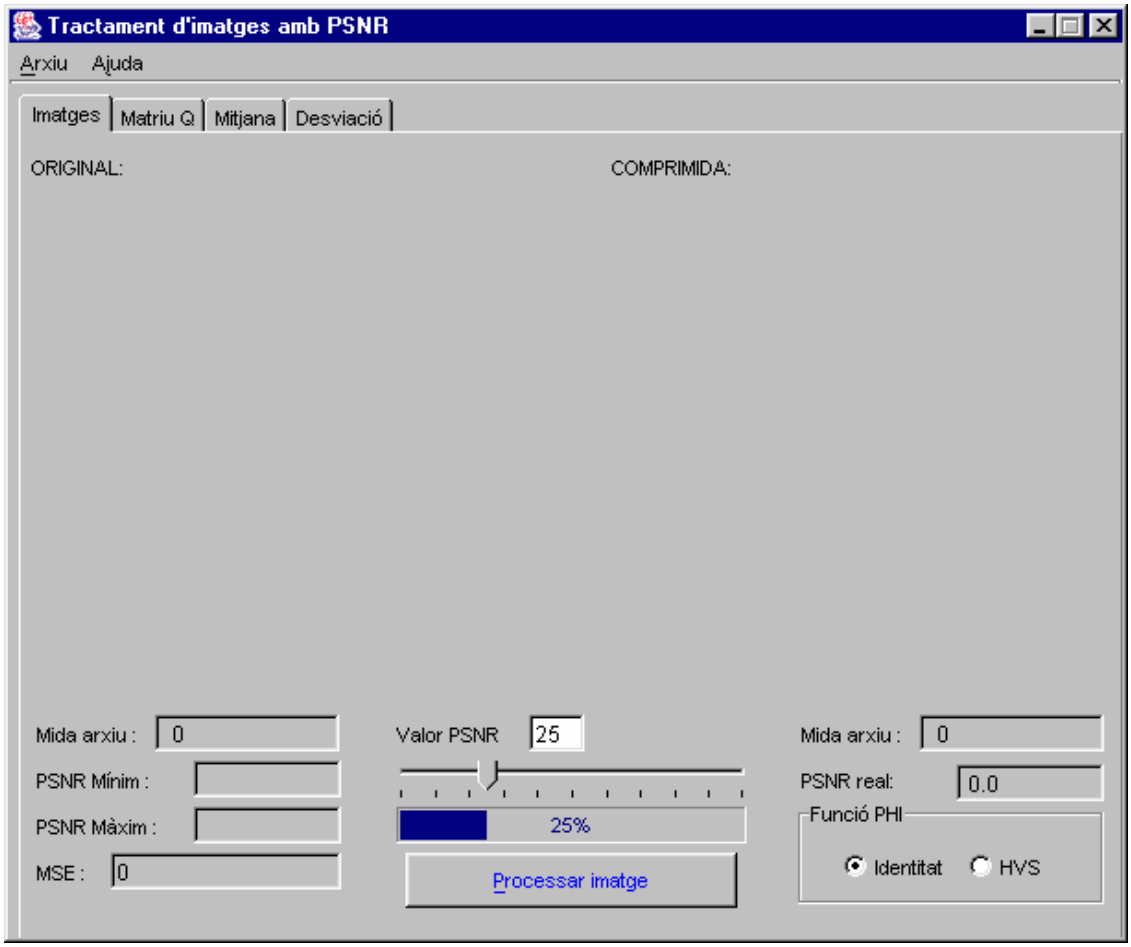
```
/************************************************************
 *
 * Mètodes auxiliars per millorar l'aspecte de l'aplicació.
 *
 ************************************************************/

public JFileChooser creaFileChooser() {

        // crea un filechooser
    JFileChooser fc = new JFileChooser();

        // Especifico el directori amb les imatges.
        // Per defecte faig que sigui el directori actual
        // on està executant-se l'aplicació.
    fc.setCurrentDirectory(new File(System.getProperty("user.dir")));

    return fc;
}

/**
 * INICI de l'aplicació. Aquest mètode és el que es crida quan s'executa l'aplicació.
*/
public static void main(java.lang.String[] args){

    try{
        /* look and feel del sistema que estiguem utilitzant Windows, Linux, ... */
        UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

        /* Create the frame */
        tfc atfc = new tfc();

        /* Calcula la mida de la pantalla */
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

        /* Centra la finestra en la pantalla */
        Dimension frameSize = atfc.getSize();
        if (frameSize.height > screenSize.height)
                frameSize.height = screenSize.height;
        if (frameSize.width > screenSize.width)
                frameSize.width = screenSize.width;
        atfc.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height -
                frameSize.height) / 2);

        /* Afegeix un controlador de finestres pels possibles events de la finestra */
        atfc.addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosed(java.awt.event.WindowEvent e) {
                System.exit(0);
            };
        });
        atfc.setVisible(true);

    } catch (Throwable exception) {
        System.err.println("Ha ocurrit una Excepció en main() de tfc");
        exception.printStackTrace(System.out);
    }
}

/**
*/
private void mostrarIndexAjuda(java.awt.event.ActionEvent arg1) {
    try {
        this.showAjuda();
    } catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
    }
}
/**
*/
private void mostrarQuantA(java.awt.event.ActionEvent arg1) {
    try {
        this.showAboutBox();
    } catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
    }
}
```

```
/**
 * MenuItemObrir. OBRIR ARXIU
 * Acció quan es pulsi per obrir un arxiu.
 * @param arg1 java.awt.event.ActionEvent
 */
private void ObrirArxiu(java.awt.event.ActionEvent arg1) {
    try {
        JFrame fi = new JFrame();
        JFileChooser fd =  creaFileChooser();

                        // Presento el quadre de dialeg.
        fd.showDialog(fi, "Obrir" );
                        // Obtinc el nom absolut del fitxer.
        NomFitxer = new String(
                            (fd.getSelectedFile()).getAbsolutePath());

        File f = new File( NomFitxer );
        getJLabOrig().setText("ORIGINAL: "+ (fd.getSelectedFile()).getName() );
        getJTextTamOrig().setText( Long.toString(f.length()) );

        carregaImatge(NomFitxer, true);
        descriptors(); //Calcula els descriptor estadísitics
        getJTextPsnrMin().setText( Float.toString(MSEmaxim(MatriuMS)));
        getJTextPsnrMax().setText( Float.toString(MSEminim(MatriuMS)));
        getJTextMSE().setText(new Double(Math.pow(255,2)/Math.pow(10,
               ((double)Double.valueOf(getEtiqValorPSNR().getText()
                   ).doubleValue()/10))).toString().substring(0,6));

        calculaPSNR();
        getJPsnrReal().setText("PSNR real : "+PsnrReal);

        NomFitxerSortida="out.jpg";
        processarImatge();
    } catch (java.lang.Throwable ivjExc) {
        System.out.println("Excepció a l'obrir el fitxer amb la imatge.");
        handleException(ivjExc);
    }
}
/**
 * Comment: ACCIO quan s'hagi pulsat 'Desar a'
 */
public void save_AsMenuItem_ActionPerformed(java.awt.event.ActionEvent actionEvent) {

    JFrame fi = new JFrame();
    JFileChooser fd = new JFileChooser();
                        // Selecciono el directori actual
    fd.setCurrentDirectory(new File(System.getProperty("user.dir")));
                        // Presento el quadre de dialeg.
    fd.showSaveDialog(fi);
                        // Obtinc el nom absolut del fitxer.
    String NomFitxer = new String(
                            (fd.getSelectedFile()).getAbsolutePath());
                        // Presento el resultat per veure el fitxer.
    JOptionPane.showMessageDialog( this,
                      "Has escrit:   "+ NomFitxer, "TFC       Desar Fitxer",
                      JOptionPane.INFORMATION_MESSAGE );

    NomFitxerSortida=fd.getSelectedFile().getName();
    getJLabComp().setText("COMPRIMIDA: "+NomFitxerSortida);
    crearJPEG();
    return;
}

/**
 * Comment
 */
public void saveMenuItem_ActionPerformed(java.awt.event.ActionEvent actionEvent) {

    String resposta = JOptionPane.showInputDialog( this,
                "Vols desar l'arxiu "+NomFitxerSortida, "TFC Desar fitxer",
                 JOptionPane.DEFAULT_OPTION );

    getJLabComp().setText("COMPRIMIDA: "+NomFitxerSortida);
    crearJPEG();
    return;
}
```

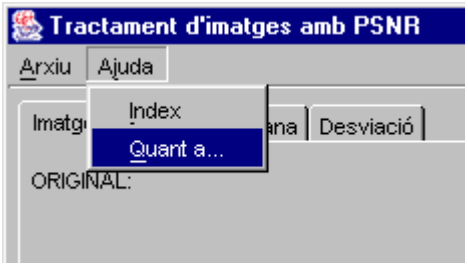
```
public void showAboutBox() {
    /* Create the AboutBox dialog */
    tfcAboutBox atfcAboutBox = new tfcAboutBox();
    Dimension dialogSize = atfcAboutBox.getPreferredSize();
    Dimension frameSize = getSize();
    Point loc = getLocation();
    atfcAboutBox.setLocation((frameSize.width - dialogSize.width) / 2 + loc.x,
                        (frameSize.height - dialogSize.height) / 2 + loc.y-50);
    atfcAboutBox.setModal(true);
    atfcAboutBox.show();
}

public void showAjuda() {
        /* Crea una finestra de tipus dialog per mostrar l'Ajuda */
    tfcAjuda atfcAjuda = new tfcAjuda();

    Dimension dialogSize = atfcAjuda.getPreferredSize();
    Dimension frameSize = getSize();
    Point loc = getLocation();
    atfcAjuda.setLocation((frameSize.width - dialogSize.width) / 2 + loc.x,
                    (frameSize.height - dialogSize.height) / 2 + loc.y-100);
    atfcAjuda.setModal(true);
    atfcAjuda.show();
 }

}
/************************************************
         Final de la classe tfc
************************************************/
```

Annex C - Pantalles de l'aplicació

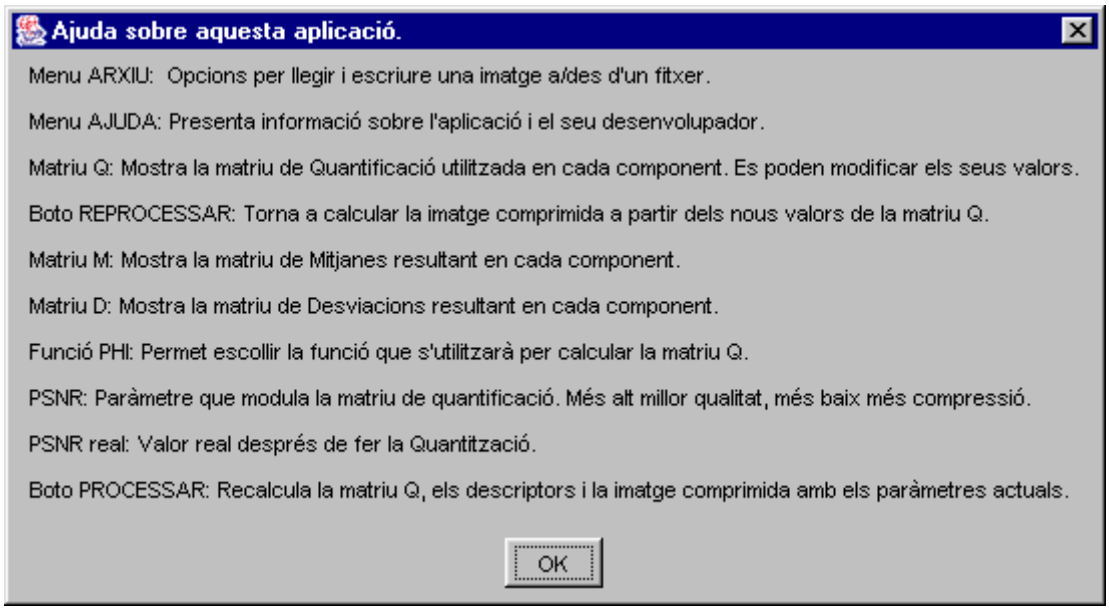
Pantalla inicial

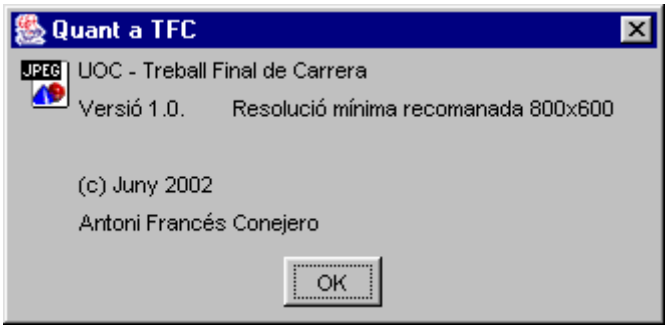


Menús arxiu i ajuda



Menú ajuda

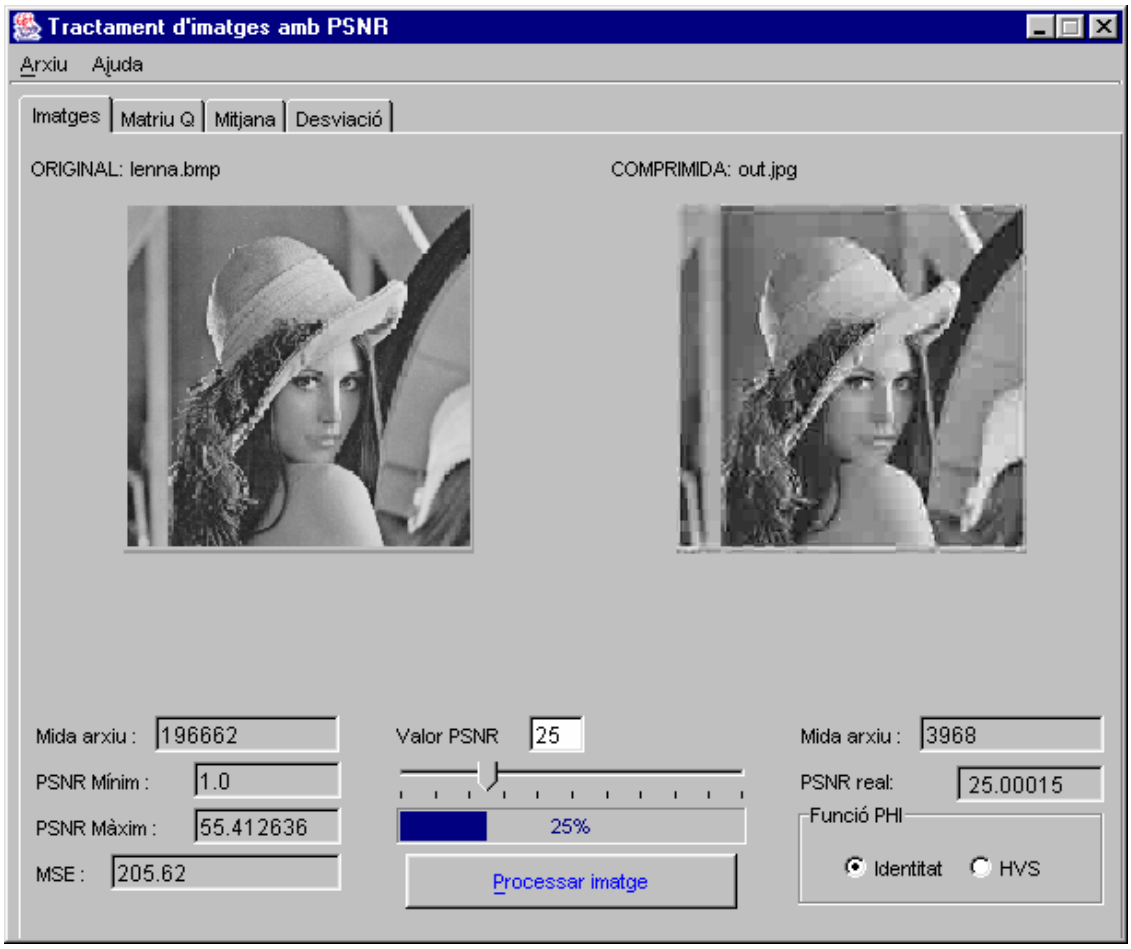




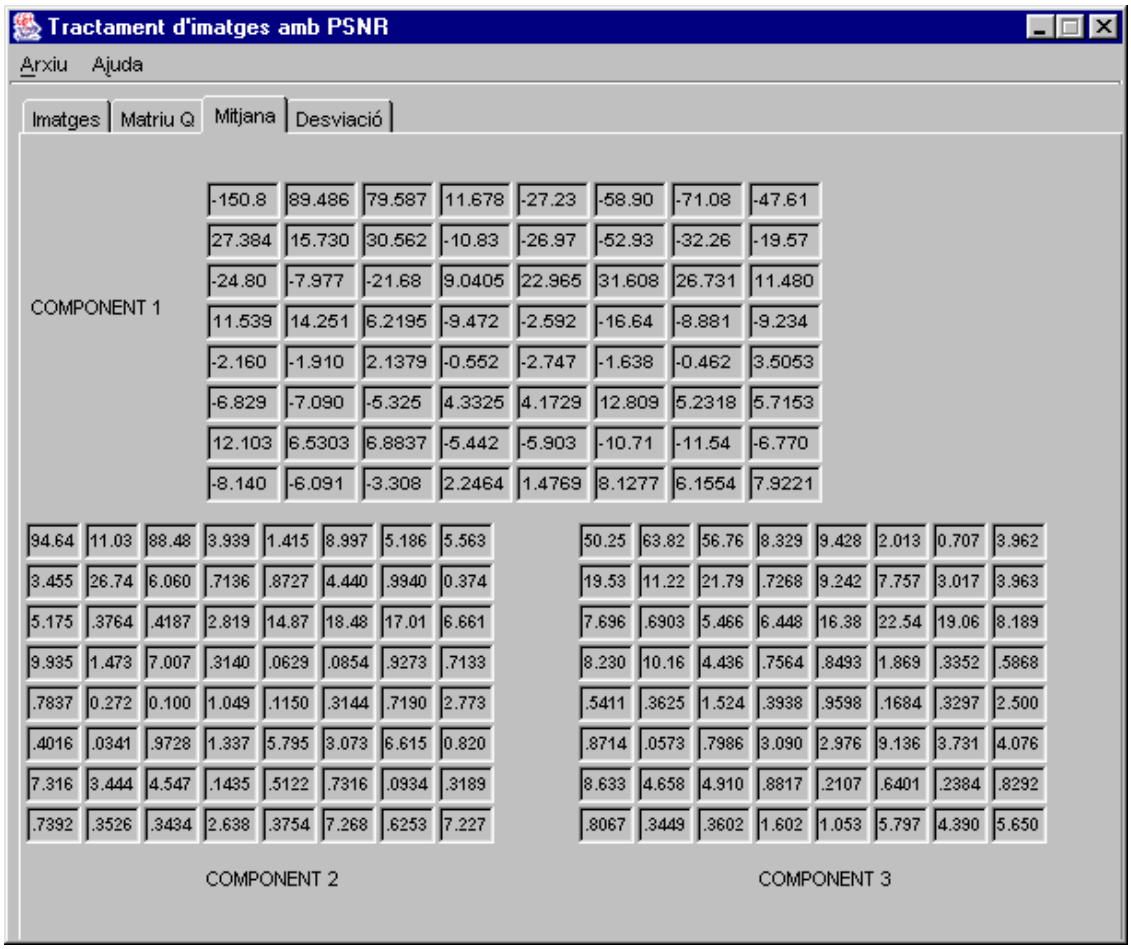
Quadre de diàleg per obrir una imatge

Quadre de diàleg per desar una imatge comprimida

Pantalla amb les imatges

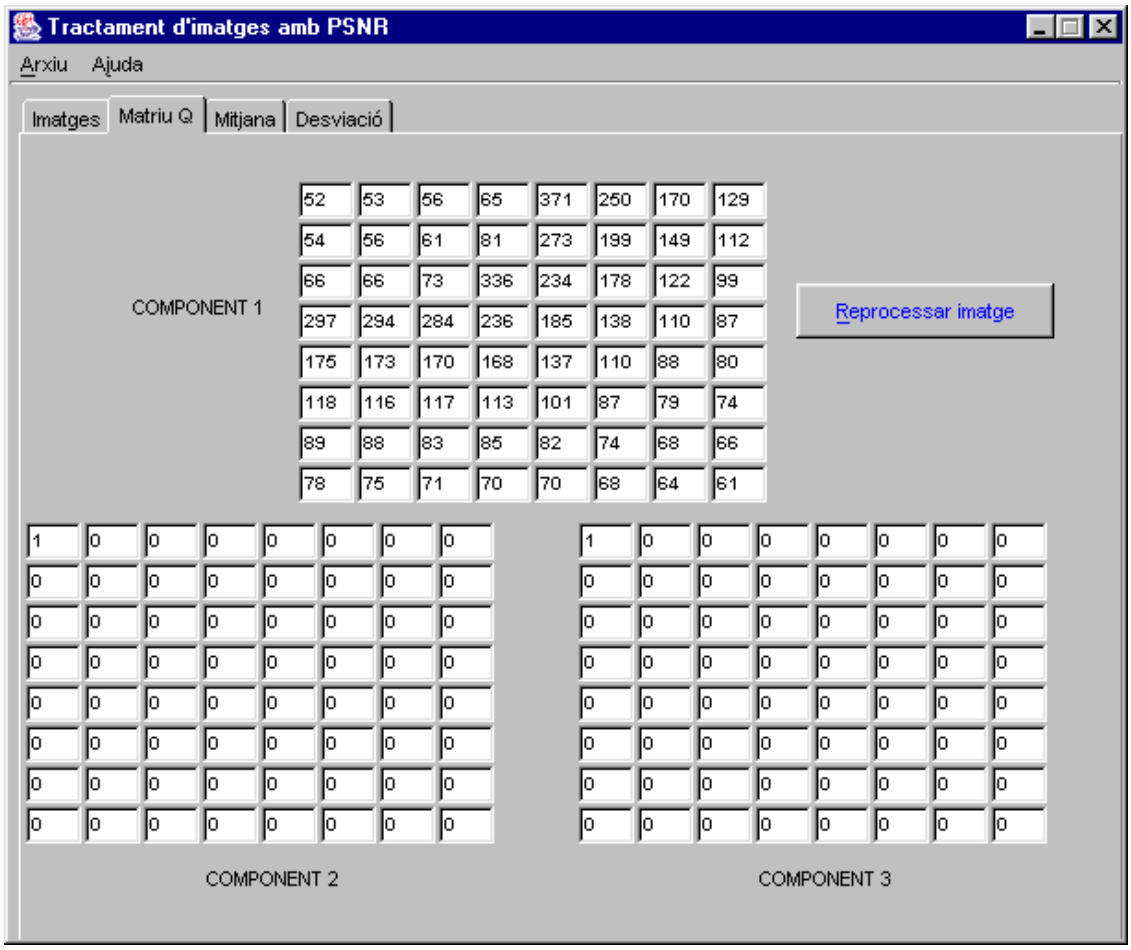


Pantalla amb la matriu de mitjanes, una per cada component



igualment per les desviacions.

Pantalla amb la matriu de quantització



permet modificar els valors en cada component i recalcular la imatge comprimida amb els nous valors.

Annex E - Exemple amb la imatge lenna.bmp.

Imatge original: tamany en bytes 270.333

Imatge comprimida	PSNR	Tamany en bytes
	20	6.955
	25	8.750
	30	13.600