CODIFICACIÓ I TRANSMISSIÓ INTERACTIVA D'IMATGES EN DISPOSITIUS MÒBILS.

Memória del projecte de final de carrera corresponent als estudis d'Enginyeria Superior en Informàtica presentat per Juan Muñoz Gómez i dirigit per Joan Bartrina Rapesta.

Bellaterra, 16 de Juny de 2008

El firmant, Joan Bartrina Rapesta , professor del Departament d'Enginyeria de la Informació i de les Comunicacions de la Universitat Autònoma de Barcelona

CERTIFICA:

Que la present memòria ha sigut realitzada sota la seva direcció per Juan Muñoz Gómez

Bellaterra, 16 de Juny de 2008

Firmat: Joan Bartrina Rapesta

A les tres dones amb més paciència del món

Agraïments

Aquest projecte ha pogut ser realitzat gracies a moltes persones, però en especial vull donar les gràcies al meu director de projecte, Joan Bartrina, per la seva gran paciència, dedicació i esforç realitzat. També vull agrair a tots els membres del grup GICI per les seves explicacions que han sigut importantissimes.

Índex

Índex de figures

Índex de taules

Capítol 1

Introducció

Recentment, la gran utilització de dispositius mòbils i la facilitat d'accés a la xarxa permeten als usuaris a accedir a grans bases de dades d'imatges. Aquestes grans bases de dades es composen d'imatges d'una mida considerable i, emmagatzemar-les, obrir-les o fins i tot transferir-les comporta el consum de molts recursos. Per reduir, el temps de transmissió, l'espai utilitzat, etc. s'utilitzen diferents sistemes de compressió d'imatges que ens permeten accedir a la part que ens interessa de la imatge sense haver-la de tenir tota en memòria i emmagatzemar-la utilitzant el mínim d'espai de disc.

El Joint Photographic Experts Group ha impulsat el desenvolupament del estàndard JPEG2000[JPEG] per la compressió i manipulació d'imatges, amb la intenció de cobrir el major nombre de requeriments citats anteriorment. JPEG2000 té la capacitat de poder comprimir eficientment imatges de volums superiors als Tera-Bits, aconseguint raons de compressió molt efectives i permeten la recuperació, pràcticament immediata, de qualsevol zona de la imatges a la resolució i qualitat desitjada.

1.1 Transmissió interactiva d'imatges

Podem parlar de interactivitat en imatges quan codifiquem una imatge i la descodifiquem de diferents maneres. Es a dir, codificar una vegada una imatge i per

exemple, només descodificar la porció de la imatge que necessitem, o bé aquesta mateixa imatge codificada descodificar-la a una mida més petita a l'original amb un nivell de qualitat menor. La transmissió interactiva d'imatges es pot definir com la transmissió de dades que es necessiten per poder descodificar aquesta imatge sol·licitada.

1.2 Motivacions del projecte

Cada vegada hi ha més necessitat i modalitats diferents d'accés a la informació. Les noves tecnologies estan permeten desenvolupar noves aplicacions basades en noves maneres de comunicació, amb el temps, s'estan convertint en necessitats dels ciutadans en general. Entre totes les fonts d'informació, tenen un paper molt destacat les imatges. En alguns centres (diagnòstic mèdic, teledetecció, meteorològics,...) les imatges són adquirides i utilitzades en el funcionament diari de les seves activitats, siguen d'especial importància tan la forma en que s'emmagatzema que es requereixi un accés a aquestes imatges des de qualsevol ubicació (àmbits urbans, treball de camp, etc.). La transmissió no només permet compartir imatges, sinó també especificar dinàmicament els requisits desitjats pel receptor referents a la resolució, la qualitat, les regions d'interès, etc. de la imatge, es diu que la transmissió és interactiva. Un estudi de l'ús de les imatges en centres on s'utilitzen extensament ens permet deduir que es convenient dissenyar aplicacions per a dispositius mòbils que permetin la transmissió interactiva d'imatges per canals segurs, permeten l'intercanvi d'imatges i meta-dades.

1.3 Objectius

L'objectiu d'aquest projecte són introduir-se en el món de la codificació d'imatges i més profundament en la transmissió interactiva d'aquestes i introduir aquests conceptes en el món del dispositius mòbils. Per poder assolir aquest objectiu, el projecte es basa en el desenvolupament d'un client JPIP en un dispositiu mòbil. Per el correcte desenvolupament d'aquest projecte, aquest objectiu l'hem frag-

mentat en tres subobjectius:

- Conèixer l'estàndard JPEG2000 i el seu protocol JPIP per poder entendre com funciona la nostra aplicació.
- Conèixer les diferents plataformes mòbils del mercat i les possibilitats d'implementació, mitjançant un estudi d'elles i comparar les seves possibilitats.
- Adaptació d'un client JPIP i el desenvolupament d'un visor en un dispositiu mòbil.

Una vegada definits els objectius per poder assolir-los el projecte ha estat dividit es set fases ben diferenciades:

1. Planificació del projecte.
2. Estudi del estàndard JPEG2000 i del seu protocol de transmissió d'imatges JPIP.
3. Estudi de les diferents plataformes mòbils i possibilitats de desenvolupament.
4. Redacció de l'informe previ.
5. Adaptació del client JPIP facilitat a una plataforma mòbil.
6. Implementació i test del visor.
7. Redacció de la documentació.

1.4 Contingut de la memòria

La memòria està estructurada en en set capítols i dos annexos. El segon i tercer capítol presenten JPEG2000, JPIP i les plataformes mòbils que hi ha al mercat. El capítol quatre, cinc i sis exposen la implementació realitzada, els resultats obtinguts i la planificació realitzada en aquest projecte. En l'ultim capítol, mostrarem

les conclusions i les línies futures. L'annex 1 d'aquesta memòria és un petit manual de l'aplicació, que explica el seu funcionament i com s'ha d'instal·lar, i en l'annex 2 trobem l'informe previ realitzat.

Capítol 2

JPEG2000

JPEG2000 és l'últim estàndard desenvolupat per el Joint Photographic Experts Group (JPEG). Aquest estàndard està adreçat a la descodificació, transmissió, seguretat i manipulació d'imatges i vídeo. Es considera que ha sigut l'avanç més significatiu en el món de la codificació d'imatges donat que aporta una idea bàsica, codificar una vegada i descodificar de moltes diferents.

JPEG2000 està dividit en 12 parts diferents, cadascuna descrivint les funcionalitats que vol incorporar l'estàndard.

- Part 1: Nucli del sistema de codificació.
- Part 2: Extensions del sistema de compressió.
- Part 3: JPEG2000 Motion, defineix el format del fitxer MJP2, per a seqüències d'imatges en moviment.
- Part 4: Es refereix a les proves conforme a JPEG2000 Part 1.
- Part 5: Programari de referència.
- Part 6: Format d'imatge compost.
- Part 7: Realment no existeix aquesta part, però podria descriure una possible implementació hardware.

- Part 8: JPSEC, Aspectes de seguretat a JPEG2000.
- Part 9: JPIP, protocol de transmissió interactiva d'imatges.
- Part 10: JP3D, compressió d'imatges volumètriques.
- Part 11: JPWL, aplicacions wireless.
- Part 12: Format d'arxiu multimèdia.

Actualment totes les parts són estandarditzades excepte la Part 7 que s'ha abandonat i la Part 10 que està en període d'estandardització.

En aquest capítol explicarem la part 1 i la part 9 de l'estàndard, que són les necessàries per comprendre i poder implementar un client del protocol JPIP, ja que aquest requereix del descodificador (part 1) per recuperar la imatge que ens envia el servidor.

2.1 Part 1: Descodificador i codificador

La part 1 [ISO00] de l'estàndard fa referència al nucli del sistema de codificació i descodificació d'imatges, la sintaxi d'un codestream JPEG2000, així com la organització del fitxer jp2.

El sistema de compressió JPEG2000 es basa en la Discrete Wavelet Transform [ABMD92] (DWT) i en el paradigma de compressió Embedded Block Coding with Optimized Truncation [Tau00] (EBCOT), basat en la divisió en blocs, que un cop codificats s'anomenen code-blocks, de cada una de les subbandes generades per la DWT.

La primera fase del codificador es el level shift, es basa en desplaçar els valors de l'imatge per tal de tenir una distribució de valors centrada en el 0. Una vegada desplaçats els valors, s'aplica la transformada de color, només en el cas d'imatges a color, i acte seguit s'aplica la transformada Wavelet. L'objectiu de la DWT és

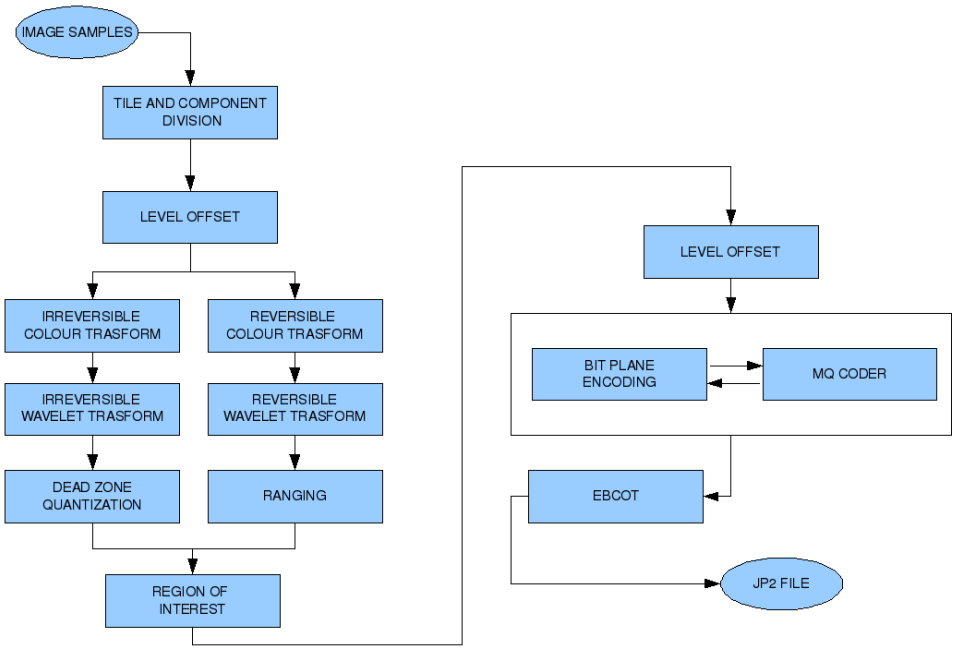
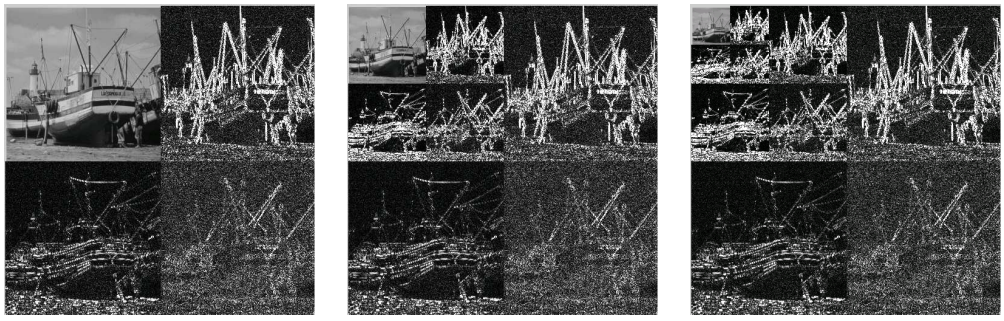


Figura 2.1: Diagrama JPEG2000



1 nivell de resolució 2 nivells de resolució 3 nivells de resolució

Figura 2.2: Imatge boats transformada a diferents nivells de resolució

descorrelacionar l'imatge. Això s'aconsegueix aplicant filtres pasa-alts i filtres pasa-baix a totes les files i columnes de la imatge generant una imatge dividida en quatre subimatges, anomenades subbandes, on la subbanda de a dalt a l'esquerra es la imatge original a un nivell de resolució més petit (1/4 de la mida original). Podem veure un exemple d'una imatge després d'haver passat per el procés de transformació a la figura 2.2. Es poden aplicar dos tipus de filtres en el procés de la DWT: el LeGall 5/3, i el Daubechies 9/7. El primer tipus s'utilitza per fer una codificació sense pèrdua (lossless) i la segona per fer una codificació amb pèrdua (lossy).

Una vegada aplicada la transformada comença l'etapa de quantització, en aquesta fase els valors en punt flotant resultants de la transformada es converteixen en valors enters per poder ser correctament interpretats en les etapes de codificació. Finalitzada aquesta fase, cada subbanda de la imatge es dividida en blocs de mida potència de 2 (amb un valor mínim 2^5).

Arribats a aquest punt, amb les subbandes dividides en blocs, comença la part de codificació pròpiament dita. Cadascun dels blocs es codifiquen utilitzant un codificador de plans de bits, on cadascuna de les matrius de bits en les que es transforma la nostra matriu d'enters definida per el nostre bloc. Cada pla de bits es codifica en tres passos de codificació: Significance Propagation, Magnutide Refinement i Clean-Up.

- **Significance Propagation:** Una posició d'un bitplane es significant si el seu valor es 1. En aquesta fase només es codifiquen els bits que tenen un dels seus 8 veïns significants.
- **Magnitude Refinement:** Només es codifiquen els bits que han esdevinguts significants en l'etapa anterior.
- **Clean-Up:** Es codifiquen la resta de bits.

A més de generar un bitstream, en aquesta fase es generen els contextos associats a cada bit que es codifica. Aquests seran utilitzats en la fase del codificador aritmètic MQ[TM02, Chapter 12.1] per ajustar millor les probabilitats del mateix, i així codificar més eficientment. Aquest codificador aritmètic es basa en

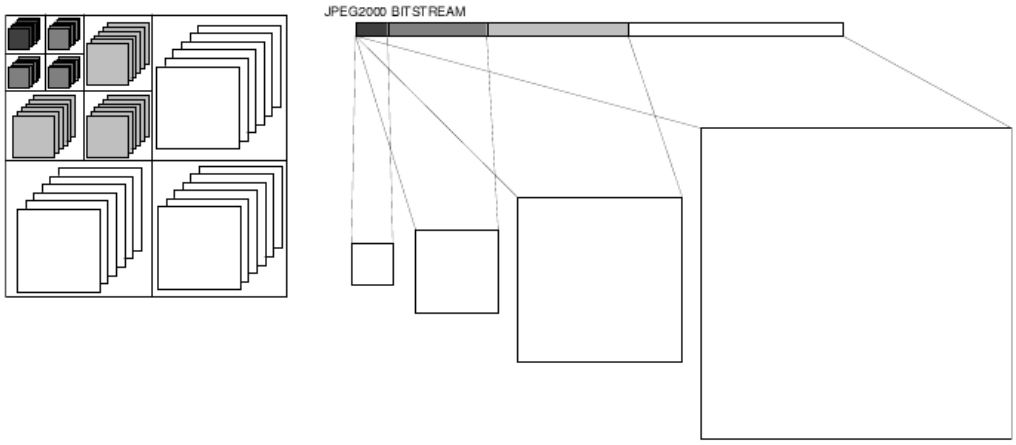


Figura 2.3: Creació d'un bitstream progressiu per resolució

la representació de la codificació com a un valor d'un interval, determinat per la freqüència d'aparició del símbol aparegut.

A continuació el Post Compression Rate Distorsion [Tau01] (PCRD-Opt) inclòs en el paradigma EBCOT , selecciona els bitstreams més òptims en quant a qualitat per al bit-rate [1] especificat.

Una vegada obtingut el bitstream resultant del PCRD-Opt podem ordenar-lo en diferents progressivitats. La progressivitat més interessant és la que organitza el bitstream per qualitat, és a dir, en els primers bits de la tira de dades tenim els que ens aporten més qualitat alhora de recuperar la imatge. A més de la progressivitat per qualitat tenim tres tipus més de progressivitat, per resolució (veure figura 2.3), on els primers bits aporten informació de cada nivell de resolució, per localització espacial, on es construeixen segons els precintes que necessitem per descodificar la zona espacial que ens interessa, on cada precinte són les agrupacions dels codeblocks del mateix nivell de resolució de les diferents subbandes que defineixen la mateixa regió de la imatge, i l'última progressivitat és per component, on es construeix el bitstream per components.

L'última fase de la codificació es la construcció del fitxer de la imatge codificada. Les imatges en JPEG2000 poden ser en dos tipus de fitxer. Un fitxer jpc on

[1] Nombre de bits per mostra.

només tenim les dades necessàries per descomprimir la imatge, o un fitxer jp2, on a més de la informació que conté el fitxer jpc hi han unes capçaleres que poden guardar meta-dades de la imatge.

2.2 Part 9: JPIP

JPIP [ISO05] és la Part 9 de l'estàndard JPEG2000 i descriu el protocol de transmissió interactiva d'imatges per aconseguir extreure eficientment les dades de les imatges JPEG2000. El protocol defineix les interaccions client-servidor basades en peticions del client i respostes del servidor.

Aquest protocol pot ser utilitzat sobre diferents protocols de transport encara que la recomanació de l'estàndard diu que utilitzem JPIP sobre el protocol HTTP utilitzant connexions TCP.

2.2.1 Arquitectura

El protocol JPIP, com podem observar a la figura 2.4, es basa en una arquitectura client-servidor, on el servidor conté un repositori d'imatges i els clients fan les diferents peticions al servidor.

Les peticions del client es basen en una View-Window per definir uns paràmetres de resolució, mida, localització, components, capes de qualitat i altres paràmetres de l'imatge per generar la petició. La resposta del servidor es basa en l'enviament de un stream de data-bins que seran rebuts per part del client. Inicialment el client pot enviar una petició al servidor per rebre només la informació de la imatge, per així poder generar una petició més refinada posteriorment.

Un altre element arquitectural és la caché de dades que conté el client on l'estructura d'aquesta es creada pel servidor, i enviada a al client, la qual s'anirà emplenant a mesura que el client va rebent les dades sol·licitades. Aquesta caché es basa en l'estructura de l'imatge organitzada en data-bins, aquests data-bins són els encarregats d'organitzar la informació de cada regió especifica de l'imatge, i per aquest motiu han de ser identificats de forma única. La funció de la caché és

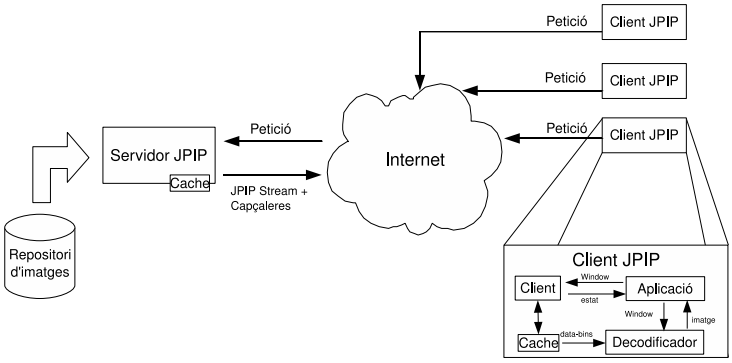


Figura 2.4: Arquitectura JPIP

guardar les data-bins rebuts, així no haver de tornar a sol·licitar-los si ja han estat enviats pel servidor prèviament.

2.2.2 Funcionament de JPIP

Com hem comentat abans el protocol JPIP es basa en les peticions que fa el client al servidor i les respostes que aquest envia al client. Aquestes peticions identifiquen la View-Window que volem visualitzar. Aquesta View-Window està composta per la regió espacial de la imatge que l'usuari vol visualitzar, el nivell de qualitat i el nivell de resolució que vol visualitzar.

Una vegada aquesta petició arriba al servidor, aquest genera una resposta. Aquesta resposta conté: l'estat de la petició, és a dir, si la petició és correcte, si el fitxer no es troba, etc, una capçalera amb la informació de l'imatge (qualitat, resolució, etc) i, el bitstream de la imatge sol·licitada. Aquesta resposta només conté informació de la regió, nivell de resolució, i nivell de qualitat que hem sol·licitat.

2.2.3 Implementacions de JPIP

Kakadu JPIK: Implementació realitzada per un dels autors del JPEG2000, David Taubman. Aquesta implementació esta realitzada en C++ i funciona sota Windows. Es pot descarregar una versió dels binaris a : http://www.kakadusofware.com

AccuRad JPIPStream: Implementació realitzada per l'empresa AWARE[Awa] i és una versió de pagament.

CADI: Implementació desenvolupada íntegrament al Group on Interactive coding of Images [GIC] de la UAB i es la que utilitzarem en el projecte. Està desenvolupada en Java, és totalment modular i té llicència GPL. Aquesta és la implementació que utilitzarem en aquest projecte. Es pot descarregar una versió a la web : http://cadi.sf.net

Capítol 3

Plataformes Mòbils

En aquest capítol farem una breu introducció al món de les plataformes mòbils, el desenvolupament d'aplicacions en Java per aquests dispositius i explicarem les diferents alternatives de desenvolupament que vam seleccionar per realitzar aquest projecte.

Una plataforma mòbil és un dispositiu de computació de mida reduïda que ens permet l'accés a la informació d'una forma semblant a la que ens ofereix un ordinador personal, es a dir, és un ordinador de mides reduïdes. En el mercat de les plataformes mòbils podem trobar diferents tipus de dispositius, PDA/PocketPcs o aparells de comunicació mòbil. Tot i que aquests dispositius tenen una finalitat diferent, últimament s'estan creant dispositius híbrids com les SmartPhone, que són dispositius de comunicació amb les funcionalitats que ens aporta una PDA/-PocketPC.

Abans de parlar dels diferents entorns de desenvolupament trobats, en introduirem el desenvolupament d'aplicacions amb J2ME [Mica]. J2ME (veure figura 3.1) és la versió de Java enfocada al desenvolupament per dispositius mòbils, conté dos tipus de màquines virtuals, la Kilobyte Virtual Machine (KVM) i la Compact Virtual Machine (CVM). Cadascuna de les màquines pot treballar amb les seves configuracions respectives i diferents perfils. En aquest cas, una configuració és el conjunt mínim d'APIs Java que permeten desenvolupar aplicacions per un determinat grup de dispositius depenent de les seves limitacions. D'aquestes con-

figuracions existeixen dues possibles, la Connected Limited Device Configuration (CLDC), i la Connected Device Configuration (CDC).

- CLDC: Aquesta configuració està orientada a dispositius dotats de connexió (wireless, bluetooth, GSM, etc) i amb limitacions en aspectes com la capacitat gràfica, còmput i memòria. Aquests dispositius han de complir unes característiques determinades, com per exemple, disposar com a mínim de 160KB de memòria per la màquina virtual.

- CDC: Aquesta configuració està orientada a dispositius amb una determinada capacitat de computació i de memòria, com per exemple les PDA. Aquesta configuració conté una màquina virtual similar a la què conté J2SE però amb limitacions en l'apartat gràfic i memòria de la màquina virtual.

Les configuracions en cap moment s'encarreguen del cicle de vida de les aplicacions, ni de la interfície gràfica o del tractament dels events, d'aquesta tasca s'encarreguen els perfils. Els perfils són un altre subconjunt de APIs orientat a un àmbit d'aplicació determinat, es a dir, els perfils donen a una configuració una funcionalitat específica. Cada configuració té uns perfils determinats, en el cas del CDC tenim el Personal Profile (PP) i el Foundation Profile (FP), i per CLDC tenim el perfil Mobile Information Device Profile (MIDP).

- PP: És un subconjunt de les llibreries de la J2SE que proporciona un entorn amb un complet suport gràfic amb les llibreries AWT. L'objectiu d'aquest perfil és dotar a la CDC d'una interfície gràfica.

- FP: Aquest perfil està orientat a dispositius que no requereixen d'una interfície gràfica, conté la majoria dels paquets de la J2SE però exclou els paquets AWT i SWING.

- MIDP: Aquest perfil defineix les capacitats del dispositiu i especifica les APIs relacionades amb el control de l'aplicació i interfície d'usuari, entre d'altres.

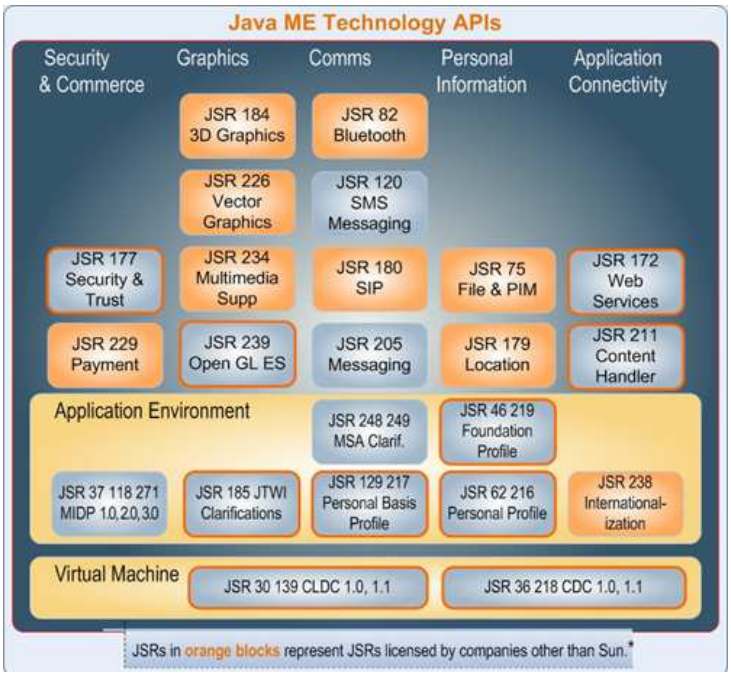


Figura 3.1: Configuracions, profiles i extensions de J2ME

3.1 Estudi de viabilitat

Una vegada introduïda la programació en Java per a plataformes mòbils, farem un estudi de viabilitat d'aquest projecte. Per poder realitzar aquest projecte es va fer un estudi de les diferents plataformes mòbils que hi ha actualment al mercat, que han de complir uns determinats requeriments no-funcionals presentats a continuació:

- Java : Com hem comentat abans la nostra plataforma ens ha d'oferir la possibilitat d'executar aplicacions fetes en Java, ja que la nostra aplicació d'origen està feta en Java, i com a conseqüència el nostre projecte també.

- Accés a Internet via Wireless : El protocol JPIP funciona sobre la suite de protocols TCP/IP.

- Simulador software de la plataforma : Per tal de reduir costos del projecte volem utilitzar un simulador enlloc de la plataforma física.

- L'entorn de desenvolupament i la plataforma han de ser programari lliure, encara que aquest requeriment no és imprescindible es valorarà.

Una vegada feta llista de requeriments no-funcionals del projecte, vam seleccionar les tres plataformes que vam trobar més viables per la realització d'aquest projecte.

- Nokia N95
- J9 WebSphere IBM
- Google Android

3.1.1 Nokia N95

Aquesta plataforma basada en el sistema operatiu Symbian ens ofereix un entorn de desenvolupament amb la configuració CLDC 1.1 i el MIDP 2.0. A més a més inclou extensions necessàries per poder comunicar-se via Wireless, BlueTooth i llibreries multimèdia per la càrrega d'imatges i gràfics 2D/3D. El desenvolupament per aquesta plataforma es pot realitzar amb Eclipse Me[Foub] o Netbeans with Mobility Pack[Micb], que son uns entorns de desenvolupament específics per desenvolupar aplicacions per aquest dispositiu. A més a més també ens hem de descarregar el simulador de la plataforma S60 de la pàgina del Forum Nokia, que és el portal de desenvolupament per plataformes de Nokia[Nok].

3.1.2 IBM WebSphere

Com podem observar a la figura 3.2, IBM WebSphere [IBM] ens ofereix un entorn de desenvolupament molt complert amb la possibilitat d'utilitzar les dues configuracions de Java possibles per a dispositius mòbils, CDC i, CLDC. També ofereix la possibilitat d'interactuar amb les llibreries d'interfície gràfica AWT. La plataforma IBM WebSphere és compatible amb qualsevol dispositiu mòbil amb Windows Mobile o Windows CE. L'entorn de desenvolupament és el WebSphere Everyplace Micro Environment basat en Eclipse[Foua] i és un entorn privatiu.

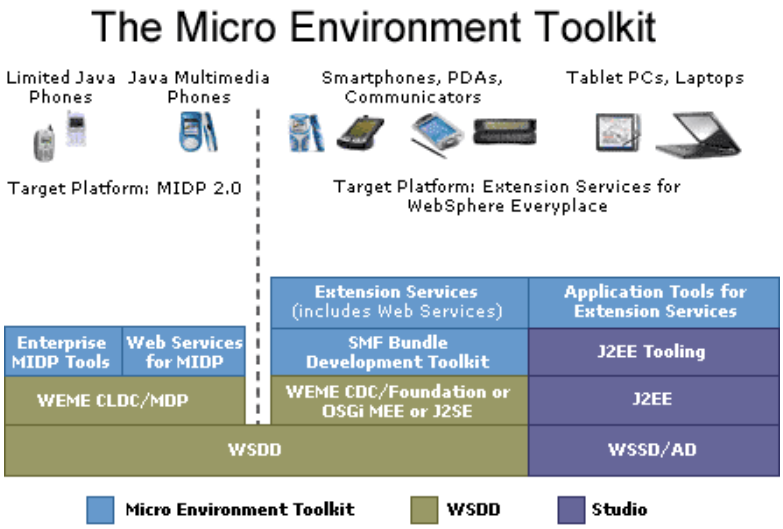


Figura 3.2: Possibilitats de desenvolupament amb IBM WebSphere

3.1.3 Google Android

Android [Inc07] és la nova plataforma per dispositius mòbils creada recentment per Google que inclou un sistema operatiu amb un kernel Linux (veure figura 3.3). Aquesta plataforma ofereix un entorn de desenvolupament que inclou una API de Java basada en la de Sun Microsystems. Inclou noves llibreries per la per la càrrega d'imatges, gràfics 2D/3D, i comunicació via Wireless. L'entorn de desenvolupament utilitzat és Eclipse amb un plugin per poder utilitzar les seves llibreries i per fer iniciar el simulador des del mateix entorn.

3.2 Conclusions

Després d'haver estudiat les tres plataformes es va optar per utilitzar la plataforma Google Android ja que era la única de les plataformes estudiades que assolia tots els requeriments no funcionals presentats de programari lliure. J9 WebSphere IBM va ser descartada encara que és la que ofereix una API i una solució més estesa, però tant la màquina virtual com el seu entorn de desenvolupament són de

Figura 3.3: Esquema de la plataforma Android

pagament i incompleix un dels requeriments que té aquest projecte, en canvi la plataforma de Nokia te l'entorn de desenvolupament gratuït però considerem que no és suficientment potent per el desenvolupament de la nostra aplicació.

Capítol 4

Cadi Mobile

Com hem comentat abans CADI és la implementació de la part 9 de l'estàndard JPEG2000, realitzada per el Group on Interactive Coding on Images (GICI) de la Universitat Autònoma de Barcelona (UAB), de la part 9 del estàndard JPEG200, el protocol JPIP. Aquesta implementació està formada per tres aplicacions, CADI-Client, CADIServer , i CADIViewer. En aquest projecte només ens hem centrat en la part del client, que és el què s'encarrega d'enviar les peticions als servidor i processar les dades rebudes.

Aquest client s'ha integrat dins d'un visor que hem desenvolupat per la plataforma Android, que en la segona secció d'aquest capítol explicarem.

4.1 Migració de CADI

En aquesta secció explicarem els canvis que s'han fet en el CADIClient per poder integrar-lo dins de la plataforma Android. Gràcies a que la API de Android és bastant complerta hem aconseguit no restar-li funcionalitat i integrar-lo completament.

Abans de començar a parlar dels canvis que s'han hagut de realitzar en CADIClient comentarem que a partir d'aquest moment quan parlem d'un package de l'aplicació ho farem de forma relativa, es a dir, que tots els packages que anirem esmentant de l'aplicació pengen del package *com.google.android*.

El primer canvi que hem realitzat va ser modificar el tipus de dades que CADIClient retorna com a imatge. CADIClient retorna una imatge de tipus *BufferedImage* de la llibreria *java.awt.Image* però, Android no conté cap llibreria del package *java.awt*. Per poder resoldre aquest inconvenient vam analitzar la utilitat de la classe *BufferedImage* en l'aplicació CADI. Una vegada analitzada la utilitat d'aquest tipus de objecte, es va detectar que només s'utilitzava per introduir els valors de la matriu imatge, retornada per la descodificació, i poder-se mostrar per pantalla, així que necessitàvem una classe que :

1. permetés assignar els valors RGB que ens dona la matriu,
2. fos visible des de una interfície gràfica o que pugues ser cridada des de una altre classe per poder ser visualitzada.

Per resoldre aquestes dues necessitats vam trobar la classe *Bitmap* de la llibreria *android.graphics* que ens cobreix totes les necessitats abans esmentades. La classe *Bitmap* no té constructor sinó que s'ha de cridar a un mètode estàtic d'aquesta classe per poder crear un objecte d'aquest tipus (veure figura 4.1). Els paràmetres d'aquest mètode són la resolució de la imatge (xSize, ySize) i un valor booleà depenent si la imatge te canal alpha o no.

```
Bitmap image = Bitmap.createBitmap(xSize, ySize, hasAlpha);
```

Figura 4.1: Mètode per crear Bitmaps

En canvi per assignar els valors de la matriu de l'imatge a aquesta classe no té un mètode directe sinó que primer s'ha de construir un valor sencer que contingui els valors RGB del pixel seleccionat, es a dir, que es codifica el RGB en 4 bytes. El primer pel canal alpha i els tres següents per el vermell, verd i blau. Aquest sencer es crea amb el mètode estàtic *Color.rgb(R,G,B)* on R,G,B han de ser sencers. En el nostre cas la nostra matriu ens retorna flotants i hem de canviar-los a sencers. A la figura 4.2 trobem el codi necessari per implementar l'assignació de color a la classe *Bitmap*.

```
int color;
for (int y = 0; y < ySize; y++) {
  for (int x = 0; x < xSize; x++) {
    for (int z = 0; z < zSize; z++) {
      if (zSize == 1)
        rgb[0]=rgb[1]=rgb[2]=(int)(imageSamplesFloat[z][y][x] +
            0.5);
      else
        rgb[z] = (int)(imageSamplesFloat[z][y][x] + 0.5);
    }
    color = Color.rgb(rgb[0],rgb[1],rgb[2]);
    bitmapImage.setPixel(x, y, color);
  }
}
```

Figura 4.2: Codi per assignar els valors de la matriu al bitmap

```
  private Bitmap imageSamplesToBitmappedImage(float[][][]
     imageSamplesFloat)
```

Figura 4.3: Prototip de la funció imageSamplesToBitmappedImage

Una vegada modificat el tipus de dades de la imatge que et retorna el CADIClient hem de modificar els mètodes que fan la conversió d'aquesta classe. La classe modificada es *Jpeg2KLogicalTarget* del package *cadi.Client.ClientLogicalTarget*. Aquesta classe s'encarrega de crear el LogicalTarget amb qui treballara JPIP el què hem de modificar és, el mètode *decode* i a més hem de crear la funció *imageSamplesToBitmappedImage* amb el prototip que mostrem a la figura 4.3. La primera funció s'encarrega de cridar a tots els mètodes utilitzats en la descodificació de l'imatge i el mètode creat s'encarrega d'agafar les dades de la matriu imatge que et retorna la descodificació i introduir-la en l'objecte de la classe *Bitmap*.

També s'ha modificat el mètode *getLogicalTargetDescription()* d'aquesta classe, que s'encarrega de retornar un string que conté tota la informació del LogicalTarget que estem treballant en format html per poder mostrar-ho en l'aplicació.

El sistema de logs de CADI també s'ha modificat ja que Android no conté els mètodes *System.println()* o *System.print()* per fer la sortida dels logs, sinó que inclou una classe especifica per fer els logs, la classe *Log* del package *android.Util*. Aquesta classe inclou mètodes per tots els possibles nivells de log descrits a la

Nivell de Log	Mètode a utilitzar
Debug	Log.println(int Priority,String Tag,String msg)
Vervose	Log.v(String Tag,String msg)
Assert	Log.d(String Tag,String msg)
Info	Log.i(String Tag,String msg)
Warn	Log.w(String Tag,String msg)
Error	Log.e(String Tag,String msg)

Taula 4.1: Diferents mètodes per fer logs a la plataforma Android

taula 4.1.

Per últim es va crear una nova classe afegida al package *cadi.Common.Util* utilitzada per poder fer proves de rendiment i consum de memòria. Aquesta classe esta formada per quatre mètodes estàtics que calculen el temps i el consum de memòria en cada fase de la descodificació i el temps total d'aquesta.

4.2 Cadi Mobile

Una vegada adaptat CADIClient a Android, explicarem la implementació de l'aplicació gràfica a la plataforma Android, però abans de començar a explicar la aplicació, farem una breu introducció al desenvolupament d'aplicacions en Android, per tal de poder entendre millor el seu funcionament.

Les aplicacions a la plataforma Android poden estar formades per quatre tipus blocs:

- Activity
- Intent Receiver
- Service
- Content Provider

Per crear les aplicacions a Android no són necessàries que siguin tots els blocs però si que han d'estar formades per una combinació d'aquests.

Activity: Una activity normalment és cada una de les finestres que té la nostra aplicació. Habitualment s'implementa tota la funcionalitat en una sola classe que està definida per una finestra principal formada per un seguit de Views que responen a events. Aquestes Views són els blocs bàsics per la implementació d'una interfície d'usuari.

Intent Receiver: Els intent receivers permeten que un codi específic s'executi en el moment que l'aplicació rep un input extern, com pot ser rebre trucada o un sms.

Service: Els service executen un codi durant un llarg període de temps i no necessiten una interfície gràfica, són semblants als dimonis dels sistemes unix-like, s'estan executant sense que l'usuari se n'adoni.

Content Provider: S'encarrega d'emmagatzemar les dades de l'aplicació en una base de dades, a més, la classe content provider, conté els mètodes necessaris per poder compartir aquestes dades amb altres aplicacions.

Una vegada explicats els quatre blocs bàsics amb els que podem desenvolupar una aplicació per Android, ara ja podem explicar el disseny de la nostra aplicació.par

La nostra aplicació està formada principalment per una Activity principal anomenada *CadiMobile*, que s'encarrega de la visualització de la imatge, i de fer les crides corresponents al CADIClient. A més a més tenim dues Activities més una encarregada de generar els formularis de la petició, i una altre que ens visualitza la descripció de la imatge que hem demanat.

A continuació descriurem les classes principals i els mètodes que contenen en la nostra aplicació.

CadiMobile

Aquesta classe es la principal de l'aplicació. S'hi defineixen els components del visor, els ítems del menú, paràmetres de l'imatge, i tots els mètodes que controlen

la interfície d'usuari.

```
public void onCreate(Bundle icicle)
```

Aquest mètode és el primer que s'executa de tota la aplicació, s'encarrega de d'inicialitzar la interfície gràfica.

```
View.OnKeyListener getKeyListener = new View.OnKeyListener()
```

Declaració d'una interfície per capturar events del teclat. Aquesta detecta quina tecla s'ha pressionat i realitza l'acció associada a cada tecla.

```
public void setTitle(CharSequence title)
```

Es modifica el valor del títol de l'aplicació.

```
public boolean onCreateOptionsMenu(Menu menu)
```

Mètode per generar el menú amb els ítems definits a la classe.

```
public boolean onMenuItemSelected(int featureId, Item item)
```

Aquest mètode es crida quan fem clic en un ítem del menú.

```
void loadImage(String fileUrl)
```

Generem una petició a partir del valor que tenim en la url.

```
void loadImageServer(String server,int port, String image)
```

Es genera una petició amb els valors que retorna el formulari de petició.

```
void loadImage()
```

Fem una petició quan es modifica qualsevol paràmetre de l'imatge (nivell de resolució, qualitat, etc).

```
protected void onActivityResult(int requestCode, int
    resultCode, String data, Bundle extras)
```

Mètode que es crida al retornar d'una subactivity, en el cas que retorni de la subactivity del formulari per crear la petició, crida la funció loadImageServer i, genera la petició amb els valors retornats.

Image

Aquesta classe s'encarrega del control del ImageView que mostra la imatge, i captura els events que es fan dins d'aquest ImageView, com són el zoom-in o el zoom-out, o el moviment de la imatge per la pantalla.

```
protected void onDraw(Canvas canvas)
```

Aquest mètode es crida en el moment de dibuixar a la pantalla. Fixa la imatge a dibuixar i els límits de la imatge dins del sistema de visualització.

```
public boolean onKeyDown(int keyCode, KeyEvent event)
```

Mètode que es crida en el moment de prémer en una tecla, determinem quina tecla s'ha pressionat i es realitza la acció determinada. En aquest cas al prémer el cursor podem moure la imatge per la pantalla o fer zoom quan premem el botó central del dispositiu.

```
public void zoomInOut()
```

Aquest mètode es crida quan des del menú es fa clic en l'opció de zoomIn/zoomOut, i modifica una variable per determinar si s'ha de fer zoom-in o zoom-out.

```
    public void setBitmap(Bitmap im)
```

En aquest mètode definim la imatge que es visualitzarà i determinem els seus

límits dins del nostre sistema de visualització.

ImageProperties

En aquesta classe visualitzem la descripció de la imatge que hem sol·licitat. Només conté el mètode *onCreate(Bundle icicle)* que com en el cas de *CadiMobile*, es crida al inici de l'activity, i genera la interfície a visualitzar i el listener pel botó que et permet retornar al activity principal.

RequestForm

Aquesta activity genera un formulari per fer la petició, com la classe anterior, només conté el mètode *onCreate(Bundle icicle)* que genera la interfície gràfica i, el listener pel botó que et permet retornar al activity principal per enviar la petició al servidor.

Capítol 5

Resultats

Aquest capítol està dividit en tres parts ben diferenciades. En la primera part mostrarem les proves realitzades en funció del temps computacional, en la segona la càrrega de memòria i en el tercer apartat farem un anàlisi dels resultats exposats i propostes de millora.

Totes les proves presentades en aquest capítol s'han realitzat amb l'emulador de la plataforma Android en un ordinador amb un processador Intel Core 2 Duo a 1,66 Ghz, 2GBytes de memòria i un sistema operatiu GNU/Linux. Les imatges utilitzades en aquestes proves són les següents :

- **512lena-lt0-ln50-wl5.jpc** (figura 5.1,a) Imatge de 512 x 512 pícsel escala de grisos, codificada amb cinc nivells de resolució i 50 capes de qualitat.
- **n1-lt0-ln50-wl5.jpc** (figura 5.1,b) Imatge de 2048 x 2560 pícsel escala de grisos, codificada amb cinc nivells de resolució i 50 capes de qualitat.

Per als dos tipus de proves els resultats es presenten en taules i gràfiques. A les files de les taules, troben les peticions realitzades representades en tres valors. Aquests valors identifiquen la component sol·licitada, nivell de resolució i capes de qualitat, i a les columnes d'aquestes tenim les fases de la descodificació. Aquestes fases son les següents:

- BD : Descodificació dels blocs

a) b)

Figura 5.1: Imatges utilitzades a les proves realitzades

- IB : Construcció de la imatge
- Deq : Dequantització
- IWT: Transformada Wavelet Inversa
- CD : Transformada Inversa de Color
- RR: Recuperació del rang
- LU: Level unshift
- RC : Chequeig del rang

5.1 Temps Computacional

Aquestes proves es basen en fer peticions de les dos imatges esmentades anteriorment amb diferents valors en els paràmetres de la petició. Ens serveixen per poder analitzar el cost temporal de la descodificació de les imatges.

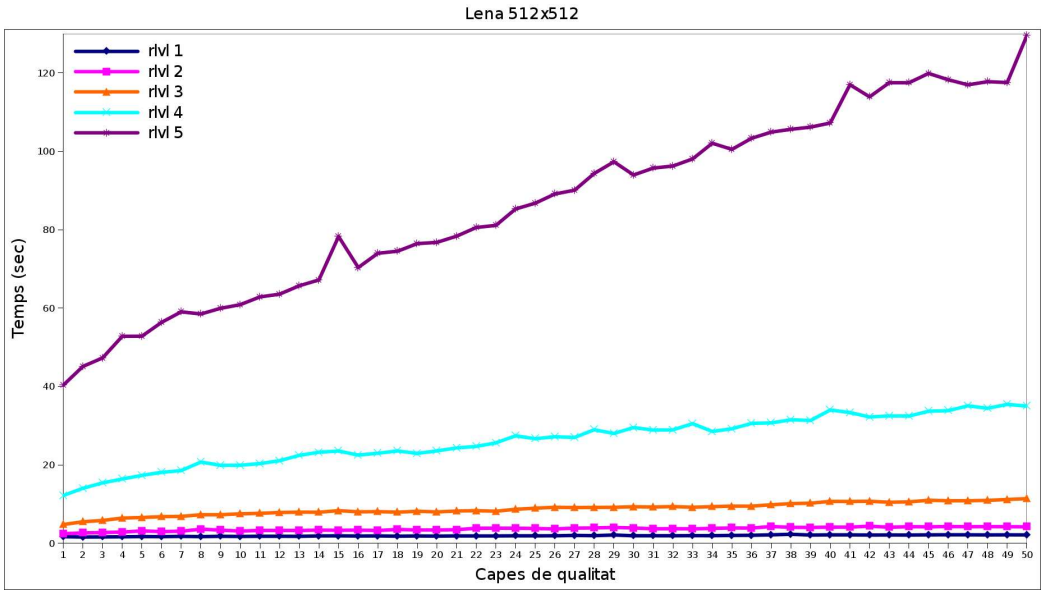


Figura 5.2: Temps (segons) per descodificar l'imatge 512lena-lt0-ln50-wl5.jpc a diferents nivells de resolució.

Els resultats es presenten de dues maneres diferents. A les taules 5.1 i 5.2 mostrem les peticions i els temps de descodificació, en segons, associats a cada fase i el temps total de la descodificació, i després a les gràfiques 5.2 i 5.3 l'evolució dels temps de comput en base al les capes de qualitat i el numero de nivells de resolució.

Com hem observat aquests resultats no són gaire satisfactoris. En l'aspecte de temps de comput podem observar a la taula 5.1 o a la taula 5.2 que a mesura que augmentem els nivells de resolució augmenten el temps de comput. Aquests temps augmenten de manera considerable ja que ha de descodificar molta més informació. A mesura que augmentem les capes de qualitat dins d'un mateix nivell de resolució, aquest temps augmenta notòriament. També hem de comentar que la fase on el descodificador requereix de més temps, gairebé un 70 % del procés, és en la fase de descodificació dels codeblocks. La resta de fases dins d'un mateix nivell de resolució tenen un temps de comput semblant ja que l'estructures que han de processar són semblants.

Petició	BD	IB	Deq	DWT	CD	RR	LU	RC	Total
1 1 1	0,2880	0,0250	0,0250	0,0410	0,0020	0,0070	0,0270	0,0220	1,0100
1 1 10	0,3940	0,0230	0,0210	0,0410	0,0000	0,0000	0,0140	0,0090	1,2050
1 1 20	0,4350	0,0220	0,0210	0,0410	0,0000	0,0000	0,0040	0,0060	1,1880
1 1 30	0,4640	0,0240	0,0210	0,0410	0,0010	0,0000	0,0030	0,0060	1,2760
1 1 40	0,5050	0,0230	0,0210	0,0400	0,0000	0,0000	0,0030	0,0060	1,4370
1 1 50	0,5090	0,0230	0,0210	0,0420	0,0000	0,0010	0,0040	0,0060	1,5030
1 2 1	0,7050	0,0980	0,0830	0,1990	0,0000	0,0000	0,0160	0,0240	1,7650
1 2 10	1,2970	0,0910	0,0830	0,2000	0,0000	0,0010	0,0150	0,0620	2,5410
1 2 20	1,3630	0,0920	0,1180	0,1990	0,0000	0,0000	0,0150	0,0240	2,6730
1 2 30	1,5100	0,0910	0,0830	0,2010	0,0000	0,0000	0,0160	0,0240	2,9010
1 2 40	1,8940	0,0950	0,0830	0,2070	0,0010	0,0000	0,0160	0,0250	3,4120
1 2 50	1,8970	0,0920	0,0830	0,2150	0,0000	0,0010	0,0150	0,0250	3,6050
1 3 1	1,3950	0,3600	0,3380	0,8400	0,0000	0,0000	0,0550	0,0880	4,3960
1 3 10	3,6120	0,3640	0,3370	0,8410	0,0000	0,0010	0,0560	0,0870	6,6350
1 3 20	4,1890	0,3570	0,3310	0,8460	0,0000	0,0000	0,0550	0,0880	7,3120
1 3 30	5,0810	0,3640	0,3330	0,8470	0,0000	0,0000	0,0940	0,0870	8,4820
1 3 40	5,7440	0,3750	0,3300	0,8550	0,0000	0,0000	0,0550	0,0910	9,2170
1 3 50	6,3110	0,3590	0,3300	0,8100	0,0000	0,0000	0,0600	0,0870	9,9400
1 4 1	2,0870	1,4740	1,2910	3,5580	0,0000	0,0000	0,2190	0,3850	12,3820
1 4 10	8,9520	1,5130	1,3480	3,5890	0,0010	0,0010	0,2180	0,3480	19,4300
1 4 20	11,8510	1,4880	1,3110	3,5700	0,0000	0,0000	0,2190	0,3480	22,3550
1 4 30	15,8040	1,4870	1,3060	3,6140	0,0000	0,0000	0,2230	0,3490	26,5800
1 4 40	19,1440	1,5510	1,3440	3,7560	0,0000	0,0010	0,2300	0,3680	30,4840
1 4 50	21,7380	1,5480	1,3110	3,8920	0,0010	0,0000	0,2360	0,3660	33,3390
1 5 1	2,1080	6,2040	5,3780	14,8240	0,0000	0,0000	0,9270	1,4930	42,6580
1 5 10	19,3250	6,0730	5,5510	14,6600	0,0000	0,0010	0,9630	1,4920	59,5130
1 5 20	35,3030	6,1900	5,2410	15,4130	0,0000	0,0000	0,9840	1,5270	76,5230
1 5 30	50,1940	6,3160	5,2640	15,9420	0,0000	0,0000	0,9840	1,5230	93,4520
1 5 40	64,5940	6,2960	5,5670	16,4830	0,0000	0,0000	0,9900	1,5500	110,2360
1 5 50	75,7970	6,2930	5,2020	17,7020	0,0000	0,0000	0,9350	1,5080	122,3780

Taula 5.1: Temps (segons) per descodificar l'imatge 512lena-lt0-ln50-wl5.jpc a diferents nivells de resolució

Petició	BD	IB	Deq	DWT	CD	RR	LU	RC	Total
1 1 1	4,111	0,176	0,486	0,778	0,002	0,002	0,084	0,134	7,74
1 1 10	6,405	0,176	0,492	1,085	0	0	0,083	0,133	10,132
1 1 20	8,224	0,494	0,477	0,774	0	0,001	0,083	0,134	12,093
1 1 30	8,98	0,176	0,507	0,776	0	0	0,082	0,133	12,559
1 1 40	10,945	0,17	0,462	0,77	0	0	0,082	0,133	14,79
1 1 50	10,309	0,206	0,544	0,867	0,001	0	0,082	0,133	14,548
1 2 1	8,307	0,682	2,07	4,298	0	0,001	0,358	0,533	20,386
1 2 10	18,901	0,683	1,93	4,584	0	0	0,333	0,573	30,976
1 2 20	26,833	0,706	1,938	4,682	0	0	0,334	0,544	39,805
1 2 30	28,265	0,739	1,984	4,509	0	0	0,353	0,534	41,25
1 2 40	34,931	0,744	1,924	4,481	0	0,001	0,328	0,576	47,926
1 2 50	36,197	0,688	1,927	4,567	0	0	0,328	0,572	50,122
1 3 1	20,851	3,208	8,413	20,288	0	0	1,327	2,205	71,839
1 3 10	59,758	4,114	8,332	24,446	0	0	1,33	2,165	115,15
1 3 20	89,82	3,39	7,972	22,479	0	0	1,33	2,162	144,61
1 3 30	101,773	3,23	7,677	22,826	0	0	1,368	2,167	157,405
1 3 40	129,151	3,728	7,957	22,643	0	0	1,319	2,134	188,645
1 3 50	137,815	3,163	8,505	28,282	0	0	1,481	2,338	209,353

Taula 5.2: Temps (segons) per descodificar l'imatge n1-lt0-ln50-wl5.jpc a diferents nivells de resolució

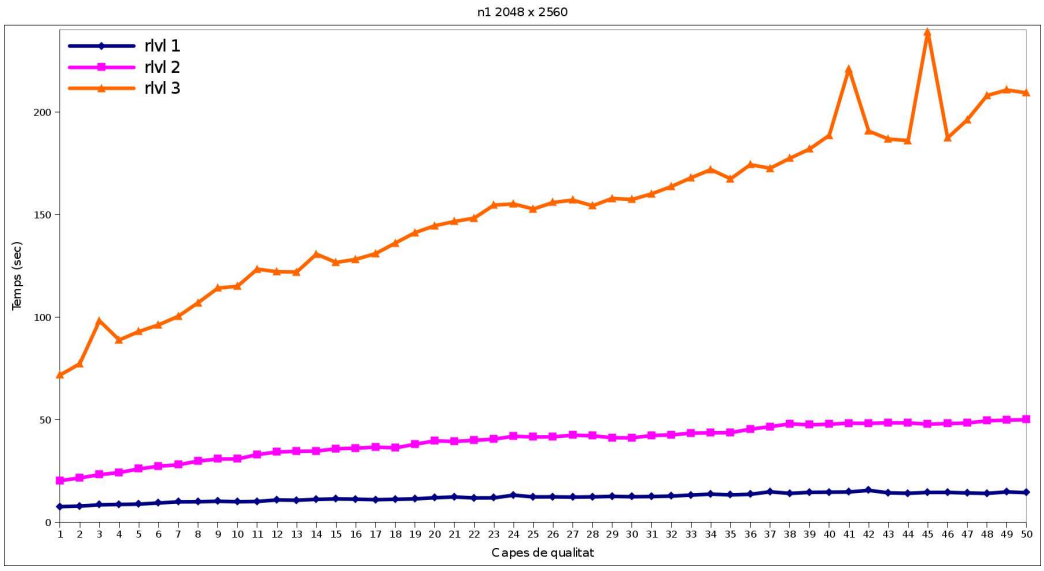


Figura 5.3: Temps (segons) per descodificar l'imatge n1-lt0-ln50-wl5.jpc a diferents nivells de resolució.

5.2 Carrega de Memòria

Com a l'apartat anterior aquestes proves es basaven en fer peticions de les dos imatges esmentades amb diferents valors en els paràmetres de la petició, i ens serveixen per poder analitzar la quantitat de memòria necessària a les diferents fases de la descodificació de les imatges.

Els resultats es presenten de dues formes diferents, primer a les taules 5.3 i 5.4 es presenten les dades on apareixen les fases de la descodificació i els consums de memòria associats, i després a les gràfiques 5.4 i 5.5 es presenten unes gràfiques de barres amb les dades resumides per poder visualitzar-les millor.

Si analitzem la figura 5.4 podem observar que a cinc nivells de resolució l'aplicació ens consumeix uns 8MB a cada fase, que són de la mida que ocupen en memòria les estructures amb les que treballem. Però a la 5.5 ens trobem que només tenim dades dels tres primers nivells de resolució. Això està ocasionat per què als nivells de resolució quatre i cinc, la memòria que necessita el descodificador a la fase del BD és major que amb la que la maquina virtual pot treballar i no permet descodificar l'imatge amb aquest paràmetres.

5.3 Propostes de millora

Una vegada plantejats i analitzats els resultats, es presenten dos tipus de problemes. El primer problema detectat és l'elevat cost computacional del procés de descodificació d'una imatge. Aquest problema té una solució directe, necessitem que el dispositiu tingui un processador més potent, ja que l'emulador que ens ofereix l'entorn de desenvolupament d'aquesta plataforma emula un processador ARM9[1] i possiblement no sigui el que definitivament s'utilitzarà en el dispositiu físic. Una altre possible solució, es basa en utilitzar prefetching strategies [DDVI+07], que es basen en que l'usuari quan demana una imatge, automàticament es descarregui el següent nivell de resolució i el descodifiqui en mode backgroud mitjançant un

[1] Processadors RISC de baix consum utilitzats en dispositius mòbils

Petició	**BD**	**IB**	**Deq**	**DWT**	**CD**	**RR**	**LU**	**RC**
1 1 1	1795	1795	1795	1795	1859	1859	1859	1859
1 1 10	1628	1637	1639	1661	1663	1664	1665	1666
1 1 20	1663	1672	1674	850	851	852	854	855
1 1 30	863	872	873	896	897	898	900	901
1 1 40	879	888	890	912	914	915	916	918
1 1 50	926	935	937	959	961	962	963	965
1 2 1	1639	1661	1682	958	959	961	962	963
1 2 10	1771	1793	1795	960	961	963	964	965
1 2 20	1833	952	954	1050	1051	1052	1054	1055
1 2 30	963	986	988	1083	1084	1086	1087	1089
1 2 40	1055	1077	1079	1174	1176	1177	1178	1180
1 2 50	1131	1154	1156	1251	1252	1254	1255	1256
1 3 1	1803	1056	1127	1497	1498	1499	1501	1502
1 3 10	2172	2245	2247	1379	1381	1382	1383	1384
1 3 20	2302	1273	1275	1644	1646	1647	1648	1650
1 3 30	1251	1324	1326	1696	1697	1699	1700	1701
1 3 40	1389	1462	1465	1834	1835	1836	1838	1839
1 3 50	1575	1648	1650	2019	2021	2022	2024	2025
1 4 1	2371	1681	1950	2088	2089	2091	2092	2094
1 4 10	3603	3873	3875	2827	2828	2830	2831	2832
1 4 20	3901	4171	4174	3306	3307	3308	3310	3311
1 4 30	2291	2561	2563	3992	3994	3995	3997	3998
1 4 40	2438	2709	2711	2177	2178	2179	2180	2182
1 4 50	3020	3290	3293	2659	2661	2662	2663	2665
1 5 1	3109	4158	3717	3751	3753	3754	3755	3757
1 5 10	6508	7557	7559	6352	6354	6355	6357	6358
1 5 20	6767	7393	7396	7094	7095	7097	7098	7099
1 5 30	6538	7586	7589	6088	6090	6091	6092	6094
1 5 40	6965	7539	7542	5376	5377	5379	5380	5381
1 5 50	6900	7607	7610	7015	7016	7018	7019	7020

Taula 5.3: Memòria (KB) consumida en les fases per la imatge 512lena-lt0-ln50-wl5.jpc

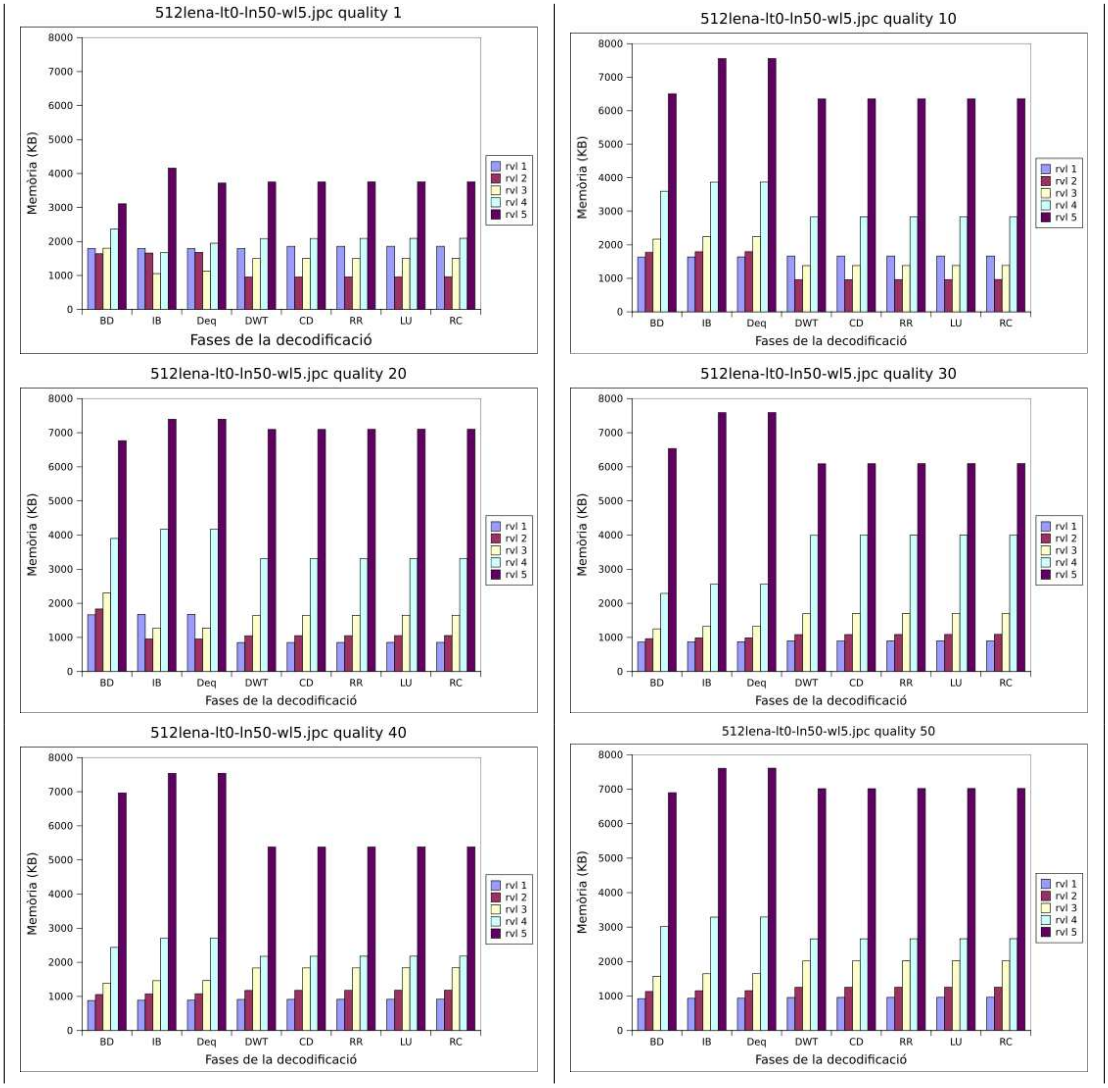


Figura 5.4: Memòria (KB) consumida per fases de la imatge 512lena-lt0-ln50-wl5.jpc

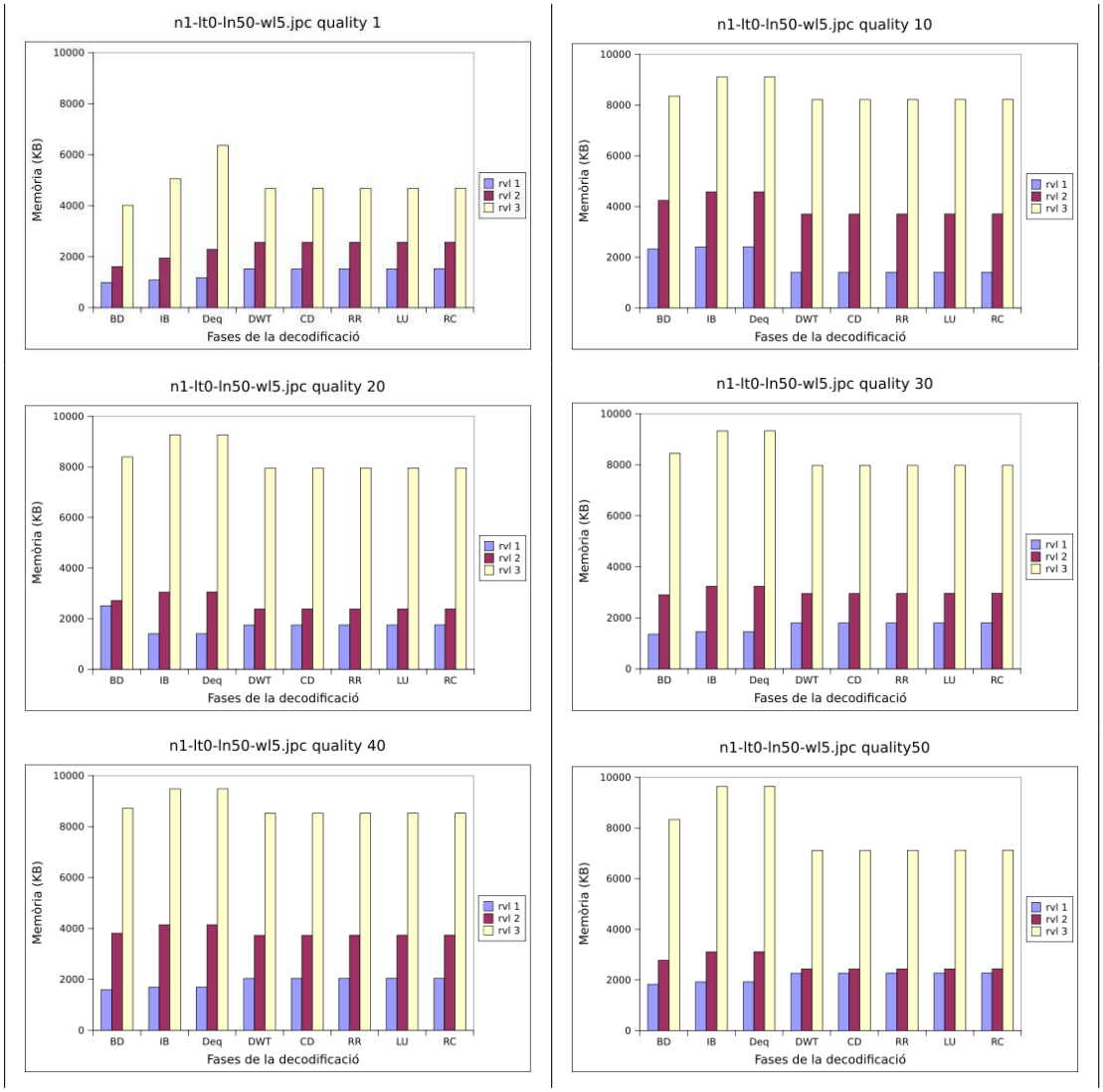


Figura 5.5: Memòria (KB) consumida per fases de la imatge n1-lt0-ln50-wl5.jpc

Petició	BD	IB	Deq	DWT	CD	RR	LU	RC
1 1 1	990	1083	1173	1517	1520	1523	1526	1529
1 1 10	2322	2412	2414	1398	1400	1401	1402	1404
1 1 20	2510	1401	1403	1744	1746	1747	1749	1750
1 1 30	1359	1449	1451	1793	1794	1795	1797	1798
1 1 40	1599	1689	1691	2033	2034	2036	2037	2038
1 1 50	1829	1919	1921	2263	2264	2266	2267	2269
1 2 1	1609	1945	2280	2563	2564	2566	2567	2569
1 2 10	4241	4577	4579	3698	3699	3701	3702	3703
1 2 20	2715	3052	3054	2380	2381	2383	2384	2385
1 2 30	2902	3238	3240	2949	2951	2952	2953	2955
1 2 40	3809	4146	4148	3727	3729	3730	3732	3733
1 2 50	2772	3108	3110	2436	2437	2439	2440	2441
1 3 1	4008	5051	6358	4670	4672	4673	4674	4676
1 3 10	8353	9104	9106	8210	8212	8213	8214	8216
1 3 20	8394	9266	9268	7951	7953	7954	7955	7957
1 3 30	8442	9324	9327	7970	7971	7972	7974	7975
1 3 40	8714	9478	9480	8519	8521	8522	8523	8524
1 3 50	8328	9637	9639	7114	7115	7116	7118	7119

Taula 5.4: Memòria (KB) consumida en les fases per la imatge n1-lt0-ln50-wl5.jpc

service[2], mentre l'usuari treballa amb l'imatge que ell mateix ha demanat. De la mateixa manera que es descarrega el següent nivell de resolució, podem descodificar l'imatge amb un nivell més de qualitat. Totes aquestes opcions les podria configurar l'usuari de l'aplicació en funció de les seves necessitats.

El segon problema detectat, l'alt consum de memòria, no té una solució tant directa com la primera, ja que la màquina virtual té un límit de memòria amb la que pot treballar, per aquest motiu s'ha de plantejar un canvi en el descodificador JPEG2000 que utilitza CADI. Actualment quan l'usuari demana una regió de l'imatge, aquest construeix l'estructura com si l'usuari hagués demanat tot el nivell de resolució. El que es planteja per solucionar aquest problema és que el descodificador només utilitzi l'espai de memòria que necessita la regió que l'usuari ha demanat. Es a dir, si l'usuari té una pantalla de 480x320 pícsels i demana el tercer nivell de resolució d'una imatge de 2048 x 2560, l'estructura amb la que treballa el descodificador no sigui de 512x640 sinó de 480x320 (veure figura 5.6), on

[2]tipus de bloc utilitzat en el desenvolupament d'aplicacions en Android, explicat en el capítol 4 d'aquesta memòria

Esquerra **Dreta**

Figura 5.6: Carrega en memòria de l'imatge a descodificar. **Esquerra**: imatge a carregar amb tot el tercer nivell de resolució, **Dreta**: imatge a carregar en funció de la mida de la pantalla.

aquest valor serà lo màxim que es podrà visualitzar en la seva finestra. El problema d'aquest mètode es que aquesta regió que demanaríem s'hauria de tractar com si fos una imatge nova i no com a part d'una imatge. Per resoldre aquest problema fem un pas previ a la fase de la descodificació on generem la nova estructura i anem introduint les dades que rebem.

Capítol 6

Planificació i Costos

En aquesta capítol s'analitzarà la planificació realitzada inicialment i la planificació resultant i els costos que ofereix aquest projecte en cas d'una implementació real en una empresa.

6.1 Planificació del projecte

Aquest projecte es va dividir set fases :

- Planificació del projecte.
- Estudi del estàndard JPEG200 (ISO 15444-1) i del seu protocol de transferència d'imatges JPIP (ISO 15444-9).
- Estudi de les diferents plataformes mòbils i possibilitats de desenvolupament.
- Redacció de l'informe previ.
- Adaptació del Client JPIP facilitat per el grup GICI a una plataforma mòbil.
- Implementació i test del visor.
- Creació de la documentació.

Tasca	Data inici prevista	Data fi prevista	Data inici real	Data fi real
Planificació	01/11/07	02/11/07	01/11/07	02/11/07
Documentació				
Estudi de les plataformes mòbils	05/11/07	30/11/07	05/11/07	30/11/07
Estudi JPEG2000	03/12/07	21/12/07	03/12/07	21/12/07
Estudi JPIP	24/12/07	11/01/07	24/12/07	11/01/07
Redacció informe previ	09/12/07	11/01/07	09/01/08	11/01/07
Implementació				
Adaptació Client CADI	02/01/08	31/01/08	02/02/08	31/03/08
Implementació Visor	01/02/08	25/02/08	02/04/08	13/04/08
Test	26/02/08	26/03/08	16/04/08	30/04/08
Redacció Documentació				
Memòria	01/04/08	11/06/08	01/05/08	09/06/08
Manual d'usuari	08/05/08	12/05/08	09/06/08	11/06/08

Taula 6.1: Temporització del projecte

Recurs	Cost del recurs (€/ hora)
Director	21
Tècnic	13

Taula 6.2: Recursos del projecte

En la taula 6.1 es presenta la temporització planificada inicialment per aquest projecte i la temporització final. Com podem observar en la taula no s'ha complert la planificació inicial per diversos motius.

- Manca de documentació i suport teòric de la plataforma Android.
- Problemes amb la fiabilitat del emulador, ja que a vegades l'aplicació no s'iniciava o es penjava durant de l'execució.
- Es comença un mes tard l'adaptació de CADI a la plataforma Android, ocasionat per els exàmens del primer semestre.
- Es va trigar dos mesos enlloc d'un en adaptar CADI, ocasionat per els problemes trobats (temps computacional i carrega de memòria) , explicats al capítol 5.

Activitat	**Encarregat**	**Temps (dies)**	**Temps (hores)**	**Preu** (€)
Cadi Mobile		170	785	11.522,00
Planificació	Director [50%]	2	8	168,00
Documentació		50	200	3.240,00
Estudi JPEG2000	Tècnic [50%]	15	60	780,00
Estudi JPIP	Tècnic [50%]	15	60	780,00
Estudi plataformes mòbils	Director [50%]	20	80	1.680,00
Implementació		39	184	2.392,00
Adaptació Client CADI	Tècnic	22	176	2.288,00
Implementació Visor	Tècnic	11	88	1.848,00
Test	Tècnic	22	176	2.288,00
Redacció Documentació		30	130	1.690,00
Memòria	Tècnic [50%]	30	120	1.560,00
Manual d'usuari	Tècnic [50%]	2,5	10	130,00

Taula 6.3: Costos del projecte

6.2 Costos del projecte

Per aquest projecte s'han utilitzat dos tipus de recursos, un director de projecte, que s'encarrega de verificar la implementació, documentació i planificar aquest projecte i un tècnic que s'encarrega de l'estudi del JPEG2000, del JPIP, i de la implementació del projecte. Els costos, presentats a la taula 6.3, s'han calculat en base a les dades presentades a la taula 6.2 i la temporització presentada a la taula 6.1. Les dades que presentem a la taula 6.2 corresponen a les dades dels sous bruts d'un director de projectes i d'un enginyer tècnic en informàtica. A la taula 6.3 podem trobar la feina realitzada per cada recurs, amb el seu percentatge de dedicació, els dies i les hores corresponent empleades i el cost de cada fase de l'implementació.

Capítol 7

Conclusions i línies futures

En aquest últim capítol presentarem les conclusions d'aquest projecte, i les línies de continuïtat.

7.1 Conclusions

Com hem explicat a la introducció, aquest projecte tenia tres objectius ben diferenciats:

- Estudi del estàndard JPEG2000 i del protocol JPIP.
- Estudi de les diferents plataformes mòbils i les possibilitats de desenvolupament.
- Adaptació del client JPIP i implementació d'un visor.

Clarament aquests objectius han sigut assolits. Donat que el client CADI no optimitza els recursos, els resultats de les proves tant de temps com de consum de memòria són elevats. Tot això es degut a que aquesta implementació no està orientada a oferir un alt rendiment, sinó una alta flexibilitat alhora d'afegir noves prestacions. Si volem que aquesta aplicació sigui més eficient s'haurà de fer un disseny orientat a rendiment i no tan modular com l'actual.

També s'ha de dir que el realitzar aquest projecte amb la plataforma Android a sigut força interessant ja que ofereix moltes possibilitats de desenvolupament.

7.2 Línies futures

El treball realitzat en aquest projecte ens ofereix diverses línies de treball futur. En primer lloc podríem millorar la interfície gràfica incloent noves funcionalitats, com fer edicions o anotacions en la imatge aprofitant les meta-dades que s'inclouen en les capçaleres dels fitxers jp2, i poder-la guardar en memòria o pujar-les al servidor. També com a millora de l'aplicació es vol poder seleccionar regions de la imatge amb el touchscreen que incorpora el dispositiu, per poder crear peticions que només incloguin la regió seleccionada i així explotar millor les possibilitats que ens ofereix JPIP.

Finalment, l'ultima millora proposada és modificar l'esquema de descodificació de CADI, com hem explicat al capítol de resultats, per així millorar els resultats obtinguts de recursos i poder realitzar peticions d'imatges d'una mida més gran.

Apèndix A

Manual d'usuari

A.1 Instal·lació de CadiMobile

Per instal·lar la nostra aplicació, primer hem de compilar el projecte amb la comanda ant [Apa].

Una vegada compilat hem d'anar al directori on tenim instal·lat el simulador d'Android i iniciar-lo amb la següent instrucció,

```
./emulator -skin HVGA-L
```

o en el cas de que vulguem visualitzar els logs que ens mostra CadiMobile

```
./emulator -skin HVGA-L -logcat CADI | grep CADI
```

Una vegada executat l'emulador ara podem instal·lar la nostra aplicació dins l'emulador amb la comanda

```
./adb install ../../../src/CadiMobile/bin/CadiMobile.apk
```

A.2 Funcionament de CadiMobile

El funcionament de CadiMobile es molt senzill, s'utilitza el cursor del dispositiu per realitzar totes les funcionalitats i botó de menú. A la figura A.1 observem les parts de la interfície gràfica.

Figura A.1: Pantalla inicial de CadiMobile

Per fer una petició al servidor hem de fer clic a l'opció `request` del menú o en el cas de que volguéssim fer una petició des de la url, primer introduïm la petició amb el teclat i després fem clic a l'opció `send URL`.

En el formulari de petició que mostrem a la figura A.2 hem d'omplir el camp `Server Name` amb la ip o el nom de host de la maquina servidor, al camp `port` el numero de port per on escolta el servidor i la imatge que volem que sigui transmesa.

Una vegada tenim la imatge descodificada, podem augmentar els paràmetres de nivell de resolució, numero de capes de qualitat i zoom-in o zoom-out dins del submenú `Image` que hi ha al menú principal (veure figura A.3). També en el submenú `Image`, hi podem accedir a les propietats de la imatge on ens mostra els paràmetres de codificació (figura A.4).

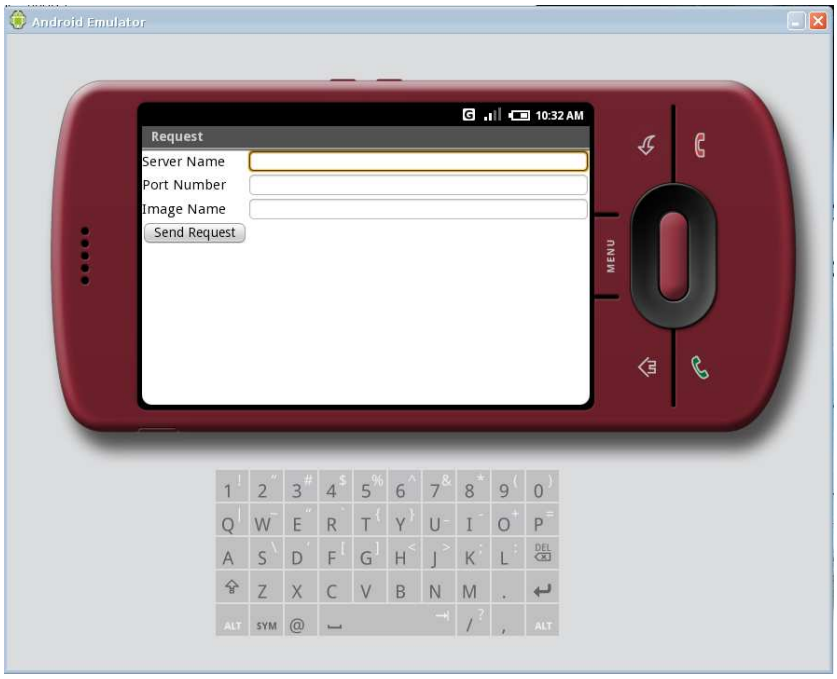


Figura A.2: Pantalla de request

Figura A.3: Submenú d'opcions Image

Figura A.4: Propietats de la Imatge

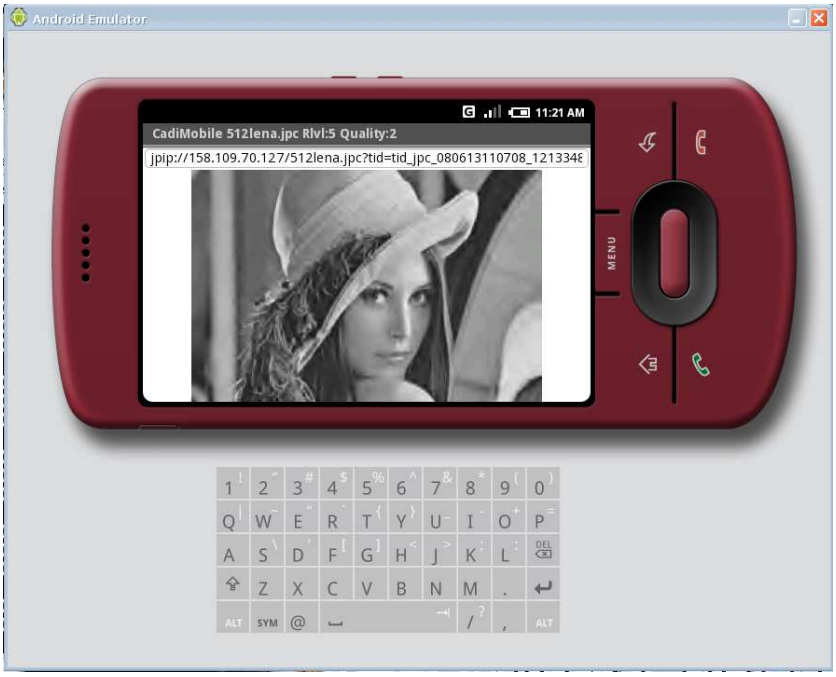


Figura A.5: Imatge 512lena-lt0-ln50-wl5.jpc amb cinc nivells de resolució i 2 capes de qualitat

Apèndix B

Informe previ

B.1 Objectius del projecte

Recentment la utilització de dispositius mòbils i la facilitat d'accés a la xarxa permeten als usuaris accedir a grans bases de dades d'imatges. Per reduir, el temps de transmissió, l'espai utilitzat, etc. s'utilitzen diferents sistemes de compressió d'imatges. L'objectiu del projecte és, fer un estudi de diferents plataformes mòbils, integració d'un descodificador de JPEG2000 en un dispositiu mòbil, i implementació d'un sistema de visualització d'imatges, amb la finalitat de permetre una transmissió interactiva d'imatges.

Per poder assolir aquest objectiu hem de dividir el projecte en tres fases:

- Estudi de les diferents plataformes mòbils i possibilitats de desenvolupament.

- Estudi del estàndard JPEG200 (ISO 15444-1) i del seu protocol de transmissió d'imatges JPIP (ISO 15444-9).

- Disseny, implementació i test del descodificador/visor per a la plataforma seleccionada.

B.2 Breu introducció a l'estat del art del tema proposat

Actualment pel sistema de compressió d'imatges JPEG2000 existeix un estàndard per a la seva transmissió interactiva, anomenat JPIP.

Existeixen diferents implementacions del protocol JPIP on cal destacar tres d'elles :

- Kakadu JPIK: Implementació realitzada per un dels autors del JPEG2000, David Taubman. Aquesta implementació està realitzada en C++ y té versions per Windows. Es pot descarregar una versió : http://www.kakadusofware.com
- AccuRad JPIPStream: Implementació realitzada per l'empresa AWARE i és una versió de pagament.
- CADI : Implementació desenvolupada íntegrament al Group on Interactive Coding of Images (GICI) de la UAB i és la que utilitzarem en el projecte. Està desenvolupada en Java i té llicència GPL. Es pot descarregar una versió a la web : http://cadi.sf.net

B.3 Estudi de viabilitat del projecte

Per poder realitzar aquest projecte es va fer un estudi de les diferents plataformes mòbils que hi ha actualment al mercat. Aquestes plataformes han de complir uns determinats requisits presentats a continuació:

- Java : La nostra plataforma ha d'oferir la possibilitat d'executar aplicacions fetes en Java.
- Accés a Internet via Wireless.
- Simulador software de la plataforma : Per reduir costos del projecte volem utilitzar un simulador enlloc de la plataforma física.
- L'entorn de desenvolupament i la plataforma han de ser Open Source.

Després d'una primera selecció es va decidir fer un estudi més intensiu de tres plataformes concretes.

- J9 WebSphere IBM : Ofereix un entorn de desenvolupament molt complert amb la possibilitat d'utilitzar les dues configuracions de Java possibles per a dispositius mòbils, CDC i, CLDC. També ofereix la possibilitat d'interactuar amb les llibreries d'interfície gràfica AWT. La plataforma J9 és compatible amb qualsevol dispositiu mòbil amb Windows Mobile o Windows CE.

- Nokia N95: Aquesta plataforma ofereix un entorn de desenvolupament amb la configuració CLDC 1.1 i el MIDP 2.0. Inclou extensions necessàries per poder comunicar-se via Wireless, BlueTooth i llibreries multimèdia per la càrrega d'imatges i gràfics 2D/3D. Aquesta plataforma funciona amb sistema operatiu Symbian

- Google Android : Nova plataforma per dispositius mòbils creada recentment per Google que inclou un sistema operatiu amb kernel Linux. Aquesta plataforma ofereix un entorn de desenvolupament que inclou una API de Java basada en la de Sun Microsystems amb noves llibreries per la carrega d'imatges, gràfics 2D/3D, i comunicació via Wireless.

Després d'haver estudiat les tres plataformes es va optar per utilitzar la plataforma Google Android ja que era l'única de les plataformes estudiades que assolia tots els requeriments no funcionals presentats anteriorment. J9 WebSphere IBM la plataforma i el entorn de desenvolupament eren de pagament, i la plataforma de Nokia no era una plataforma Open Source.

Bibliografia

[ABMD92] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image coding using wavelet transform. *IEEE Transactions on Image Processing*, 1(2):205–220, April 1992.

[Apa] Apache. Apache ant. http://ant.apache.org/.

[Awa] Aware Inc. http://www.aware.com/.

[DDVI+07] A. Descampe, C. De Vleeschouwer, M. Iregui, B. Macq, and F. Marques. Prefetching and caching strategies for remote and interactive browsing of jpeg2000 images. *Image Processing, IEEE Transactions on*, 16(5):1339–1354, May 2007.

[Foua] Eclipse Foundation. Eclipse. http://www.eclipse.org/.

[Foub] Eclipse Foundation. Eclipse me. http://www.eclipseme.org.

[GIC] GICI. Group on Interactive Coding of Images (GICI). http://gici.uab.es. Department of Information and Communications Engineering. Universitat Autònoma de Barcelona.

[IBM] IBM. Websphere everyplace micro environment. http://www-306.ibm.com/software/wireless/weme/.

[Inc07] Google Inc. Google Android. http://code.google.com/android, 2007.

[ISO00] JPEG 2000 image coding system - Part 1: Core coding system, Information technology 15444-1. International Organization for Sandardization / International Electrotechnical Commission (ISO/IEC);

International Telecommunication Union-Telecom Standardization (ITU-T), December 2000.

[ISO05] JPEG 2000 image coding system - Part 9: Interactivity tools, APIs, and protocols. Information technology 15444-9. International Organization for Sandardization / International Electrotechnical Commission (ISO/IEC); International Telecommunication Union-Telecom Standardization (ITU-T), December 2005.

[JPEG] Joint Photographic Experts Group. Jpeg2000. http://www.jpeg.org/jpeg2000/.

[Mica] Sun Microsystems. Java 2 Micro Edition. http://java.sun.com/. Java 2 Micro Edition.

[Micb] Sun Microsystems. Netbeans. http://www.netbeans.org.

[Nok] Nokia. Forum nokia. http://forum.nokia.org.

[Tau00] David S. Taubman. High performance scalable image compression with EBCOT. *IEEE Transactions on Image Processing*, 9(7):1158–1170, July 2000.

[Tau01] R. Taubman, D.; Rosenbaum. Rate-distortion optimized interactive browsing of jpeg2000 images. In *Proceedings of IEEE International Conference on Image Processing 2003*, volume 3, pages 765–768 vol.2, Barcelona, September 2001. International Multimedia Conference, IEEE.

[TM02] D.S. Taubman and M.W. Marcellin. *JPEG2000: Image Compression Fundamentals, Standards, and Practice*, volume 642. Kluwer International Series in Engineering and Computer Science, 2002.

Firmat: Juan Muñoz Gómez

Bellaterra, 16 de Juny de 2008

Resum

JPEG2000, és el nou estàndard de compressió d'imatges impulsat pel Joint Photographics Experts Group, el qual defineix un protocol eficient per la transmissió interactiva d'imatges, anomenat JPIP. El Group on Interactive Coding of Images (GICI) té una implementació d'aquest protocol, CADI. En aquest projecte es realitza l'implementació del client d'aquest protocol en un dispositiu mòbil. Per això s'ha realitzat un estudi de les plataformes mòbils que hi ha actualment en el mercat. Finalment s'han proposat millores, en el descodificador, per reduir el temps de computació i la carrega de memòria.

Resumen

JPEG2000, el nuevo estándar de compresión de imágenes impulsado por el Joint Photographics Experts Group, el cual define un protocolo eficiente para la transmisión interactiva de imágenes, llamado JPIP. El Group on Interactive Coding of Images (GICI) tiene una implementació de este protocolo, CADI. En este proyecto se realiza la implementación del cliente de este protocolo en un dispositivo móvil. Para ello se ha realizado un estudio previo de todas las plataformas móviles que hay actualmente en el mercado. Finalmente se han propuesto mejoras, en el proceso de descodificación, para reducir el tiempo de computación y el uso de memoria.

Abstract

JPEG2000 is the new compresion standard defined by the Joint Photographic Experts Group committee, which defines an efficient protocol, called JPIP, devised for the interactive transmission of images. The Group on Interactive Coding of Images (GICI) has developed CADI, a novel implentation of the JPIP protocol. In this project, it has been implemented a client for this protocol in a mobile device. Previously to the implementation, it has been carried out a market research on mobile devices available currently. Finally, it has been proposed a few improvements, in the decoding process, aimed to reduce timing performance and memory usage.