Títol: Sistema multi-projector per a realitat virtual

Volum: 1/1
Alumne: David Aguilera Moncusí

Director/Ponent: Pere Brunet Crosa
Departament: Llenguatges i Sistemes Informàtics
Data: Gener 2009

DADES DEL PROJECTE

Títol del projecte:	Sistema multi-projector per a realitat virtual
Nom de l'estudiant:	David Aguilera Moncusí
Titulació:	Enginyeria Superior en Informàtica
Crèdits:	37,5
Director/Ponent:	Pere Brunet Crosa
Departament:	Llenguatges i Sistemes Informàtics

MEMBRES DEL TRIBUNAL *(nom i signatura)*

President:	Isabel Navazo Alvaro
Vocal:	Germán Santos Boada
Secretari:	Pere Brunet Crosa

QUALIFICACIÓ

Qualificació numèrica:

Qualificació descriptiva:

Data:

Sistema multi-projector per a realitat virtual

David Aguilera Moncusí

Gener 2009

Agraïments

En primer lloc vull agrair a Rodrigo Pizarro tota la feina que ha fet i les ganes que ha posat en aquest projecte; no només va ser ell qui me'n va parlar i em va motivar per començar-lo, sinó que també ha sigut un gran company i amic durant tot el seu desenvolupament.

També vull agrair els consells i correccions que hem tingut per part del nostre tutor, Pere Brunet, i l'ajuda que ens ha donat Jordi Moyes en els temes constructius. Menció especial mereixen l'Antonio Villegas, el Marc Loan i el David Sancho, els quals sempre han tingut un moment per a donar-me un cop de mà.

Finalment, m'agradaria mencionar a la meva família, ja que són ells els qui sempre han estat al meu costat recolzant-me.

A tots vosaltres, i a tots aquells que segur que oblido, moltes gràcies.

Índex

Índex de figures

CAPÍTOL 1

Introducció

En aquest primer capítol es pretén explicar *grosso modo* tant el tema que es tracta en el present projecte de final de carrera com l'organització en què es presenta la memòria.

Així doncs, per una banda s'exposa quina és la meta que s'ha intentat assolir, es descriuen breument quins objectius se'n deriven i es comenta el perquè d'un projecte com aquest.

Per altra banda, també es detalla com s'ha organitzat la memòria, comentant les raons que han determinat aquesta distribució de la informació i no una altra, tenint en compte que la idea és fer que la seva lectura resulti el més amena possible.

1.1 Descripció del projecte

El projecte consisteix en dissenyar un sistema *immersiu* de realitat virtual totalment *portable*, construir-ne un prototipus i comprovar i validar el seu funcionament.

El prototipus constarà de dues *unitats independents* amb un projector i una càmera digital cadascuna, necessaris el primer per a la projecció d'imatges i la segona per a realitzar el calibratge del sistema.

A més a més, per tal d'augmentar el nivell d'immersió, també inclourà un mecanisme que permeti fer el seguiment actiu de l'usuari[1].

1.2 Motivació

Una de les motivacions d'aquest projecte és aconseguir tenir un dispositiu de tipus pantalla la mida del qual sigui totalment arbitrària. Aquest tipus de dispositiu pot resultar de molta utilitat en àmbits molt diferents i variats:

Videojocs Els usuaris cada cop volen pantalles més grans i amb més resolució. Aquest projecte en concret se centra precisament en l'àmbit de les aplicacions 3D, potenciant l'efecte immersiu que s'aconsegueix amb la utilització d'una pantalla de grans dimensions i tècniques com el *head tracking*.

[1] Aquest està basat en la videoconsola de Nintendo® *Wii*® (vegeu els capítols 8 i 9)

Medicina Les pantalles que s'utilitzen actualment en aplicacions mèdiques han de concentrar una quantitat d'informació molt gran en dispositius 2D força reduïts; afegint-hi una tercera dimensió i eliminant les restriccions de mida, la feina de molts especialistes pot veure's simplificada en gran mesura, atès que disposaran d'una eina molt més potent de visualització.

Publicitat, actes multitudinaris, etc. La publicitat o esdeveniments com ara concerts, esports o espectacles a l'aire lliure, els quals ja disposen de pantalles de gran mida, podrien disposar d'una solució molt més barata i amb una qualitat d'imatge molt superior.

En qualsevol cas, es presenta un sistema altament flexible i portable, amb un manteniment senzill gràcies a la utilització dels mòduls que composen la solució final, els quals simplifiquen el muntatge de tota la infraestructura.

1.3 Objectius

Tal i com ja s'ha introduït a l'apartat 1.1, el principal objectiu del projecte és *desenvolupar un sistema immersiu de realitat virtual*. Aquest té una sèrie de característiques que el fan singular:

Portabilitat El sistema ha de poder utilitzar-se en diferents ubicacions i no ha d'estar, per tant, lligat a un emplaçament fix.

Calibratge automàtic La portabilitat del sistema requereix que es pugui adaptar fàcilment a cadascun dels diferents entorns on s'utilitzi. Per tal de cobrir aquesta necessitat de la forma més còmoda possible, es vol que sigui capaç de configurar-se automàticament[2].

Head tracking Per tal d'assolir un nivell de realisme més elevat, el sistema implementa un mecanisme que fa un seguiment de la posició de l'usuari, de tal manera que el que se li mostra és coherent amb la seva ubicació real relativa a la pantalla.

Aplicació d'usuari Es necessita una aplicació que utilitzi tota la infraestructura desenvolupada i que demostri el correcte funcionament del projecte.

Construcció d'un prototipus Cal muntar el suport físic que implementarà tot el sistema (tàndem projector - càmera, gestionat amb un ordinador portàtil).

1.4 Organització de la memòria

El projecte de final de carrera consta de dues parts ben diferenciades la conjunció de les quals permet dur-lo a terme:

Calibratge del sistema La infraestructura a la que el projecte pretén arribar requereix d'un calibratge acurat dels diferents components.

Desenvolupament d'una aplicació 3D El sistema de realitat virtual requereix una aplicació que aprofiti la infraestructura sobre la qual està corrent i que aprofiti al màxim tots els recursos disponibles.

[2] Aquesta configuració automàtica inclou els processos de calibratge geomètric i cromàtic, els quals es duen a terme mitjançant processos de visió per computador.

Per aquest motiu, s'estima oportú que la memòria respecti aquesta diferenciació dels paquets de treball i que, per tant, s'organitzin en diferents capítols, on cadascun d'ells comprengui una de les fases. Així, els capítols 7 i 9 descriuen les diferents etapes d'especificació, disseny i implementació efectuades.

La memòria també recopila els capítols introductoris a cadascun dels conceptes aplicats al projecte.

Així, els capítols 5 i 6 introdueixen el concepte d'*homografia* i el *patró de disseny de xarxa* aplicats a la primera part del projecte.

El capítol 8 explica amb més detall què és el *seguiment de l'usuari* i com s'ha implementat.

Finalment, també hi ha capítols de caire administratiu, com els capítols 4 i 11, on es detallen la planificació inicial del projecte i es compara amb l'execució final i els costs que se'n deriven.

1.4.1 El calibratge i el desenvolupament d'una aplicació

El fet de partir de dues fases que estan tan diferenciades facilita pensar que qualsevol d'elles és prou important per si sola i té suficient entitat com per a considerar-la un projecte auto contingut, de tal manera que seria la unió d'ambdós projectes la que representaria la totalitat del PFC. En aquest cas s'ha cregut convenient interpretar-ho d'aquesta manera, i s'ha decidit, doncs, donar-los-hi la categoria de projecte.

Aquesta decisió permet treballar-hi de forma independent, documentant-los per separat i fent que tasques com l'anàlisi de requisits, el disseny del sistema o la fase de proves es vinculin a cadascun d'ells sense que s'interfereixin. Això facilita enormement el desenvolupament del projecte, ja que la feina queda dividida des de l'inici en ens més petits. A més a més, també en simplifica la seva comprensió, atès que s'estudia, en cada cas, una part més petita del total.

És important subratllar, però, que no estan completament desvinculats l'un de l'altre; de fet, i com ja s'ha insinuat anteriorment, el *desenvolupament de l'aplicació 3D* depèn del *calibratge del sistema*. Així, es pot intuir fàcilment que s'haurà de treballar pensant en què cap de les dues parts està aïllada totalment i que, conseqüentment, s'ha de definir molt bé el nexe d'unió entre elles.

1.4.2 Treball en equip

La feina que s'ha de fer per a construir amb èxit aquest projecte és molt gran. Aquesta complexitat fa que sigui impensable fer-ho una única persona i, per això, s'han distribuït les tasques entre dues: Rodrigo Pizarro i jo.

Ara bé, tot i que el propòsit final del projecte és únic i comú, els objectius que ha d'assolir cadascú de nosaltres són lleugerament diferents. Aquestes diferències es posaran de manifest a les respectives memòries, havent-hi força referències creuades.

CAPÍTOL 2

Sistema de realitat virtual

En aquest capítol es descriu què és un sistema de realitat virtual i quines tècniques hi ha actualment per tal d'augmentar el grau d'immersió que té l'usuari. També en detallarem alguns exemples concrets[1] com, per exemple, la *cave*.

2.1 Què és un sistema de realitat virtual?

La *realitat virtual* és una tecnologia que permet a un usuari interaccionar amb un entorn simulat per ordinador, essent aquest real o imaginari. La majoria d'aquests entorns se centren en l'aspecte visual d'aquesta realitat, ja sigui a través de la pantalla d'un ordinador o mitjançant pantalles estereoscòpiques.

La definició anterior deixa molt clar quines són les dues claus que caracteritzen aquests sistemes i que els fan singulars: la *recreació* d'un món simulat i la *interacció* amb ell. D'aquesta manera, hom pot intuir que com més *realista* sigui la simulació que es presenta davant de l'usuari, i més *natural* la manera d'interaccionar-hi, millor serà el sistema dissenyat.

2.2 Què és un sistema immersiu?

Un sistema de realitat virtual consisteix en un *entorn simulat* amb el que *l'usuari interacciona*. L'objectiu final d'aquests sistemes és aconseguir que l'usuari tingui la sensació d'*estar dins d'aquest món virtual* i, com es pot deduir fàcilment, la manera d'aconseguir-ho és estimulant-li els seus sentits, atès que, de fet, és com els humans interaccionen amb el món real.

Sistema immersiu és un tipus de sistema que potencia l'estimulació d'un o més sentits de l'usuari per tal d'augmentar la sensació de pertànyer al món virtual amb què interacciona.

A dia d'avui, el sentit que més s'està explotant en entorns virtuals és el de la vista, però hi ha investigacions que estudien com excitar la resta.

[1] Cal tenir present que, avui en dia, la majoria de sistemes de realitat virtual exploten la immersió visual i, per tant, ens centrarem en aquest tipus.

2.2.1 Tècniques per aconseguir la immersió

En aquesta secció es mostren algunes de les tècniques emprades per tal d'aconseguir o augmentar el grau d'immersió, classificant-les segons el sentit que estimulen.

Visió

Quan hom pensa en la realitat virtual s'imagina, molt possiblement, un món tridimensional en una pantalla qualsevol, com els que es poden veure en, per exemple, els videojocs. Si ens fixem en aquesta indústria, segurament tindrem la idea preconcebuda de què com més elevat sigui el detall i realisme de les imatges (nombre de polígons, textures, il·luminació, etc.), millor serà l'experiència que tindrà l'usuari.

Paradoxalment, no es requereixin unes imatges virtuals de qualitat fotogràfica[2], sinó que només cal que l'entorn es mostri tal i com ho faria si fos real, utilitzant tècniques com induir la sensació de profunditat a l'usuari, o fent un seguiment actiu de la seva posició:

Projecció en estèreo Aquesta tècnica consisteix en projectar dues imatges lleugerament diferents de l'escena que s'està visualitzant; una per cada ull.

La tècnica es basa en què la percepció que té un humà de la realitat és lleugerament diferent per cada ull, de tal manera que s'aconsegueix la sensació de profunditat. Oferint-li les dues perspectives del món virtual aconseguim aquest mateix efecte.

Per tal de dur a terme la projecció en estèreo tenim dues tècniques:

- *Estèreo passiu*: es disposa de dos projectors, associats a cada ull, projectant ambdues imatges de forma superposada. La llum que emet cadascun d'ells es polaritza de forma diferent aplicant-los-hi un filtre, de tal manera que l'usuari, equipat amb unes ulleres de vidres també polaritzats, vegi a cada ull només una de les dues imatges.
- *Estèreo actiu*: en aquest cas es disposa d'un únic projector, el qual dibuixarà les imatges de l'ull esquerra i de l'ull dret de forma alternativa. La manera que té l'usuari de distingir-les i separar-les és utilitzant unes ulleres que, sincronitzades amb el projector, tapen un dels dos ulls alternativament.

Aquest tipus de visionat ha de tenir en compte la posició de l'espectador respecte a la pantalla, ja que la distància i la posició de la càmera al món virtual ha de correspondre's aproximadament amb la realitat. Si no es fes, l'usuari podria marejar-se o veure reduïda la sensació d'immersió, atès que les imatges podrien aparèixer deformades i les distàncies irreals.

La posició de l'espectador es pot assumir estàtica, fent-se, en aquest cas, una suposició de la ubicació real de l'usuari, o fer-ne un seguiment actiu (anomenat *head tracking*), tal i com veurem a continuació.

Head tracking Aquesta tècnica es veu en més detall al capítol 8 de la pàgina 79. A grans trets, consisteix en fer un seguiment actiu de la posició real de l'usuari en relació a la pantalla i, aprofitant aquesta informació, fer que la perspectiva que se li dóna del món virtual sigui la que toca.

[2] Això no vol dir, però, que augmentar el realisme no millori l'experiència que s'obté en utilitzar-los. Detalls com les ombres o les reflexions són molt importants.

Oïda

Una de les primeres aproximacions és el *so estèreo*, el qual reprodueix el so amb dos canals, un per a cada oïda, aconseguint diferenciar si el so ve de la dreta o de l'esquerra.

Actualment hi ha molts sistemes, força més evolucionats que l'anterior, que estimulen convenientment aquest sentit, essent molts d'ells heretats del cinema. L'exemple més clar són els *sistemes de so envolupant* que, disposant diferents altaveus al voltant de l'usuari, aconsegueixen un so tridimensional.

Una altra tècnica molt potent, però que no ha tingut molt ressò, és l'*holofonia*. L'objectiu és aconseguir un so envolupant però sense tenir la necessitat de disposar de molts altaveus que voregin a l'usuari. La manera que té d'aconseguir-ho és enregistrant les fonts de so de forma independent per a cada orella, utilitzant dos micròfons omnidireccionals situats al cap d'un *dummy*, tal i com s'ubicarien realment les orelles en un cap humà. L'èxit d'aquesta tècnica està en què, se suposa, imita la forma en què el cervell humà processa el so.

Tacte

Els dispositius que treballen amb aquest sentit s'anomenen *haptics*. Una de les aproximacions més comuns és un braç robotitzat, un dels extrems del qual s'agafa amb la mà. L'usuari pot agafar aquest extrem per moure el braç robotitzat que, treballant de forma conjunta amb algun programa que visualitzi on es troba la mà dins del món virtual, oferirà resistència quan el moviment efectuat impacti contra la superfície de l'objecte, donant una sensació de tacte.

Aquests sistemes encara tenen bastants problemes i inconvenients:

- Per una banda, aquest braç és força car i, per tant, només assequible pels centres d'investigació.
- Per altra banda, per a poder notar i resseguir una superfície virtual, la detecció de col·lisions ha de ser molt ràpida, o l'usuari notaria un tremolor que faria desaparèixer la sensació d'immersió. Això és degut a què el tacte és molt més sensible que la vista en el que es refereix a canvis.

Tot i tenir aquests petits problemes, són tècniques amb molt potencial, sobretot en aplicacions com la medicina, el disseny de cotxes, treball a distància, etc.

Olfacte

Finalment, també hi ha línies d'investigació obertes amb el sentit de l'olfacte, però és una via molt poc explorada. S'han construït alguns prototipus i proposat idees com integrar sensors i emissors d'olfacte en mòbils, dispositius connectats a l'ordinador que emeten olors, o fins i tot webs que suportin aquests dispositius i enviïn informació per utilitzar-los.

De totes maneres, les dificultats tècniques d'aquests aparells, i també la del programari per utilitzar-los, fan que encara sigui una idea del futur.

Cal dir, però, que es tracta de dispositius amb un gran potencial al darrere, perquè aquest sentit és un gran estimulant del cervell, essent un dels que més records evoca i reaccions agradables, o no, produeix a les persones.

2.3 Exemples

En aquest darrer apartat del capítol es mostren uns quants exemples dels sistemes de realitat virtual comentats.

2.3.1 La *cave*

Una *cave* consisteix en «una sala» de $3 \times 3 \times 3m$ on es projecten imatges a quatre cares del cub (tres parets i un terra). L'usuari se situa a l'interior d'aquest espai i s'endinsa, d'aquesta manera, al món virtual, tal i com es pot veure a la figura 2.1.

Figura 2.1: Exemple d'una *cave*, amb dos usuaris ubicats al seu interior.

L'aplicació 3D que gestiona la *cave* hi projecta el món virtual, de tal manera que no es noti que hi ha parets; l'usuari, envoltat d'aquestes imatges, tindrà la sensació de ser-hi dins.

Per tal de millorar-ne l'experiència, les caves han evolucionat i, actualment, incorporen la visió estèreo i el seguiment de l'usuari (vegeu la secció 2.2.1).

2.3.2 La *Power Wall*

La *power wall* es podria descriure com l'evolució d'un cinema tradicional. Es tracta d'una pantalla tan gran com es pugui, generalment amb estereovisió passiva i, fins i tot, so envolupant.

L'aplicació que la gestiona suposa que l'usuari està visualitzant la pantalla des del centre i, tot i que això no és exacte, atès que s'acostumen a asseure en cadires i, per tant, no és possible que totes elles estiguin, òbviament, al centre, es tracta d'una aproximació vàlida (especialment perquè els espectadors són estàtics).

Figura 2.2: Exemple d'una *Power Wall*.

2.3.3 Ulleres de realitat virtual

Les ulleres de realitat virtual van ser una de les primeres aproximacions que es van construir. La idea es tenir dues pantalles molt petites separades i que cada ull només en pugui veure'n una, muntant-les sobre unes ulleres.

Aquest tipus de dispositiu permet fer *head tracking* mesurant els girs del cap amb acceleròmetres, suposant que l'usuari està quiet a una cadira, o es pot mesurar amb un sistema com el de la *cave*, i llavors l'usuari es pot moure.

Els inconvenients són que les pantalles muntades a les ulleres han de ser molt bones, perquè han de tenir una resolució prou alta i ser suficientment grans com per a què l'usuari no hagi de forçar la vista i la qualitat de la imatge sigui bona. També hi ha altres problemes molt més difícils de solucionar, com que l'aplicació típicament només té coneixement de la posició del cap, però no de la resta del cos i, per tant, no el pot ensenyar.

Figura 2.3: Exemple d'unes ulleres de realitat virtual.

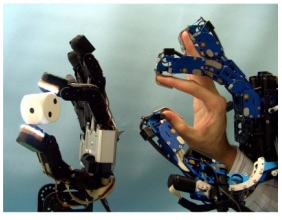
Actualment, s'està investigant l'evolució d'aquestes ulleres per tal d'eliminar aquests problemes. La idea és utilitzar unes ulleres normals, però incloure uns petits projectors a les patilles. D'aquesta manera, i fent el seguiment de l'usuari, podem solucionar el problema de què es maregi per no veure el seu propi cos, aprofitant que els vidres serien transparents i se'l podria veure. També se solucionarien els problemes de què la pantalla no és prou gran. A la figura 2.3 podem veure unes ulleres de realitat virtual actuals.

2.3.4 Dispositius *haptics*

Aquests dispositius se centren en el sentit del tacte. A la figura 2.4 que hi ha tot seguit s'hi poden veure uns quants exemples.

(a) Exemple de braç robotitzat *haptic*.

(b) Exemple de robot *haptic* petit.



(c) Exemple de dispositiu *haptic* per a la mà.

Figura 2.4: Diferents exemples de dispositius *haptics*.

La figura 2.4(a) mostra un braç robotitzat. L'usuari agafa la punta més fina amb la mà i la va movent lliurement fins que troba un objecte virtual, de tal manera que el robot ofereix resistència al moviment en aquell sentit i l'usuari nota l'impacte. Es tracta d'elements força cars i, conseqüentment, difícilment assequibles, però ja s'està començant a fabricar-ne de més petits, com el de la figura 2.4(b).

Altres tipus de dispositius són les *mans robotitzades*, les quals estan connectades a una altra mà que té l'usuari. Aquest darrer pot moure la seva pròpia mà lliurement; la mà robotitzada que duu posada està sincronitzada amb la que està tocant l'objecte a distància, tal i com podem veure molt més clarament a la figura 2.4(c).

CAPÍTOL 3

Propòsit del projecte

Aquest capítol, a diferència del primer, pretén descriure amb un nivell de detall força més elevat les característiques del sistema que es vol desenvolupar, de tal manera que es comprengui millor el seu abast i l'entorn en què s'ubica. Això implicarà analitzar quin és el context actual en el món dels sistemes de realitat virtual, com aquest projecte pretén canviar-lo i discutir les alternatives existents que pugui haver-hi ja.

Per altra banda, també es parla de les persones que es veuen afectades pel projecte, descrivint el seu rol i el nivell d'implicació que hi tindran.

3.1 Context actual

Al capítol 2 es comenta què és un sistema de realitat virtual i algunes de les tècniques emprades actualment per tal de millorar la immersió que ofereix. També s'hi poden veure alguns exemples de sistemes de realitat virtual, tals com una *cave* o una *power wall*.

Els sistemes de realitat virtual que hi ha avui en dia no estan a l'abast del públic general, atès que acostumen a tenir uns costos molt elevats i requerir d'unes instal·lacions específiques. Tenint en compte que un parell dels objectius a acomplir són la *portabilitat* i la *reducció de costos*, el projecte es perfila com una solució apta tant per a entitats que ja disposen de sistemes de realitat virtual "clàssics" com per al públic en general.

De fet, la idea d'aquest PFC naix amb l'objectiu de dissenyar un sistema que pugui competir amb una *cave*, solucionant-ne els problemes que veurem tot seguit.

3.2 Problema a resoldre

Una *cave* consisteix en «una sala» de $3 \times 3 \times 3m$ on es projecten imatges a quatre cares del cub (tres parets i un terra). L'usuari se situa a l'interior d'aquest espai i s'endinsa, d'aquesta manera, al món virtual.

Tenint en compte que l'ideal que persegueix el nostre projecte és esdevenir el substitut d'una *cave*, és necessari comprendre quins són els principals problemes que aquesta té:

- La ≪sala≫ (la *cave* en si) ha d'estar ubicada a un emplaçament molt més gran que permeti ubicar-hi les pantalles de projecció, els projectors i l'equip informàtic.
- Per tal d'evitar les ombres que produiria l'usuari al passar entre el projector i la pantalla de projecció, la imatge és retroprojectada. Això vol dir que les pantalles han de ser especials per a què la imatge es pugui veure amb prou lluminositat.
- Els projectors han de ser capaços d'enviar imatges amb una resolució força gran, i projectar en superfícies de $3 \times 3m$. Es tracta, doncs, de projectors d'una gran envergadura i d'elevat cost.
- Donat el cost dels projectors, no és viable plantejar-se tenir-ne dos per cada pantalla projectada (essent, d'aquesta manera, un total de vuit). Per tant, per tal de generar imatges estèreo s'ha d'utilitzar l'*estèreo actiu*; això implica que cada ull només veu les imatges la meitat del temps, amb la conseqüent pèrdua de lluminositat que això implica.

L'àmbit d'ús d'una *cave* és molt reduït, al tractar-se d'una solució molt restrictiva: la portabilitat d'aquest sistema (entenent portabilitat com la possibilitat de desmuntar-lo i muntar-lo en una altra ubicació) és pràcticament nul·la.

Aconseguir un substitut de la *cave* és una tasca titànica, i més tenint en compte els objectius plantejats (vegeu la pàgina 2). Amb els termes en què es defineix ara mateix el projecte no es disposa d'una pantalla tant gran com pot ser la d'una *cave*[1], ni tampoc de més d'una paret que envolti a l'usuari. Tampoc s'hi implementa la visió en estèreo[2].

El PFC, doncs, es perfila com el *predecessor* d'un sistema molt més ampli i ambiciós, el qual sí que pot arribar a competir, realment, amb una *cave*.

3.3 Resultats esperats

Es vol un sistema que sigui *altament portable*, que funcioni en *qualsevol lloc* de *forma automàtica* i que tingui un bon nivell d'*immersió*; és a dir, que assoleixi els objectius descrits a l'apartat 1.3.

Acomplir aquests objectius vol dir solucionar els principals problemes que planteja una *cave*, o molts dels sistemes de realitat virtual d'avui en dia: la portabilitat i el cost de la infraestructura.

Al capítol 7 es descriuen les tècniques utilitzades per tal d'aconseguir que la infraestructura desenvolupada es calibri a una ubicació donada qualsevol de forma automàtica, permetent que la intervenció de l'usuari sigui pràcticament nul·la.

Al capítol 9, en canvi, es munta una aplicació damunt d'aquesta infraestructura, per tal de demostrar-ne el correcte funcionament. A més a més, s'hi integra el *head tracking* per a poder millorar l'experiència de l'usuari a l'utilitzar el sistema.

3.4 Alternatives al projecte

Actualment, no hi ha sistemes de realitat virtual que disposin de les característiques amb què volem dotar el nostre projecte. El principal handicap que tenen aquests sistemes és la portabilitat que, o és pràcticament inexistent, o no són tant flexibles com el que presentem.

[1] Tot i que no resultaria molt complicat d'aconseguir (vegeu [CSWL02]).

[2] Atès que el sistema plantejat admet la projecció d'imatges sobre qualsevol paret, s'hauria d'emprar l'estèreo actiu (el passiu és inviable, ja que es perdria la polarització de la llum). La utilització de l'estèreo actiu no representa cap dificultat teòrica, i només requereix la generació de *frames* diferents per a cada ull, sense haver de calibrar parelles de projectors (el projector de l'ull esquerre amb el del dret).

3.5 *Stakeholders* i usuaris

Els *stakeholders* són tota entitat afectada o necessària pel desenvolupament i ús d'un sistema. Tractant-se d'un projecte de realitat virtual i caire certament acadèmic, el nombre d'*stakeholders* que hi ha és força reduït.

Els rols que s'han identificat són:

Administrador És aquella persona que s'encarrega de *muntar* i *configurar* la infraestructura del sistema.

Controlador És aquella persona que supervisa l'ús que l'usuari està fent del sistema.

Usuari És la persona que utilitza el sistema i que, per tant, s'"endinsa" en el món virtual.

Es pot considerar que l'*administrador*, tal qual s'ha definit, està, de fet, realitzant dos rols diferents: el que s'encarrega de la part física de la infraestructura, muntant els projectors, les càmeres, etc., i el que s'encarrega de la part programàtica del sistema, configurant-lo convenientment (executant els processos de calibratge, l'aplicació d'usuari, etc.). Per altra banda, el *controlador* és una figura que pot ser-hi present o no, ja que la seva participació no és estrictament necessària.

CAPÍTOL 4

Planificació del projecte

La *planificació* d'un projecte és una de les parts més importants de la *gestió de projectes*.

El primer que cal fer és definir l'àmbit del projecte[1], de tal manera que, a continuació, es puguin definir les tasques que s'hauran d'assolir per a completar-lo, així com la relació que hi ha entre elles. Tot plegat fa possible estimar la duració total que necessitarà el projecte, determinar les tasques crítiques que si duren més d'allò que s'ha previst provocarien que s'acabés més tard, calcular el cost que suposarà desenvolupar-lo o coordinar els recursos que es necessitaran.

En aquest capítol es mostren les diferents tasques en què hem decidit descompondre el projecte i les seves dependències, qui s'encarrega d'elles i les fites que s'han d'assolir.

4.1 Introducció

El projecte s'inicia formalment a mitjans de juliol de 2008, moment en què es matricula, tot i que el procés d'estudi i desenvolupament va iniciar-se abans, a principis de l'any 2008, que és quan es contacta amb Pere Brunet i es decideix inscriure'l a la facultat.

Aquest capítol inclou totes les tasques realitzades des del moment en què es va inscriure fins a la seva fi.

4.2 Tasques a dur a terme

La taula que tenim a continuació mostra totes les tasques en què s'ha dividit el projecte i indica, per cadascuna d'elles, qui és el responsable de fer-la. Les tasques es corresponen a les del diagrama de Gantt de la figura 4.2.

[1] Vegeu el capítol 3

Nom	Assignat a
Tasques administratives	
Matriculació del PFC	Ambdós
Redacció informe previ	Ambdós
Entrega informe previ	Ambdós
Redacció memòria	Ambdós
Revisió memòria	Ambdós
Correccions memòria	Ambdós
Impressió i enquadernació memòria	Ambdós
Verificació i entrega e la memòria	Ambdós
Pactar data de presentació	Ambdós
Preparació de la presentació	Ambdós
Presentació	Ambdós
Estudi previ	
Lectura dels *papers*	Ambdós
Definició d'objectius del projecte	Ambdós
Especificació del mòdul de calibratge	
Acotar objectius del mòdul	Ambdós
Definició dels casos d'ús	David
Anàlisi de requisits	David
Definició de les classes	David
Disseny del mòdul de calibratge	
Estudi d'integració xarxa al codi Java	David
Modificació diagrama classes per integrar-hi patrons	David
Definició dels algorismes de coordinació	David
Desenvolupament calibratge geomètric	
Implementació	
Impl. de les classes	Ambdós
Impl. dels algorismes de visió per computador	Rodrigo
Impl. del mòdul de xarxa	David
Impl. dels algorismes de calibratge	Ambdós
Integració dels components	Ambdós
Testos	
Test conjunt dels components	David
Proves de calibratge geomètric	David
Desenvolupament calibratge cromàtic	
Implementació	
Impl. de les classes	David
Impl. dels algorismes de calibratge	David
Integració amb el mòdul geomètric	David
Testos	
Test de l'algorisme de calibratge	David
Test amb calibratges reals	David
Correccions necessàries al calibratge	
Correcció del calibratge geomètric	David
Correcció del calibratge cromàtic: intensitats	David
Correcció del calibratge cromàtic: zona d'intersecció	David

(continua a la següent pàgina)

(ve de la pàgina anterior)

Escriptura de resultats	
Definició de resultats a compartir	Ambdós
Proves d'escriptura	Ambdós
Especificació de l'aplicació d'usuari	
Acotar objectius del mòdul	Ambdós
Definició dels casos d'ús	David
Anàlisi de requisits	David
Estudi de l'aplicació [CL]	Rodrigo
Definició de les classes	Rodrigo
Disseny de l'aplicació d'usuari	
Estudi dels diversos patrons de xarxa	Rodrigo
Modificació diagrama classes per integrar-hi patrons	Rodrigo
Desenvolupament del mòdul de visualització	
Implementació	
Estudi de C#	Rodrigo
Reorganització del codi	Rodrigo
Integració de la xarxa	Rodrigo
Lectura dels resultats del calibratge	Rodrigo
Implementació dels *shaders*	Rodrigo
Proves	
Proves de xarxa	Rodrigo
Proves de *shaders*	Rodrigo
Proves de tota l'aplicació	Rodrigo
Prototipus	
Estudi de les necessitats necessitats	Ambdós
Adquisició / reutilització de materials	Ambdós
Construcció dels prototipus	Ambdós

Taula 4.1: Tasques a realitzar des de la matriculació del projecte fins al final.

4.3 Planificació estimada

La primera planificació que es va dur a terme no plantejava les tasques en què s'hauria de dividir l'*aplicació d'usuari*, i per aquest motiu el diagrama de Gantt de la figura 4.1 estima tot el procés com una única tasca la duració de la qual és d'uns, aproximadament, dos mesos.

4.4 Planificació corregida

A mitjans de juliol de 2008, moment en què es matricula el projecte, es decideix refer el diagrama de Gantt, per tal de, per una banda, incloure-hi els retards que hi havien hagut durant els primers mesos de desenvolupament i, per l'altra, de detallar les tasques que composen la segona part del projecte. El diagrama de la figura 4.2 mostra les següents particularitats:

- S'han adaptat les estimacions de les tasques dutes a terme a les seves duracions reals.

- S'adapten els inicis de la resta de tasques, degut a què les duracions inicialment previstes de les primeres tasques s'han estès.
- S'hi introdueix la planificació de la segona part del projecte: l'aplicació d'usuari.

4.5 Duració real

Finalment, la figura 4.3 mostra un diagrama amb la duració real que ha tingut el projecte, per tal de poder-ho comparar amb la planificació que s'havia fet.

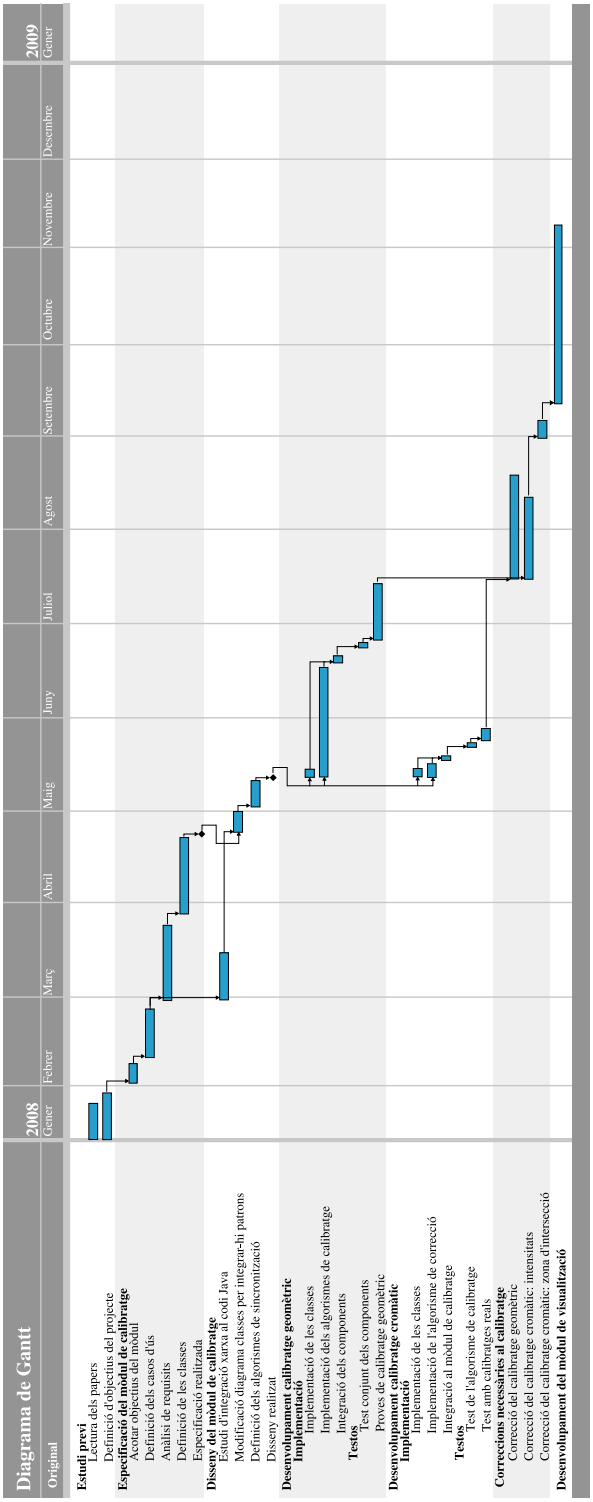


Figura 4.1: Diagrama de Gantt amb la planificació estimada a l'inici del projecte, quan es va inscriure.

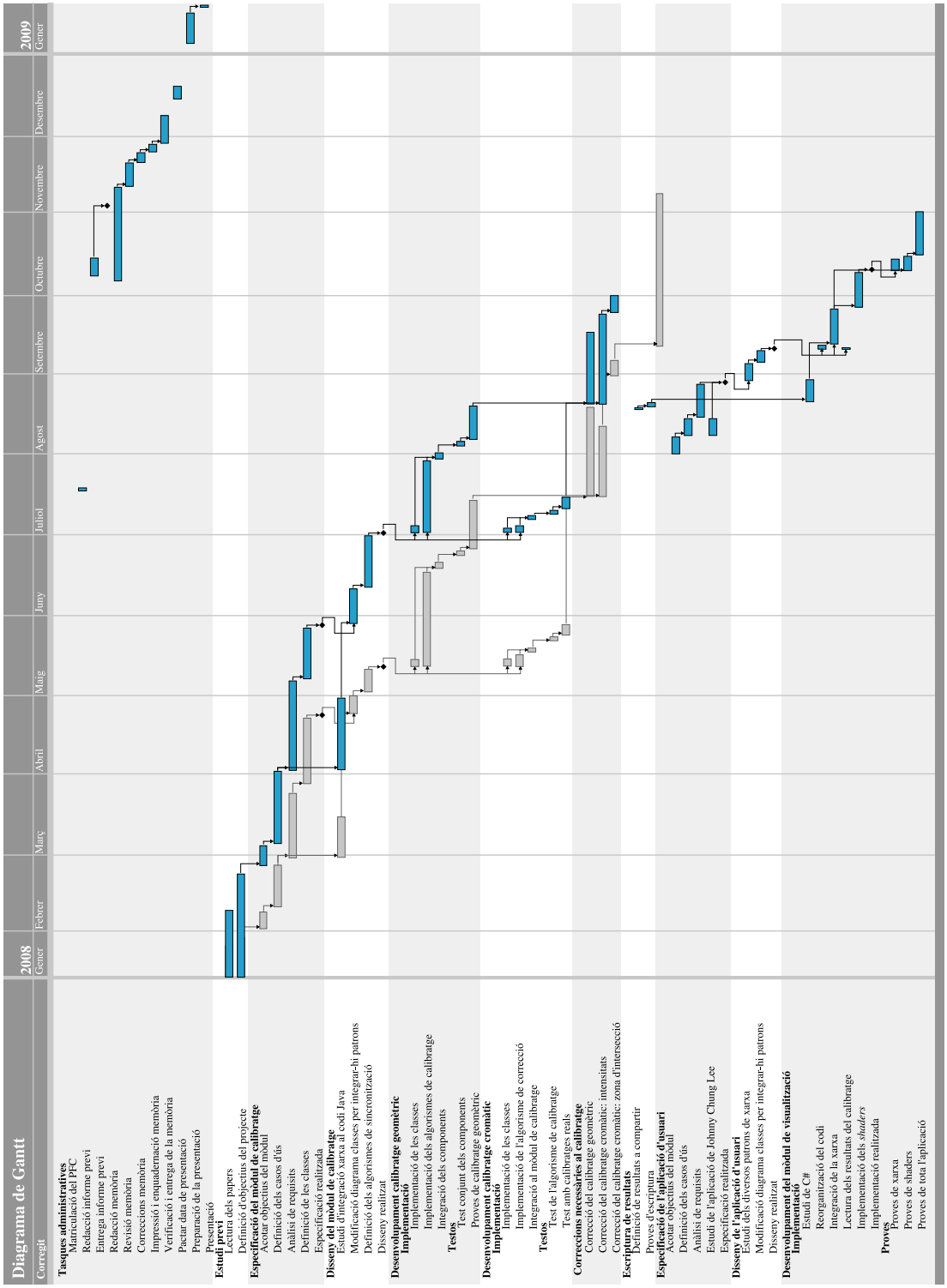


Figura 4.2: Diagrama de Gantt actualitzat al moment de matricular el projecte, corregint les duracions.

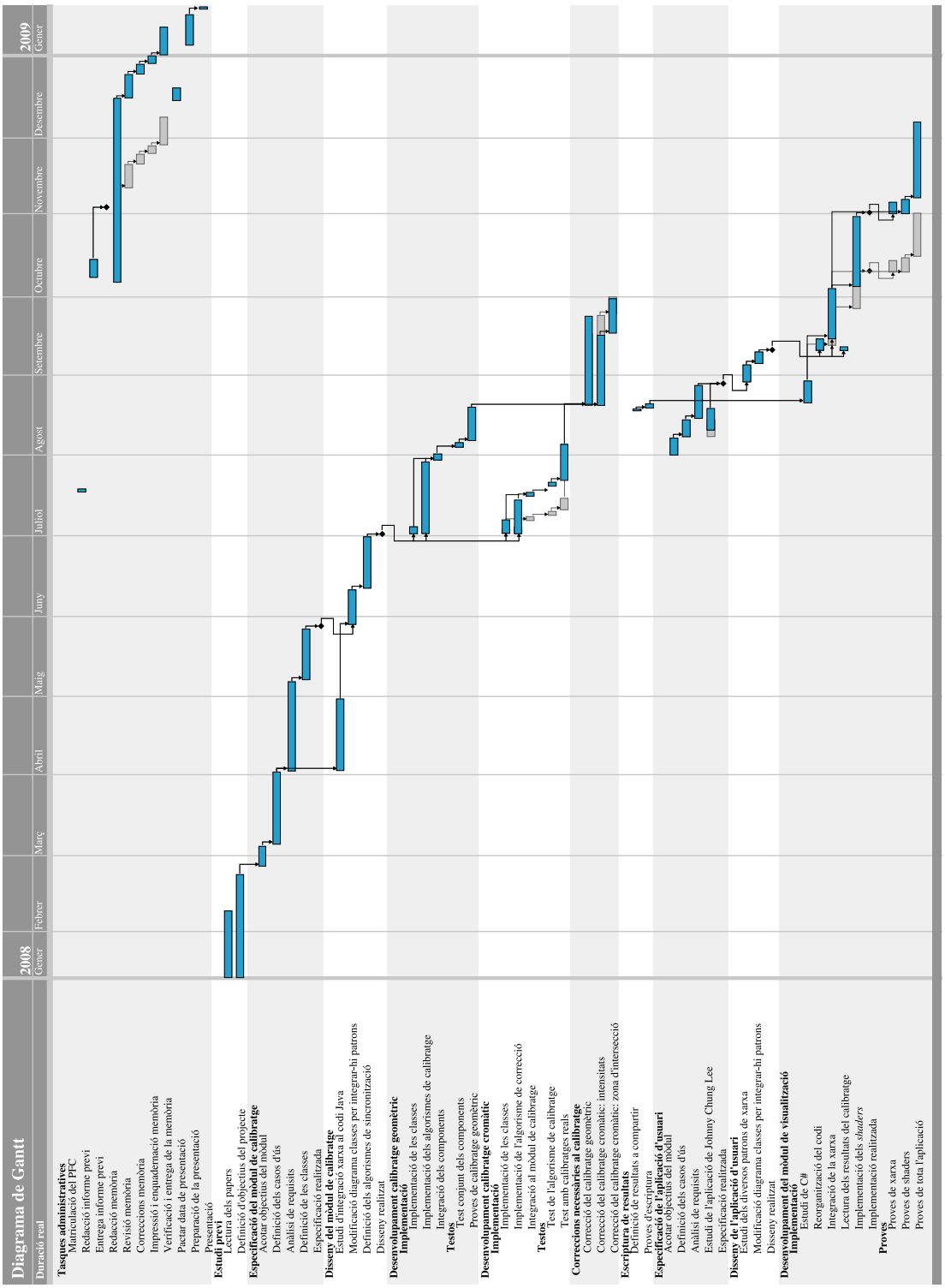


Figura 4.3: Diagrama de Gantt final, on es mostra la durada real de cadascuna de les tasques.

CAPÍTOL 5

Homografies

En aquest apèndix s'explica què són les homografies, per a què serveixen i com encaixen dins d'aquest projecte.

També s'explica amb força detall com s'ha implementat el seu càlcul amb les eines disponibles al sistema (càmeres i projectors), comentant les diferents iteracions que s'han dut a terme per a arribar a la solució final. Això vol dir que es comença descrivint com calcular una homografia quan només es té un projector i una càmera, i com s'acaba arribant a fer-ne el còmput amb N projectors i N càmeres.

5.1 Què són les homografies?

Una *homografia* és un concepte emprat en l'àmbit de la geometria projectiva. Aquesta defineix la relació que hi ha entre dues imatges, de tal manera que donat un punt qualsevol d'una d'elles, aquest correspon a un, i només un, punt de l'altra.

Aquesta correspondència entre dos punts se sol expressar amb una matriu H que té la següent forma:

$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix}$$

Sigui $P = (x_p, y_p, 1)$ un punt qualsevol que pertany a la primera figura i $Q = (x_q/r, y_q/r, 1)$ un altre que pertany a la segona, i saben que entre aquests dos punts existeix la correspondència abans esmentada, es té que:

$$\begin{pmatrix} x_q \\ y_q \\ r \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x_p \\ y_p \\ 1 \end{pmatrix}$$

5.2 Com calcular una homografia

Per tal de calcular una homografia pla-a-pla es poden emprar dos mètodes, la *solució lineal no homogènia* i la *solució homogènia*. A continuació es descriuen ambdós mètodes.

5.2.1 Mètode d'estimació homogènia

Escrivint l'homografia H en forma de vector $h = (h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}, h_{33})^T$, les equacions homogènies per a n punts esdevenen $Ah = 0$, essent A la matriu de mida $2n \times 9$ següent:

$$A = \begin{pmatrix} x_{p_1} & y_{p_1} & 1 & 0 & 0 & 0 & -x_{p_1}x_{q_1} & -y_{p_1}x_{q_1} & -x_{q_1} \\ 0 & 0 & 0 & x_{p_1} & y_{p_1} & 1 & -x_{p_1}y_{q_1} & -y_{p_1}y_{q_1} & -y_{q_1} \\ x_{p_2} & y_{p_2} & 1 & 0 & 0 & 0 & -x_{p_2}x_{q_2} & -y_{p_2}x_{q_2} & -x_{q_2} \\ 0 & 0 & 0 & x_{p_2} & y_{p_2} & 1 & -x_{p_2}y_{q_2} & -y_{p_2}y_{q_2} & -y_{q_2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{p_n} & y_{p_n} & 1 & 0 & 0 & 0 & -x_{p_n}x_{q_n} & -y_{p_n}x_{q_n} & -x_{q_n} \\ 0 & 0 & 0 & x_{p_n} & y_{p_n} & 1 & -x_{p_n}y_{q_n} & -y_{p_n}y_{q_n} & -y_{q_n} \end{pmatrix}$$

El vector h que minimitza els residus de $\det(A)$, essent $\det(H) = 1$, és obtingut pel vector propi del menor valor propi d'A^TA. Aquest vector propi es pot obtenir directament de la descomposició de valor singular (SVD, per les seves sigles en anglès) d'A^TA.

5.2.2 Solució lineal no homogènia

En aquest mètode, un dels elements de l'homografia té un valor fixat, generalment la unitat, computant la solució dels altres vuit elements via una pseudo inversa. Com a desavantatge es té que s'obtenen resultats força pobres si el valor de l'element escollit hauria d'haver sigut zero.

En el cas que es discuteix en aquest projecte, quan diem que $Q = HP$, es considera que el símbol d'igualtat vol dir *igualtat ignorant el factor d'escalat*. Aquest no afecta a l'equació i, per tant, només vuit dels valors d'aquesta homografia són significatius.

Per tant, si assumim que

$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{pmatrix}$$

per tal de calcular els vuit elements que manquen d'H només caldrà resoldre el següent sistema:

$$\begin{pmatrix} x_{p_1} & y_{p_1} & 1 & 0 & 0 & 0 & -x_{p_1}x_{q_1} & -y_{p_1}x_{q_1} \\ 0 & 0 & 0 & x_{p_1} & y_{p_1} & 1 & -x_{p_1}y_{q_1} & -y_{p_1}y_{q_1} \\ x_{p_2} & y_{p_2} & 1 & 0 & 0 & 0 & -x_{p_2}x_{q_2} & -y_{p_2}x_{q_2} \\ 0 & 0 & 0 & x_{p_2} & y_{p_2} & 1 & -x_{p_2}y_{q_2} & -y_{p_2}y_{q_2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{p_n} & y_{p_n} & 1 & 0 & 0 & 0 & -x_{p_n}x_{q_n} & -y_{p_n}x_{q_n} \\ 0 & 0 & 0 & x_{p_n} & y_{p_n} & 1 & -x_{p_n}y_{q_n} & -y_{p_n}y_{q_n} \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{pmatrix} = \begin{pmatrix} x_{q_1} \\ y_{q_1} \\ x_{q_2} \\ y_{q_2} \\ \vdots \\ x_{q_n} \\ y_{q_n} \end{pmatrix}$$

En cas de què el sistema estigui sobre determinat (és a dir, $n > 4$), s'ha de calcular la solució utilitzant mínims quadrats.

5.3 Càlcul amb un projector i una càmera

Aquest és el cas més senzill amb el que podem treballar. L'objectiu serà associar les coordenades dels punts que es veuen a la càmera amb les coordenades que s'han enviat.

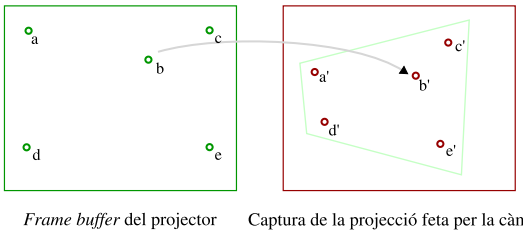


Figura 5.1: Correspondència entre punts enviats per un projector i punts capturats per una càmera.

Quan es disposa de la informació de correspondències s'aplica el mètode vist anteriorment per tal de computar l'homografia H. Amb ella, donat un punt qualsevol P_p del projector, podem saber el punt P_c associat que obtindríem en el sistema de coordenades de la càmera.

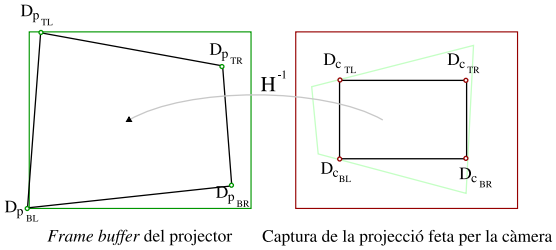


Figura 5.2: Càlcul de les coordenades de la pantalla final per a un projector i una càmera.

Tenint en compte que l'objectiu és construir una pantalla quadrada des del punt de vista de la càmera, es poden calcular les coordenades dels quatre vèrtexs ($D_{c_{TL}}$, $D_{c_{TR}}$, $D_{c_{BL}}$, $D_{c_{BR}}$) en el sistema de coordenades d'aquesta i aplicar H^{-1} per a conèixer quina zona del *frame buffer* del projector s'hauria de pintar (vegeu la figura 5.2).

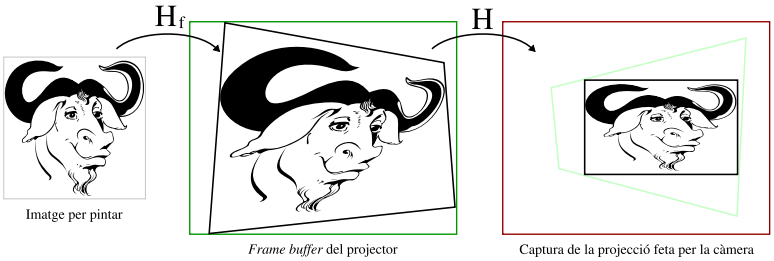


Figura 5.3: Diagrama de les transformacions que apliquem a la imatge inicial per a què al ser capturada per la càmera es vegi correctament, sense deformacions.

Finalment, es calcula una nova homografia H_f que converteixi les coordenades d'una imatge normalitzada ($x \in [0-1]$ i $y \in [0-1]$) cap a les coordenades ($D_{p_{TL}}$, $D_{p_{TR}}$, $D_{p_{BL}}$, $D_{p_{BR}}$). Tal i com es pot veure a la imatge 5.3, això permet que quan la imatge sigui projectada pel projector

i capturada per la càmera, el que aquesta última vegi sigui la imatge inicial, sense deformacions de perspectiva[1].

5.4 Càlcul amb dos projectors i una càmera

A continuació, s'afegeix un nou projector al muntatge descrit anteriorment, complicant-lo lleugerament. Tal i com es pot apreciar a la figura 5.4, ara la càmera capta els patrons de dos projectors i, per tant, es disposa de la geometria de les dues àrees projectades (en verd i vermell).

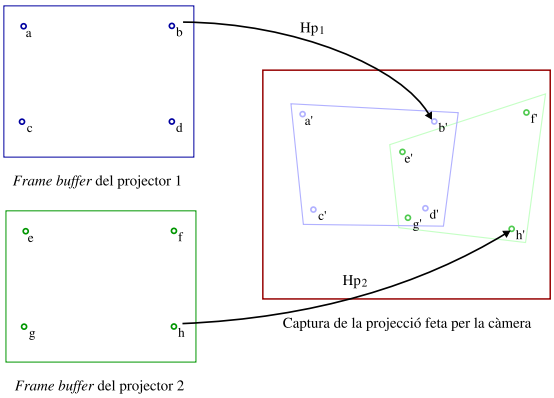


Figura 5.4: Correspondència entre punts enviats per dos projectors i punts capturats per una càmera.

De manera anàloga al cas anterior, es pot calcular una homografia H_{P_1} que converteixi els punts del *frame buffer* del projector P_1 a coordenades de la càmera, així com una homografia H_{P_2} que faci el mateix per al projector P_2. Tot seguit, caldrà calcular la geometria de la pantalla on es dibuixaran les imatges, tenint en compte que l'àrea que la contindrà és la unió de les àrees projectades per cada mòdul (vegeu la figura 5.5).

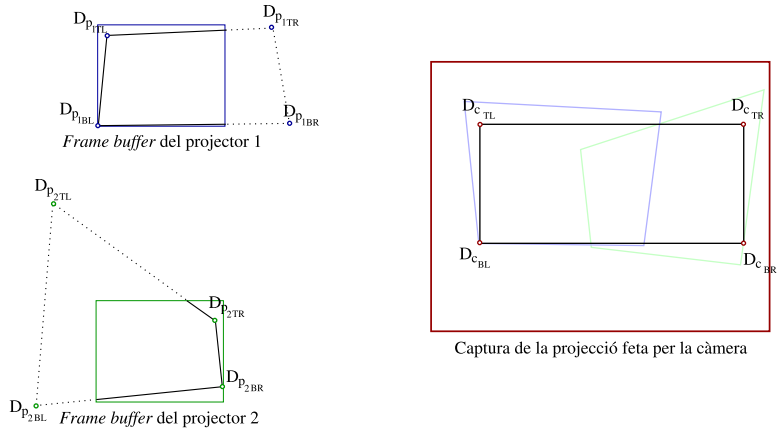


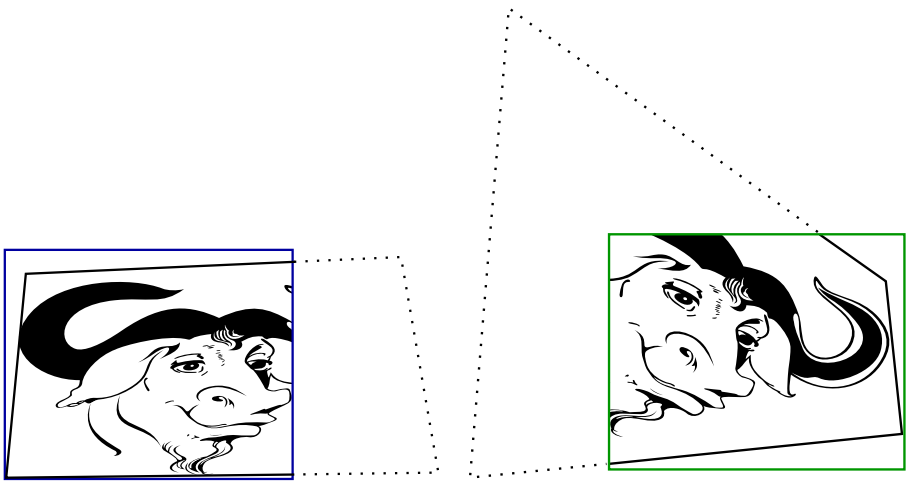
Figura 5.5: Càlcul de les coordenades de la pantalla final utilitzant dos projectors i una càmera.

Aplicant les homografies inverses es poden obtenir les coordenades ($D_{c_{TL}}$, $D_{c_{TR}}$, $D_{c_{BL}}$, $D_{c_{BR}}$) de la pantalla en els respectius sistemes de coordenades de cada projector. En aquest

[1] Això no vol dir que no hi pugui haver deformacions de relació d'aspecte.

cas, és possible que algunes d'aquestes coordenades "quedin fora" del *frame buffer*. Això és totalment normal, atès que ara un projector només dibuixarà, en general, una part de la imatge, donant per fet que la resta de la imatge la pintarà l'altre.

Finalment, només cal calcular les homografies finals $H_{f_{P_1}}$ i H_{fP_2} que, de la mateixa manera que en el cas d'un projector i una càmera, situaran la imatge correctament dins dels *frame buffers*.



(a) Imatge final situada a l'interior del *frame buffer* del projector 1.

(b) Imatge final situada a l'interior del *frame buffer* del projector 2.

Figura 5.6: Imatge final situada a l'interior dels *frame buffers*.

5.5 Càlcul amb N projectors i una càmera

Si se suposa que tenim N projectors enlloc de dos, el problema a resoldre és el mateix i no suposa cap complicació addicional.

En aquest cas, és necessari calcular les N homografies que ens permetran passar les coordenades de la pantalla final (calculades en el sistema de coordenades de la càmera) a les coordenades de cadascun dels projectors i computar, aleshores, les homografies finals que situïn una imatge normalitzada a cadascun dels N *frame buffers*.

Ara bé, cal tenir en compte que quan el nombre de projectors va augmentant, les dimensions que podrà assolir la pantalla final també creixen. Disposar d'una única càmera implica que arribarà un punt en què aquesta no tindrà un camp de visió prou ampli com per a veure tota l'àrea projectada. Tot i que allunyant-la s'aconseguiria veure-ho tot, la resolució de la càmera no permetria fer-ho amb prou detall.

La manera de solucionar-ho consisteix en disposar de més d'una càmera, de tal manera que cadascuna d'elles visualitzi una petita porció del total d'àrea projectada.

5.6 Càlcul amb N projectors i M càmeres

El cas més general consisteix en disposar d'N projectors i M càmeres, de tal manera que es podrà construir una pantalla tan gran com es desitgi.

Hem vist que si s'utilitza un elevat nombre de projectors i es pretén alinear-los utilitzant una única càmera, aquesta ha d'estar situada molt lluny per a què pugui capturar tota l'àrea projectada, perdent el nivell de detall que ens ofereix cada captura. Utilitzar més càmeres, en canvi, fent que cadascuna d'elles capturi només una part d'aquesta àrea, permet situar-les més a prop i, d'aquesta manera, aprofitar millor la resolució que tenen.

El càlcul de la geometria de la pantalla continguda dins d'un conjunt d'àrees projectades té una particularitat evident que ara cal subratllar: s'ha realitzat en un únic sistema de coordenades. Quan només es disposava d'una càmera, i aquesta realitzava les captures de cada projector, s'obtenia la geometria que projectava cadascun d'ells en un sistema de coordenades SC_{cam}. El fet de tenir totes les àrees representades al mateix espai permetia computar fàcilment la geometria de la pantalla al SC_{cam}.

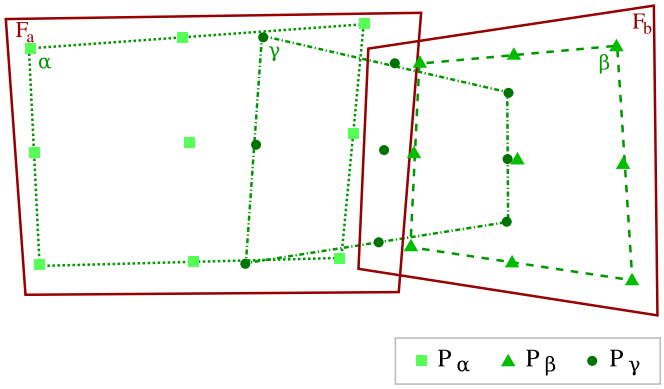


Figura 5.7: Exemple d'una paret amb $N = 3$ projeccions en verd i $M = 2$ captures en vermell.

Tenir M càmeres implica tenir M sistemes de coordenades diferents. Tenint en compte que l'objectiu és calcular les coordenades de la pantalla final, i que per a fer-ho necessitem tenir totes les àrees projectades en un únic sistema de coordenades, és prou clar que tenir la informació en M sistemes de coordenades diferents és un problema que cal resoldre.

Per tal d'aconseguir-ho les captures tenen una particularitat que permet solucionar el problema dels M sistemes de coordenades: les zones fotografiades no són disjuntes.

Siguin a i b les càmeres de què disposem i α, β i γ els projectors. Definim F_c com la captura que ha fet la càmera c, P_p com un dels punt del projector p i P_{pc} com un dels punts del projector p capturat per la càmera c.

Si ens fixem en l'exemple de la figura 5.7, les captures F_a i F_b tenen en comú que ambdues veuen punts P_γ, però que, en canvi, només a veu punts P_α i només b veu punts P_β. El problema, doncs, és que no tenim els tres conjunts de punts en el mateix sistema de coordenades: o bé ho estan P_α i P_γ, i P_β queda fora, o bé P_β i P_γ, i és P_α qui no hi està present. Tenint en compte que necessitem tenir-los tots representats al mateix espai, haurem de moure el tercer conjunt de punts (P_β en el primer cas, P_α en el segon) cap al sistema de coordenades dels altres dos.

Definim H_{pc} com l'homografia que transforma un punt en el sistema de coordenades del *frame buffer* de p a un altre punt en el sistema de coordenades de c, la qual s'ha calculat com hem descrit en aquest capítol. Per altra banda, també definirem H_{cp} com l'homografia que converteix un punt en el sistema de coordenades de la càmera c a un punt del *frame buffer* de p; és a dir, $H_{pc}^{-1} = H_{cp}$.

A l'exemple que estem tractant, les homografies de què disposem[2] són $H_{\alpha a}$, $H_{\gamma a}$, $H_{\gamma b}$ i $H_{\beta b}$.

[2]No tenim les homografies $H_{\alpha b}$ ni $H_{\beta a}$ perquè, evidentment, no s'han pogut calcular; la càmera a no veu els punts del projector β ni la càmera b els d'α.

Si volem tenir tots els punts en el sistema de coordenades de, per exemple, a, només caldrà aplicar les homografies convenientment. Per a passar obtenir un punt P_α al sistema de coordenades d'a utilitzarem directament l'homografia $H_{\alpha a}$; per a fer-ho amb punts P_γ utilitzarem $H_{\gamma a}$; i, finalment, per a fer-ho amb els punts P_β haurem d'aplicar una composició de matrius:

El que volem és poder passar qualsevol punt P_β al sistema de coordenades d'a[3], però, com ja hem vist, no disposem de l'homografia $H_{\beta a}$. El que farem, doncs, és passar un P_β qualsevol al sistema de coordenades de b aplicant $H_{\beta b}$. Tot seguit, aplicant $H_{\gamma b}^{-1}$ passem aquest punt al sistema de coordenades del projector γ. Finalment, només ens cal aplicar $H_{\gamma a}$ a aquest nou punt per tal de tenir P_β al sistema de coordenades d'a. Formalment, tenim que:

$$H_{\beta a} = H_{\beta b} H_{b\gamma} H_{\gamma a}$$

La solució descrita és vàlida per a qualsevol nombre de projectors i càmeres, sent sempre possible transformar un punt qualsevol d'un projector donat al sistema de coordenades de qualsevol càmera. Ara bé, el fet d'anar concatenant les matrius implica un subtil problema, i és que es va *acumulant error*. Tal i com s'han construït les homografies de què es parteix, aquestes són susceptibles de tenir petits errors de precisió; a mesura que anem movent un punt d'un sistema de coordenades a un altre, tot aplicant-li les homografies, anem introduint-hi més i més error.

[CSWL02] introdueix el concepte *camera homography tree*, amb el qual s'aconsegueix una solució que minimitza l'error resultant. Tot seguit, es comenta breument la idea que hi ha al darrere.

5.6.1 *Camera homography trees* [CSWL02]

Sigui $G(V, E)$ un CHG (*Camera Homography Graph*, Graf d'homografies de càmera), on cada vèrtex del conjunt V és la captura d'una càmera i una aresta E correspon a una homografia directament computable entre dues vistes, o, altrament dit, una aresta connecta dos vèrtexs només si tenen, com a mínim, un projector en comú.

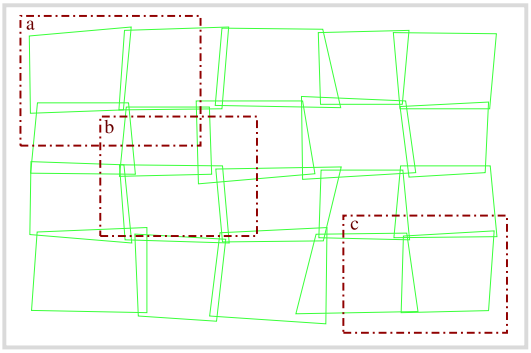
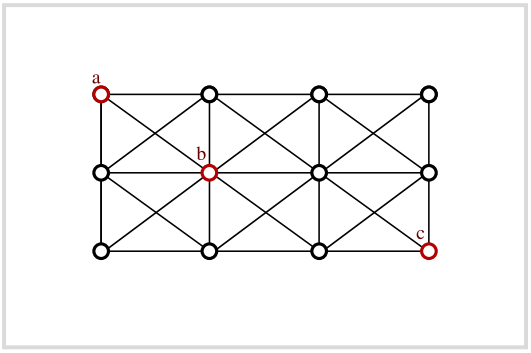


Figura 5.8: Àrees projectades en verd i tres captures en vermell de les, en aquest exemple, dotze realitzades per les càmeres.

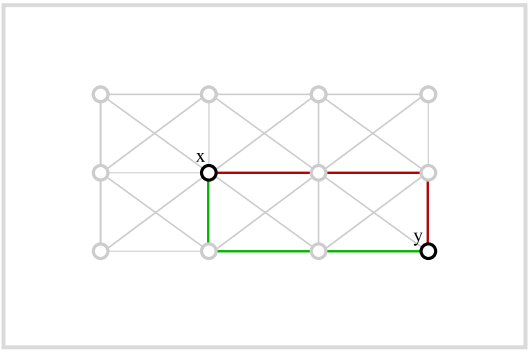
Quan s'ha construït aquest graf (figura 5.9(a)), i tenint en compte que el CHG és connex, és possible calcular l'homografia entre dos vèrtexs qualsevol composant una cadena d'homografies que estigui en algun dels camins (vegeu la figura 5.9(b)). En teoria, un cicle qualsevol al CHG hauria de ser l'homografia identitat, atès que es parteix d'un vèrtex x i s'arriba al propi vèrtex

[3] És indiferent obtenir els punts P_β al sistema de coordenades d'a, que obtenir els P_α en b.

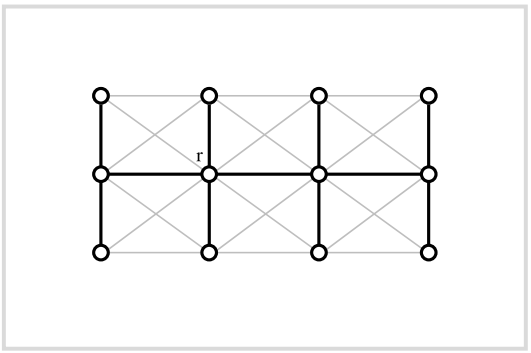
x. Quan es dóna la propietat que per a qualsevol cicle del graf l'homografia retornada és la identitat, llavors es diu que tenim un *CHG consistent*. El problema és que en el cas d'aquest projecte, degut a petits errors en la captura i el processament d'homografies, això no és cert; tenim un *CHG inconsistent*. Això vol dir que l'homografia calculada entre dos vèrtexs depèn del camí escollit.



(a) Construcció del graf CHG.



(b) Exemple de camins entre dos vèrtexs x i y qualsevol.



(c) Arbre expandit a partir del vèrtex r.

Figura 5.9: Diagrama de la construcció d'un *camera homography tree* a partir d'un CHG.

Per definició, un *arbre* és un graf sense cicles. Conseqüentment, si el CHG obtingut és un arbre, aquest serà sempre consistent, atès que no hi haurà dos camins diferents entre dos vèrtexs.

L'objectiu és, doncs, construir un arbre a partir del graf inicial, que minimitzi la distància entre un vèrtex qualsevol i l'arrel de l'arbre, i que minimitzi la distància entre dos vèrtexs qualsevol[4].

Un algorisme concret que ho aconsegueix i que, a més a més, assoleix un nivell de precisió de *subpixel* està descrit a [CSWL02].

[4]Minimitzant la distància entre vèrtexs es minimitza la propagació de l'error.

CAPÍTOL 6

Xarxa

El projecte consta de dues parts ben diferenciades, cadascuna d'elles amb les seves característiques i requisits. La part de xarxa que s'implementa per al mòdul de calibratge és diferent de l'emprada a l'aplicació d'usuari, atès que persegueixen objectius diferents.

En aquest capítol s'explica amb detall els conceptes que s'han aplicat per a configurar la xarxa del sistema de calibratge, i només es presenta la idea que s'amaga darrere de la de l'aplicació d'usuari[1].

6.1 Xarxa física

Una de les principals metes que persegueix aquest projecte és dissenyar un sistema de realitat virtual *portable* i, per tant, és evident que disposar d'una xarxa física flexible és molt important.

Tenint en compte que es vol que tots els mòduls càmera-projector siguin iguals i independents, i pensant en aconseguir la màxima flexibilitat, es decideix que les comunicacions es faran mitjançant una xarxa sense fils. Això permet estalviar-se l'haver de muntar una instal·lació més complexa, per a la qual s'hagi de passar un cablejat, i on es depengui, probablement, d'altres dispositius (*switches*, *routers*, etc.).

Habitualment, muntar una xarxa sense fils també inclou la utilització de maquinari addicional, com *routers wi-fi* o punts d'accés. Ara bé, degut a què no es vol dependre de més maquinari del que ja tenim, es decideix muntar una xarxa *ad hoc*; és a dir, un dels mòduls és designat com l'encarregat d'oferir accés a la xarxa sense fils a la resta, i es configura el seu ordinador convenientment.

Per veure com habilitar una xarxa *ad hoc* i fer que la resta d'ordinadors s'hi connectin, vegeu l'annex C.

[1] Podeu veure amb més detall com s'ha implementat la xarxa de l'aplicació d'usuari a [PL].

6.2 Sistema de calibratge

El sistema de calibratge és un procés lent, el qual requereix una bona sincronització entre tots els mòduls i on hi ha un flux continu d'informació molt variada entre ells. Els principals objectius que té són:

- Sincronització de mòduls per al calibratge geomètric (un projecta patrons; la resta els captura) segons convingui.
- Possibilitat d'enviar i rebre molta i molt variada informació.
- Control d'errors (si un mòdul no pot dur a terme la seva feina, cal detectar-ho).

El punt més important és el primer: és necessari trobar una solució que permeti designar un mòdul com a "proveïdor" de patrons i faci que la resta de mòduls (i, de fet, ell mateix) els "consumeixin", capturant-los i processant-los per tal de generar les corresponents homografies.

6.3 Aplicació d'usuari

L'aplicació d'usuari té uns objectius molt diferents al sistema de calibratge. Es tracta d'una aplicació 3D en temps real i, per tant, el temps emprat en qualsevol comunicació de xarxa és crític.

La informació que flueix entre els mòduls és:

- Posició de l'usuari respecte la pantalla (resultat del *head tracking*), per tal de sincronitzar els mòduls i visualitzar correctament l'escena.
- Geometria de l'escena 3D.

De manera anàloga al cas anterior, l'aplicació d'usuari tindrà un mòdul que actuarà com a director, el qual s'encarregarà d'organitzar tot el flux d'informació[2]. La seva feina serà passar la posició de l'usuari el més ràpid possible a la resta de mòduls per a què s'actualitzin i tots dibuixin l'escena de forma coherent.

Caldrà utilitzar un protocol de comunicacions en temps real anomenat UDP (*User Datagram Protocol*, Protocol de datagrames d'usuari). Aquest tipus de protocol funciona perquè la informació de què es disposa s'actualitza molt ràpidament i no importa si algun mòdul en un moment determinat no rep un paquet, ja que se sap que en un breu instant de temps se sobreescriurà amb noves dades.

6.4 Patró *master - worker*

Per tal de construir la xarxa a nivell d'aplicació per a la part de calibratge s'utilitza el patró *master - worker*, atès que s'adequa perfectament bé a les nostres necessitats.

Aquest patró consta de dues entitats: un *Master* o "cap" i un o més *Workers* o "treballadors". El *master* és qui inicia el procés de còmput i genera un conjunt de tasques que s'han de dur a terme, organitzant-les en un espai comú. Quan un treballador es connecta, demana alguna tasca per processar, fa els càlculs pertinents i retorna el resultat al cap.

Un dels principals avantatges d'aquest patró és que l'algorisme balanceja automàticament la càrrega del sistema. Això és degut a què la feina està compartida i, mentre n'hi hagi, els treballadors l'aniran processant.

[2] De fet, és totalment lògic i coherent seguir aquesta arquitectura tenint en compte que només un d'ells tindrà associat el *Wiimote*® i, per tant, coneixerà la ubicació de l'usuari.

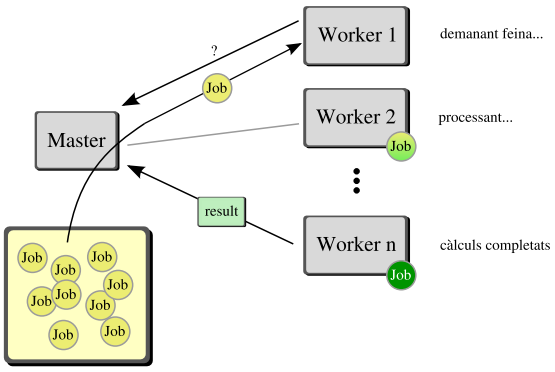


Figura 6.1: Esquema del funcionament del patró *master - worker*.

6.4.1 Afegint-hi el patró *callback*

El patró *master - worker* funciona bé quan són els treballadors qui van a buscar la feina; de fet, el *Master* no sap quants en té ni necessita saber-ho. Per al calibratge, en canvi, és un requisit indispensable que en sigui conscient, atès que els ha de sincronitzar entre ells. El que caldrà implementar, doncs, és un patró lleugerament modificat on els rols s'inverteixin: el client farà una connexió inicial al servidor per a registrar-se com a treballador i, tot seguit, es posarà en *mode passiu*; el servidor serà qui, de forma activa, donarà tasques als treballadors. A mesura que el servidor vagi tenint tasques pendents de processar, les enviarà als treballadors que estiguin desocupats (*call [the client] back*).

Així doncs, implementar el mòdul de calibratge aplicant aquesta tècnica permet resoldre els problemes que plantejava la xarxa: serà possible sincronitzar els projectors, ja que de forma selectiva decidirem qui projecta i qui captura, i es podrà fer que processin en paral·lel les homografies quan vulguem.

6.5 Implementació del patró

Java disposa de diferents mecanismes per tal de comunicar ordinadors remots. Tenint en compte que la implementació del mòdul de calibratge s'ha realitzat íntegrament en aquest llenguatge, s'ha utilitzat l'RMI (*Remote Method Invocation*, Invocació de mètodes remots) per a la xarxa, ja que forma part de l'estàndard de Java i permet construir molt fàcilment una aplicació distribuïda quan els dos extrems a comunicar són en Java[3].

6.5.1 RMI

Mitjançant l'RMI, un programa Java pot exportar un objecte i fer-lo accessible des de la xarxa. En general, qualsevol aplicació RMI es composa de dues parts:

- Un *servidor*, el qual crea uns quants objectes remots, els fa accessibles a la xarxa i espera que el client els invoqui.
- Un *client*, que obté la referència als objectes i els invoca.

[3]Si necessitem comunicar extrems codificats amb tecnologies diferents, haurem d'utilitzar altres protocols, com ara CORBA (*Common Object Request Broker Architecture*, Arquitectura comú d'intermediaris per a peticions a objectes) o SOAP (*Simple Object Access Protocol*, Protocol d'accés a objectes simple).

Per tal de distribuir objectes a la xarxa utilitzant aquesta tecnologia, cal definir una interfície per a l'objecte i donar-ne una implementació. La interfície és com la signatura d'aquest objecte, i descriu quines operacions tenim disponibles. La implementació, en canvi, dóna una solució concreta a cadascun dels mètodes que descriu la interfície. Quan un client obté la referència a un objecte remot, només podrà invocar aquells mètodes descrits a la interfície.

Master - worker: interfícies i implementacions

Per a la implementació en RMI de les classes `Master` i `Worker` de la figura 6.2 serà necessari definir, en primera instància, un parell d'interfícies que duran els mateixos noms i que permetran que els objectes siguin accessibles des de la xarxa:

`Master` Només descriu dos mètodes, que serveixen per a què un *worker* es doni d'alta o de baixa al sistema: `register (T worker)` (el qual retorna l'identificador que se li ha atorgat al *worker*) i `unregister (T worker)`.

`Worker` També disposa d'únicament dos mètodes: `getId()`, el qual retorna l'identificador que se li ha atorgat, i `quit()`, el qual provoca que el treballador es doni de baixa del *master*.

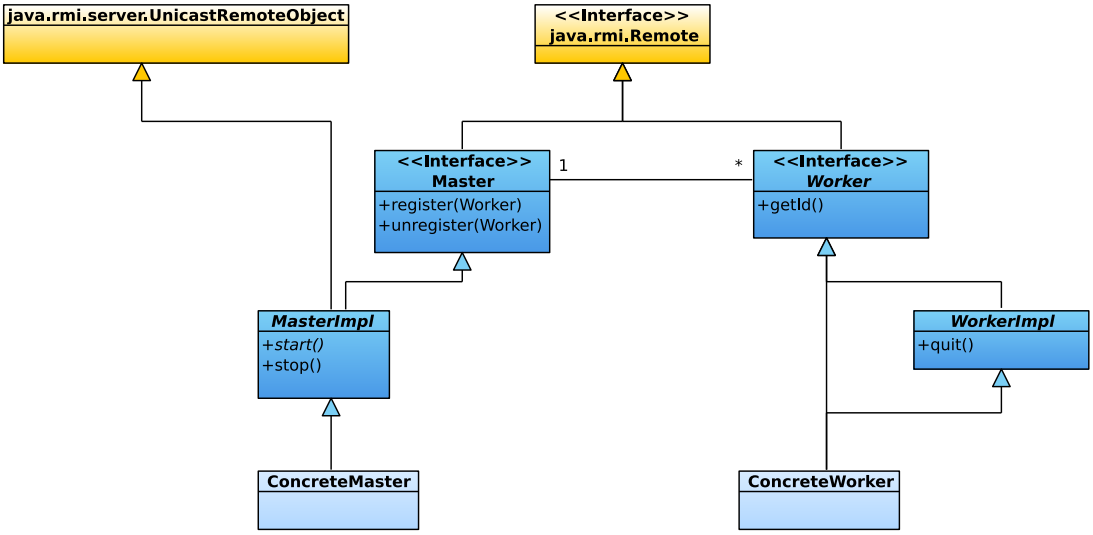


Figura 6.2: Diagrama del patró *master - worker* implementat en Java RMI.

Per tal d'estalviar feina a les classes que acabin implementant aquestes interfícies, i donant per fet que el comportament dels mètodes que hi ha definits ha de ser sempre el mateix, es defineixen sengles classes que donin unes implementacions per defecte:

`MasterImpl` Implementa els mètodes descrits a la interfície `Master`, donant suport a l'addició i eliminació de referències a treballadors.
La seva constructora inicialitza correctament l'objecte i el fa accessible a la xarxa.
A més a més, defineix un parell de mètodes addicionals molt útils:

- Un mètode abstracte d'arrencada (`start()`), el qual pretén oferir a les seves subclasses un punt d'entrada comú i personalitzable.

- Un mètode d'aturada (`stop()`), el qual desconnecta els *workers* que estan relacionats amb ell i elimina l'objecte de la xarxa.

`WorkerImpl` A més a més d'implementar correctament els mètodes descrits a la seva interfície, s'encarrega de registrar-se automàticament al *master* corresponent.

Tot plegat serveix per a disposar d'un codi que sigui re-utilitzable, genèric i que faciliti el disseny d'una aplicació distribuïda.

A l'apartat 7.3.4 es descriuen els passos que s'han seguit per tal d'integrar el patró al mòdul de calibratge.

CAPÍTOL 7

Part 1: calibratge

En aquest capítol es descriu de forma exhaustiva la primera part del PFC, que consisteix en el calibratge del sistema de realitat virtual, tot seguint els passos clàssics del desenvolupament en cascada d'una aplicació (anàlisi de requisits, especificació i disseny, implementació, integració i proves).

En primera instància, es determinen quins objectius dels descrits al capítol 1 cal acomplir en aquest punt del projecte. Tot seguit, caldrà fer un anàlisi acurat dels requisits funcionals i no funcionals que té aquesta part i que definiran, de forma no ambigua, les seves característiques.

Després, un cop s'hagi definit el marc de treball, es mostra l'especificació del sistema, el disseny concret que s'ha fet (aplicant el patró de xarxa descrit al capítol 6) i s'expliquen els detalls de la implementació que s'ha seguit.

Finalment, es comenten alguns dels tests que hem dut a terme per tal d'assegurar-nos de què tot funciona correctament.

7.1 Introducció i objectius

L'objectiu principal que s'ha d'acomplir en aquesta fase dels descrits a la introducció del projecte és el *calibratge automàtic*. En altres paraules, que donada una disposició qualsevol dels projectors envers una paret plana (i sempre que aquests dos tinguin una àrea en comú), es pugui calcular automàticament una pantalla el més gran possible al seu interior.

A més a més, i tenint en compte que, en general, interessa que aquesta pantalla estigui correctament alineada amb la percepció que té l'usuari de què és vertical i què és horitzontal, cal oferir-li una *eina que permeti modificar la geometria del quadrilàter que contindrà la pantalla* per tal de què, si ho creu convenient, l'alineï manualment[1].

Per tant, els dos punts clau són:

- Calibratge automàtic.
- Eina per a corregir la pantalla resultant.

[1] El sistema ignora què és horitzontal i què vertical i, per tant, no pot alinear-hi automàticament la pantalla.

També és important recordar que els resultats que obtinguem aquí s'han de passar cap a l'*aplicació d'usuari* descrita al capítol 9:

- Exportació dels resultats del calibratge cap a l'aplicació d'usuari.

7.2 Especificació

L'especificació d'un sistema qualsevol consisteix en descriure de la millor manera possible *què* és el que ha de fer. Mitjançant diverses tècniques, com per exemple l'anàlisi de requeriments, es pot obtenir una definició molt acurada de què s'espera del sistema, de forma que no hi hagi ambigüitats i s'hi pugui treballar en un espai ben acotat.

En aquesta secció parlarem, per una banda, de les restriccions que hi ha al voltant de la primera part del projecte i ens centrarem en els *casos d'ús* que hi trobem, així com en els *requisits*[2] que se'n desprenen. Per l'altra, també mostrarem el diagrama de classes que té el sistema de calibratge i comentarem els punts més importants que el composen.

7.2.1 Restriccions

Abans d'entrar en matèria i començar a descriure els casos d'ús del sistema, els requisits, etc, és molt important que es delimiti acuradament què ha de fer aquest mòdul, què no ha de fer, quins supòsits assumim, etc.

Supòsits

En primera instància, cal posar de manifest aquelles parts del sistema que s'assumeixen són certes i que, per tant, no ens han de preocupar:

- Un mòdul projector - càmera disposa de, com a mínim, una càmera.
- El sistema operatiu sobre el que s'executarà la nostra aplicació és *Microsoft Windows XP*.
- Els ordinadors disposen de xarxa sense fils, i es pot configurar com s'especifica a l'annex C.
- La paret sobre la qual es realitza el calibratge és completament plana i blanca.

Què no fa el sistema

Per tal de delimitar bé l'àmbit d'ús del sistema de calibratge, cal indicar quins són els seus límits, què no pot fer:

- El sistema no pot calibrar els projectors si aquests projecten a més d'una paret.
- El nombre màxim de mòduls projector - càmera que es pot enregistrar al sistema és de dos.
- El sistema només corregeix les intensitats dels projectors, però no pot "dissimular" el fons sobre el qual s'està projectant (de fet, ja hem dit que es projecta contra una paret blanca).

[2] El mètode de *Volere* ofereix una plantilla (fitxa) per tal de presentar els requisits que hem decidit utilitzar, de tal manera que se'n recopili la informació indispensable i esdevinguin unitats atòmiques perfectament definides.

- El sistema no calcula automàticament una pantalla que estigui alineada amb els eixos vertical i horitzontal del món real.

- El sistema no calcula la pantalla d'àrea més gran continguda a la zona projectada.

7.2.2 Casos d'ús

El primer que s'ha de fer per tal de determinar els casos d'ús que tindrà el mòdul de calibratge és definir la jerarquia d'usuaris del sistema, la qual ens la mostra la figura 7.1. Bàsicament, es tracta dels *stakeholders* identificats al punt 3.5, però correctament organitzats.

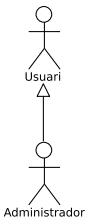
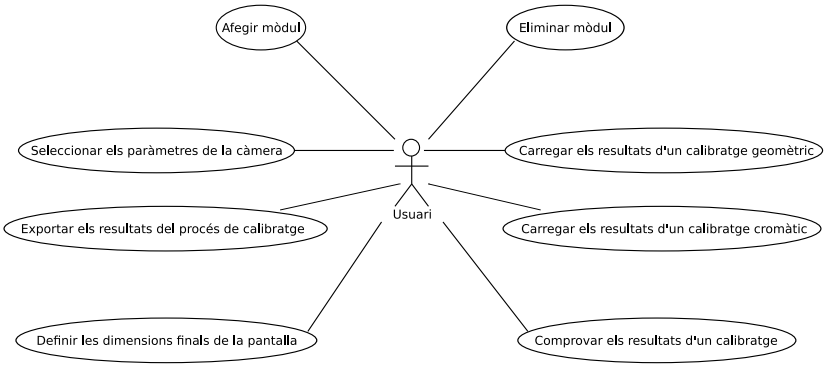
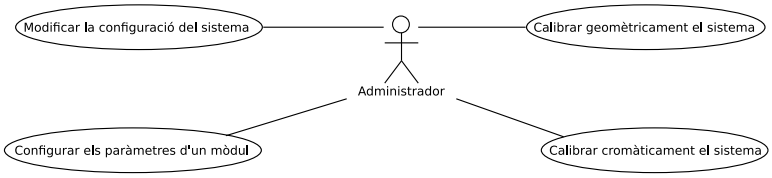


Figura 7.1: Jerarquia d'usuaris del mòdul de calibratge.

A continuació, mostrem els casos d'ús que pot dur a terme cada usuari:



(a) Casos d'ús de l'usuari "usuari".



(b) Casos d'ús de l'usuari "administrador".

Figura 7.2: Diagrama de casos d'ús.

Finalment, podem veure un llistat complet dels casos d'ús que caracteritzen aquest mòdul, descrivint-los de forma exhaustiva. Per cadascun d'ells tenim una breu descripció de què fa,

indiquem qui el duu a terme, especifiquem les precondicions[3] que tingui i un criteri de validació[4] i, finalment, mostrem la interacció entre l'usuari i el sistema.

1. Afegir mòdul	
Descripció	
El sistema ha de permetre enregistrar els mòduls físics (tàndems projector i càmera) que estiguin disponibles per al procés de calibratge.	
Actor principal	Usuari
Precondició	
-	
Criteri de validació	
El sistema manté una referència al mòdul indicat.	
Curs típic d'esdeveniments	
Usuari	**Sistema**
1. L'usuari decideix afegir un nou mòdul al sistema. 2. L'usuari posa en funcionament el mòdul en qüestió. 3. L'usuari indica que vol enregistrar el mòdul al sistema.	
	4. El sistema enregistra el mòdul.
Cursos alternatius	
4a. El sistema d'enregistrament no està disponible: 4a1. El mòdul avisa a l'usuari de què no s'ha pogut enregistrar.	

2. Eliminar mòdul	
Descripció	
El sistema ha de permetre donar de baixa mòduls que hi estiguin enregistrats, per tal d'evitar que s'utilitzin en el procés de calibratge.	
Actor principal	Usuari
Precondició	
El mòdul ha d'estar registrat al sistema.	
Criteri de validació	
El sistema no manté cap referència al mòdul indicat.	
Curs típic d'esdeveniments	
Usuari	**Sistema**
1. L'usuari decideix donar de baixa un mòdul del sistema. 2. L'usuari indica que vol desconnectar el mòdul en concret.	
	3. El sistema esborra la referència al mòdul. 4. El mòdul queda inactiu.
Cursos alternatius	

[3]Condicions que s'han de complir abans de què es dugui a terme el cas d'ús. Si alguna d'elles no fos certa, el cas d'ús no es podria realitzar.

[4]Quin és el resultat que s'espera obtenir després de l'execució d'aquest cas d'ús, cosa que ens permet determinar si ha funcionat correctament o no

3. Modificar la configuració del sistema	
Descripció	
El sistema ha de permetre modificar la configuració per defecte que s'empra en el procés de calibratge.	
Actor principal	Administrador
Precondició	
-	
Criteri de validació	
Els paràmetres de calibratge (vegeu l'annex A) que utilitzi el sistema són els especificats per l'administrador.	
Curs típic d'esdeveniments	
Usuari	**Sistema**
1. L'administrador decideix que s'han de canviar els paràmetres de configuració del sistema.	
2. L'administrador introdueix els nous paràmetres de configuració.	
	3. El sistema actualitza els paràmetres amb els valors indicats per l'administrador.
Cursos alternatius	

4. Seleccionar els paràmetres de la càmera	
Descripció	
L'usuari ha de poder seleccionar quina càmera del mòdul s'ha de poder utilitzar i amb quins paràmetres treballarà.	
Actor principal	Usuari
Precondició	
-	
Criteri de validació	
La càmera que s'emprarà en el procés de calibratge i els paràmetres amb els quals funcionarà són els que que ha especificat l'usuari.	
Curs típic d'esdeveniments	
Usuari	**Sistema**
1. L'usuari inicia el programari que controla un mòdul.	
	2. El sistema mostra informació de les càmeres disponibles a l'usuari.
3. L'usuari selecciona una de les càmeres disponibles.	
	4. El sistema registra la selecció de l'usuari.

(continua a la següent pàgina)

(ve de la pàgina anterior)

	5. El sistema mostra informació de les resolucions disponibles amb què pot treballar la càmera a l'usuari.
6. L'usuari selecciona una de les resolucions disponibles.	
	7. El sistema registra la selecció de l'usuari.
Cursos alternatius	
3a. L'usuari no selecciona cap càmera:	
3a1. El sistema atura el programari que controla el mòdul.	
6a. L'usuari no selecciona cap resolució:	
6a1. El sistema atura el programari que controla el mòdul.	

5. Carregar els resultats d'un calibratge geomètric

Descripció

L'usuari ha de poder recuperar els resultats d'un calibratge geomètric anterior per tal d'estalviar-se el procés d'alineament dels projectors quan aquests no veuen variada la seva disposició.

Actor principal	Usuari
Precondició	
El nombre de mòduls que hi ha registrats al sistema al moment de carregar els resultats és el mateix que el que hi havia quan es van desar.	
Criteri de validació	
El sistema disposa dels resultats de calibratge computats anteriorment i els mòduls estan llestos per treballar-hi.	
Curs típic d'esdeveniments	
Usuari	**Sistema**
1. L'usuari decideix carregar els resultats d'un calibratge geomètric anterior.	
2. L'usuari indica al sistema que vol fer-ho.	
	3. El sistema pregunta a l'usuari quins resultats vol.
4. L'usuari indica al sistema els paràmetres que ha de carregar.	
	5. El sistema carrega els resultats indicats.
Cursos alternatius	

6. Carregar els resultats d'un calibratge cromàtic

Descripció

L'usuari ha de poder recuperar els resultats d'un calibratge cromàtic anterior per tal d'estalviar-se el procés de correcció del color quan els projectors no veuen variada la seva disposició.

(continua a la següent pàgina)

(ve de la pàgina anterior)

Actor principal	Usuari
Precondició	
El nombre de mòduls que hi ha registrats al sistema al moment de carregar els resultats és el mateix que el que hi havia quan es van desar.	
Criteri de validació	
El sistema disposa dels resultats de calibratge computats anteriorment i els mòduls estan llestos per treballar-hi.	
Curs típic d'esdeveniments	
Usuari	**Sistema**
1. L'usuari decideix carregar els resultats d'un calibratge cromàtic anterior.	
2. L'usuari indica al sistema que vol fer-ho.	
	3. El sistema pregunta a l'usuari quins resultats vol.
4. L'usuari indica al sistema els paràmetres que ha de carregar.	
	5. El sistema carrega els resultats indicats.
Cursos alternatius	

7. Exportar els resultats del procés de calibratge

Descripció	
El subsistema de calibratge ha de poder comunicar-se amb l'aplicació d'usuari (veure capítol 9) i, per tant, tots els resultats que es computin han de ser accessibles per al segon.	
Actor principal	Usuari
Precondició	
S'han realitzat correctament els calibratges geomètric i cromàtic.	
Criteri de validació	
S'han desat els resultats que espera com a entrada l'aplicació d'usuari correctament, segons el format pactat[5].	
Curs típic d'esdeveniments	
Usuari	**Sistema**
1. L'usuari decideix que vol iniciar l'aplicació d'usuari.	
2. L'usuari indica al sistema que vol exportar els resultats del calibratge al format que necessita l'aplicació.	
	3. El sistema desa els resultats en el format pactat a la ubicació especificada a la seva configuració.
Cursos alternatius	

[5]La definició d'aquest format el podeu veure a l'apartat A.

8. Calibrar geomètricament el sistema

Descripció

El sistema ha de poder calibrar la geometria dels projectors, estan aquests ubicats físicament de forma arbitrària, per tal d'aconseguir calcular una pantalla virtual que hi estigui continguda.

Actor principal Administrador

Precondició

- Existeix una àrea de solapament entre les zones projectades per ambdós projectors.
- Hi ha algun mòdul registrat al sistema.

Criteri de validació

El sistema coneix la disposició dels projectors i ha calculat les homografies[6] necessàries per a poder dibuixar correctament a l'interior de la pantalla.

Curs típic d'esdeveniments

Usuari	Sistema
1. L'administrador decideix que vol corregir l'alineació dels projectors.	
2. L'administrador indica al sistema que iniciï el procés de calibratge geomètric.	
	3. El sistema inicia el procés de calibratge.
	4. El sistema computa els resultats.
	5. El sistema pregunta a l'usuari on vol desar els resultats que ha obtingut.
6. L'administrador indica la ubicació on vol que es desin aquests resultats.	
	7. El sistema desa els resultats.

Cursos alternatius

9. Calibrar cromàticament el sistema

Descripció

El sistema ha de corregir les intensitats dels projectors, per tal d'evitar que un projector brilli més que l'altre o que la zona de solapament d'ambdós projectors sigui perceptible.

Actor principal Administrador

Precondició

- El sistema disposa d'un calibratge geomètric vàlid per a la disposició actual dels projectors (computat o carregat).
- El nombre de mòduls registrats al sistema es corresponen a aquest calibratge geomètric.

Criteri de validació

(continua a la següent pàgina)

[6] En aquesta fase d'especificació del projecte no acostuma a ser bona idea parlar de solucions tecnològiques a un problema donat (com gestionar la correcció geomètrica dels projectors, en aquest cas). De totes maneres, tenint en compte que l'ús de les homografies és una imposició inicial del projecte i que, per tant, no es contempla cap altra solució, considerem vàlid mencionar-les.

(ve de la pàgina anterior)
El sistema corregeix la intensitat dels projectors, fent que resulti impossible percebre que la pantalla calculada estigui formada per més d'un projector.

Curs típic d'esdeveniments

Usuari	Sistema
1. L'administrador decideix que vol corregir les intensitats de llum que els projectors aporten a la pantalla calculada.	
2. L'administrador indica al sistema que iniciï el procés de calibratge cromàtic.	
	3. El sistema inicia el procés de calibratge.
	4. El sistema computa els resultats.
	5. El sistema pregunta a l'usuari on vol desar els resultats que ha obtingut.
6. L'administrador indica la ubicació on vol que es desin aquests resultats.	
	7. El sistema desa els resultats.

Cursos alternatius

---

10. Definir dimensions finals de la pantalla

Descripció

L'usuari ha de poder corregir la geometria de la pantalla que el sistema ha computat automàticament, per tal d'adaptar-la a les seves necessitats[7].

Actor principal Usuari

Precondició

- El sistema disposa d'un calibratge geomètric vàlid per a la disposició actual dels projectors (computat o carregat).
- El nombre de mòduls registrats al sistema es corresponen a aquest calibratge geomètric.

Criteri de validació

El sistema ha computat unes noves homografies per tal d'adaptar les imatges a la nova geometria indicada per l'usuari de la pantalla.

Curs típic d'esdeveniments

Usuari	Sistema
1. L'usuari decideix que vol canviar la geometria actual de la pantalla continguda a l'àrea de projecció dels projectors.	
2. L'usuari indica al sistema que vol canviar-la.	
	3. El sistema pregunta a l'usuari quines noves coordenades vol per a cadascun dels quatre punts que conformen la pantalla.
4. L'usuari introdueix les noves coordenades.	
	5. El sistema computa les noves homografies segons la informació que li ha donat l'usuari.

(continua a la següent pàgina)

[7] En general, aquesta correcció manual que aplica l'usuari vol dir alinear la pantalla als eixos vertical i horitzontal de la realitat

(ve de la pàgina anterior)

	6. El sistema pregunta a l'usuari on desar els resultats.
7. L'usuari indica la ubicació on vol que el sistema els desi.	
	8. El sistema desa els resultats a la ubicació explicitada.
Cursos alternatius	

---

11. Comprovar els resultats del calibratge

Descripció

El sistema ofereix una eina a l'usuari que li permeti comprovar quin és el resultat que s'obtindria si es dibuixés una imatge qualsevol a la pantalla que s'ha obtingut de l'alineament geomètric (i cromàtic, si s'hagués dut a terme també).

Actor principal Usuari

Precondició

- El sistema disposa d'un calibratge geomètric vàlid per a la disposició actual dels projectors (computat o carregat).
- El nombre de mòduls registrats al sistema es corresponen a aquest calibratge geomètric.

Criteri de validació

S'ha enviat a pintar a la pantalla computada pel sistema de calibratge la imatge que l'usuari vol per a comprovar els resultats obtinguts.

Curs típic d'esdeveniments

Usuari	Sistema
1. L'usuari vol comprovar quin resultat s'obté amb el calibratge que ha donat el sistema.	
2. L'usuari indica al sistema quina imatge vol utilitzar per a dur a terme el test.	
	3. El sistema carrega la imatge i la projecta convenientment per a què estigui continguda a la pantalla "virtual" computada.
Cursos alternatius	

---

12. Configurar els paràmetres d'un mòdul

Descripció

Hi ha casos en què pot ser necessari modificar els diferents paràmetres que controlen un mòdul (vegeu l'annex A) per a adaptar-los a les condicions de l'entorn de calibratge.

Actor principal Administrador

Precondició

-

Criteri de validació

(continua a la següent pàgina)

(ve de la pàgina anterior)
Els paràmetres del mòdul són els que ha introduït l'usuari.

Curs típic d'esdeveniments	
Usuari	**Sistema**
1. L'usuari s'adona de què s'han de modificar els paràmetres d'algun mòdul. 2. L'usuari explicita a quin mòdul vol canviar-los-hi. 3. L'usuari indica quins són els nous paràmetres.	
	4. El sistema comprova que els paràmetres siguin vàlids. 5. El sistema actualitza els paràmetres al mòdul.
Cursos alternatius	
4a. El sistema detecta que els paràmetres són invàlids i cancel·la l'actualització.	

7.2.3 Anàlisi de requisits

L'SRS (*Software Requirements Specification*, anàlisi de requisits) és una eina que permet posar de manifest d'una forma clara i concisa aquelles funcionalitats que el sistema ha d'oferir, així com aclarir aquelles restriccions a les que estigui subjecte.

El resultat d'aquest anàlisi no només emmarca perfectament l'àmbit de treball, sinó que també es converteix en una eina que farà possible comprovar de forma inequívoca si el projecte fa tot el que ha de fer.

És important comentar que l'SRS conté només requisits funcionals i no funcionals, però no esbossa cap tipus de solució subjecta a la tecnologia, essent totalment independent de la que finalment es proposi.

Per tal de fer l'anàlisi de requisits hem seguit la plantilla del mètode *Volere*[Vol], però convenientment adaptada per tal d'evitar generar una documentació excessivament extensa.

7.2.4 Requisits funcionals

A partir dels casos d'ús que hem vist a l'apartat 7.2.2, podem descriure de forma concisa quins són els requisits que tindrà el nostre PFC.

Un requisit funcional descriu de manera no ambigua i concisa alguna característica, "habilitat", funcionalitat que el sistema ha d'oferir o satisfer. El conjunt de requisits funcionals serveixen per a acotar i explicar de forma detallada *com* s'han de resoldre els casos d'ús, però mantenint-nos sempre al marge de solucions tecnològiques concretes.

Tot seguit, es mostra el llistat complet de requisits que té el sistema, seguint la plantilla del mètode *Volere*. Per tal d'organitzar millor la memòria i facilitar la lectura i comprensió dels requisits, aquests es presenten organitzats en subapartats, cadascun dels quals representa un petit conjunt dels casos d'ús definits anteriorment.

Requisits per als mòduls

En aquesta primera secció, descriurem aquells requisits que estan directament relacionats amb els mòduls projector - càmera que formen el nostre sistema, i que es deriven de casos d'ús com

el 1 o 2.

Requisit 1			
Tipus:	Funcional	**Casos d'ús associats:**	∼[8]
Descripció			
El sistema hauria de poder ser conscient dels mòduls dels que disposa.			
Justificació			
Per tal de realitzar el calibratge, el sistema ha de conéixer quins mòduls hi ha disponibles.			
Condició de satisfacció			
El sistema té constància de què un determinat mòdul està disponible.			
Satisfacció del client:	5	**Insatisfacció del client:**	5
Dependències:	4	**Conflictes:**	-

Requisit 2			
Tipus:	Funcional	**Casos d'ús associats:**	1
Descripció			
El sistema hauria de permetre afegir mòduls nous.			
Justificació			
Degut a què tenim un sistema altament modular i portable, és necessari que els mòduls que el composen es puguin donar d'alta en qualsevol moment i sense donar per fet que són quelcom que sempre està disponible.			
Condició de satisfacció			
El sistema té una referència cap al mòdul donat.			
Satisfacció del client:	5	**Insatisfacció del client:**	5
Dependències:	4, 1	**Conflictes:**	-

Requisit 3			
Tipus:	Funcional	**Casos d'ús associats:**	2
Descripció			
El sistema hauria de permetre eliminar mòduls nous.			
Justificació			
La naturalesa portable del sistema fa que sigui possible que, en un moment determinat, algun dels mòduls deixi d'estar disponible i, per tant, s'ha de poder donar de baixa del registre de mòduls disponible que es manté.			
Condició de satisfacció			
El sistema ja no té cap referència cap al mòdul donat.			
Satisfacció del client:	5	**Insatisfacció del client:**	5
Dependències:	4, 1	**Conflictes:**	-

[8] Aquells requisits que siguin necessaris per a pràcticament qualsevol cas d'ús, i per tal de no sobrecarregar la plantilla, duràn com a indicador de *cas d'ús associat* el símbol "∼".

Requisit 4			
Tipus:	Funcional	**Casos d'ús associats:**	∼
Descripció			
Els mòduls que té el registre haurien de poder identificar-se de forma única.			
Justificació			
[9]			
Condició de satisfacció			
Cada mòdul té un identificador únic al sistema, sense que hi hagi repeticions.			
Satisfacció del client:	3	**Insatisfacció del client:**	3
Dependències:	-	**Conflictes:**	-

Requisit 5			
Tipus:	Funcional	**Casos d'ús associats:**	12
Descripció			
El sistema hauria de permetre modificar els paràmetres[10] d'un mòdul.			
Justificació			
Hi ha casos en què, per tal de realitzar bé el calibratge, i degut a les condicions de l'entorn (com, per exemple, la quantitat de llum de la sala), el sistema ha de poder modificar les característiques de funcionament del mòdul.			
Condició de satisfacció			
Els paràmetres del mòdul són els que ha introduït l'usuari.			
Satisfacció del client:	5	**Insatisfacció del client:**	1
Dependències:	6	**Conflictes:**	-

Requisit 6			
Tipus:	Funcional	**Casos d'ús associats:**	12
Descripció			
El sistema hauria de poder validar que els paràmetres introduïts per l'usuari són vàlids.			
Justificació			
-			
Condició de satisfacció			
El sistema accepta els paràmetres especificats si aquests són vàlids (veure A per a saber què són paràmetres vàlids).			
Satisfacció del client:	3	**Insatisfacció del client:**	3
Dependències:	-	**Conflictes:**	-

[9] En cas de què la justificació sigui suficientment trivial, s'ometrà.
[10] Veure el cas d'ús 12 per a saber de quins paràmetres es tracta.

Requisits de configuració

Tot seguit, detallem aquells requisits que descriuen i delimiten la forma en què el nostre sistema es pot configurar[11], de tal manera que puguem adaptar-ne el funcionament a les necessitats específiques que tinguem.

Alguns requisits que podrien quedar implícits al 7, com per exemple el 8 o 9, s'han decidit explicitar, atès que considerem són prou importants com per a estar ben descrits.

Requisit 7			
Tipus:	Funcional	**Casos d'ús associats:**	3
Descripció			
El sistema hauria de permetre modificar els seus paràmetres de configuració.			
Justificació			
El sistema s'ha de poder adaptar a diferents condicions en què pot arribar a executar-se (sobretot, això afecta a les condicions d'il·luminació).			
Condició de satisfacció			
-			
Satisfacció del client:	5	**Insatisfacció del client:**	3
Dependències:	-	**Conflictes:**	-

Requisit 8			
Tipus:	Funcional	**Casos d'ús associats:**	3
Descripció			
L'usuari hauria de poder seleccionar quin tipus de patró s'emprarà per a realitzar el calibratge.			
Justificació			
Diferents patrons i, per tant, diferents maneres de detectar-los, poden modificar característiques com el temps de calibratge o adaptar-se millor a les condicions en què s'està executant el sistema.			
Condició de satisfacció			
El patró que s'empri per a calibrar el sistema serà el que ha escollit l'usuari.			
Satisfacció del client:	5	**Insatisfacció del client:**	3
Dependències:	-	**Conflictes:**	-

Requisit 9			
Tipus:	Funcional	**Casos d'ús associats:**	3
Descripció			
L'usuari hauria de poder modificar les característiques que defineixen un patró, per tal d'adaptar-lo i modificar-lo segons li convingui.			
Justificació			
El fet de poder modificar patrons dóna flexibilitat al sistema, atès que s'admet que es pugui canviar la forma en què aquest darrer interacciona amb ells (veure el capítol A).			
Condició de satisfacció			

(continua a la següent pàgina)

[11] Alguns d'aquests els trobem descrits a l'apartat de requisits associats al mòdul, com poden ser el 5 o 6

(ve de la pàgina anterior)
El patró que s'empri per a calibrar el sistema estarà definit segons hagi indicat l'usuari.

Satisfacció del client:	5	**Insatisfacció del client:**	1
Dependències:	8	**Conflictes:**	-

Requisit 10

Tipus:	Funcional	**Casos d'ús associats:**	4

Descripció
El sistema hauria de mostrar a l'usuari de quines càmeres disposa un mòdul determinat.
Justificació
-
Condició de satisfacció
-

Satisfacció del client:	3	**Insatisfacció del client:**	4
Dependències:	-	**Conflictes:**	-

Requisit 11

Tipus:	Funcional	**Casos d'ús associats:**	4

Descripció
El sistema hauria de mostrar a l'usuari les resolucions disponibles que té la càmera seleccionada.
Justificació
L'usuari ha de poder escollir amb quina resolució vol que el sistema realitzi els càlculs.
Condició de satisfacció
-

Satisfacció del client:	3	**Insatisfacció del client:**	4
Dependències:	12	**Conflictes:**	-

Requisit 12

Tipus:	Funcional	**Casos d'ús associats:**	4

Descripció
L'usuari hauria de poder seleccionar quina càmera cal utilitzar d'entre les que el sistema disposi.
Justificació
-
Condició de satisfacció
La càmera que s'emprarà és la que hagi seleccionat l'usuari.

Satisfacció del client:	5	**Insatisfacció del client:**	5
Dependències:	10	**Conflictes:**	-

Requisit 13			
Tipus:	Funcional	**Casos d'ús associats:**	4
Descripció			
L'usuari hauria de poder seleccionar les característiques amb què ha de treballar la càmera seleccionada per al procés de calibratge (resolució de la imatge i format en què es desarà).			
Justificació			
-			
Condició de satisfacció			
La càmera realitzarà les captures segons les opcions que l'usuari hagi seleccionat.			
Satisfacció del client:	5	**Insatisfacció del client:**	2
Dependències:	12	**Conflictes:**	-

Requisits del procés de calibratge

La tasca principal d'aquest mòdul de calibratge és, precisament, alinear els projectors i corregir la intensitat de llum que aporten a la pantalla final. El que mostrem tot seguit són els requisits que ha d'acomplir el mòdul per tal de què aquest procés de calibratge acompleixin les característiques definides al capítol 1.3.

Els casos d'ús que es veuen afectats en aquest cas són el 8, el 9 i el 10.

Requisit 14			
Tipus:	Funcional	**Casos d'ús associats:**	8, 9
Descripció			
L'usuari ha de poder seleccionar la ubicació d'on carregar o on desar les dades que genera el sistema.			
Justificació			
-			
Condició de satisfacció			
El sistema demana a l'usuari que indiqui una ubicació d'on carregar o on desar els resultats i és la que empra per a fer-ho.			
Satisfacció del client:	5	**Insatisfacció del client:**	1
Dependències:	1	**Conflictes:**	-

Requisit 15			
Tipus:	Funcional	**Casos d'ús associats:**	8
Descripció			
El sistema hauria de poder dur a terme un calibratge geomètric dels projectors.			
Justificació			
-			
Condició de satisfacció			
El sistema ha computat les homografies corresponents a la disposició actual dels projectors.			
Satisfacció del client:	5	**Insatisfacció del client:**	5
Dependències:	1	**Conflictes:**	-

Requisit 16			
Tipus:	Funcional	**Casos d'ús associats:**	8
Descripció			
El sistema hauria de poder desar els resultats que s'han obtingut de l'execució d'un calibratge geomètric.			
Justificació			
En cas de disposar d'una configuració dels projectors que sigui equivalent a una que s'havia calibrat anteriorment, i per tal de poder carregar-ne els resultats sense haver de repetir el procés de calibratge, aquests haurien d'haver-se desat.			
Condició de satisfacció			
Els resultats de calibratge s'han desat i estan disponibles per a ser carregats més endavant.			
Satisfacció del client:	5	**Insatisfacció del client:**	2
Dependències:	15, 14	**Conflictes:**	-

Requisit 17			
Tipus:	Funcional	**Casos d'ús associats:**	9
Descripció			
El sistema hauria de poder corregir la intensitat de llum que aporten els projectors a la pantalla.			
Justificació			
Ens interessa que l'usuari no pugui percebre que la pantalla final està composada amb dos projectors i, per tant, s'han de poder igualar la intensitat de llum que aporta cadascun d'ells, així com evitar que la zona de solapament entre ells "brilli" més.			
Condició de satisfacció			
La quantitat de llum que hi ha a la zona projectada que pertany a la pantalla calculada al procés de calibratge és uniforme i, per tant, no es pot percebre que hi hagi més d'un projector.			
Satisfacció del client:	5	**Insatisfacció del client:**	4
Dependències:	1, 15	**Conflictes:**	-

Requisit 18			
Tipus:	Funcional	**Casos d'ús associats:**	9
Descripció			
El sistema hauria de poder desar els resultats que s'han obtingut de l'execució d'una correcció cromàtica d'intensitats.			
Justificació			
En cas de disposar d'una configuració dels projectors que sigui equivalent a una que s'havia calibrat anteriorment, i per tal de poder carregar-ne els resultats sense haver de repetir el procés de calibratge, aquests haurien d'haver-se desat.			
Condició de satisfacció			
Els resultats de la correcció cromàtica s'han desat i estan disponibles per a ser carregats més endavant.			

(continua a la següent pàgina)

(ve de la pàgina anterior)

Satisfacció del client:	5	**Insatisfacció del client:**	2
Dependències:	17, 14	**Conflictes:**	-

Requisit 19			
Tipus:	Funcional	**Casos d'ús associats:**	10
Descripció			
L'usuari hauria de poder modificar manualment les coordenades de les quatre cantonades de la pantalla final.			
Justificació			
El procés de calibratge automàtic no té manera de saber quins són els eixos vertical ("columna") i horitzontal ("terra") de la realitat, i a l'usuari li interessa que la pantalla hi estigui alineada.			
Condició de satisfacció			
L'usuari pot modificar les coordenades dels quatre punts que defineixen la pantalla final i el sistema ha computat homografies noves que adapten el calibratge geomètric a la nova geometria de la pantalla final.			
Satisfacció del client:	5	**Insatisfacció del client:**	3
Dependències:	15	**Conflictes:**	-

Requisit 20			
Tipus:	Funcional	**Casos d'ús associats:**	10
Descripció			
El sistema hauria de poder desar les coordenades de la pantalla final que l'usuari ha editat manualment[12].			
Justificació			
-			
Condició de satisfacció			
Les noves homografies que s'han computat per tal d'adaptar el calibratge geomètric a la nova geometria de la pantalla final s'han desat correctament.			
Satisfacció del client:	5	**Insatisfacció del client:**	3
Dependències:	19	**Conflictes:**	-

Càrrega de resultats

Som conscients de què el procés de calibratge és lent i costós i, per tant, és interessant que sempre que puguem ens l'estalviem. Per tant, resulta interessant tenir la opció de carregar resultats computats anteriorment i estalviar-nos haver-los de calcular novament. Requisits com el 16 o el 18 ens asseguren de què disposarem de resultats per carregar.

[12]De fet, modificar les coordenades de la pantalla final és dur a terme una part del calibratge geomètric i el resultat que se'n deriva és, de fet, el mateix que s'obté amb un calibratge geomètric. Per tant, i a efectes pràctics, podem considerar-los el mateix.

En aquest petit apartat, descrivim els requisits que, precisament, ens permeten recuperar els resultats que haguéssim desat anteriorment i treballar-hi.

Requisit 21			
Tipus:	Funcional	**Casos d'ús associats:**	5
Descripció			
El sistema hauria de poder utilitzar els resultats d'un calibratge geomètric anterior.			
Justificació			
-			
Condició de satisfacció			
-			
Satisfacció del client:	5	**Insatisfacció del client:**	2
Dependències:	16, 14	**Conflictes:**	-

Requisit 22			
Tipus:	Funcional	**Casos d'ús associats:**	6
Descripció			
El sistema hauria de poder utilitzar els resultats d'una correcció cromàtica anterior.			
Justificació			
-			
Condició de satisfacció			
-			
Satisfacció del client:	5	**Insatisfacció del client:**	2
Dependències:	18, 14	**Conflictes:**	-

Exportar resultats

Com ja sabem, el projecte consta de dues parts, i en aquest capítol només estem fixant-nos en la primera. Ara bé, ambdues parts estan fortament lligades, atès que els resultats que aquesta primera genera són l'entrada que necessita la segona per a funcionar. Per aquest motiu, tenim el cas d'ús 7 que descriu la necessitat d'exportar els resultats a un format que la segona part entengui.

En aquest punt, tractarem els requisits que d'ell se'n deriven:

Requisit 23			
Tipus:	Funcional	**Casos d'ús associats:**	7
Descripció			
El sistema ha de poder exportar els resultats de la part de calibratge per a què els pugui utilitzar l'aplicació d'usuari.			
Justificació			
-			
Condició de satisfacció			

(continua a la següent pàgina)

(ve de la pàgina anterior)
Els resultats s'han desat amb el format pactat (vegeu A) entre les dues parts.

Satisfacció del client:	5	**Insatisfacció del client:**	5
Dependències:	15, 17	**Conflictes:**	-

Requisit 24			
Tipus:	Funcional	**Casos d'ús associats:**	7

Descripció

L'usuari ha de poder seleccionar la ubicació on el sistema ha de desar els resultats exportats al format que requereix la segona part, per tal que aquesta darrera els pugui trobar.

Justificació

L'aplicació d'usuari disposa de la seva pròpia configuració i està desvinculada de la de la primera part. Per tant, cal que l'exportació de resultats es faci allà on l'aplicació d'usuari espera trobar-los.

Condició de satisfacció

-

Satisfacció del client:	2	**Insatisfacció del client:**	5
Dependències:	-	**Conflictes:**	-

Altres requisits

Finalment, descrivim els requisits relatius al cas d'ús que manca per definir, el 11.

Requisit 25			
Tipus:	Funcional	**Casos d'ús associats:**	11

Descripció

L'usuari hauria de poder veure quin és el resultat del calibratge dut a terme pel sistema, mitjançant el dibuixat d'una imatge donada.

Justificació

-

Condició de satisfacció

El sistema projecta una imatge aprofitant la infraestructura muntada (és a dir, aplicant, com a mínim, les homografies computades i corregint les intensitats dels projectors, si s'ha dut a terme la correcció cromàtica).

Satisfacció del client:	5	**Insatisfacció del client:**	2
Dependències:	8, 9	**Conflictes:**	-

Requisit 26			
Tipus:	Funcional	**Casos d'ús associats:**	11

Descripció

(continua a la següent pàgina)

(ve de la pàgina anterior)
L'usuari hauria de poder seleccionar quina imatge vol utilitzar per a comprovar el resultat del calibratge.

Justificació

-

Condició de satisfacció

La imatge que projecta el sistema és la que ha especificat l'usuari.

Satisfacció del client:	3	**Insatisfacció del client:**	1
Dependències:	-	**Conflictes:**	-

7.2.5 Requisits no funcionals

En aquest apartat es mostren els requisits no funcionals que caracteritzen el mòdul de calibratge. Aquest tipus de requisits serveixen per a imposar restriccions al disseny o la implementació del sistema, fent-se càrrec de temes com l'eficiència, la seguretat, etc.

Tenint en compte que les imposicions a les que està subjecte el projecte són reduïdes, no hi ha molts requisits d'aquest tipus:

Requisit 27

Tipus:	Plataforma	**Casos d'ús associats:**	-

Descripció

El sistema ha de funcionar en un entorn distribuït.

Justificació

Els mòduls de què disposem són, tots ells, ens independents. Per tant, i ja de partida, ens trobem en un entorn distribuït amb el qual el nostre sistema s'ha d'enfrontar.

Condició de satisfacció

-

Satisfacció del client:	5	**Insatisfacció del client:**	5
Dependències:	-	**Conflictes:**	-

Requisit 28

Tipus:	Plataforma	**Casos d'ús associats:**	-

Descripció

El sistema ha de funcionar amb una xarxa sense fils.

Justificació

-

Condició de satisfacció

-

Satisfacció del client:	5	**Insatisfacció del client:**	5
Dependències:	-	**Conflictes:**	-

Requisit 29			
Tipus:	Plataforma	**Casos d'ús associats:**	-
Descripció			
El sistema ha d'executar-se sobre el sistema operatiu Windows XP.			
Justificació			
-			
Condició de satisfacció			
-			
Satisfacció del client:	1	**Insatisfacció del client:**	1
Dependències:	-	**Conflictes:**	-

Requisit 30			
Tipus:	Portabilitat	**Casos d'ús associats:**	-
Descripció			
El sistema ha de poder executar-se en qualsevol entorn vàlid[13].			
Justificació			
-			
Condició de satisfacció			
-			
Satisfacció del client:	5	**Insatisfacció del client:**	5
Dependències:	-	**Conflictes:**	-

Requisit 31			
Tipus:	Manutenció	**Casos d'ús associats:**	-
Descripció			
El sistema ha de poder-se actualitzar amb facilitat i el seu manteniment ha de resultar prou senzill.			
Justificació			
-			
Condició de satisfacció			
Per tal d'afegir nous algorismes per a realitzar els càlculs no s'ha de tocar el codi existent.			
Satisfacció del client:	5	**Insatisfacció del client:**	1
Dependències:	-	**Conflictes:**	-

Requisit 32

(continua a la següent pàgina)

[13]Un entorn vàlid és aquell que disposa d'una paret blanca, plana, on s'hi pugui projectar còmodament. Les condicions d'il·luminació no són extremes, de tal manera que la càmera pot fotografiar correctament els patrons, sense saturar-se.

(ve de la pàgina anterior)

Tipus:	Docs.	**Casos d'ús associats:**	-
Descripció			
Tot el codi generat ha de tenir una bona documentació.			
Justificació			
-			
Condició de satisfacció			
Tots els mètodes i classes del codi han d'estar correctament documentats, explicant què fan i els seus paràmetres i atributs.			
Satisfacció del client:	3	**Insatisfacció del client:**	3
Dependències:	-	**Conflictes:**	-

7.2.6 Diagrama de classes

En aquest apartat es descriu el diagrama de classes que dóna suport a tot el sistema, mostrant els diferents conceptes que el composen i les relacions entre ells.

Una de les eines que hi ha per a representar-lo és l'UML (*Unified Modelling Language*, Llenguatge de modelatge unificat), el qual permet expressar aquests conceptes i relacions de forma visual (vegeu l'annex B).

Per tal de simplificar la comprensió del diagrama, es divideix en petites unitats i es descriuen una a una, comentant els punts més importants.

Quant a la notació

El sistema de calibratge està composat per moltes classes que són "eines de càlcul" i, per tant, la relació amb la resta d'entitats és atípica; hi estan relacionades, perquè les utilitzen, però només com a entrades i sortides del procés que duen a terme. Un exemple és un `Finder`, que utilitza una imatge prèviament filtrada per tal d'obtenir les coordenades dels punts que hi apareixen.

Tenint en compte que l'objectiu d'aquests diagrames és facilitar la comprensió del sistema i definir-lo de la millor manera possible, s'ha decidit adaptar la notació clàssica d'UML a les nostres necessitats i representar aquest tipus de relacions (mitjançant línies discontínues).

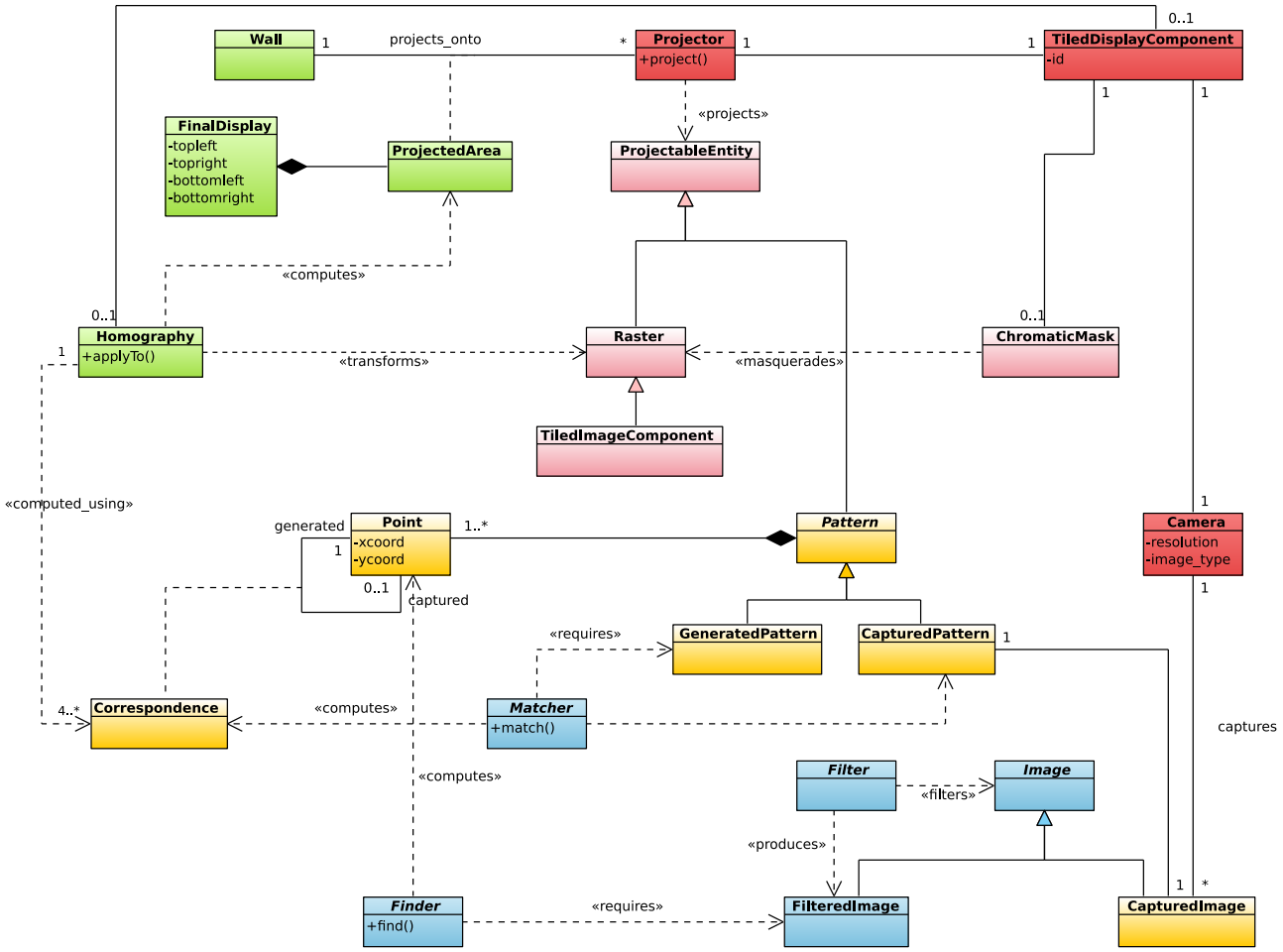


Figura 7.3: Diagrama de classes d'especificació del mòdul de calibratge.

Mòdul projector - càmera

La figura 7.4 mostra com es representen dins del sistema cadascun dels mòduls físics de què disposem realment. Això permet notificar al sistema de què un mòdul consta d'un projector i una càmera, que són les entitats que s'encarregaran de mostrar informació (patrons, imatges, etc.) i de capturar-la, respectivament, i treballar-hi còmodament.

Es tracta d'una part del sistema que no té cap complicació, però que val la pena comentar, perquè és una bona manera de començar a entendre la totalitat del diagrama.

Figura 7.4: Detall del diagrama de classes d'especificació: mòdul projector - càmera

Patrons

En aquest apartat es pot veure la zona del diagrama de classes relativa als patrons i la seva relació amb les captures que fa la càmera.

La imatge 7.5 mostra que un patró és una entitat que és la composició d'un conjunt de punts; és a dir, no té sentit parlar d'un patró si aquest no té punts associats. Hi ha un tipus especial de patrons, els `CapturedPattern`, que estan associats, a més a més, amb una imatge capturada.

Qualsevol instància d'un `CapturedPattern` apareix al sistema després de que la càmera hagi realitzat una captura; a partir d'aquesta, i aplicant els algorismes de visió per computador, es calculen les coordenades dels punts que s'hi veuen.

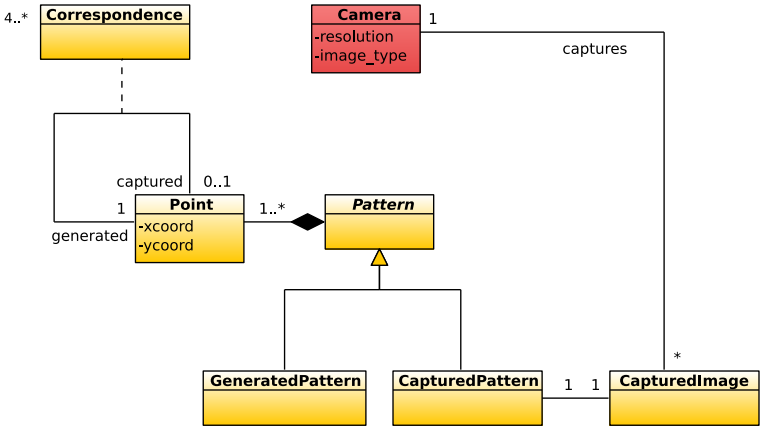


Figura 7.5: Detall del diagrama de classes d'especificació: patrons i captures

Un altre detall que s'hi aprecia és la classe `Correspondence`. Aquesta representa la relació que hi ha entre un punt capturat (calculat, doncs, a partir d'una `CapturedImage`) i un dels punts que hi havia descrits inicialment.

Noteu que un punt capturat sempre ha d'estar relacionat amb un punt generat (no hauria estat capturat si prèviament no hagués estat enviat), però no passa el mateix al revés; un punt generat no té perquè haver estat capturat (ja que pot haver-se projectat fora de l'abast de la càmera que ha realitzat la captura).

Les correspondències són molt importants per al mòdul de calibratge, atès que són l'entrada que requereix una homografia per a poder ser calculada.

Homografies

Les homografies són una part vital d'aquest mòdul ja que no només són el resultat que s'espera obtenir de l'alineament geomètric, sinó que s'utilitzen durant tot el procés de calibratge. Utilitzant-les es pot conèixer virtualment les àrees que queden cobertes pels diferents projectors i, així, realitzar diversos càlculs necessaris, com el còmput de la geometria de la pantalla final o de les màscares cromàtiques que caldrà aplicar.

Les homografies, tot i ser un ens ben senzill, estan relacionades amb moltes altres entitats del sistema:

- Poden, eventualment, estar relacionades amb un `TiledDisplayComponent`. Això permetrà que quan el segon vulgui projectar correctament el fragment d'imatge que li correspon, conegui l'homografia que ha d'aplicar.

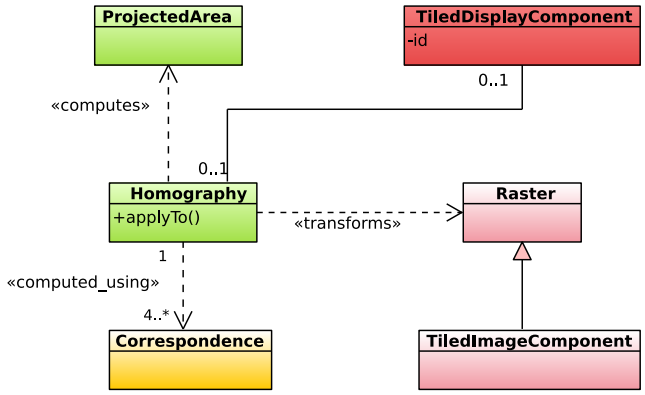


Figura 7.6: Detall del diagrama de classes d'especificació: les homografies

- Una homografia serveix per a calcular l'àrea que queda projectada per un `TiledDisplayComponent` determinat.
- Una homografia ha de poder modificar la geometria d'una imatge per a què al projectar-la "es vegi bé". Per aquest motiu, també està relacionada amb instàncies de `Raster`.

Màscares

Les màscares són l'entitat que permet dur a terme la correcció cromàtica. El que fan és modificar el color de la imatge que s'envia a projectar.

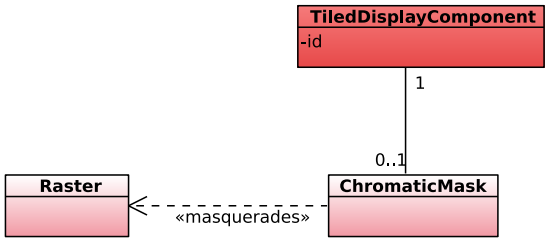


Figura 7.7: Detall del diagrama de classes d'especificació: màscares cromàtiques

Tal i com es pot veure, la màscara cromàtica s'aplica a qualsevol *raster* que es vulgui projectar. De fet, però, en general interessarà aplicar-la a una imatge que ja s'hagi deformat convenientment a l'aplicar-li una homografia.

Processos de càlcul

El mòdul de calibratge ha de dur a terme diferents càlculs per tal d'aconseguir computar les homografies, que són l'objectiu final (acompanyades de les respectives màscares). Per tal de fer-ho, i com podeu veure al capítol 5, es necessita:

Conjunt de punts Els quals s'obtenen a partir del patró generat i del patró capturat.

Correspondències entre ells Ja que, per a calcular una homografia, el que realment cal és disposar de correspondències entre els dos conjunts de punts anteriors (els capturats i els generats).

Així doncs, es té que, per una banda, cal un ens que a partir d'una imatge (i, per tant, de processos de *visió per computador*) sigui capaç d'obtenir el conjunt de punts capturats i, per l'altra, quelcom que pugui determinar les correspondències entre ambdós conjunts. La figura 7.12 mostra, precisament, la representació d'ambdós conceptes.

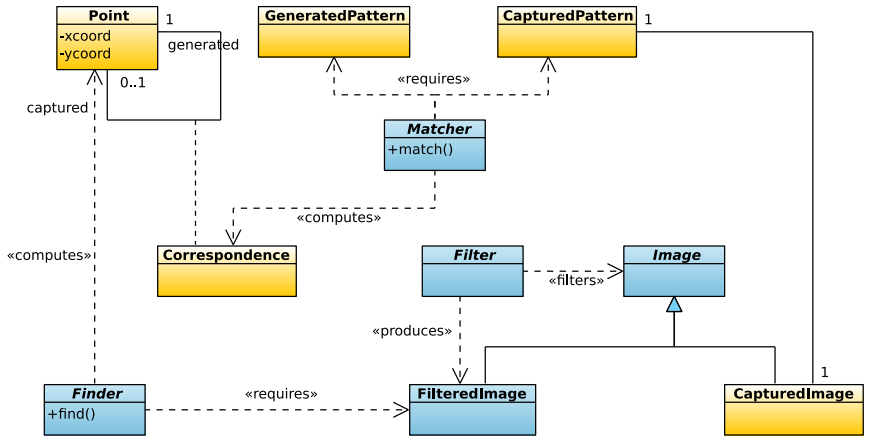


Figura 7.8: Detall del diagrama de classes d'especificació: processos de càlcul

7.3 Disseny

A l'apartat anterior s'ha especificat el sistema de calibratge, veient els requisits que ha d'acomplir i mostrant el diagrama de classes on es posa de manifest les diferents entitats que tenim i la relació entre elles. Tot això sempre des d'un punt de vista genèric, sense entrar mai en solucions tecnològiques concretes. S'ha vist, doncs, *què* ha de fer.

En aquest apartat, en canvi, ens centrarem en el seu disseny; és a dir, descriurem *com* es faran les coses. Així, veurem l'arquitectura que tindrà finalment el nostre sistema i comentarem diferents problemes que s'ha previst podrien aparèixer en el moment d'implementar la solució final i les solucions que s'han introduït al disseny per tal d'evitar-los.

7.3.1 Complicacions previstes

Per a poder calcular les homografies cal tenir una correspondència entre els punts d'un patró original i els d'un patró capturat. Com ja hem avançat a l'apartat 7.2.6, per tal d'obtenir els punts capturats és necessari aplicar tècniques de visió per computador i després haurem de calcular les correspondències. Com més complex és el patró concret que s'utilitza, més difícil és dur a terme ambdues tasques.

Per a solucionar aquest problema es presenta el concepte *component de patró*, el qual permetrà crear patrons tant complexos com es vulgui però, alhora, farà que treballar-hi sigui senzill. Així, es defineix un patró qualsevol com la composició de petits components [d'un patró], els quals seran petites entitats amb les que es pugui operar fàcilment, i mitjançant la combinació dels quals es podrà disposar d'un patró tant complex com es vulgui.

Exemple

Per tal d'entendre millor com utilitzar els *components de patró*, plantegem el següent exemple.

Es disposa d'un patró on hi ha tres línies horitzontals i tres de verticals, però els processos de visió per computador no poden diferenciar si són línies verticals o horitzontals. Per tal de poder utilitzar aquest patró amb els algorismes disponibles:

- Es defineix el patró amb dos *components*: el primer d'ells, només contindrà les línies horitzontals, i l'altre, les verticals.
- S'apliquen els processos de visió per computador a cadascun dels *components*, els quals són tant senzills com ens interessi. Gràcies a haver-ho separat, sabem que el primer component capturat conté les línies horitzontals i l'altre les verticals.
- Es combinen les dades obtingudes de cadascun dels *components* i es composa la informació original que realment interessa del patró (en aquest cas, els punts d'intersecció entre les línies).

Com podem veure, el fet d'haver separat el patró complex en patrons senzills permet operar-hi còmodament. Aquesta solució requereix la introducció d'alguns canvis al diagrama de classes, ja que els mòduls de càlcul (vegeu la secció 7.2.6) han de tenir en compte que ara els patrons són més complexos i que han de treballar amb el concepte *component d'un patró*.

7.3.2 Arquitectura del sistema

L'arquitectura del sistema és una descripció dels diferents mòduls que el formen i de les relacions entre ells. En aquesta secció es determinarà l'organització final del sistema de calibratge, tot aplicant *patrons de disseny*:

Patró de disseny és una solució genèrica a un problema conegut i comú al disseny de programari. No és quelcom traduïble directament a codi font, però sí que dóna una plantilla o descripció de com solucionar el problema en diferents situacions.

Arquitectura en tres capes

Una de les arquitectures més comuns a l'hora de desenvolupar una aplicació és l'*arquitectura en capes*, la qual agrupa els components del sistema en diferents *capes*, de tal manera que cadascun d'ells només pot comunicar-se amb altres components de la mateixa capa o les contigües.

És molt habitual trobar sistemes que segueixen aquesta arquitectura, amb un model de tres capes: la de *presentació*, on hi ha els components que interaccionen amb l'usuari; la de *domini*, on hi ha tota la lògica del sistema, encarregada de controlar la validesa de la informació que gestiona, ordenar l'execució d'accions, etc.; i la de *gestió de dades*, la qual procura que tota la informació que gestiona el nostre sistema i hagi de ser persistent ho sigui.

El sistema de calibratge que desenvolupem no és una "aplicació típica", sinó que es tracta d'un mòdul de càlcul, el qual té unes entrades i unes sortides clares. Tot i que ofereix certa interacció amb l'usuari (presentació), l'opció de desar i recuperar resultats (gestió de dades) i tota la lògica de procés (en certa manera, domini), s'ha decidit no complicar de forma innecessària el nostre sistema seguint aquesta arquitectura.

Arquitectura distribuïda

El sistema a desenvolupar ha de ser, forçosament, distribuït, atès que és un dels objectius i requisits que té. Per tant, des del punt de vista de com organitzar el sistema de forma distribuïda, sí que s'ha seguit una arquitectura concreta.

D'entre les diferents arquitectures distribuïdes que es poden aplicar, s'ha utilitzat la que ens ofereix el patró *master - worker*, descrit al capítol 6.

Tenint en compte que el calibratge dels projectors és un procés que ha d'estar correctament sincronitzat, ja que a cada instant de temps només pot estar enviant patrons un únic mòdul i la resta han d'estar capturant, és fàcil veure que el patró *master - worker* es una molt bona solució a les necessitats que té: un dels mòduls actuaria com a *master* i la resta (inclòs el propi *master*) com a *workers*.

7.3.3 Plataforma física

La plataforma física d'un sistema fa referència a les tecnologies, tant de programari com de maquinari[14], que utilitzem per a construir i fer funcionar l'aplicació.

Per tal de programar el mòdul de calibratge s'utilitza la *tecnologia Java*. Java és un llenguatge de programació orientat a objectes desenvolupat per *Sun Microsystems*, amb una sintaxi similar a *C* o *C++*. Algunes de les seves característiques són:

- Es tracta d'un *llenguatge interpretat*.
- És independent de la plataforma.
- És un llenguatge orientat a objectes.
- La gestió de memòria és automàtica (no disposa de punters).
- És una de les tecnologies més emprades actualment i, per tant, disposa de moltes "llibreries" amb molts propòsits diferents. Això vol dir que pràcticament qualsevol cosa que necessitem la podem arribar a trobar implementada ja.

Els motius per a escollir aquest llenguatge de programació és que és molt fàcil fer *debug* del codi, que disposa d'un enorme ventall de "llibreries" amb les que treballar (en concret, de visió per computador, de càlcul matemàtic, etc.), i que ja hi havíem treballat anteriorment i, per tant, el coneixem amb profunditat.

L'única contrapartida que té és que al tractar-se d'un llenguatge interpretat (funciona sobre una màquina virtual) és més lent executant-se que un llenguatge compilat de forma nativa (el qual sí que s'executa directament a l'ordinador). De totes maneres, aquest no és un problema crític per a nosaltres, ja que sabem que el procés de calibratge és lent i no ens importa el temps que tardem en fer-lo.

7.3.4 Diagrama de classes

Tot seguit, es mostra el diagrama de classes modificat, per tal d'incloure-hi tots els canvis proposats (adopció d'una determinada arquitectura per al sistema i resolució de problemes futurs).

[14] La construcció dels prototipus està descrita de forma exhaustiva al capítol 10.

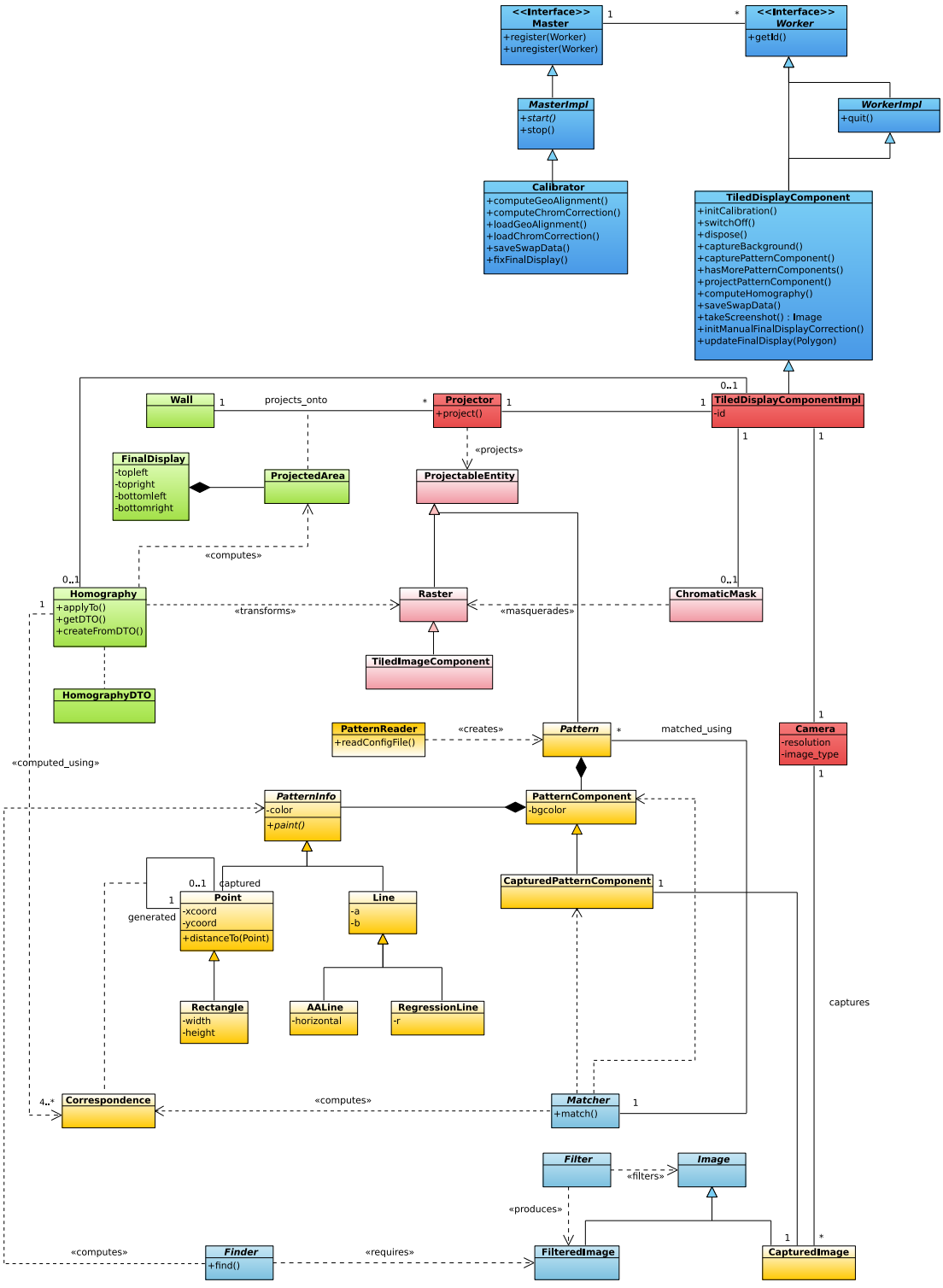


Figura 7.9: Diagrama de classes de disseny del mòdul de calibratge.

Integració del patró *master - worker*: arquitectura distribuïda

Per tal d'integrar el patró de xarxa al sistema el primer que s'ha de fer és definir una nova entitat, que serà la que actuarà com a *master*. D'aquesta classe hi haurà una, i només una, instància. Per tal de reduir l'acoblament entre un `TiledDisplayComponent` i el *master*, el primer implementa el *patró façana*, pel qual les operacions més comuns estan accessibles directament a través d'ell, sense que el segon hagi d'interaccionar amb altres objectes (com la càmera o el projector). A part dels avantatges que normalment aporta aquest patró, en aquest cas, en un entorn distribuït, és molt important utilitzar-lo, atès que es minimitza el nombre d'objectes remots que s'han de comunicar.

Un cop definides les classes `TiledDisplayComponent` i `Calibrator`, és el moment de definir la jerarquia d'herències que permetrà distribuir els objectes emprant la tecnologia RMI de Java. Tal i com s'ha avançat al capítol 6, per a fer-ho n'hi ha prou en què heretin de `Worker`/`WorkerImpl` i `MasterImpl`.

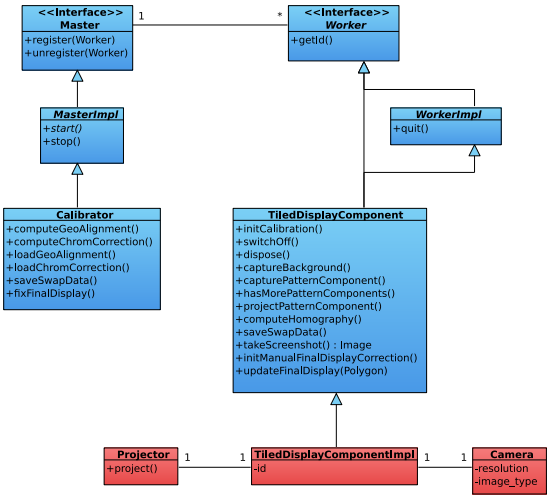


Figura 7.10: Detall del diagrama de classes de disseny: patró *master - worker*

Patrons i components de patró

Els patrons són un dels elements del model que més modificacions han rebut respecte els descrits a l'etapa d'especificació.

Ara, un `Pattern` és una composició de `PatternComponents`. Això solucionarà el problema descrit a l'apartat 7.3.1, ja que es treballa amb els components de patró (suposadament senzills), enlloc d'amb els patrons directament (en general, més complexos).

També es pot veure que ja no és el patró qui té la informació dels punts, sinó que això es desplaça al `PatternComponent`, que és a qui s'aplicaran els processos de visió per computador (vegeu l'apartat 7.3.4). De fet, no només s'ha canviat la relació entre els punts i el patró, sinó que s'ha ampliat el tipus d'informació que té un patró; no només són punts, sinó que també poden representar línies[15].

És fàcil veure que el resultat és una manera d'organitzar la informació que dóna la màxima flexibilitat per a representar patrons, sense sacrificar la facilitat en treballar-hi.

[15] Més endavant ja es traduiran com a punts, que és el que es necessita per a calcular homografies.

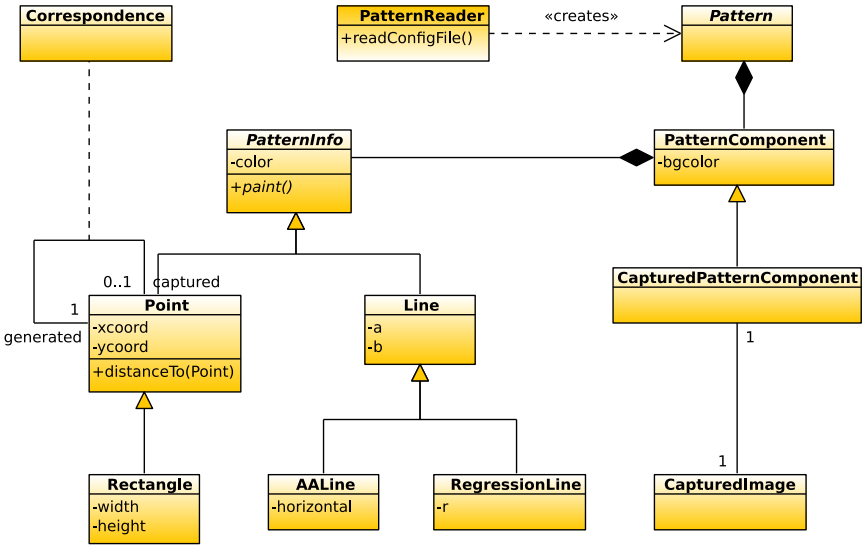


Figura 7.11: Detall del diagrama de classes de disseny: patrons i components de patró

Processos de càlcul

Degut a què s'ha canviat la definició dels patrons, és necessari canviar també la definició de les entitats que fan els processos de visió per computador i calculen les correspondències entre punts.

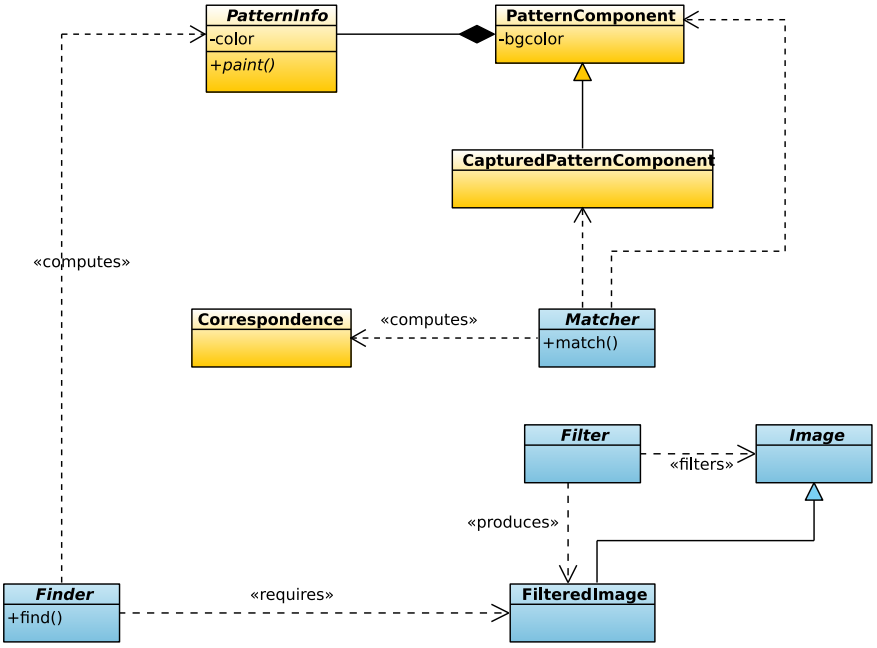


Figura 7.12: Detall del diagrama de classes de disseny: processos de càlcul

L'exemple més clar és el del tàndem `Finder/Matcher`. Un patró que projecti línies utilitzarà un `Finder` que, donada una captura, sàpiga trobar les línies, i el *Matcher* trobarà les correspondències entre les línies enviades i les projectades. Ara bé, el resultat d'aquest ha de ser quelcom útil per a calcular una homografia i, per tant, tot i que estigui preparat per a cercar correspondències entre línies, ha de retornar correspondències entre punts[16].

7.4 Implementació

La implementació del sistema consisteix en, bàsicament, la traducció de tot el que s'ha definit a la secció de disseny al llenguatge Java. Aquest procés de traducció no sempre és directe, i hi ha alguns detalls més complexos que val la pena comentar.

Els principals problemes que cal resoldre són la utilització real de la càmera, la codificació d'alguns dels mòduls de càlcul utilitzats (que dependrà de les "llibreries" emprades) i alguna millora en el mòdul de xarxa.

7.4.1 Utilització de la càmera

Per tal de poder utilitzar la càmera des de Java i disposar de la màxima resolució i qualitat que ofereix, s'ha utilitzat la llibreria *DirectShow*. Per a més informació sobre com s'ha integrat al projecte, vegeu [PL].

7.4.2 Processat en paral·lel

Tal qual s'ha definit l'arquitectura del sistema, es disposa d'un programa que coordina què fa cada mòdul en un moment determinat. Així, si es mira el pseudocodi descrit a l'apartat 7.5.2, es pot veure que el *master* selecciona un mòdul per a què projecti patrons i, de forma seqüencial, fa que la resta realitzin les captures i acabin computant les respectives homografies.

Després d'haver provat la implementació del patró s'ha detectat que a l'executar qualsevol dels mètodes que estan a un *worker* des del *master* no disposem de paral·lelisme; cal esperar a què acabi el processat a la màquina remota per tal de poder continuar amb la següent instrucció, fins i tot quan la nova sentència no depèn de l'anterior. Això vol dir que quan, per exemple, el *worker 1* està processant una homografia, cal esperar a què el mètode acabi abans de poder-li dir al *worker 2* que també ho faci, quan és evident que podrien estar realitzant els còmputs en paral·lel.

Per a solucionar-ho el que s'ha fet és invocar l'operació en un altre fil d'execució del *master* i controlar que aquest acabi. Atès que això caldrà fer-ho en més d'un mètode remot, s'ha decidit aplicar un patró de disseny més: *Adapter Pattern* o *Wrapper Pattern*.

L'*Adapter Pattern* el que fa és traduir la interfície d'una classe a una que sigui compatible amb allò que el client espera. Les crides que el client fa a la seva interfície es tradueixin a crides a la interfície original, tot afegint-hi una quantitat mínima de codi. Un exemple del patró és implementar el tipus abstracte de dades "Pila" a través d'una "Llista":

```
0  /* The client class should instantiate adapter objects */
   /* by using a reference of this type */
   interface Stack<T>
   {
     void push (T o);
5    T pop ();
     T top ();
```

[16] En un patró de línies, un cop es disposa de les correspondències entre aquestes es pot fer les interseccions entre les rectes al patró enviat i al capturat i disposar, així, de les correspondències entre punts.

```
}

/* DoubleLinkedList is the adaptee class */
class DList<T>
{
  public void insert (DNode pos, T o) { ... }
  public void remove (DNode pos) { ... }

  public void insertHead (T o) { ... }
  public void insertTail (T o) { ... }

  public T removeHead () { ... }
  public T removeTail () { ... }

  public T getHead () { ... }
  public T getTail () { ... }
}

/* Adapt DList class to Stack interface is the adapter class */
class DListImpStack<T> extends DList<T> implements Stack<T>
{
  public void push (T o) {
    insertTail (o);
  }

  public T pop () {
    return removeTail ();
  }

  public T top () {
    return getTail ();
  }
}
```

Codi 7.1: Exemple del patró *Adapter Pattern* amb una Pila i una Llista [WP].

Seguint aquesta idea, definim la classe `ThreadedTDC` com un *wrapper* a `TiledDisplayComponent`, el qual executa els mètodes que ens interessa en un fil d'execució diferent, aplicant, d'aquesta manera, el patró `Adapter Pattern`. A més a més, ofereix eines que permeten controlar si ja ha acabat l'execució de l'operació remota i coordinar, així, el funcionament del programa.

El processat en paral·lel d'instruccions acostuma a ser un tema delicat, on la sincronització entre cada fil d'execució esdevé un punt crític. Per a aquest projecte, però, tot el que s'executa en paral·lel són tasques atòmiques, amb la qual cosa el control de sincronisme que cal dur a terme es limita a controlar quan s'acaba el processat.

7.4.3 Configuració i personalització del sistema

Els requisits del sistema indiquen clarament que l'usuari ha de poder modificar diversos atributs del sistema, com per exemple el patró que s'emprarà per a dur a terme el calibratge. Hi ha diferents maneres d'introduir les preferències de l'usuari al sistema: interfícies gràfiques d'usuari, mitjançant fitxers de configuració, etc.

La solució escollida és la utilització de *fitxers de configuració* en format XML (*eXtensible Markup Language*, Llenguatge de marques extensible), degut a què tenim molts paràmetres que són susceptibles de ser modificats i dissenyar una interfície gràfica genèrica que permeti modificar-los representa un sobrecost innecessari.

En general, els processos de visió per computador requereixen certa participació de l'usuari per tal d'afinar millor les opcions que ofereixen els seus algorismes i aconseguir resultats més

acurats. Tenint en compte això, i sabent que el nostre mòdul de calibratge permet afegir *filters* o *finders* (vegeu els diagrames de classes de les pàgines dels capítols 7.2.6 o 7.3.4), els quals també poden ser configurats per l'usuari (cadascun d'ells amb les seves pròpies opcions), disposar d'una manera fàcil i còmoda d'introduir tota aquesta informació sense modificar el codi font és vital.

Format del fitxer de configuració

Els paràmetres que es poden modificar de l'aplicació són els següents:

Patró emprat El patró que s'emprarà per a dur a terme el calibratge.

Paràmetres dels *filters* Els filtres transformen les imatges capturades de patrons, aplicant-los-hi diferents tipus de modificacions. Cada filtre defineix els seus propis paràmetres a modificar.

Paràmetres de *finders* Els *finders* serveixen per a trobar un tipus de dada concret en una captura. És a dir, si s'ha enviat un patró de punts, s'encarregarà de buscar els punts a la captura; si és de línies, cercarà línies.

Algorisme *final-display* Classe que implementa l'algorisme per a cercar l'àrea de la pantalla final (vegeu 7.5.3).

Imatges Per tal de comprovar què fa el sistema, i amb quines captures treballa, permet desar-les en diferents estats del processat dels algorismes de visió per computador.

Altres paràmetres La resta de paràmetres que es poden modificar, per defecte, són els que tenim descrits a A.

El format general de qualsevol paràmetre a modificar consisteix en una tupla que conté *nom del paràmetre* i el *valor* que se li atorga.

Per tal de veure exemples de fitxers de configuració, vegeu l'annex A.

Definició i configuració de patrons

La manera de crear nous patrons i configurar-los és, també, mitjançant fitxers de tipus XML:

Matcher S'hi indica quina classe de tipus `Matcher` és la que serà capaç de relacionar els punts d'un patró amb els punts del patró capturat.

Components d'un patró Se'n poden posar tants com es vulgui.
Cadascun d'ells tindrà informació sobre punts o línies, que són els tipus d'informació bàsica amb què, segons el diagrama de disseny, pot treballar un patró.

Per veure un exemple comentat d'un patró, vegeu també l'annex A.

7.4.4 Gestió dels resultats

Finalment, el requeriments del sistema indiquen què ha de poder fer amb els resultats que genera. En concret, defineixen que aquests s'han de poder *desar* i *recuperar*, així com *exportar* al format que necessita l'aplicació d'usuari.

Desar i carregar resultats

S'ha decidit que la manera en què els resultats d'un calibratge es desarien i es carregarien sigui mitjançant *fitxers*, enlloc d'emprar, per exemple, sistemes gestors de bases de dades.

Per a fer-ho, s'aprofita una de les funcionalitats del llenguatge de programació emprat, Java, per tal de desar i recuperar informació: els *serializable objects*. Aquest tipus d'objectes s'especifiquen com a `serializable` i permeten ser *desats en* i *carregats d*'un fitxer binari al disc dur.

Exportar resultats per a l'aplicació d'usuari

Per a exportar els resultats del procés de calibratge també s'utilitzen fitxers. Ambdós sistemes (*calibratge* i *aplicació d'usuari*) tenen accés al sistema de fitxers del sistema operatiu i no requereixen cap mena de configuració addicional per treballar-hi, justificant, així, la utilització d'aquesta solució.

A diferència del cas anterior, el format dels fitxers d'intercanvi el definim nosaltres. Es tracta d'un conjunt de fitxers de text pla que contenen les diferents informacions que requereix l'aplicació d'usuari. Podeu veure el format concret que utilitzen a l'apartat A.3.

La ubicació d'aquests *fitxers d'intercanvi* ha de ser coneguda per ambdues aplicacions, atès que el que una desi a un lloc, l'altra haurà de poder-ho llegir. Per tal de coordinar-les, es defineix un nou fitxer de configuració que indica quines són aquestes ubicacions. Podeu veure la definició concreta d'aquest fitxer a l'annex A.

7.5 Implementació d'algorismes

En aquesta secció es mostren els pseudocodis dels principals algorismes de coordinació de què disposa el nostre sistema. Això permetrà comprendre millor com funcionen i com interaccionen els diferents components que hem introduït.

7.5.1 Visió per computador

La implementació dels algorismes de visió per computador (corresponents a les classes `Filter`, `Matcher` i `Finder`) estan comentats de forma exhaustiva al propi codi. De totes maneres, a la memòria del meu company [PL] trobareu explicacions més acurades quan s'escaigui.

La idea general de com interaccionen aquestes classes és la següent:

1. Donada una captura de pantalla, aquesta es passa com a paràmetre al `Finder`.
2. El `Finder` aplica els filtres que tingui configurats.
3. Quan es disposa d'una imatge correctament filtrada, cerca la informació que hi ha per tal de poder representar-la analíticament.
4. Finalment, el `Matcher` relaciona la informació del component de patró enviat i el capturat.

El `Finder` que més bé funciona és el `PolychromeRegressionLinesFinder`. Aquest aplica el filtre `BasicColorsFilter`, el qual funciona de la següent manera:

- Substrau el fons a la imatge capturada (per tal de tenir-ne una altra que sigui pràcticament negra i només contingui les línies).
- Aplica un filtre de binarització al resultat per tal de conèixer a quines zones de la imatge hi ha línies.

- A aquells *pixels* de la imatge binaritzada que són de color blanc, comprova quin és el color original que tenien. Per a determinar-lo, comprova quin dels tres components RGB és major, en quin grau ho és respecte la resta i si és superior a un cert llindar; si es compleixen totes aquestes condicions, determina que aquell és el color.

La imatge filtrada serveix al `Finder` per a aplicar un algorisme de regressió lineal als tres conjunts de punts que s'han obtingut: el conjunt de *pixels* vermells, el de *pixels* verds i el de *pixels* blaus.

Finalment, el `Matcher` cerca la relació entre les línies en funció del patró de component que les contenen i del seu color.

7.5.2 Obtenció d'homografies

L'algorisme d'obtenció d'homografies consta de dues fases ben diferenciades:

- Enviament i captura de patrons.
- Processat d'homografies.

En primera instància, s'encarrega d'anar iterant sobre tots els `Workers`, de tal manera que en cada moment només un d'ells sigui el que projecti i la resta (inclòs ell) els que capturin. Cada cop que es completa la projecció i captura de patrons per part d'un mòdul, tots els mòduls intenten calcular les respectives homografies que indicaran com passar les coordenades del mòdul projector a la seva càmera.

```
0  /* Itera sobre tots els Workers, per tal que tots ells projectin
   els seus PatternComponents i els altres els vagin capturan */
   funcio obteHomografies() retorna Graf<Homografia> {
      Graf<Homografia> homografies = nou Graf<Homografia>();
      pertot Worker w fer
5        homografies.afegeix(computaHomografiesPer(w));
      fi

      retorna homografies;
   }
10
   /* Quan s'ha decidit quin Worker projecta, se li fa projectar tots
   els seus PatternComponents i la resta de Workers (ell inclos) els
   capturen */
   funcio computaHomografiesPer(Worker w) retorna Llista<Homografia> {
15    mentre w.tePatternComponents() fer
         w.projectaPatternComponent();
         pertot Worker w2 fer
            w2.capturaPatternComponent();
         fi
20       w.seguentPatternComponent();
      fi

      pertot Worker w2 fer
         /* Aquest procediment s'executa en un fil apart, de tal manera
25       que tots els calculs es realitzin en paral.lel */
         w2.computaHomografia();
      fi

      Llista<Homografia> homografies = nova Llista<Homografia>();
30    pertot Worker w2 fer
         si w2.posibleComputarHomografia() llavors
            homografies.afegeix(w2.obteHomografia());
```

```
        fi
    fi

    retorna homografies;
}
```

Codi 7.2: Pseudocodi per a obtenir les homografies

7.5.3 Càlcul automàtic de la pantalla final

Quan el sistema disposa de totes les homografies, pot calcular en un mateix sistema de coordenades la geometria de l'àrea que queda projectada (vegeu 5), de tal manera que es és possible calcular un rectangle que hi estigui contingut. Aquest rectangle serà la pròpia geometria de la "pantalla final".

```
/* Calcula la geometria de la pantalla final */
funcio calculaPantalla (Area area_projectada) retorna Rectangle {
    Rectangle r = area_projectada.calculaCaixaEnglobant();
    mentre no area_projectada.conte(r) fer
        redueix(r);
    fi
    retorna ajusta(r, area_projectada);
}

/* Donat un rectangle contingut dins d'una area, el fa creixer tant com
pugui, pero sense que deixi d'estar-hi contingut */
funcio ajusta (Rectangle r, Area a) retorna Rectangle {
    Rectangle rok;
    fer {
        rok = r;
        augmenta(r);
    } mentre a.conte(r);

    Rectangle r1 = rok;
    creixVertical(r1, a);
    creixHoritzontal(r1, a);

    Rectangle r2 = rok;
    creixHoritzontal(r2, a);
    creixVertical(r2, a);

    si r1.area() > r2.area() llavors
        retorna r1;
    sino
        retorna r2;
}

funcio creixHoritzontal(Rectangle r, Area a) retorna Rectangle {
    Rectangle rok;
    fer {
        rok = r;
        fesAmple(r);
    } mentre a.conte(r);
    retorna rok;
}

funcio creixVertical(Rectangle r, Area a) retorna Rectangle {
    Rectangle rok;
    fer {
        rok = r;
        fesAlt(r);
```

```
    } mentre a.conte(r);
    retorna rok;
}
```

Codi 7.3: Pseudocodi per a calcular la geometria de la pantalla final

Correcció assistida de la pantalla final

El problema de fer-ho automàticament és que el rectangle calculat només es veu com a tal, i no com a un quadrilàter qualsevol, des de la càmera el sistema de coordenades de la qual hem utilitzat per a computar-lo. En altres paraules, si aquella càmera fa una captura quan s'està dibuixant el perímetre de la pantalla final, aquest estarà alineat amb els eixos vertical i horitzontal de la càmera, però l'usuari, molt probablement, vegi un "rectangle deformat".

A l'usuari li interessa que la pantalla sigui un rectangle "a la realitat"; és a dir, que estigui alineada horitzontalment amb el "terra" i verticalment amb una "columna". Per aquest motiu, se li ofereix una eina que li permet modificar les quatre cantonades de la pantalla de forma interactiva (vegeu l'apartat D.1.5).

7.5.4 Correcció cromàtica

Les regions de la pantalla que estan il·luminades per més d'un projector brillen molt més que la resta, essent clarament perceptibles per l'usuari. Per tal d'aconseguir que no ho siguin, cal baixar la contribució de llum que hi aporten els projectors.

A la figura 7.13 es poden veure diferents maneres d'aplicar la correcció d'intensitats; la primera fila mostra el projector de l'esquerra, la segona el de la dreta i la darrera el resultat que obtenim.

La subfigura 7.13(a) mostra el cas ideal; si el calibratge és perfecte, qualsevol de les tres aproximacions dóna un bon resultat. La resta d'imatges (apagant completament part dels projectors, deixant que contribueixin al cinquanta per cent o utilitzant un degradat), exemplifiquen els errors que es produirien a la intensitat final de tota la pantalla si el calibratge no ha sigut prou acurat.

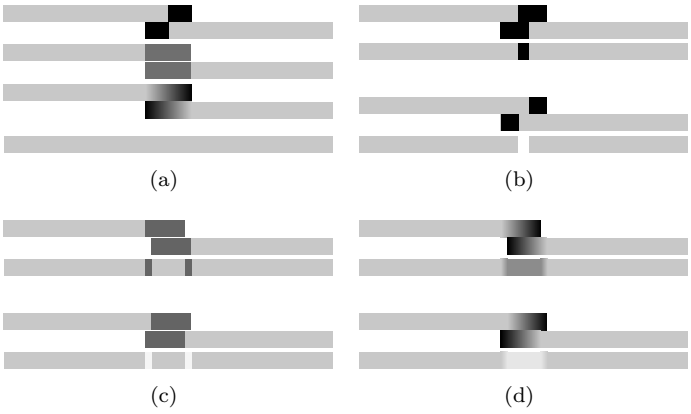


Figura 7.13: Efecte obtingut aplicant diferents correccions cromàtiques[BRR+02].

La solució de la figura 7.13(d), utilitzant degradats, és la millor. A diferència de la resta, no veurem clarament una franja on la intensitat de llum sigui molt superior o molt inferior

a la resta, sinó que només percebrem, a l'àrea de solapament, un petit augment o una petita disminució d'aquesta intensitat.

L'algorisme[BRR+02] que s'explica a continuació dóna com a resultat unes màscares amb aquests degradats a l'àrea de solapament.

Definim una màscara per a cada projector, la qual assigna un pes entre $[0.0, 1.0]$ a la intensitat de cada *pixel* que aquest projecta. Ja hem dit que un mateix punt pot ser il·luminat per més d'un projector; en qualsevol cas, donat un punt qualsevol, la intensitat total que aquest rep ha de sumar 1.0.

L'algorisme que calcula els degradats utilitza una estratègia per la qual els *pixels* que pertanyen fora de la pantalla final són zero, els que estan en una àrea de no solapament valen u, i la resta tenen valors propers a 1. En concret, per a trobar el pes $A_m(u)$ associat al *pixel* $u = (x, y, 1)$ del projector P_m, avaluem tots els *pixels* d'un projector en funció de la seva distància a l'aresta més propera (o, més ben dit, al *pixel* no visible més proper). Si tenim en compte que les homografies les hem calculat amb coordenades normalitzades de projector, de tal manera que x i y varien entre $[0, 1]$, llavors tenim que la distància d'un *pixel* a l'aresta més propera per a un projector P_i és

$$d_i(u) = w(x, y)min(x, y, 1 - x, 1 - y)$$

on $w(x, y)$ val 1 si $u, v \in [0, 1]$, o 0 altrament. Això simplifica el problema d'assignació de pesos a resoldre una funció *min*.

A més a més, basant-nos en les correspondències implícites de *pixels* al llarg dels projectors (via les homografies), podem assegurar que el pes dels *pixels* il·luminant una mateixa superfície suma la unitat. El pes $A_m(u)$ associat al *pixel* u al projector P_m s'avalua de la següent forma:

$$A_m(u) = \frac{d_m(u)}{\sum_i d_i(H_{ri}H_{rm}^{-1}u)}, i = 1, ..., n$$

Problemes amb la correcció cromàtica

Tal i com es comenta a l'apartat de tests d'aquest capítol, la correcció cromàtica, a la pràctica, no funciona correctament. En concret, el resultat que obtenim aplicant l'algorisme anteriorment descrit sempre dóna com a resultat una àrea de solapament clarament més fosca que la resta, com la que es veu al primer cas de la figura 7.13(d).

Per tal de solucionar-ho s'ha afegit un segon pas a l'algorisme de correcció cromàtica on es fa una captura amb les càmeres d'un fons blanc aplicant-hi les màscares calculades a la primera iteració. Això permet veure com la zona de solapament és, efectivament, més fosca, i permet calcular *com de més* n'és, de fosca, respecte la resta de pantalla. Un cop calculat aquest factor, s'utilitza per tal d'augmentar la intensitat de tots els *pixels*.

En cas de no voler dur a terme aquest segon pas, és possible introduir el valor del factor de correcció manualment a través del fitxer de configuració per tal d'evitar haver-lo de calcular.

7.6 Integració i proves

El darrer pas que s'ha de dur a terme per a donar per acabat el subsistema de calibratge és integrar tots els components desenvolupats i realitzar els testos corresponents. En aquesta fase s'unifica tot el treball d'enginyeria que s'ha realitzat per tal de determinar si la implementació que es dóna del sistema satisfà els requisits i les restriccions especificats a la secció 7.2.

El que s'ha fet ha estat, primerament, realitzar les *proves unitàries* dels diferents components clau que composen el nostre sistema, per tal d'assegurar que, de forma independent, funcionen

correctament. A continuació, es van integrant tots ells i es valida que el funcionament sigui l'esperat. Finalment, un cop es disposa de tot el sistema integrat, cal comprovar que els casos d'ús queden correctament satisfets i que es compleixen els requisits.

En aquest capítol es comenten els punts més importants de les proves unitàries i d'integració que s'han realitzat.

7.6.1 Proves unitàries

Les proves unitàries són un mètode emprat per a verificar que les diferents unitats de codi funcionen correctament. Una unitat és la part més petita de la qual se'n pot comprovar el funcionament.

Generació i projecció de patrons

S'ha comprovat que la classe encarregada de llegir un patró descrit en un fitxer XML vàlid genera un patró que s'adequa a la classe `Pattern`.

També s'ha verificat que la classe `Projector` mostra correctament els diferents `PatternComponent` que pot rebre.

Xarxa

La xarxa és un dels punts crítics del sistema. Un cop s'està segur de què els diferents mòduls estan connectats a la mateixa xarxa (vegeu la secció C.5) és necessari comprovar que la xarxa a nivell d'aplicació també funciona.

Les proves que s'han dut a terme han consistit en crear un *master* i enregistrar-hi fins a dos *workers*, per comprovar que el sistema de donar d'alta i de baixa mòduls funciona correctament.

També s'ha verificat que els objectes definits com a `java.io.Serializable` poden ser passats d'un extrem a l'altre de la xarxa.

Còmput de les homografies

L'objectiu final del mòdul és el còmput de les homografies; per tant, s'ha verificat que, donades com a mínim quatre parelles de correspondències de punts, el sistema calculava correctament la homografia resultant.

Gestió dels resultats

S'ha comprovat que totes les funcions de lectura i escriptura de dades funcionessin correctament. Per una banda, s'ha verificat que, efectivament, aquells objectes que havien de poder-se desar i recuperar aprofitant els objectes `Serializable` de Java ho són. Per altra banda, també s'ha comprovat que els fitxers que havien de seguir un format determinat i ubicar-se en un lloc concret seguien les especificacions establertes.

7.6.2 Proves d'integració

Les proves més importants i exhaustives que s'han realitzat són les d'integració.

El primer problema important amb el que ens hem enfrontat ha sigut el calibratge geomètric dels projectors, el qual depèn del correcte funcionament del mòdul de xarxa, de què els filtres funcionin correctament i la seva configuració sigui l'adequada, etc.

Degut a la enorme complexitat que implica aquest procés, el que hem fet ha sigut simplificar-ho al cas més senzill i, de forma iterativa, augmentar-ne la complexitat. Així, començàvem

comprovant que tot funcionés bé amb una càmera i un projector (estalviant-nos, doncs, el mòdul de xarxa), continuàvem afegint-hi el mòdul de xarxa, però treballant només amb una càmera i, finalment, executant el "cas real". Tot seguit, veurem amb més detall cada prova.

Per altra banda, un cop hem comprovat que l'alineació geomètrica era correcta, hem aprofitat per comprovar si la correcció cromàtica, a la pràctica, funcionava realment. Els tests duts a terme ens han servit per comprovar que, a l'hora de la veritat, la correcció cromàtica donava problemes i que havíem d'adaptar-la lleugerament.

En concret, la zona de solapament, tot i que en teoria hauria de quedar correctament il·luminada amb els "degradats" que s'obtenen a partir de les màscares, es veu clarament més fosca que la resta d'àrea de la pantalla[17].

Una càmera - un projector

En el primer cas d'alineació geomètrica, ha servit per comprovar que tots els mòduls (a excepció del de xarxa) funcionaven correctament. S'ha pogut verificar que les captures es realitzaven com s'esperava, que els filtres corregien bé les imatges i que els mòduls de còmput podien obtenir els punts de la captura i fer les correspondències entre aquests i els del patró original.

Una càmera - dos projectors

El següent pas era assegurar que la xarxa funcionava. Per a fer-ho, s'ha afegit un nou mòdul del qual se n'utilitzava només el projector.

En aquest cas, s'ha pogut comprovar que quan al projector remot se li ordenava qualsevol cosa (com per exemple pintar un nou `PatternComponent`), aquest l'acomplia correctament.

De totes maneres, en aquest punt, el procés de les homografies i, per tant, l'execució en paral·lel dels còmputs, encara no s'havia validat.

Dues càmeres - dos projectors

Finalment, només és necessari validar que tot el procés d'alineació geomètrica sigui vàlid. En aquest cas, els mòduls ja són com toca (una càmera i un projector per cadascun d'ells), i s'encarreguen de projectar i capturar tots ells (òbviament coordinats pel *Master*).

D'aquesta manera s'ha pogut verificar que la coordinació (qui captura i qui projecta) era vàlida i no hi havia col·lisions, així com que els processos de càlcul d'homografies es realitzen en paral·lel i no en sèrie.

[17] La solució aplicada la podeu veure a 7.5.4.

CAPÍTOL 8

Seguiment de l'usuari

En aquest capítol s'explica amb detall en què consisteix el *head tracking* i perquè utilitzar-lo representa una millora qualitativa molt gran pel que fa al nivell d'immersió que s'obté.

També es descriu la manera en què s'implementa i integra al nostre sistema.

8.1 Què és el *head tracking*?

La traducció literal de *head tracking* vol dir *seguiment del cap*. Es tracta d'una tècnica que permet a l'ordinador conèixer la posició del cap de l'usuari respecte a alguna posició concreta. En el nostre cas, ens interessa conèixer-la respecte la pantalla on es visualitza l'entorn 3D.

Disposar d'aquesta informació permet millorar la sensació d'immersió que el sistema ofereix a l'usuari, atès que allò que se li mostra és congruent amb la seva posició. Això vol dir que l'entorn virtual respon de manera natural als moviments que fa l'usuari, oferint-li informació addicional sobre la forma i dimensions d'un objecte tridimensional.

La figura 8.1 mostra la idea que hi ha darrere del seguiment de l'usuari.

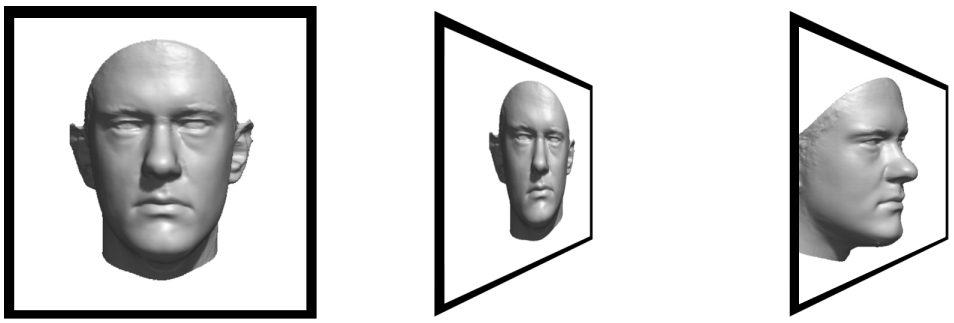
Si es parteix de la imatge 8.1(a), la qual se suposa s'està observant des del davant, i l'usuari es desplaça lleugerament cap a un costat, el que veurà quan no hi ha el *head tracking* activat és el que es veu a la imatge 8.1(b). En canvi, si s'activa, el sistema serà conscient de què l'usuari està mirant la pantalla des d'una nova perspectiva. La figura 8.1(c) mostra el perfil de la cara, doncs és el que l'usuari esperaria veure des de la seva nova ubicació.

8.2 Videoconsola *Wii*® (*Nintendo*®)

El comandament de la consola *Nintendo*® *Wii* (*Wiimote*®) consta, entre d'altres sensors, d'una càmera d'infrarrojos. Aquesta càmera té un angle d'obertura aproximada de 45º i una resolució de 1024×768 *pixels*. També consta d'un transmissor *Bluetooth* per a comunicar les dades.

La idea per a realitzar el *head tracking* amb el *Wiimote*®[1] és fer que controli en tot moment la posició de dos punts infrarrojos, els quals suposarem associats a la posició del cap de l'usuari. A

[1] Vegeu [CL].



(a) Imatge d'una cara en primer pla, vista des del davant.

(b) Imatge de la cara vista des del lateral, sense el *head tracking* activat.

(c) Imatge de la cara vista des del lateral, amb el *head tracking* habilitat.

Figura 8.1: Exemple de *head tracking* (imatges obtingudes de [BV99]).

partir d'una posició inicial donada, per la qual s'assumirà que l'usuari està mirant la pantalla des del centre, calcularem la posició relativa del cap respecte la pantalla quan aquests es desplacin.

Quan hom està jugant a la *Wii*®, el que fa és controlar el *Wiimote*® i se situen els LED (*Light Emisor Diode*, Díodes emissors de llum) infrarrojos sota la pantalla. En el nostre cas, però, la configuració serà inversa: el *Wiimote*® prop de la pantalla, enfocant cap a l'usuari, i en el cap d'aquest hi posarem dos LED infrarrojos.

8.3 Integració amb el projecte

Per tal d'integrar el *head tracking* al projecte cal resoldre, principalment, dos problemes:

- Comunicació entre l'aplicació d'usuari i el *Wiimote*®.
- Gestió i càlculs de la informació que ens proporciona el *Wiimote*®.

Per tal de comunicar el comandament i l'ordinador s'ha utilitzat la *llibreria WiimoteLib*, amb la qual ja s'han realitzat diversos projectes. Es tracta d'una *llibreria* molt, que fins i tot és capaç de gestionar més d'un comandament alhora.

Finalment, només cal veure com processar la informació que s'obté, cosa que s'explica al següent subapartat.

8.3.1 Càlculs

Per tal de fer el seguiment d'un objecte a l'espai (la posició del cap) no n'hi ha prou en disposar de només dos punts. Per a poder fer-ho d'aquesta manera s'han d'assumir una sèrie de simplificacions i s'ha de ser conscient de què el que s'obté és només una aproximació a la ubicació real.

L'objectiu d'aquest apartat és, doncs, trobar una fórmula per a obtenir la posició Hx, Hy, Hz, tenint com a dades la distància entre els dos LED i la imatge que rebem des del *Wiimote*®.

Sabent que estem en un espai com a la figura 8.2, centrat a la càmera del *Wiimote*®, i que aquesta ens dóna una imatge com la de la figura 8.3, definim WP_1 i WP_2 com els dos punts que veiem des del *Wiimote*® i $pointDist$ com la distància en el pla entre ells (vegeu la figura 8.4):

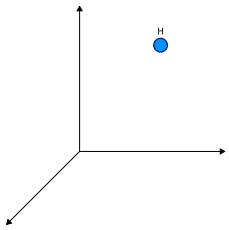


Figura 8.2: Posició del cap.

Figura 8.3: Imatge de la càmera del *Wiimote*®.

$$pointDist = \sqrt{(WP1_x - WP2_x)^2 + (WP1_y - WP2_y)^2}$$

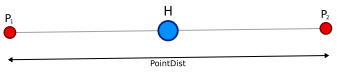


Figura 8.4: Perspectiva des de la càmera del *Wiimote*®.

Si tenim en compte que la càmera té una resolució horitzontal de 1024 *pixels* i una obertura de 45°, podem deduir l'angle que hi ha entre dos *pixels* consecutius tal i com es descriu a l'equació 8.1.

$$radiansPerPixel = \frac{\pi/4}{1024} \tag{8.1}$$

En primer lloc, calculem l'angle α del triangle rectangle que formen W, P_2 i H amb 8.2 (vegeu la figura 8.5). Recordem que estem suposant que l'usuari sempre mira cap a la pantalla i que, per tant, el triangle serà sempre rectangle.

$$\alpha = radiansPerPixel \cdot \frac{pointDist}{2} \tag{8.2}$$

A partir d'aquest angle α podem calcular la distància entre la càmera i H com:

$$H_z = \frac{dotDistance/2}{\tan(\alpha)/screenHeight}$$

Ara calculem les coordenades del punt mig entre WP_1 i WP_2.

$$avgX = \frac{WPx_1 + WPx_2}{2}$$

$$avgY = \frac{WPy_1 + WPy_2}{2}$$

Aquest punt $(avgX, avgY)$ ens serveix per a situar les coordenades (x, y) d'H. Calculant quants *pixels* en x està desplaçat $avgX$ del centre de la captura del *Wiimote*® (recordem

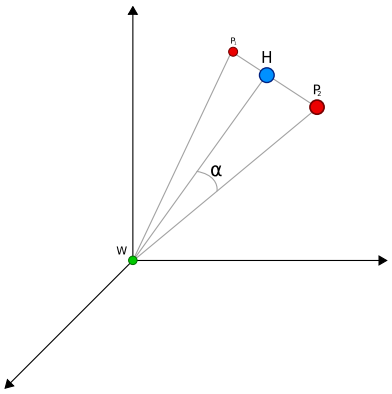


Figura 8.5: Angle α.

que la resolució és de 1024×768), i convertint aquest desplaçament en un angle mitjançant $radiansPerPixel$, podem aplicar una funció trigonomètrica simple (8.3) per trobar H_x.

$$H_x = \sin(radiansPerPixel * (avgX - 512)) * headDist \tag{8.3}$$

Per a obtenir H_y la idea que seguim és essencialment la mateixa, però amb alguna particularitat que cal tenir en compte.

La ubicació ideal del *Wiimote*® respecte la pantalla és just al centre, però no l'hi podem posar perquè l'interposaríem entre l'usuari i aquesta. Per a evitar-ho, l'haurem de col·locar per sobre o per sota de la pantalla, cosa que introduirà un cert error que caldrà corregir. L'aplicació d'usuari desenvolupada al capítol 9 mostra un cub virtual d'aresta 1. Tenint en compte que hem de situar la càmera virtual al centre de la pantalla, i que el *Wiimote*®, en realitat, està desplaçat mitja aresta amunt o mitja aresta avall de la pantalla, haurem d'aplicar un $factor$ que valgui 0.5 o -0.5 respectivament, per a fer la correcció.

A més a més, degut a la diferència d'altures que hi pot haver entre els usuaris, hem introduït un mecanisme per a establir la posició central de repòs; és a dir, l'usuari se situa a una posició natural davant la pantalla i indica al sistema que aquest és el seu centre, de tal manera que l'aplicació pugui guardar el desplaçament vertical *offset* observat. D'això se'n diu *zeroing y position*, i de l'angle *offset* que trobem en diem $cameraVerticaleAngle$:

$$relativeVerticalAngle = radiansPerPixel * (avgY - 384)$$

$$H_y = factor + \sin(relativeVerticalAngle + cameraVerticaleAngle) * headDist$$

CAPÍTOL 9

Part 2: aplicació final

Al capítol 7 s'ha explicat de forma exhaustiva el mòdul de calibratge del sistema, tot descrivint les fases clàssiques del desenvolupament de programari.

L'*aplicació final* ha estat principalment desenvolupada per Rodrigo Pizarro, i és per aquest motiu que la descripció més acurada d'aquest capítol és a la seva memòria [PL].

No obstant, tot seguit es mostren les principals tasques d'especificació de l'aplicació[1]. En concret, es defineixen els casos d'ús i es determinen els requisits (funcionals i no funcionals) que el caracteritzen.

9.1 Introducció i objectius

En primer lloc, és important recordar quins dels objectius descrits a la introducció són els que s'han d'assolir en aquesta part del projecte. De moment, el *calibratge automàtic* i, conseqüentment, la *portabilitat* del sistema, són objectius coberts al mòdul de calibratge.

El que ara es vol aconseguir, doncs, és una *aplicació* que demostri que la infraestructura que "s'ha acabat de calibrar" funciona correctament:

- Implementar una aplicació d'usuari.
- Dotar-la de *Head tracking*.

Tenint en compte que aquesta aplicació d'usuari ha d'estar executant-se sobre una plataforma concreta[2], i que la informació sobre com treballar-hi la proporciona el mòdul de calibratge, cal acomplir un objectiu addicional:

- Importació dels resultats del calibratge a l'aplicació d'usuari.

[1] Degut al meu perfil d'*enginyeria del software*.

[2] Aquesta "plataforma" és tota la infraestructura muntada, la qual ofereix una pantalla on dibuixar les imatges com la composició de dos projectors.

9.1.1 Adaptant una aplicació ja existent...

Abans de començar des de zero el desenvolupament de l'aplicació d'usuari es va dur a terme una procés de cerca per tal de trobar diferents alternatives de programes amb seguiment d'usuari. El resultat d'aquesta exploració és l'aplicació [CL].

Tenint en compte que aquest programa és una *demo* senzilla que implementa *head tracking*, es va decidir analitzar-ne el codi. Degut a la seva simplicitat i a què els requisits de maquinari addicional que requereix per disposar de seguiment de l'usuari (un *Wiimote*® i dos LED) no suposen un sobrecost molt important, es va decidir utilitzar-lo com a base de la qual partir, adaptant-lo convenientment.

9.2 Especificació

Com ja s'ha comentat a l'apartat 7.2, l'especificació d'un sistema descriu *què* fa, enlloc de *com* ho fa. L'aplicació d'usuari que es vol desenvolupar no serà des de zero, sinó que se n'adaptarà una de ja existent. Això vol dir que l'especificació serà lleugerament diferent a la que s'ha fet per al mòdul de calibratge.

Per una banda, veurem les restriccions que té l'aplicació que es vol obtenir i descriurem els casos d'ús i els requisits que ha de tenir per tal d'acomplir els objectius exposats. Per altra banda, també s'exposaran els casos d'ús que el sistema del qual es parteix tenia implementats, de tal manera que la informació que ofereixi aquesta especificació sigui el més acurada possible al programa que finalment s'obté.

9.2.1 Supòsits

- Es disposa d'un calibratge geomètric i cromàtic vàlid.
- La disposició dels mòduls projector-càmera coincideix amb el calibratge que tenim.
- Per tal de comunicar-nos amb els ordinadors remots, es té una llista de direccions IP vàlida, on cadascuna d'elles identifica un mòdul.
- La xarxa sense fils és pròpia i s'assumeix que no hi ha terceres persones connectades; no ens preocupa, doncs, la seguretat.
- L'ordinador que tindrà associat el comandament *Wiimote*® disposa d'una interfície *Bluetooth*® per a connectar-s'hi.
- Es disposa d'un *driver* que permet treballar amb el comandament *Wiimote*®.

9.2.2 Què no fa el sistema

En general, quan es delimita un projecte no només es diu *què fa*, sinó que també se sol indicar *què no fa*. En aquest cas en concret, però, només cal indicar clarament quines funcionalitats tenia l'aplicació de partida i quines se li afegeixen per tal d'assolir els objectius descrits anteriorment, assumint que farà *només* allò.

9.2.3 Casos d'ús

Els casos d'ús del sistema es divideixen en dues categories: aquells que ja estaven presents a l'aplicació [CL] i aquells que s'hi afegeixen per tal de què acompleixi els objectius.

El diagrama 9.1 mostra en blanc els casos d'ús originals i en gris els afegits.

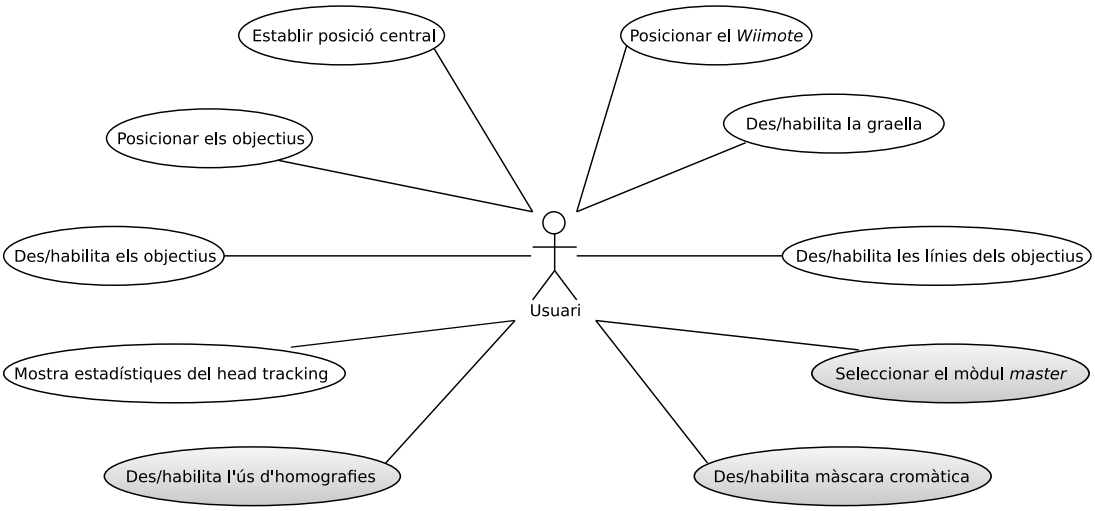


Figura 9.1: Diagrama de casos d'ús de l'*aplicació d'usuari*.

Casos d'ús presents a l'aplicació original

L'aplicació de partida ofereix unes quantes funcionalitats a l'usuari que val la pena comentar:

13. Establir posició central	
Descripció	
L'aplicació necessita conèixer quina és la posició inicial des de la qual l'usuari es mirarà la pantalla, la qual podríem considerar equivalent al "centre" d'una *cave*.	
Actor principal	Usuari
Precondició	
El *Wiimote*® veu els dos LED de l'usuari.	
Criteri de validació	
Quan l'usuari se situa a la posició senyalada com "central", l'aplicació mostra l'escena totalment centrada.	
Curs típic d'esdeveniments	
Usuari	**Sistema**
1. L'usuari decideix modificar la posició inicial des de la qual mira l'escena. 2. L'usuari se situa a algun lloc de la sala i indica al sistema que aquesta nova posició és la que s'ha de considerar com a inicial.	
	3. El sistema enregistra la nova posició com a posició central. 4. El sistema ubica la càmera virtual al centre, per tal de què la imatge mostrada estigui centrada.
Cursos alternatius	
-	

14. Posicionar Wiimote®	
Descripció	
Com hem avançat al capítol 8, per tal de calcular bé la posició de l'usuari, la càmera hauria d'estar situada al centre de la pantalla, però això el molestaria i no li deixaria veure les imatges. Per a solucionar-ho, el *Wiimote*® se situa físicament per sobre o per sota de la pantalla, amb la qual cosa caldrà corregir el desplaçament introduït.	
Actor principal	Usuari
Precondició	
-	
Criteri de validació	
El sistema coneix la ubicació real del *Wiimote*® respecte la pantalla.	
Curs típic d'esdeveniments	
Usuari	**Sistema**
1. L'usuari situa el *Wiimote*® per sobre o per sota de la pantalla. 2. L'usuari indica al sistema on l'ha posat.	
	3. El sistema enregistra aquesta posició, de tal manera que podrà aplicar les correccions pertinents.
Cursos alternatius	
-	

15. Posicionar objectius	
Descripció	
Els objectius que mostra l'aplicació poden ubicar-se aleatòriament a diferents posicions del món virtual.	
Actor principal	Usuari
Precondició	
-	
Criteri de validació	
Els objectius s'han redistribuït de forma aleatòria per l'espai.	
Curs típic d'esdeveniments	
Usuari	**Sistema**
1. L'usuari vol canviar la posició dels objectius. 2. L'usuari indica al sistema que vol que els objectius s'ubiquin a noves posicions de l'escenari.	
	3. El sistema redistribueix aleatòriament tots els objectius.
Cursos alternatius	
-	

16. Des/habilita graella	
Descripció	
La sala virtual que es dibuixa està representada amb "filferros". Les parets que la composen no són llises, sinó que poden (o no) mostrar una graella.	
Actor principal	Usuari
Precondició	
-	
Criteri de validació	
Es mostra o s'oculta la graella de les parets.	
Curs típic d'esdeveniments	
Usuari	**Sistema**
1. L'usuari vol amagar (si es veia) o mostrar (si no) la graella de les parets. 2. L'usuari indica al sistema que vol canviar l'estat de la graella.	
	3. El sistema mostra o amaga la graella de les parets.
Cursos alternatius	
-	

17. Des/habilita objectius	
Descripció	
La sala virtual pot o no contenir objectius, i l'usuari hauria de poder seleccionar si els vol veure o no.	
Actor principal	Usuari
Precondició	
-	
Criteri de validació	
Es mostren o s'oculten els objectius de la sala.	
Curs típic d'esdeveniments	
Usuari	**Sistema**
1. L'usuari vol amagar (si es veien) o mostrar (si no) les dianes de la sala. 2. L'usuari indica al sistema que vol canviar l'estat dels objectius.	
	3. El sistema oculta els objectius si es veien, o els mostra (calculant noves posicions) si no.
Cursos alternatius	
-	

18. Des/habilita les línies dels objectius

(continua a la següent pàgina)

(ve de la pàgina anterior)

Descripció	
Tots els objectius poden tenir una línia que va des del seu centre, on estan ubicats "flotant" a l'espai, fins a la paret del fons (uneix el punt central de l'objectiu amb aquest mateix punt projectat al pla de la paret del fons). L'usuari pot fer que aquestes línies es mostrin o s'ocultin.	
Actor principal	Usuari
Precondició	
-	
Criteri de validació	
Es mostren o s'oculten les línies dels objectius.	
Curs típic d'esdeveniments	
Usuari	**Sistema**
1. L'usuari vol amagar (si es veien) o mostrar (si no) les línies de les dianes. 2. L'usuari indica al sistema que vol canviar l'estat de les línies dels objectius.	
	3. El sistema oculta les línies dels objectius si es veien, o les mostra si no.
Cursos alternatius	
-	

19. Mostra/oculta estadístiques del head tracking

Descripció	
L'usuari pot estar interessat en conèixer què tal està funcionant el *head tracking* del sistema, així com algunes estadístiques internes de l'aplicació.	
Actor principal	Usuari
Precondició	
-	
Criteri de validació	
El sistema mostra algunes estadístiques (nombre de LED que s'estan veient, *frame rate* de l'aplicació, etc.) si no es veien, o les amaga si estaven visibles.	
Curs típic d'esdeveniments	
Usuari	**Sistema**
1. L'usuari vol amagar (si es veien) o mostrar (si no) les estadístiques del *head tracking*. 2. L'usuari indica al sistema que vol canviar l'estat de les estadístiques del *head tracking*.	
	3. El sistema oculta les estadístiques si es veien, o les mostra si no.
Cursos alternatius	
-	

Nous casos d'ús

Un dels objectius del projecte era aconseguir una aplicació d'usuari que utilitzés tota la infraestructura desenvolupada; en altres paraules, que "dibuixés" l'escena virtual a la pantalla computada durant els processos de calibratge. A més a més, també es volia que implementés algun mecanisme per a fer el seguiment de l'usuari.

Sabent que partim d'una aplicació que dibuixa una escena virtual i que, amb el *Wiimote*® fa *head tracking*, l'únic que cal fer és afegir-li els components necessaris per a què funcioni de forma "distribuïda", de tal manera que l'escena sigui pintada pels diferents projectors.

Tot seguit, es llisten els casos d'ús específics de l'aplicació modificada:

20. Seleccionar el mòdul master	
Descripció	
L'usuari hauria de poder determinar quin mòdul actua com a *master*, de tal manera que la resta siguin de tipus *slave*.	
Actor principal	Usuari
Precondició	
-	
Criteri de validació	
El mòdul seleccionat actua com a *master*.	
Curs típic d'esdeveniments	
Usuari	**Sistema**
1. L'usuari decideix quin mòdul actua com a *master*. 2. L'usuari associa a l'ordinador del mòdul el *Wiimote*® via *Bluetooth*. 3. L'usuari inicia l'aplicació.	
	4. El sistema detecta que té un *Wiimote*® associat. 5. El sistema configura el mòdul per a què treballi com a *master*.
Cursos alternatius	
-	

21. Des/habilita màscara cromàtica	
Descripció	
L'usuari hauria de poder aplicar les correccions cromàtiques que s'han calculat al mòdul de calibratge.	
Actor principal	Usuari
Precondició	
-	
Criteri de validació	
El mòdul càmera-projector aplica la màscara cromàtica si no s'estava emprant, o deixa d'utilitzar-la altrament.	
Curs típic d'esdeveniments	
Usuari	**Sistema**

(continua a la següent pàgina)

(ve de la pàgina anterior) 1. L'usuari decideix que vol aplicar (o deixar de fer-ho) les correccions cromàtiques en un mòdul determinat. 2. L'usuari indica al mòdul càmera-projector concret que apliqui (o deixi d'aplicar) les correccions cromàtiques.	
	3. El sistema utilitza (o deixa de fer-ho) la màscara cromàtica de què disposa aquell mòdul.
Cursos alternatius	
-	

22. Des/habilita l'ús d'homografies	
Descripció	
L'usuari hauria de poder aplicar les homografies que s'han calculat al mòdul de calibratge.	
Actor principal	Usuari
Precondició	
-	
Criteri de validació	
El mòdul càmera-projector aplica l'homografia si no s'estava emprant, o deixa d'utilitzar-la altrament, de tal manera que, en el primer cas, l'escena 3D es deformi convenientment per adaptar-se a la geometria de la pantalla.	
Curs típic d'esdeveniments	
Usuari	**Sistema**
1. L'usuari decideix que vol aplicar (o deixar de fer-ho) l'homografia en un mòdul determinat. 2. L'usuari indica al mòdul càmera-projector concret que apliqui (o deixi d'aplicar) la seva homografia.	
	3. El sistema utilitza (o deixa de fer-ho) l'homografia de què disposa aquell mòdul.
Cursos alternatius	
-	

9.2.4 Requisits funcionals

En aquest apartat es descriuen els requisits funcionals que es deriven dels casos d'ús que s'han afegit a l'aplicació original (és a dir, els descrits a l'apartat 9.2.3), així com els que naixen a partir de la pròpia definició dels objectius.

S'ha decidit no posar els requisits que s'obtindrien dels casos d'ús originals, perquè compliquen innecessàriament la memòria i suposen una feina addicional que no ajudarà a l'hora d'aplicar els canvis que siguin pertinents:

Requisit 33			
Tipus:	Funcional	**Casos d'ús associats:**	∼
Descripció			
El *master* ha de conèixer les IP de tots els mòduls *slave*.			
Justificació			
Per tal de passar-los-hi la ubicació de l'usuari (del qual n'està fent el seguiment amb el *Wiimote*®), necessita conèixer la seva direcció a la xarxa.			
Condició de satisfacció			
-			
Satisfacció del client:	5	**Insatisfacció del client:**	5
Dependències:	-	**Conflictes:**	-

Requisit 34			
Tipus:	Funcional	**Casos d'ús associats:**	20
Descripció			
L'usuari hauria de poder seleccionar quin mòdul actuarà com a *master*.			
Justificació			
Un dels mòduls ha de ser qui tingui associat el *Wiimote*® i actui com a *master*.			
Condició de satisfacció			
-			
Satisfacció del client:	5	**Insatisfacció del client:**	5
Dependències:	-	**Conflictes:**	-

Requisit 35			
Tipus:	Funcional	**Casos d'ús associats:**	20
Descripció			
Els mòduls que no actuin com a *master* s'han de configurar com a *slave* automàticament.			
Justificació			
Els mòduls *slave* no tenen un *Wiimote*® associat i, per tant, algú els hi ha de proporcionar la informació de la ubicació de l'usuari.			
Condició de satisfacció			
-			
Satisfacció del client:	5	**Insatisfacció del client:**	5
Dependències:	-	**Conflictes:**	-

Requisit 36			
Tipus:	Funcional	**Casos d'ús associats:**	∼
Descripció			

(continua a la següent pàgina)

(ve de la pàgina anterior)

El sistema hauria de poder carregar la informació que ha calculat el mòdul de calibratge.

Justificació

-

Condició de satisfacció

-

Satisfacció del client:	5	**Insatisfacció del client:**	5
Dependències:	23	**Conflictes:**	-

Requisit 37

Tipus:	Funcional	**Casos d'ús associats:**	21

Descripció

L'usuari hauria de poder aplicar una màscara cromàtica en un mòdul determinat.

Justificació

-

Condició de satisfacció

La imatge projectada pel mòdul concret té aplicada una màscara a sobre.

Satisfacció del client:	5	**Insatisfacció del client:**	5
Dependències:	-	**Conflictes:**	-

Requisit 38

Tipus:	Funcional	**Casos d'ús associats:**	21

Descripció

L'usuari hauria de poder desactivar l'ús d'una màscara cromàtica en un mòdul determinat.

Justificació

-

Condició de satisfacció

La imatge projectada per un dels mòduls no té aplicada cap correcció cromàtica.

Satisfacció del client:	5	**Insatisfacció del client:**	5
Dependències:	-	**Conflictes:**	-

Requisit 39

Tipus:	Funcional	**Casos d'ús associats:**	22

Descripció

L'usuari hauria de poder aplicar en un mòdul determinat la seva homografia.

Justificació

-

Condició de satisfacció

La imatge projectada pel mòdul es deforma convenientment, segons l'homografia.

(continua a la següent pàgina)

(ve de la pàgina anterior)

Satisfacció del client:	5	**Insatisfacció del client:**	5
Dependències:	-	**Conflictes:**	-

Requisit 40			
Tipus:	Funcional	**Casos d'ús associats:**	22
Descripció			
L'usuari hauria de poder desactivar l'ús de l'homografia en un mòdul determinat.			
Justificació			
-			
Condició de satisfacció			
La imatge projectada per un dels mòduls mostra tota l'escena virtual, sense aplicar-se-li cap homografia.			
Satisfacció del client:	5	**Insatisfacció del client:**	5
Dependències:	-	**Conflictes:**	-

9.2.5 Requisits no funcionals

Finalment, es mostren els requisits no funcionals que caracteritzen l'aplicació d'usuari.

De fet, tenint en compte que aquesta part del projecte només pretén demostrar la viabilitat de tot plegat, no disposa d'uns requisits no funcionals clars. De totes maneres, s'ha decidit que els canvis que s'apliquessin al programa original tinguessin algunes propietats que els fessin útils en un futur, a part d'heretar els que ja s'han vist a la part 1.

Requisit 41			
Tipus:	Plataforma	**Casos d'ús associats:**	-
Descripció			
El sistema ha de funcionar en un entorn distribuït.			
Justificació			
Els mòduls de què disposem són, tots ells, ens independents. Per tant, i ja de partida, ens trobem en un entorn distribuït amb el qual el nostre sistema s'ha d'enfrontar.			
Condició de satisfacció			
-			
Satisfacció del client:	5	**Insatisfacció del client:**	5
Dependències:	-	**Conflictes:**	-

Requisit 42

(continua a la següent pàgina)

(ve de la pàgina anterior)

Tipus:	Plataforma	**Casos d'ús associats:**	-
Descripció			
El sistema ha de poder estendre's fàcilment.			
Justificació			
Tot i que l'aplicació d'usuari només és una *demo*, és interessant que per a desenvolupar-ne una de nova es pugui aprofitar el màxim de codi possible, sobretot el pertanyen a la xarxa o a l'aplicació d'homografies i màscares.			
Condició de satisfacció			
-			
Satisfacció del client:	5	**Insatisfacció del client:**	1
Dependències:	-	**Conflictes:**	-

9.3 Utilització dels resultats del calibratge

En aquest apartat es descriu la idea que s'ha aplicat per tal d'utilitzar els resultats que s'han obtingut al mòdul de calibratge.

Quan l'aplicació d'usuari s'inicia, disposa de:

- Una homografia.
- Una màscara cromàtica.

9.4 Utilització d'una homografia

Ja hem vist que una homografia permet transformar "una imatge en una altra". L'aplicació d'usuari, però, té un model 3D, no una imatge. Tot i que seria possible dibuixar l'escena en un *frame* i aleshores aplicar-li la homografia, és molt millor fer-ho dins del procés de transformació del model tridimensional a imatge. El principal problema que es té és la sintaxi, perquè s'haurà de manipular dades tant en 3D com en 2D.

El resultat de multiplicar un punt del model per les matrius de modelat i projecció és un punt amb tres coordenades més l'homogènia. Per a fer la correcció geomètrica es volen coordenades 2D i, per tant, o bé es crea un vector 2D homogeni per poder-lo multiplicar per la matriu 3x3, o bé es manipula la matriu per convertir-la en una 4x4. Per raons de sintaxi en HLSL (*High Level Shader Language*, Llenguatge de *shading* propietat de Microsoft)[3], la segona opció és molt més fàcil d'implementar i també més eficient.

Per a aplicar una homografia en 2D (coordenades homogènies) es fa de la següent manera:

$$\begin{pmatrix} x_q \\ y_q \\ r \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x_p \\ y_p \\ 1 \end{pmatrix}$$

Per tenir una equivalència en 3D que conservi la coordenada z, es pot manipular l'homografia i afegir-li una dimensió. Només es vol modificar x, y, conservant la z. Afegint una columna a 0 a la tercera posició s'aconsegueix el primer objectiu, perquè s'anul·la la component z. Si

[3]Per a aplicar les transformacions d'una homografia s'han utilitzat *shaders*.

s'intercala una fila a la tercera posició, amb un 1 a la tercera columna i la resta zeros, s'obté la z sense modificar.

$$\begin{pmatrix} x_q \\ y_q \\ z_q \\ r \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & 0 & h_{13} \\ h_{21} & h_{22} & 0 & h_{23} \\ 0 & 0 & 1 & 0 \\ h_{31} & h_{32} & 0 & h_{33} \end{pmatrix} \begin{pmatrix} x_p \\ y_p \\ z_p \\ 1 \end{pmatrix}$$

Aquest resultat, però, no es pot passar tal qual al *pixel shader*, perquè abans d'arribar-li el *pipeline* agafa la component homogènia i divideix tot el vector per ella, de tal manera que no tindríem z_q, sinó z_q/r. L'únic que caldrà fer abans d'acabar és dividir x_q i y_q per r i posar la component homogènia a 1. Així doncs, el punt resultant és $(x_q/r,\ y_q/r,\ z_q,\ 1)$.

9.5 Utilització d'una màscara

Quant a la màscara, la seva funció és, per una banda, dissimular la zona de solapament entre ambdós projectors i, per l'altra, fer un retallat de la imatge (ja que, segurament, no tot el *frame buffer* del projector s'estigui utilitzant. La manera de fer-ho és ubicant un rectangle davant de tota l'escena, que "tapi" tot el *frame buffer* i al qual li aplicarem la màscara com a textura, utilitzant *alpha blending*.

CAPÍTOL 10

Prototipus

En aquest capítol es descriu breument com s'han construït cadascun dels mòduls que conformen el prototipus físic del projecte.

10.1 Introducció

Fins ara, hem vist que cadascun dels prototipus consta de les següents peces:

- Un ordinador portàtil.
- Un projector
- Una càmera.

Tenint en compte que el procés de calibratge ha de ser força acurat, interessa que tots els mòduls estiguin muntats sobre alguna plataforma que eviti que es puguin moure lliurement, i que doni certa llibertat a l'hora d'ubicar-los per la sala. A més a més, el projector i, eventualment, el portàtil, necessitaran connectar-se al corrent elèctric, i no es pot assumir que qualsevol sala disposarà de suficients endolls.

Tenint en compte aquestes consideracions, es decideix construir una plataforma que, si bé senzilla, disposi de, com a mínim:

- Un suport on recolzar el projector.
- Un suport on recolzar el portàtil.
- Un lladre que permeti endollar el projector, el portàtil i, eventualment, el lladre d'un altre mòdul.
- Un suport mòbil i extensible per a la càmera.

10.2 Construcció

La figura 10.1 mostra un dels prototipus construïts. Aquest consta d'una base on es recolza el projector feta de metacrilat, el qual és prou resistent com per a desplaçar-lo per la sala sense que es trenqui. Podem veure com la base suporta el lladre que ens permet realitzar les connexions, així com un "braç" d'alumini on s'ha collat la càmera. Sobre d'aquesta, en un segon nivell, s'ubica una segona planxa de metacrilat on es pot recolzar el portàtil. Ambdues planxes, unides mitjançant les barres metàl·liques, esdevenen un mòdul ben sòlid i segur.

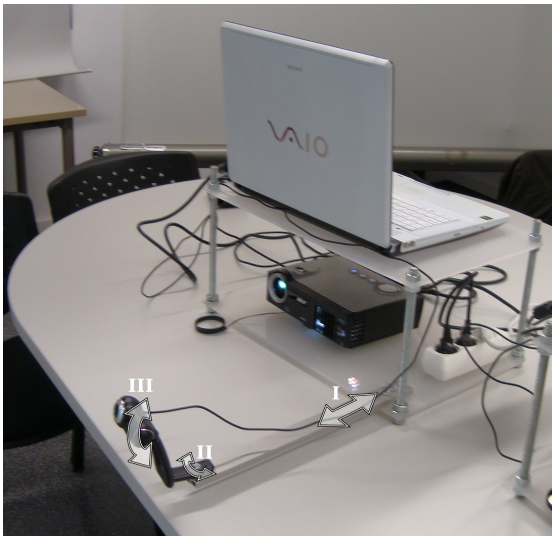


Figura 10.1: Fotografia d'un dels prototipus muntats.

El braç que suporta la càmera ofereix diversos graus de llibertat, de tal manera que la podem apropar i allunyar respecte el focus del projector (I), rotar-la lleugerament (II) o fer que enfoqui més amunt o més avall (III).

CAPÍTOL 11

Cost del projecte

En aquest capítol s'ha dut a terme un breu anàlisi de costos del projecte, on es calcula, per una banda, el cost dels treballadors que hi haurien estat implicats i, per l'altra, el del maquinari que s'ha hagut d'adquirir.

És important adonar-se de la importància que té haver realitzat la planificació correctament per tal de calcular amb precisió el cost del projecte abans de començar-lo. Haver fet una bona planificació al principi de les tasques i les seves duracions, així com preveure qui se n'hauria d'encarregar, permet estimar molt bé el cost que caldria assumir per a desenvolupar-lo.

En aquest cas, però, s'ha utilitzat la seva duració real[1], enlloc de l'estimació inicial. D'aquesta manera, el que obtenim al final és el cost real que representa.

11.1 Cost salarial

Primerament, cal notar que les tasques necessàries per a dur a terme el projecte requereixen l'acció de treballadors amb diferents habilitats. En el nostre cas, aquestes quedaven cobertes amb dos rols[2]: un *analista* i un *programador*.

11.1.1 Dedicació dels treballadors

A cadascun dels rols se l'hi assigna les tasques que li pertoca desenvolupar i, per cadascuna, en quina mesura ho fa. Això permet determinar quantes hores ha dedicat a una tasca en concret i, conseqüentment, quantes hores ha dedicat en total al projecte. Tota aquesta informació la podeu veure recopilada a les taules 11.1 i 11.3.

Així, per exemple, si algú ha dedicat dos dies de feina a una tasca determinada, és a dir, un total de setze hores, però només amb un 50% de dedicació, hi hauria dedicat un total de $16h \times 50\% = 8h$.

[1] Vegeu la secció 4.5.

[2] De fet, és possible considerar un tercer rol, encarregat de la construcció dels prototipus. Aquest, però, s'ha ignorat a l'hora de realitzar els càlculs ja que, tal i com es pot veure més endavant, s'assumeix que aquesta tasca la fa el programador.

Activitat	Dies	Dedicació	Hores
Especificació del mòdul de calibratge	67	40%	215
Disseny del mòdul de calibratge	65	40%	205
Especificació de l'aplicació d'usuari	45	30%	110
Disseny de l'aplicació d'usuari	8	40%	25
		Total d'hores	555

Taula 11.1: Hores invertides per l'analista al projecte.

Activitat	Dies	Dedicació	Hores
Desenvolupament calibratge geomètric	35	60%	170
Desenvolupament calibratge cromàtic	25	60%	120
Correccions necessàries al calibratge	30	60%	140
Escriptura de resultats	3	40%	10
Desenvolupament del mòdul de visualització	67	60%	320
		Total d'hores	760

Taula 11.2: Hores invertides pel programador al projecte.

11.1.2 Salaris

Després de conèixer la dedicació en hores al projecte de cada treballador, cal saber quina és la seva retribució per hores per tal de poder calcular-ne el cost.

Per tal de fer-nos una idea del salari mig d'un analista i d'un programador s'ha consultat [IJT]. A la següent figura es pot veure l'evolució dels salaris de cadascun d'ells:

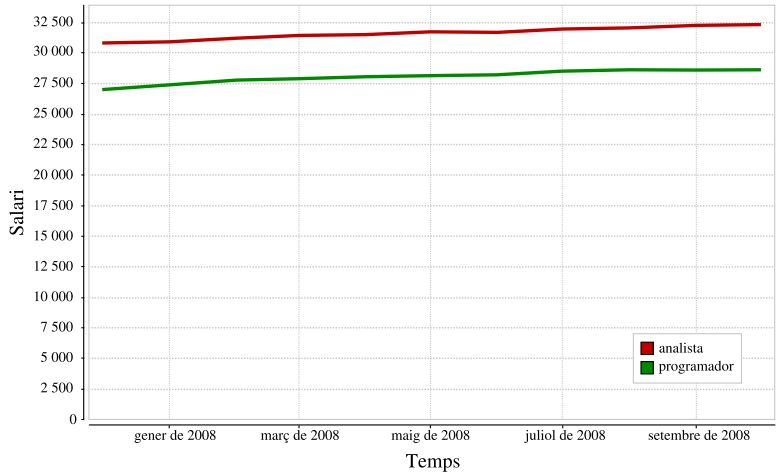


Figura 11.1: Evolució dels salaris mitjos per a un programador i un analista des del gener de 2008 fins al setembre de 2008 [IJT].

Finalment, tal i com veiem a la figura 11.1, assumim que el sou mig d'un programador és de 28 000 €/*any* i el d'un analista és de 31 000 €/*any*. Si suposem que aquestes quantitats és divideixen en 14 pagues, i que cada mes consta de quatre setmanes amb cinc dies laborables cadascuna d'elles, i amb vuit hores per dia, hem de dividir aquestes quantitats per $14 \times 4 \times 5 \times 8 = 2240$ per a conèixer el sou per hora:

Treballador	**Hores**	**Salari (€/h)**	**Total**
Analista	555	13.84	7 680
Programador	760	12.50	9 500
		Total (€)	17 180

Taula 11.3: Cost en salaris del projecte.

11.2 Cost del maquinari

Una altra part important del projecte és el cost de la infraestructura física que es munta. És necessari disposar de projectors, adquirir les càmeres, etc. La següent taula mostra el cost de cadascun d'aquests components i el total que representa per al projecte:

Concepte	**Unitats**	**Preu unitat (€)**	**Preu (€)**
Portàtil	2	1 200	2 400
Projector	2	800	1 600
Wiimote®	1		60
Càmera Logitech	2	70	140
Prototipus	2		
Connector elèctric	2	2.35	(4.70)
Planxa de metacrilat d'$1m^2$	1	ND	(-)
Guia $1m$	1	ND	(-)
Cargols, arandeles, etc.	-	-	(7.00)
		Cost total	4 210

Taula 11.4: Cost dels components necessaris per a muntar els prototipus.

11.3 Cost total

Finalment, després d'haver analitzat detalladament les principals "despeses" que comporta el projecte, ja és possible calcular-ne el *cost total*:

$$17\,180\,€ + 4\,210\,€ = \mathbf{21\,390\,€}$$

CAPÍTOL 12

Feina futura

Després d'haver exposat com s'ha construït el sistema i les seves característiques, és hora d'explicar possibles ampliacions que poden fer-s'hi o canvis que s'hi poden aplicar per tal de millorar-lo.

Per una banda, ja s'ha comentat a la introducció que aquest projecte és el predecessor d'una eina que pot ser molt més potent i ambiciosa. Tots els canvis que puguin afegir-s'hi es converteixen en ampliacions que l'hi aproparan.

Per altra banda, el codi actual no està exempt de petits errors o punts millorables i, per això, també es vol dedicar un espai on es posin de manifest de forma general i on s'esbossin possibles solucions.

12.1 Millora del codi actual

12.1.1 Precisió del calibratge geomètric

Un dels punts febles del codi actual és la precisió del calibratge geomètric. El codi de què disposem ara mateix no aconsegueix un calibratge la precisió del qual sigui a nivell de *pixel* o, fins i tot, de *subpixel*. Si es tractés d'un producte comercial no serien admissibles els petits errors amb què ens enfrontem ara.

Per tal d'aconseguir millorar aquesta situació, cal millorar les tècniques de visió per computador que utilitzem, millorant els filtres que apliquem a les imatges per tal d'eliminar el màxim de soroll possible.

També es pot aprofitar la redundància que tenim al realitzar captures des de dues càmeres, enlloc de només basar-se en les captures que ha realitzat una d'elles.

12.1.2 Temps de calibratge

El temps emprat en el procés de calibratge és força gran, al tractar-se d'un procés lent. Fins i tot quan s'assumeix que no es tracta d'un punt crític, val la pena pensar com calibrar el sistema més ràpidament.

Per una banda, per a reduir els temps total cal millorar l'eficiència dels algorismes que es van aplicant, per tal de fer-los més ràpids.

Per altra, cada mòdul consta d'un ordinador que pot executar càlculs i, per tant, on s'hi poden distribuir els còmputs que es vulguin. Tot i que això ja es fa en alguns casos, com el càlcul d'homografies, n'hi ha molts altres que només s'executen al mòdul que actua com a *master*. Si aquestes operacions es duguessin a terme des d'una perspectiva distribuïda, s'aconseguiria reduir els temps de calibratge i s'aprofitaria millor la infraestructura de què es disposa.

12.1.3 Seguretat a la xarxa

Un altre punt important que caldria millorar si es tingués una aplicació comercial seria la seguretat en xarxa.

Per a aquest projecte no ha sigut necessari contemplar-la, atès que s'ha suposat que l'aplicació s'executa en un entorn aïllat segur. Si es volgués utilitzar fora d'aquest context, però, seria un requisit imprescindible.

12.2 Possibles ampliacions del projecte

Les ampliacions del projecte estan encaminades a transformar-lo en un autèntic substitut de la *cave*. Això permet intuir que les diferents ampliacions que es proposaran són les petites diferències que hi ha entre ella i el sistema desenvolupat.

12.2.1 Visió en estèreo

Una de les ampliacions més senzilles a implementar és l'estereovisió, atès que els canvis que implicaria al codi són mínims.

Caldria decidir si es fa *activa* o *passiva*, cosa que depèn del tipus de construcció final de què disposem. Tenint en compte que es vol projectar sobre parets arbitràries hauria de ser activa, perquè per a tenir-la passiva ha d'enviar-se la llum polaritzada, però les parets eliminarien la polarització que s'ha aplicat, fent-ho inviable.

De fet, recomanem fer-la activa, ja que d'aquesta manera es disposaria d'un sistema que funcionaria sota qualsevol condició d'ús.

12.2.2 Projecció sobre més d'una paret

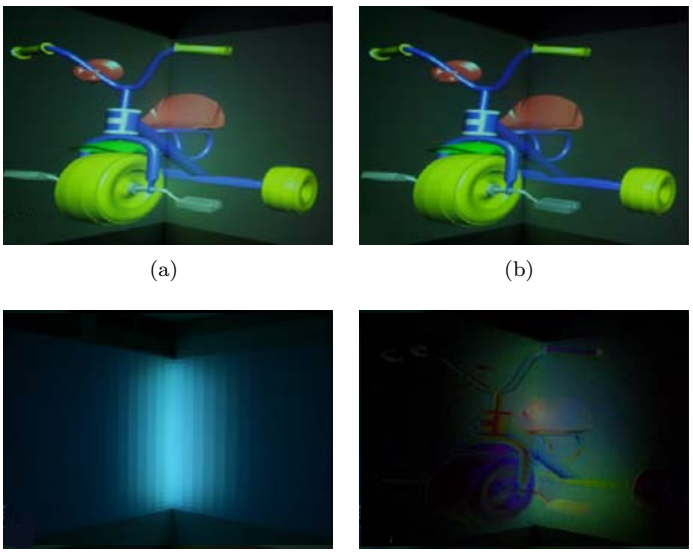
El següent pas seria implementar un calibratge que corregís la deformació que es produeix al projectar sobre parets, com a la figura 12.1.

12.2.3 Correcció cromàtica ampliada

La darrera ampliació que es proposa és augmentar el paper que juga la correcció cromàtica al sistema.

Cada projector és lleugerament diferent dels altres, degut als errors de precisió al muntatge i construcció d'aquests. Aquests petits errors provoquen que els colors que projecta difereixin lleugerament entre ells, fins i tot quan són de la mateixa marca i model. També resulta que les parets on es projecta no són perfectes; poden estar brutes, tenir desnivells, tenir zones més brillants, etc.

Si es vol un sistema realment flexible, cal tenir en compte tot això i preveure que no sempre les parets seran adequades per projectar-hi, amb la qual cosa ens veurem amb l'obligació de

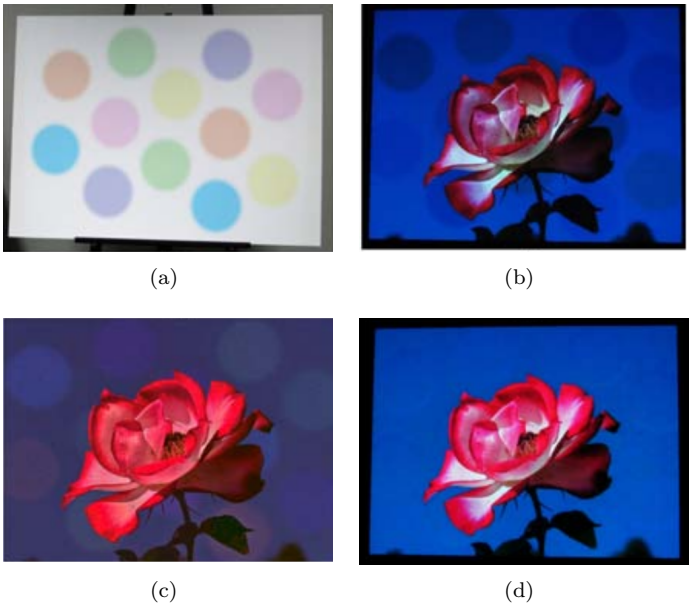


(a) (b)

(c) (d)

Figura 12.1: Exemple de projecció sobre més d'una paret [BIWG08].

solucionar-ho. Implementant alguns dels algorismes descrits a [BIWG08] es poden aconseguir resultats molt espectaculars.



(a) (b)

(c) (d)

Figura 12.2: Exemples de correccions cromàtiques més avançades [BIWG08].

CAPÍTOL 13

Conclusions

El projecte de final de carrera és la darrera activitat que realitzem a la universitat, on demostrem tot allò que hem après durant els últims anys. El tema que he escollit, la realitat virtual, sempre m'ha agradat i no hi havia pogut treballar anteriorment. Per això vaig decidir embarcar-m'hi.

Per tal d'acabar aquesta memòria i donar per finalitzat el projecte, m'agradaria comentar un parell de punts que considero importants.

13.1 Quant als objectius

Els objectius que ens havíem plantejat al principi del projecte han estat completament coberts. Hem aconseguit dissenyar i muntar un sistema de realitat virtual portable amb *head tracking*, el qual es calibra automàticament.

Ja hem vist al capítol de "Feina futura" que el producte és ampliable i millorable, però el que s'ha desenvolupat assoleix uns resultats molt bons i, sens dubte, satisfactoris.

13.2 Quant a la planificació i els costos

La planificació del projecte no ha sigut gens senzilla, sobretot perquè s'assumia que ens ocuparia dos quadrimestres i s'ignorava, al començar-lo, com es duria a terme el desenvolupament de l'aplicació d'usuari.

La planificació inicial que vam fer no anava molt desencaminada de la realitat en quant a duracions, però sí que va resultar un xic optimista. Per una banda, donava per fet que començaríem a treballar amb el projecte abans del que realment ho vam fer, i això ha comportat desviacions clares en les tasques inicials. Per altra banda, refer la planificació a mig projecte (aprofitant que s'havia d'entregar a l'*informe previ*) va ser una molt bona idea, atès que vam poder concretar les tasques que s'havien de dur a terme per a realitzar la segona part, tot tenint en compte els errors comesos al principi.

En qualsevol cas, el projecte s'ha pogut completar dins del termini previst, assolint, com ja hem dit, tots els objectius amb èxit.

13.3 Quant al futur comercial

L'empresa *i2cat* es dedica a desenvolupar i investigar en l'àmbit d'Internet. Aquesta companyia està interessada comercialment en el nostre PFC per aplicar-ho en un projecte de col·laboració cultural entre diferents regions de Catalunya anomenat "Anella Cultural".

De moment, les dues parts (universitat i empresa) hem acordat aplaçar les negociacions fins a acabar la defensa i fer l'entrega del projecte.

13.4 Personals

Primerament, el fet de realitzar el projecte entre dues persones fa que l'enfocament amb el qual es planteja el seu desenvolupament sigui totalment diferent. Cal que les tasques que ha de desenvolupar cadascú de nosaltres estiguin ben definides i que les decisions que ens afecten als dos es prenguin de manera consensuada.

Avui en dia no és habitual que la càrrega del desenvolupament d'un projecte d'enginyeria informàtica recaigui sobre una única persona, sinó que es reparteix entre equips de desenvolupadors. Per tant, considero que haver-lo fet en equip ens apropa més al món real i ha servit per a preparar-nos millor.

En segon lloc, una de les coses que he trobat més interessant i difícil de realitzar ha sigut adaptar tots els coneixements del meu perfil, l'enginyeria del *software*, a un projecte d'aquestes característiques.

El nostre projecte és bastant proper a les assignatures de gràfics que hi ha a la facultat, com VIG (Visualització i interacció gràfica) o VA (Visualització avançada), on no és habitual dur a terme totes les fases prèvies d'anàlisi i especificació d'una aplicació. Per altra banda, els exemples amb els que treballem a la universitat en l'àmbit de l'especificació acostumen a estar sempre enfocats a sistemes de gestió, on es manega informació d'empleats, clients, etc. El fet de poder integrar tots els coneixements d'enginyeria del *software* a un projecte com aquest ha resultat una tasca complexa, no trivial, però que m'ha ensenyat molt.

El resultat obtingut d'aquesta integració és una documentació exhaustiva i útil, que clarifica tots els components que composen el projecte. En aquest aspecte estic molt content del resultat, atès que tota la informació recopilada a la memòria, així com la documentació generada al codi, és molt bona. Això és molt important, si tenim en compte que aquest projecte pretén ser quelcom més gran en un futur i que, per tant, pot arribar a ser continuat per terceres persones.

Finalment, m'agradaria concloure aquesta memòria subratllant que dur a terme aquest projecte ha estat una molt bona oportunitat a nivell personal, on he après moltes coses noves i on he pogut consolidar-ne d'altres.

APÈNDIX A

Format de les dades

En aquest annex veurem els diferents paràmetres que es poden configurar de l'aplicació.

Ja hem comentat que la manera en què introduirem aquests valors al sistema serà via fitxers XML. Tot seguit veurem de quins fitxers concrets disposem, la seva sintaxi concreta i què ens permeten modificar.

A.1 Configuració de l'aplicació

El primer fitxer amb què ens trobem per a configurar la nostra aplicació és `default-configuration.xml`. Aquest especifica el patró que utilitzarem, l'algorisme de còmput de la geometria de la pantalla final i molts altres paràmetres. Tot seguit, podem veure un exemple:

```
<?xml version="1.0" encoding="UTF-8"?>

<pvrs-configuration>
  <!--
     The pattern to be used.
     Built-in flag specifies whether this xml is contained in the jar or
     it is located somewhere else.
  -->
  <pattern built-in="true" file="PatternFile.xml" />
  <display pirf-class="edu.up...NaivePolygonInscribedRectangleFinder" />


  <filters>
    <filter class="edu.up...filters.BasicColorsFilter">
      <param name="RGBMin" value="40,35,37" />
    </filter>
  </filters>

  <finders>
    <finder class="edu.up...finders.RegressionLinesFinder">
      <param name="RGBMin" value="35,30,45" />
      <param name="dispersionThres" value="0.96" />
      <param name="mindiff" value="0.70" />
```

```
      <param name="minPoints" value="500" />
    </finder>
  </finders>

  <misc>
    <!-- Misc -->
    <param name="common-config-file" value="C:\common-config.txt" />
    <param name="fb-width" value="1024" />
    <param name="fb-height" value="768" />
    <param name="camera-opts-selector" value="true" />

    <!-- Darkener factors -->
    <param name="darkener-factors-its" value="1" />
    <param name="darken-dominance" value="0.025" />
    <param name="darken-overall" value="0.7" />

    <!-- Geometric alignment -->
    <param name="homography-image" value="C:\dp\patro.jpg" />
    <param name="capture-delay" value="2500" />
    <param name="homographies-mean" value="true" />

    <!-- Chromatic corrections -->
    <param name="darken" value="0.75,0.5,0.75" />
    <param name="mask-fix-intensities" value="false" />
    <param name="ca-intensity-tune" value="0.9" />
    <param name="fix-common-area" value="true" />

    <!-- testing -->
    <param name="final-screenshot" value="C:\tmp\calibresult.jpg" />
    <param name="fd-testing" value="false" />

    <!-- Default IP -->
    <param name="default-interface" value="eth1" />
  </misc>

  <save-images dir="C:\tmp\">
    <!-- camera params -->
    <param name="captured-background" value="true" />
    <param name="captured" value="true" />
    <param name="substracted" value="true" />
    <param name="binarized" value="true" />

    <!-- other data -->
    <param name="filtered" value="true" />
    <param name="line-filtered" value="false" />
    <param name="line-processed" value="true" />
  </save-images>
</pvrs-configuration>
```

Codi A.1: Exemple de fitxer de configuració de l'aplicació.

Les principals característiques d'aquest fitxer són:

- La línia 8 especifica el fitxer XML que descriu el patró que utilitzarem per a dur a terme el calibratge[1].
- La línia 9 especifica la classe que calcula la geometria de la pantalla final. Necessitem un algorisme que calculi un rectangle dins d'un polígon arbitrari, i la classe que aquí indiquem se suposa que l'implementa. Per defecte, utilitzem `NaivePolygonInscribedRectangleFinder`.

[1] Vegeu la següent secció A.2 per a conèixer com és aquest fitxer.

- El següents dos blocs que tenim són els `<filters>` i els `<finders>`. Serveixen per a especificar els paràmetres concrets de cada `<filter>` i `<finder>` que té el nostre sistema. Què es pot configurar de cadascun d'ells depèn de la implementació concreta de cada classe, i és per això que s'utilitza un format genèric per a passar els valors a les variables que es poden modificar:
  `<param name="`*nom*`"value="`*valor*`"/>`.
  Aquest format ens permet expressar el nom d'un paràmetre i el valor que aquest adquirirà. El *filter* o *finder* cercarà el valor concret per a l'atribut aprofitant el nom del paràmetre i li correspondrà a ell fer el *parsing* adequat del valor.

- A continuació, disposem d'un bloc `<misc>`. En aquest apartat definim tot un conjunt de paràmetres miscel·lània que hem anat introduint mentre desenvolupàvem l'aplicació, el propòsit principal dels quals era poder fer un bon *debug* del codi.
  Els paràmetres més interessants a comentar que podem manipular són:

  - 29: Fitxer que conté la informació sobre les ubicacions dels fitxers d'intercanvi entre la part de calibratge i l'aplicació d'usuari (A.3).
  - 30-31: Dimensions del *frame buffer*.
  - 45: Valors per enfosquir la intensitat de llum al projectar colors, per tal d'evitar la saturació de colors de la càmera.
  - 46: Corregir les intensitats dels projectors, per a què, si un brilla molt més que l'altre, això no sigui perceptible.
  - 55: Per tal d'obtenir la IP del nostre mòdul, conèixer quina és la interfície de xarxa que s'ha de consultar.

- El darrer bloc, `save-images`, ens permet desar imatges de les captures efectuades per les càmeres en diferents estats del procés de calibratge. Això permet *debugar* el codi i veure què estant fent els algorismes de visió per computador.

A.2 Configuració d'un patró

Els patrons que podem utilitzar per a dur el procés de calibratge es defineixen via un fitxer de configuració. Això permet dissenyar nous patrons molt fàcilment, de tal manera que integrar-los a l'aplicació tingui un cost mínim:

```
0  <?xml version="1.0" encoding="UTF-8"?>

   <pattern bgcolor="0,0,0">
     <matcher classname="PolychromeReg...neMatcher" built-in="true" />

5    <!-- Horizontal lines -->
     <pattern-component>
       <hline color="255,0,0" width="2" normalized="true">0</hline>
       <hline color="0,255,0" width="2" normalized="true">0.5</hline>
       <hline color="0,0,255" width="2" normalized="true">1</hline>
10     <hline color="0,0,255" width="2" normalized="true">0.9</hline>
     </pattern-component>

     <!-- Vertical lines-->
     <pattern-component>
15     <vline color="255,0,0" width="2" normalized="true">0</vline>
       <vline color="0,255,0" width="2" normalized="true">0.5</vline>
       <vline color="0,0,255" width="2" normalized="true">1</vline>
```

```
  </pattern-component>

   <pattern-component>
     <point color="255,0,0" width="20" height="20" normalized="true">
       <xcoord>0</xcoord>
       <ycoord>0</ycoord>
     </point>
     <point color="0,255,0" width="20" height="20" normalized="true">
       <xcoord>1</xcoord>
       <ycoord>0</ycoord>
     </point>
  </pattern-component>
</pattern>
```

Codi A.2: Exemple d'un XML on es defineix un patró.

El que cal definir en un patró és:

- La classe `Matcher` que s'encarregarà de calcular les correspondències entre una captura i un `PatternComponent`. Això permet que si definim un patró molt complex, puguem implementar un `Matcher` nou que pugui treballar-hi.
- La geometria del patró (organitzada en conjunts de `pattern-component`):
  - `hline`: Línia horitzontal.
  - `vline`: Línia vertical.
  - `point`: Punt (especificant les coordenades x `xcoord` i y `ycoord`).

  Per cadascun d'aquests tipus de dades podrem especificar el `color`, l'ample `width` i, a més a més en el cas dels punts, l'alt `height`.
  Finalment, també podem definir, mitjançant l'opció `normalized`, si la ubicació de la línia o el punt dins del *frame buffer* està amb coordenades absolutes o relatives. En el primer cas, indiquem directament el *pixel* superior esquerra on s'ha d'ubicar la unitat d'informació; en l'altre, les coordenades pertanyen a l'interval [0..1], de tal manera que el patró s'escala automàticament segons les dimensions del *frame buffer*.

A.3 Exportar els resultats

Per acabar, només ens cal veure el format dels fitxers d'intercanvi de dades entre el mòdul de calibratge i l'aplicació d'usuari. Entre ambdós mòduls és necessari passar la següent informació:

- Homografia: la homografia que necessita el mòdul per a deformar convenientment l'escena 3D.
- Màscara cromàtica: fitxer en format PNG que conté la màscara que s'ha d'aplicar a l'escena 3D.
- Llista de direccions IP: l'aplicació d'usuari que actua com a *master* necessita conèixer les direccions IP de la resta de mòduls.

Degut a què hem de passar diferents blocs d'informació, definim un primer fitxer on s'especifica la ubicació de cadascun d'aquests altres fitxers. Això ens permet configurar el lloc on el mòdul de calibratge desarà els resultats i d'on, després, l'aplicació d'usuari els llegirà:

```
1 net config file@C:\Documents and Settings\Administrador\Mis documentos\
      PFC\part2\bin\Debug\netconfig.dat
2 homography file@C:\Documents and Settings\Administrador\Mis documentos\
      PFC\part2\bin\Debug\homography.dat
3 chromatic mask file@C:\Documents and Settings\Administrador\Mis
      documentos\PFC\part2\bin\Debug\chrom-mask.png
4 wii config file@C:\Documents and Settings\Administrador\Mis documentos\
      PFC\part2\bin\Debug\wii-config.dat
```

Codi A.3: Fitxer on s'especifiquen les ubicacions dels fitxers d'intercanvi, comú entre el mòdul de calibratge i l'aplicació d'usuari.

Primerament, necessitem un fitxer on escriure l'homografia calculada:

```
1 0,953562671818114 -0,046437328181886 0,775862495975432
2 0,077006505353708 -0,229297207155889 0,082166208485029
3 0,729425167793546 -0,046437328181886 1,000000000000000
```

Codi A.4: Fitxer que conté una homografia.

En segon lloc, també necessitem un fitxer amb la llista d'IP de la resta de mòduls (el primer nombre indica quants mòduls més, a més a més d'un mateix, hi ha):

```
1 1
2 192.168.1.101
```

Codi A.5: Fitxer amb la llista de direccions IP.

Finalment, només necessitem un fitxer d'imatge en format PNG (amb canal alfa) que contingui la màscara.

APÈNDIX B

Modelització conceptual

B.1 Introducció

La manera en què s'escriuen els programes avui en dia és força diferent a com es feia en un principi. A mesura que han anat passant els anys, s'han anat afegint diferents capes d'abstracció que faciliten el desenvolupament de programari, reduint les hores invertides en programar i en corregir errors.

Els primer programadors escrivien el codi directament en zeros i uns que especificaven les instruccions que l'ordinador havia d'executar. És prou evident que es tractava d'una tasca molt complicada i molt propensa a la introducció d'errors.

El naixement dels llenguatges assemblador va suposar una gran revolució en aquest aspecte. El programadors podien utilitzar un conjunt d'instruccions de codi màquina, oblidant-se de les cadenes binàries que fins aleshores havien d'utilitzar.

Més endavant es va afegir una nova capa d'abstracció amb els llenguatges de programació d'alt nivell. Molt més propers al llenguatge natural, aquests resultaven molt més intel·ligibles, de tal manera que mantenir-los i corregir-los resultava molt millor. A més a més, la introducció de compiladors per a traduir codi d'alt nivell a codi assemblador permetia, per una banda, realitzar comprovacions sintàctiques i semàntiques del codi escrit i, per altra, allunyar-se de l'arquitectura concreta de la màquina.

Avui en dia, una nova manera de programar està naixent. L'MDA (*Model-Driven Architecture*, Arquitectura dirigida per models), proposada i patrocinada per l'OMG (*Object Management Group*), defineix el model del sistema i les funcionalitats que el caracteritzen, de manera totalment independent a la tecnologia emprada.

B.2 Modelització conceptual

En l'àmbit de l'enginyeria del *software*, entenem per modelització conceptual l'activitat que especifica i descriu el coneixement que un sistema ha de conèixer sobre un cert domini. El principal objectiu és, doncs, descriure'n l'esquema conceptual.

Qualsevol sistema *software* disposa d'aquest esquema conceptual, però en molts casos aquests no està plasmat en un document físic i queda amagat entre milers de línies de codi.

És clar que per a desenvolupar qualsevol programa és necessari que tota la gent involucrada sàpiga quina informació ha de manegar i com ho ha de fer. Això vol dir que qualsevol d'ells té un esquema conceptual en ment. Evidentment, això és un problema, atès que l'objectiu és coordinar esforços i que tots es dirigeixin cap al mateix objectiu, cosa que resulta difícil si es tenen diferents idees de què s'espera.

Definir l'esquema conceptual a les primeres fases de desenvolupament de programari permet detectar possibles carències que tingui el sistema en un moment en què dur a terme els canvis necessaris suposaria un cost relativament baix.

Involucrar als usuaris per a la creació del model conceptual amb els seus requeriments és molt important, especialment si tenim en compte el *principi d'incertesa de l'enginyeria de requisits*, pel qual es té que els requisits d'un sistema no podran ser considerats com a adequats fins que l'usuari hagi pogut utilitzar-lo i validar-lo[Oli07].

B.3 Esquema conceptual

L'esquema conceptual és el resultat de la modelització conceptual. Descriu els elements més importants del domini (*entitats*), emmagatzemant informació sobre els seus atributs i sobre les associacions que hi ha entre ells (*relacions*). Es tracta, doncs, d'una descripció dels *objectes*, les *relacions* i els *atributs* d'un domini.

Els *llenguatges de modelització conceptual* són la manera que tenim de representar tota aquesta informació d'un domini. Un d'aquests llenguatges és l'UML, el qual veurem amb més detall tot seguit.

B.4 UML

En aquest apartat es descriuen molt breument les principals característiques de l'UML, per tal de poder entendre un esquema conceptual bàsic. Si voleu conèixer amb més detall aquest llenguatge, vegeu [RJB04].

B.4.1 Entitats i atributs

Una entitat és un concepte les instàncies del qual són objectes identificables individualment, els quals se suposa que existeixen al domini en un moment determinat.

Imaginem que el nostre domini és una botiga virtual com pot ser Amazon. Com a botiga que és, aquesta disposa de conceptes evidents com *productes* o *clients*.

Figura B.1: Exemple d'entitats modelades en UML.

La figura B.1 mostra com representar ambdós conceptes en UML: per a un producte s'emmagatzema informació sobre el seu nom i el seu preu, i per a un client interessa el nom, el nom d'usuari o l'adreça on viu.

Herències

Podem pensar que tenir una classe anomenada `Producte` és, potser, massa genèric. A Amazon és possible comprar llibres, CD, pel·lícules, etc. És evident que diferents tipus de productes tenen diferents atributs que volem emmagatzemar (per a un llibre pot interessar conèixer l'autor i l'any de publicació, i per a una pel·lícula el director, l'any de l'estrena i els actors), però tots ells són, al cap i a la fi, productes. UML permet representar aquesta realitat mitjançant una jerarquia de classes:

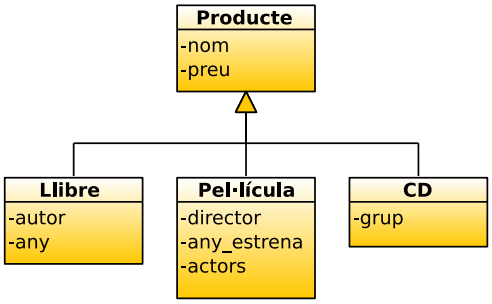


Figura B.2: Exemple d'una jerarquia de classes.

A la figura B.2 es veu com representar una jerarquia en UML. Els diferents tipus concrets de producte de què disposem hereten de la classe `Producte` i, per tant, disposen de tots els atributs definits a la super classe i, a més a més, especifiquen els atributs concrets que requereixen. Així, per exemple, un llibre, un CD i una pel·lícula disposen, tots ells, de l'atribut *nom* i l'atribut *preu*, però només una pel·lícula té un *director* o un llibre un *autor*.

B.4.2 Relacions

A qualsevol domini del món real existeix quelcom més que "entitats", i són les relacions entre ells. No només es vol saber *què hi ha* al món que estem modelitzant, sinó *com es relaciona*.

Continuant amb l'exemple d'Amazon, hem vist que disposem de dues entitats: un *client* i un *producte*. La relació entre aquests dos conceptes és ben clara i coneguda per tots nosaltres: un *client* realitza una *comanda* a l'empresa, on el que fa és demanar un conjunt de *productes*:

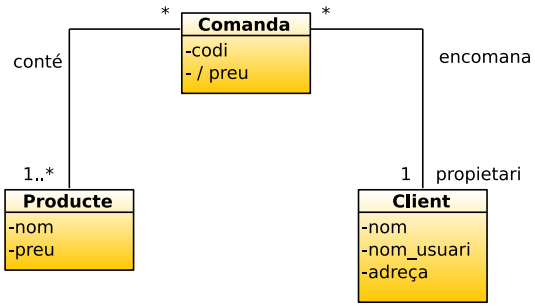


Figura B.3: Exemple de relacions en UML.

La figura B.3 mostra el nou concepte `Comanda` i com aquest es relaciona amb els altres dos. Les relacions es representen com una línia que uneix les classes, i que disposen d'una sèrie d'atributs que ens permeten restringir-ne l'àmbit d'aplicació:

- Els extrems d'una relació poden, eventualment, disposar d'un *nom de rol* (si s'omet, aquest nom se suposa que és el de la classe). Això té una funcionalitat purament semàntica, per a fer més comprensible l'esquema conceptual, o per a evitar, en alguns casos, possibles ambigüitats. Per exemple, el client d'una comanda n'és el *propietari*.
- Les relacions poden tenir noms que les descriguin. Novament, es tracta d'una eina que ens permet definir el significat de la relació. En el nostre cas, podem veure que la relació entre un client i una comanda és que el primer "encomana" una comanda i que la relació entre una comanda i els productes és que la comanda "conté" productes.
- L'altre detall important són els nombres que acompanyen els extrems de la relació. Aquests identifiquen el nombre d'instàncies amb què poden estar relacionats els conceptes. Per tal d'interpretar els nombres, cal situar-se en una classe concreta i fixar-nos en una relació; el nombre de l'extrem oposat ens indica amb quantes instàncies de la classe destí pot estar relacionada la classe en què ens hem centrat:
  - 0..1: Pot estar relacionada amb una o cap instància.
  - 1: Ha d'estar forçosament relacionada amb una instància.
  - $*$: Pot estar relacionat amb tantes instàncies com vulgui (o cap).
  - 1..$*$: Ha d'estar relacionada amb, com a mínim, una instància de l'altra classe.
  - n: Ha d'estar relacionada amb exactament n instàncies de l'altra classe.
  - $n..m$: Ha d'estar relacionada amb un total d'entre n i m instàncies de l'altra classe, ambdós inclosos, essent $n < m$.

  A l'exemple que estem tractant, donada una comanda, aquesta ha de tenir forçosament d'un (i només un) propietari (vegeu l'1 a l'extrem oposat de la relació amb client), i té un o més productes (1..$*$). Per altra banda, si mirem les relacions al revés, veurem que un client pot tenir tantes comandes com vulgui (eventualment, això pot voler dir que no en té cap, i per això posem $*$), i que una comanda pot pertànyer a tantes comandes com es vulgui (o, novament, a cap, perquè encara no l'ha encomanat ningú).

Atributs derivats

La classe `Comanda` disposa d'un atribut *preu* que està precedit per una "/". Això indica que aquest atribut és calculat; és a dir, no és quelcom que estigui emmagatzemat directament al sistema d'informació dissenyat, sinó que es calcula seguint alguna regla. En aquest cas, és prou clar que el preu d'una comanda és la suma dels preus individuals de tots els productes encomanats.

Aquest atribut posa de manifest una mancança de l'UML: amb l'esquema gràfic mostrat no tenim cap manera de descriure com calcular l'atribut. Per tal de suplir aquesta carència es podria adjuntar textualment "el *preu* d'una comanda es calcula com la suma dels preus de tots els productes que hi ha inclosos". Això, però, no s'hauria de fer amb llenguatge natural, atès que és ambigu. Cal, doncs, incloure una nou llenguatge textual que permeti representar aquest tipus de *regles de derivació*, així com eliminar possibles ambigüitats que sorgeixin de l'esquema conceptual.

B.4.3 OCL

L'OCL (*Object Constraint Language*, Llenguatge de restriccions d'objectes) és un llenguatge *formal* que permet descriure expressions als models UML. Aquestes poden representar *precondicions*, *poscondicions*, *invariants*, etc.

La comprensió de com funciona aquest llenguatge cau fora de l'abast del projecte i, per tant, no es comenta més enllà d'aquesta breu definició, però podeu trobar-ne més informació a [WK03].

APÈNDIX C

Instal·lació i configuració

Aquest annex descriu els passos que s'han de fer per tal de poder executar el nostre projecte, descrivint com instal·lar tot el que necessita l'ordinador i com configurar-lo.

Se suposa que el sistema operatiu de què disposem és un *Microsoft® Windows® XP* correctament actualitzat (Service Pack 3).

C.1 Arbre de directoris del CD

L'arbre de directoris del CD és el següent:

```
cdrom
|-- misc
|-- pfc
|   |-- doc
|   |-- part1
|   |   |-- bin
|   |   |-- lib
|   |   `-- src
|   `-- part2
`-- software
    `-- installers
```

- El directori `misc` conté vídeos d'exemple.
- El directori `pfc` conté aquesta memòria i el seu codi font, així com el codi font i compilat del mòdul de calibratge i de l'aplicació d'usuari.
- El programari addicional que cal instal·lar està correctament ubicat dins del directori `software/installers`.

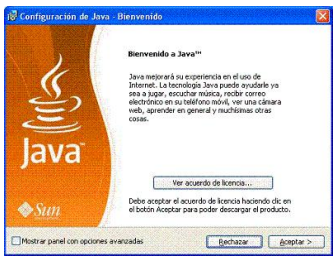
C.2 Instal·lació de Java

Abans d'executar l'instal·lador de Java és necessari que tanqueu totes les aplicacions, fins i tot el navegador web.

1. Feu doble clic a l'instal·lador `jre-6u3-rc-windows-i586.exe`.

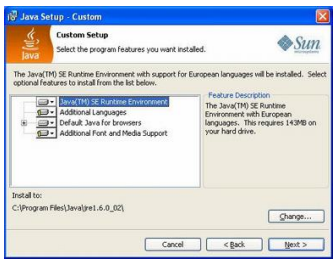
   L'instal·lador descomprimirà tots els arxius necessaris per a la instal·lació.

2. A la finestra *Configuració de Java - Benvingut*, l'instal·lador us permet veure el *Contracte de llicència*. Accepteu-lo fent clic al botó *Accepta* per continuar el procés d'instal·lació.



3. Tot seguit es mostren diferents elements que es poden instal·lar, com la *Barra de Google per a Internet Explorer* i *Google Desktop*. Escolliu el que vulgueu i feu clic a *Següent*.

4. L'instal·lador mostra una nova pantalla d'*instal·lació personalitzada*. Deixeu les opcions per defecte i feu clic a *Següent*.



5. Apareix un diàleg indicant quin és el progrés de la instal·lació.

6. Tot seguit, s'obriran diversos diàlegs per a realitzar les darreres etapes de la instal·lació. Finalment, apareixerà un missatge de confirmació, agraint-vos haver instal·lat Java.

C.3 Instal·lació de *DirectShow Java wrapper*

Descomprimiu l'arxiu `dsj.zip` al vostre escriptori i realitzeu els següents passos:

1. Assegureu-vos de tenir instal·lat *DirectX 9* i el reproductor *Windows Media Player9 9* o superior. Si no els teniu instal·lats no disposareu de tots els descodificadors necessaris i és possible que els dispositius de captura no funcionin.

2. Poseu el fitxer `dsj.dll` al directori `jre/bin` que trobareu al directori d'instal·lació de Java (generalment, `C:\Arxius de programa\Java`), o al directori de Sistema (`C:\WINDOWS\System32`).

3. Poseu el fitxer `dsj.jar` al direcotri `jre/lib/ext`.

Si després d'executar aquests passos el DSJ encara no funciona, comproveu si disposeu de més d'una instal·lació de Java. En tal cas, assegureu-vos d'haver copiat els fitxers a la ubicació de la *Java Runtime Environment* que s'està executant.

C.4 Instal·lació del *driver Blue Soleil*

1. Feu doble clic al fitxer `blue-soleil-setup.exe` per tal d'iniciar l'instal·lador.
2. Apareix una finestra on podeu seleccionar l'idioma de l'instal·lador. Seleccioneu-ne un i feu clic a *Següent*.
3. Seguiu les instruccions en pantalla.
4. Torneu a iniciar el vostre ordinador.

C.5 Configuració de la xarxa

A les xarxes amb *routers* sense fils, aquests s'encarreguen de coordinar les comunicacions entre els ordinadors que hi estan connectats. A les xarxes *ad hoc* cal designar un ordinador per a què actuï com a tal.

C.5.1 Configurar el primer ordinador

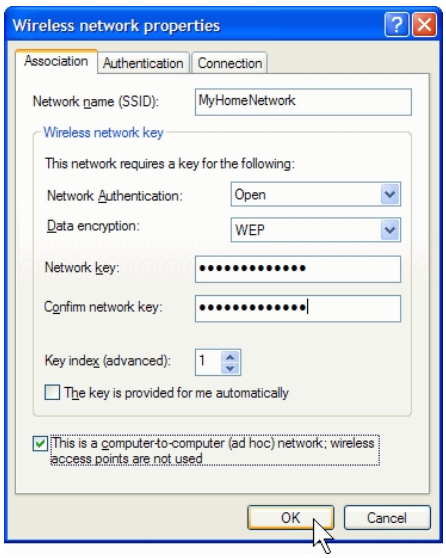
1. Feu clic a *Inici* i, a continuació, a *Panell de control*.
2. A la finestra d'*Escolliu una categoria*, seleccioneu *Treball en xarxa i connexions a Internet*.
3. Feu clic amb el botó dret a la vostra connexió a la xarxa sense fils i seleccioneu *Propietats*.

4. Al diàleg *Propietats de la connexió a la xarxa sense fils*, feu clic a la pestanya *Xarxes sense fils*.
5. Dins la pestanya *Xarxes sense fils*, a la zona de *Xarxes preferides*, feu clic a *Afegeix*.

6. A la finestra *Propietats de la xarxa sense fils*, a la pestanya anomenada *Associació*, escriviu el no de la vostra xarxa *ad hoc* al camp *Nom de la xarxa (SSID)*.

7. Deixeu sense marcar la casella *La clau és proporcionada per mi automàticament* i seleccioneu *Això és una xarxa d'ordinador - a - ordinador (*ad hoc*)*.

8. Creeu una contrasenya de 13 dígits i escriviu-la als camps *Contrasenya de la xarxa* i *Confirmeu la contrasenya de la xarxa*. Per a augmentar la seguretat, utilitzeu lletres, nombres i símbols de puntuació. Finalment, feu clic a *D'acord*.



9. Feu novament clic a *D'acord* per a desar els canvis.

C.5.2 Configurar ordinadors addicionals

1. Feu clic amb el botó dret a la icona *Xarxes sense fils* que hi ha a la part inferior dreta de la vostra pantalla, i llavors seleccioneu *Mostra les xarxes sense fils disponibles*.

2. La finestra de *Connexió a xarxa sense fils* apareix i mostra la xarxa sense fils que heu configurat prèviament amb el nom (SSID) que haguéssiu escollit. Si no la veieu, feu clic a *Refresca la llista de xarxes*, a l'extrem superior esquerre. Feu clic a la vostra xarxa i, tot seguit, al botó *Connecta* de la cantonada inferior dreta.

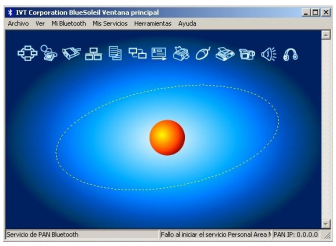
3. El sistema us demana una contrasenya. Escriviu als dos camps la contrasenya que abans heu creat i feu clic a *Connecta*.

Un cop connectats, podeu tancar la finestra *Connexió a xarxa sense fils*.
Repetiu aquests tres passos a cada ordinador que vulgueu connectar a la xarxa *ad hoc*.

C.6 Sincronitzar un *Wiimote®*

El primer que s'ha de fer per a sincronitzar un *Wiimote®* amb l'ordinador és iniciar el programa *BlueSoleilImatges obtingudes a [CP].*:

1. Feu clic amb el botó dret del ratolí a la icona de *Bluetooth* que trobareu a la safata de sistema de la barra d'inici (part inferior dreta de la pantalla).
2. Seleccioneu l'opció *Mostra vista clàssica.*
3. Una finestra com la següent apareix:



Un cop iniciat el programa, podeu agafar el comandament i sincronitzar-lo:

1. Traieu la tapa de les piles del *Wiimote®* i premeu el botó vermell.
2. Mentre les llums del *Wiimote®* parpellegin, feu doble clic a l'esfera taronja de la finestra principal de *BlueSoleil.*

3. El sistema ha reconegut tots els dispositius *Bluetooth.* En concret, el *Nintendo RVL-CNT-01*, que és el *Wiimote®*.
4. Tornem a prémer el botó vermell del comandament.
5. Amb el botó dret del ratolí, feu clic sobre el dibuix del *Wiimote®* i escolliu l'opció *Connecta* i, tot seguit, *Servei de dispositiu d'interfície humana de* Bluetooth.
6. Després d'això, el comandament ja està sincronitzat i podem posar-hi novament la tapa. El resultat és el que veiem a continuació.

C.7 Instal·lació i configuració de l'aplicació d'usuari

Per tal d'instal·lar l'aplicació d'usuari només cal que copieu el contingut del CD ubicat a `pfc/part2` a algun directori del vostre sistema.

Tot seguit, heu d'editar l'arxiu de configuració `common-config.txt`[1], de tal manera que les ubicacions que s'hi especifiquen puguin ser escrites pel mòdul de calibratge i llegides per l'aplicació d'usuari.

C.8 Instal·lació i configuració del mòdul de calibratge

Per tal d'instal·lar el mòdul de calibratge només cal que copieu els continguts del directori `pfc/part1/bin`. Això inclou els arxius `master.bat`, `lworker.bat` i `rworker.bat`, que serveixen per a iniciar el programa, així com `pfc-calibrator.jar` i `default-configuration.xml`.

El fitxer que haureu d'editar per tal de què el mòdul de calibratge es vinculi correctament a l'aplicació d'usuari és `default-configuration.xml`. Aquest conté una configuració per defecte del mòdul que funciona correctament, però caldrà que editeu el següent camp

```
<param name="common-config-file" value="C:\dp\common-config.txt" />
```

per a què el fitxer apunti al que hi ha a l'arrel de l'aplicació d'usuari.

[1] Vegeu l'annex A.

APÈNDIX D

Manual d'usuari

En aquest annex s'explica com utilitzar el sistema: per una banda, s'ensenyen els passos que cal seguir per a calibrar-lo i fer que els resultats siguin accessibles per l'aplicació d'usuari; per l'altra, es comenten les principals opcions que es poden activar de l'aplicació d'usuari.

Recordeu que tots els mòduls han d'estar connectats a la mateixa xarxa sense fils, tal i com s'explica a l'annex C.5.

D.1 Mòdul de calibratge

Per a utilitzar el mòdul de calibratge és necessari que inicieu, primer, el *master* i, tot seguit, hi connecteu tots els *workers*. Quan el sistema tingui registrats tots els mòduls, podreu començar amb el procés de calibratge.

D.1.1 Iniciar el *master*

Per tal d'iniciar el *master*, obriu un terminal:

1. Feu clic a *Inici* i, tot seguit, escolliu *Executa*.
2. Al camp *Obra*, escriviu `cmd`.
3. S'obra una finestra amb un terminal.

Tot seguit, heu d'accedir al directori on estigui instal·lat el mòdul de calibratge i executar la següent comanda:

```
C:\pfc\calibratge\> master.bat
```

D'aquesta manera, el sistema ens mostrarà el següent menú:

```
AVAILABLE OPTIONS
--Misc options--
0.- Quits
```

```
1.- Print available options
2.- Print number of registered workers

--Load/save data--
3.- Load new configuration file
4.- Load a previously computed geoalign result
5.- Load previously computed chromatic masks
6.- Ask all clients to save their data (homographies, masks, etc.)

--Geometric alignment tasks--
7.- Compute darkener factors for each client
8.- Geometric alignment
9.- Fix final display manually

--Chromatic alignment tasks--
10.- Chromatic alignment

--Testing tasks--
11.- Send an image to the projectors (geo/chrom testing)
12.- Paint projectors' contours
13.- (Dummy task)
================================================================

>> Enter an option:
```

Conèixer la IP del *master*

Per tal de poder iniciar els *workers* necessitareu conèixer l'adreça IP del *master*, ja que la necessiten per connectar-s'hi. Executeu la següent comanda en un terminal i fixeu-vos en el valor de la línia *Adreça IP*:

```
C:\> ipconfig
```

D.1.2 Iniciar un *worker*

Per tal d'iniciar un *worker* obriu un terminal i dirigiu-vos al directori on heu instal·lat el mòdul de calibratge (tal i com es descriu anteriorment). Aleshores, executeu la comanda

```
C:\pfc\calibratge\> lworker
```

si el *master* i el *worker* són al mateix ordinador, o

```
C:\pfc\calibratge\> rworker 192.168.1.1
```

si estan en ordinadors diferents. En el segon cas, heu de canviar `192.168.1.1` per l'adreça IP que tingui realment el *master*.

D.1.3 Modificar la configuració

A l'apartat d'instal·lació s'explica com modificar el fitxer `default-configuration.xml` per tal de poder utilitzar l'aplicació. El primer que cal fer quan s'hagi iniciat el *master* i hi hagi tots els *workers* connectats és carregar aquesta nova configuració:

1. Seleccioneu l'opció 3, `Load new configuration file`.
2. Indiqueu la ubicació del nou fitxer.

D.1.4 Realitzar un calibratge geomètric

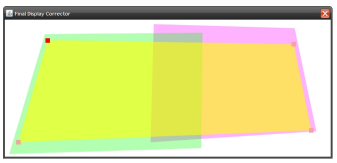
1. Seleccioneu l'opció 8, `Geometric alignment`.
2. Les pantalles de tots els ordinadors es posaran en negre i la projecció de patrons s'iniciarà. Cada vegada que un mòdul acabi de projectar el seu patró, tots els mòduls intentaran calcular les respectives homografies.
3. Quan el calibratge acabi, el sistema us preguntarà on voleu desar el resultats obtinguts (per si voleu recuperar-los més endavant). Indiqueu-li una ubicació que existeixi i un nom de fitxer qualsevol, com per exemple:

```
> Enter a filename: C:\tmp\geoalign.data
```

D.1.5 Corregir la pantalla final

Després d'haver dut a terme un calibratge geomètric *satisfactori*:

1. Seleccioneu l'opció 9, `Fix final display manually`.
2. El sistema projecta la pantalla final que hi ha calculada en aquell moment pels projectors, per tal de què pugueu veure com queda. El *master* també mostra una finestra com la següent, que permet modificar-ne les quatre cantonades.



3. Quan el resultat que veieu projectat sigui satisfactori, tanqueu la finestra per a donar per finalitzada la correcció manual.
4. El sistema us demanarà que indiqueu on desar els resultats. Indiqueu-li una ubicació que existeixi i un nom de fitxer qualsevol, com per exemple:

```
> Enter a filename: C:\tmp\geoalign-ok.data
```

D.1.6 Realitzar un calibratge cromàtic

Després d'haver dut a terme un calibratge geomètric *satisfactori*:

1. Seleccioneu l'opció 10, `Chromatic alignment`.
2. El sistema comença el procés de correcció cromàtica. En funció dels paràmetres de configuració, és possible que projecti o no algun patró.
3. Quan el sistema acabi de realitzar els càlculs us demanarà que indiqueu on desar els resultats. Indiqueu-li una ubicació que existeixi i un nom de fitxer qualsevol, com per exemple:

```
> Enter a filename: C:\tmp\chrom.data
```

D.1.7 Obrir resultats

Tal i com s'explica als passos anteriors, després de dur a terme cadascuna de les etapes de calibratge és possible desar-ne els resultats. El sistema ens permet recuperar-los en qualsevol moment:

- Per tal de recuperar una *alineació geomètrica*, seleccioneu l'opció 4, `Load a previously computed geoalign result`, i indiqueu el nom del fitxer.
- Per tal de recuperar una *correcció cromàtica*, seleccioneu l'opció 5, `Load previously computed chromatic masks`, i indiqueu el nom del fitxer.

D.1.8 Desar els resultats d'intercanvi

Per tal de desar els resultats obtinguts durant els processos de calibratge en un format que l'aplicació d'usuari entengui, i a les ubicacions especificades pel fitxer `common-config.txt`, escolliu l'opció 6, `Ask all clients to save their data (homographies, masks, etc.)`.

D.1.9 Sortir de l'aplicació

Per tal de sortir de l'aplicació, seleccioneu l'opció 0, `Quit`.

D.2 Aplicació d'usuari

En aquesta secció s'explica amb cert detall les funcionalitats que s'han afegit a l'aplicació [CL]. Les que s'han heretat, en canvi, es mostren breument en una llista.

D.2.1 Habilitar l'ús de les homografies

Cada mòdul disposa de la descripció total del món virtual. Això vol dir que poden dibuixar l'escena individualment i mostrar-la correctament.

Si es vol veure l'escena a través dels projectors, cal habilitar les homografies amb la tecla `[E]` per tal de què el que mostra cadascun d'ells sigui la porció de món que li pertoca.

D.2.2 Habilitar l'ús de les màscares

Per a poder aplicar les màscares cromàtiques s'ha de prémer la tecla `[M]`.

D.2.3 Reiniciar la posició central

Tot i que aquesta funcionalitat ja estava present a [CL], s'explica apart degut a la seva importància. Quan l'usuari s'ha situat a una posició que ell considera "natural" i que es correspon al "centre" del món virtual (mirant, òbviament, a la pantalla), ha de prémer la tecla *Espai* per a què la càmera virtual se situï a la posició inicial.

D.2.4 Altres funcionalitats

El sistema té altres funcionalitats que es poden activar i desactivar:

`[G]` Mostra o amaga la graella.

`[H]` Mostra o amaga estadístiques del sistema.

[L] Mostra o amaga les línies que van dels objectius a la paret del fons.

[R] Reubica els objectius.

[S] Mostra o amaga estadístiques relacionades amb el *Wiimote*®.

[T] Mostra o amaga els objectius.

APÈNDIX E

Resultats

Aquest annex mostra els diferents resultats que s'han obtingut tant al mòdul de calibratge com a l'aplicació d'usuari. Les imatges que hi ha pertanyen a execucions reals del sistema.

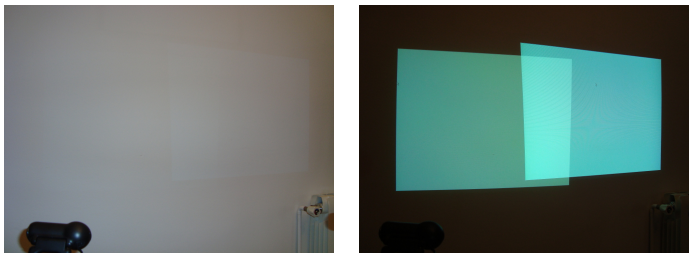
E.1 Mòdul de calibratge

A continuació ens centrarem en els resultats que dóna el mòdul de calibratge. En concret, es mostren els diversos passos que realitza l'algorisme amb el resultat que ofereixen cadascun d'ells.

Per tal de veure el procés de calibratge en funcionament, podeu mirar el vídeo que trobareu inclòs al CD adjunt a aquesta memòria.

E.1.1 Els patrons

La manera que té el sistema per tal de conèixer la disposició de les projeccions sobre la paret és enviant patrons i analitzant-los. La figura E.1.1 mostra la paret on es projectaran les àrees (E.1(a)) i la zona que ocupa cadascuna d'elles (E.1(b)).



(a) Paret on es projectarà el món virtual.

(b) Àrees ocupades pels projectors.

Figura E.1: Zona de projecció.

Després de què tots els mòduls hagin capturat el fons, s'inicia l'enviament i captura de patrons[1], com el que podeu veure a la captura de la figura E.1.1.

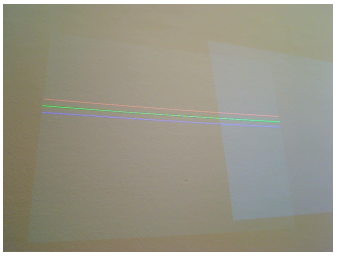
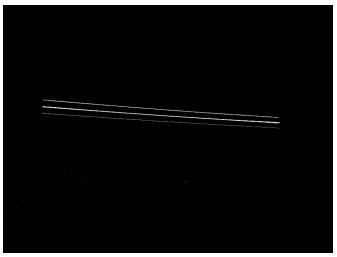


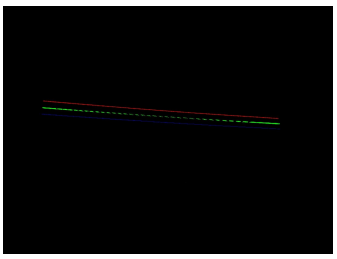
Figura E.2: Patró capturat per un dels mòduls.

Quan es disposa d'aquestes captures, s'apliquen els algorismes de visió per computador que permeten obtenir la informació dels patrons de les imatges (vegeu la figura E.1.1.

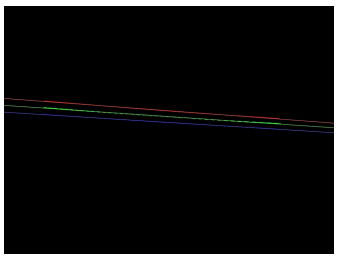
(a) Imatge filtrada fent la diferència entre la captura i el fons.



(b) Imatge filtrada aplicant una binarització a E.3(a).



(c) Imatge filtrada on es determina el color de cada línia.



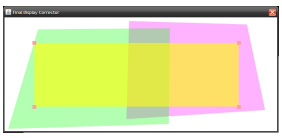
(d) Imatge filtrada on s'hi ha superposat la recta calculada analíticament, després d'aplicar un algorisme de regressió.

Figura E.3: Evolució del processat d'una captura a l'aplicar-li els diversos filtres.

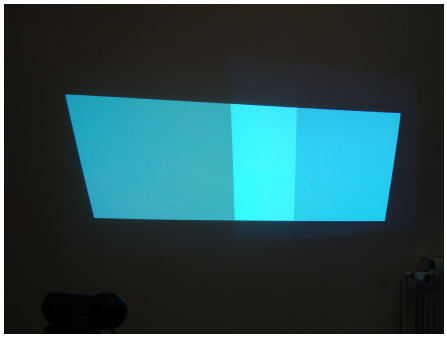
E.1.2 La pantalla final

El resultat d'haver processat els patrons és disposar de la informació sobre les àrees projectades, de tal manera que es pugui calcular la pantalla final. La figura E.4 mostra la pantalla final que s'ha calculat automàticament, la qual es veu com a un rectangle des de la càmera del mòdul 1.

[1] És important recordar que, de fet, el que s'envia i es captura són *components d'un patró*.



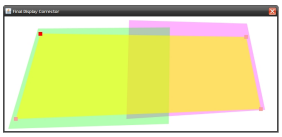
(a)



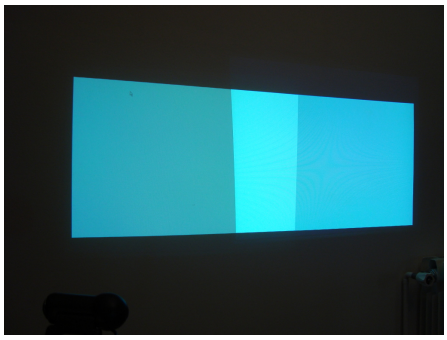
(b)

Figura E.4: Pantalla final calculada automàticament.

Si l'usuari modifica manualment les coordenades d'aquesta pantalla, podrà alinear-la amb el terra i les columnes.



(a)



(b)

Figura E.5: Pantalla final corregida manualment per l'usuari.

E.1.3 Correcció cromàtica

Finalment, quan ja es té calculada la geometria de la pantalla final, es poden calcular les màscares que faran la correcció cromàtica.

La figura següent mostra els diferents resultats que s'obtenen a l'aplicar les homografies sobre una imatge qualsevol i projectar-la, en funció de la màscara que se l'hi apliqui.

(a) No s'aplica cap màscara. (b) S'aplica la màscara amb degradats. (c) S'aplica la màscara amb degradats corregits.

Figura E.6: Resultats obtinguts amb el procés de calibratge, aplicant diferents màscares per a la correcció cromàtica.

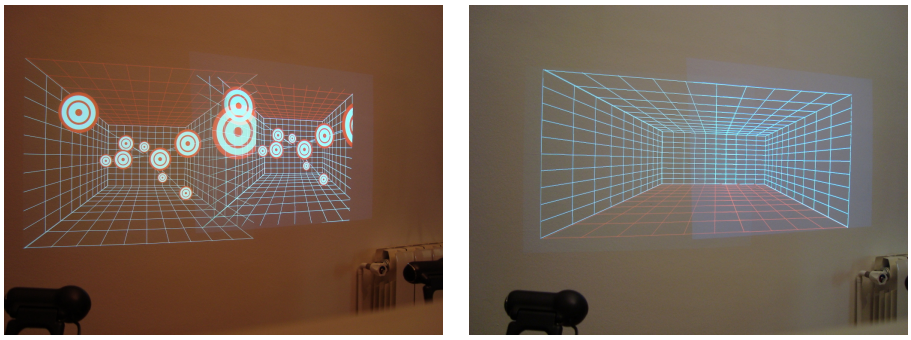
La figura E.6(b) posa de manifest el problema descrit a la secció 7.5.4. Després de corregir-ho, el resultat final és el de la figura E.6(c).

E.2 Aplicació d'usuari

Tot seguit, es mostren unes quantes captures dels resultats obtinguts a l'aplicació d'usuari. De totes maneres, en aquest cas es recomana veure el vídeo contingut al CD adjunt a la memòria, el qual permet comprendre molt millor el funcionament d'aquesta part del projecte.

E.2.1 Homografies

L'aplicació d'usuari permet activar o desactivar l'ús de les homografies. Tal i com es pot apreciar a la figura E.2.1, el sistema només és coherent si estan habilitades, de tal manera allò que es mostra és coherent.

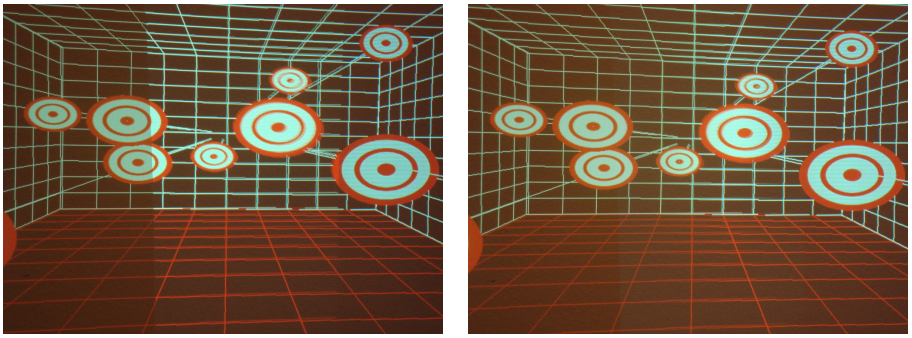


(a) (b)

Figura E.7: Exemple de l'aplicació d'usuari utilitzant o no les homografies.

E.2.2 Màscares

L'aplicació també permet habilitar i deshabilitar les màscares. Utilitzar-les és aconsellable, ja que aconsegueix dissimular els petits errors de calibratge que s'hagin pogut produir. La següent figura mostra ambdós resultats:



(a) (b)

Figura E.8: Exemple de l'aplicació d'usuari amb o sense màscares.

Bibliografia

[BIWG08] Oliver Bimber, Daisuke Iwai, Gordon Wetzstein, and Anselm Grundhöfer, *The Visual Computing of Projector-camera Systems*, SIGGRAPH '08: ACM SIGGRAPH 2008 classes (New York, NY, USA), ACM, 2008, pp. 1–25.

[BRR+02] Jeroen Van Baar, Ramesh Raskar, Ramesh Raskar, Jeroen Baar, Jin Xiang Chai, and Jin Xiang Chai, *A Low-Cost Projector Mosaic with Fast Registration*, in Proceedings of Asian Conference on Computer Vision (ACCV), Inc, 2002, pp. 114–119.

[BV99] Volker Blanz and Thomas Vetter, *A Morphable Model for the Synthesis of 3d Faces*, SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques (New York, NY, USA), ACM, 1999, pp. 187–194.

[CL] Johnny Chung Lee, *Head Tracking for Desktop VR Displays using the Wii Remote*, http://www.cs.cmu.edu/ johnny/projects/wii/.

[CP] *Usar el Wiimote en el PC*, http://en.wikipedia.org/wiki/Adapter_pattern.

[CSWL02] Han Chen, Rahul Sukthankar, Grant Wallace, and Kai Li, *Scalable Alignment of Large-Format Multi-Projector Displays Using Camera Homography Trees*, VIS '02: Proceedings of the conference on Visualization '02 (Washington, DC, USA), IEEE Computer Society, 2002, pp. 339–346.

[DCK03] Rodrigo De Castro Korgi, *El universo LATEX, Segunda Edición*, Panamericana Formas e Impresos S.A., 2003.

[IJT] *InfoJobs Trends Salarios*, http://salarios.infojobs.net/.

[JMP] *JAMA: Java Matrix Package, Online JavaDoc*, http://math.nist.gov/javanumerics/jama/doc/.

[JV] *JavaVis: A Computer Vision Library in Java, Online JavaDoc*, http://javavis. sourceforge.net/docs/javadoc/index.html.

[Lat] *Latex Documentation*, http://www.latex-project.org/guides/.

[Oli07] Antoni Olivé, *Conceptual Modelling of Information Systems*, Springer, 2007.

[OPHS] T. Oetiket, H. Partl, Hyna, and E. Schlegl, *The not-so-short Introduction to Latex*.

[PL] Rodrigo Pizarro Lozano, *Sistema multi-projector per a realitat virtual*, Master's thesis.

[RJB04] James Rumbaugh, Ivar Jacobson, and Grady Booch, *The Unified Modelling Language: Reference Manual, Second Edition*, Addison-Wesley, July 2004.

[Vol] *Volere: Requirements Specification Template*, http://www.volere.co.uk/rst.htm.

[VP] *Visual Paradigm for UML*, http://www.visual-paradigm.com/product/vpuml/.

[WK03] J. Warmer and A. Kleppe, *The Object Constraint Language: Precise Modelling with UML, Second Edition*, Addison-Wesley, 2003.

[WP] *Adapter Pattern*, http://en.wikipedia.org/wiki/Adapter_pattern.

Glossari de termes

Amazon	Companyia nord-americana de comerç electrònic.
Bluetooth	Especificació industrial per a les PAN (*Personal Area Network*, Xarxes d'àmbit personal) sense fils, que serveix per a connectar dispositius a una distància pròxima.
CHG	*Camera Homography Graph*, Graf d'homografies de càmera.
CORBA	*Common Object Request Broker Architecture*, Arquitectura comú d'intermediaris per a peticions a objectes.
Haptic	Es refereix a què estimula el sentit del tacte (per exemple, *Haptic device*, dispositiu tàctil).
HLSL	*High Level Shader Language*, Llenguatge de *shading* propietat de Microsoft.
LED	*Light Emisor Diode*, Díodes emissors de llum.
MDA	*Model-Driven Architecture*, Arquitectura dirigida per models.
OCL	*Object Constraint Language*, Llenguatge de restriccions d'objectes.
OMG	*Object Management Group*.
Pixel	*Picture element*, mínima unitat homogènia en color que forma part d'una imatge digital.
RGB	Components de color vermell, verd i blau, de les seves sigles en anglès.
RMI	*Remote Method Invocation*, Invocació de mètodes remots.
Router	Dispositiu per a la interconnexió en xarxa d'ordinadors, el qual pot comunicar més d'una xarxa.
Shader	Conjunt d'instruccions gràfiques que s'executen a la targeta gràfica.
SOAP	*Simple Object Access Protocol*, Protocol d'accés a objectes simple.
SRS	*Software Requirements Specification*, anàlisi de requisits.

Switch	Dispositiu per a la interconnexió en xarxa d'ordinadors.
UDP	*User Datagram Protocol*, Protocol de datagrames d'usuari.
UML	*Unified Modelling Language*, Llenguatge de modelatge unificat.
VA	Visualització avançada.
VIG	Visualització i interacció gràfica.
Wrapper	Literalment, "embolcall". En computació, se sol utilitzar per a emmascarar un fragment de codi, de tal manera que les crides que s'hi fan siguin a través d'una interfície coneguda.
XML	*eXtensible Markup Language*, Llenguatge de marques extensible.