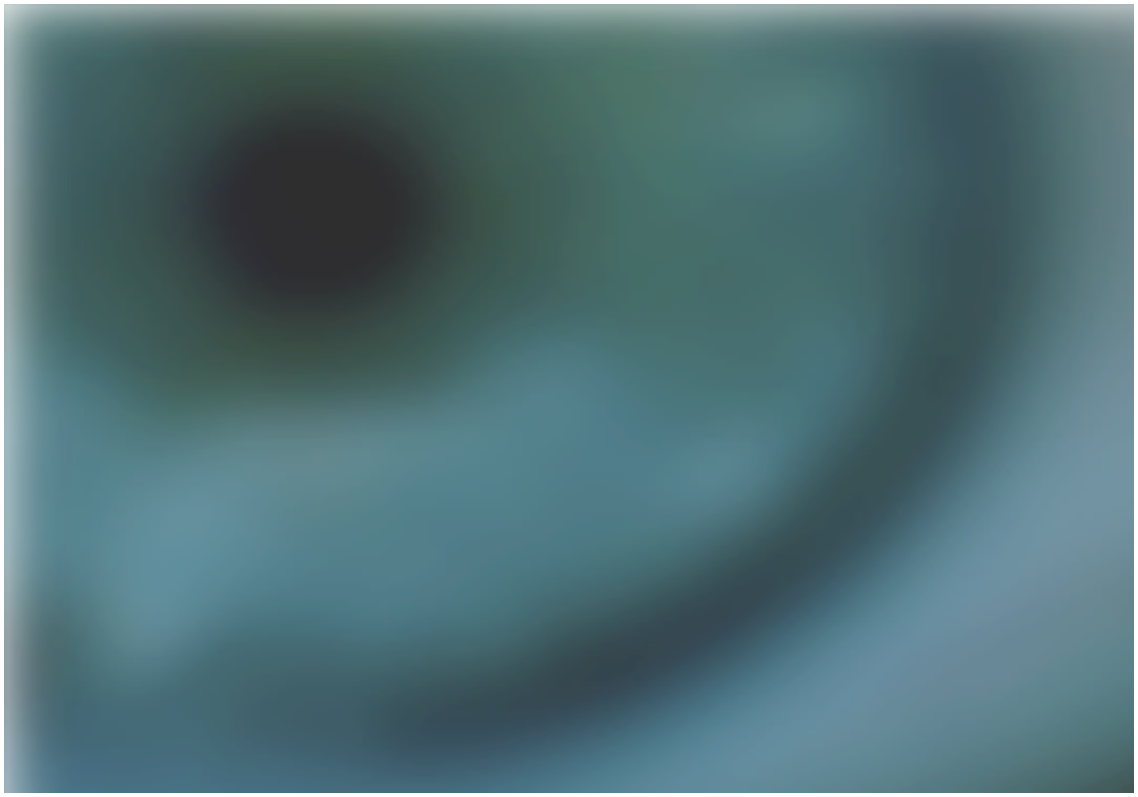
Universitat Oberta de Catalunya
www.uoc.edu

PFC – Visió per Computador



Sistema automàtic per a fisioterapeutes.

Autor: Oscar Sánchez Montaner

Enginyeria Informàtica.

Consultors: Sergio Escalera

José Ramírez

11 de Juny de 2010

... *gracias Papá* i gracies Mama per confiar sempre en mi!

"*Daisy, Daisy, give me your answer, do...*" anava dient HAL 9000 mentre moria a la pel·lícula *2001: A Space Odyssey*. Un dels defectes o virtuts dels éssers humans ha estat la capacitat d'humanitzar el que ens envolta. Des dels límits de la ciència ficció, l'Arthur C. Clarke ens va portar un computador amb una consciència de si mateix i pròpia amb la capacitat de percebre els seu entorn a través d'unes càmeres. Ja fa uns anys que hem traspassat el límit de la frontera de la ciència ficció i ja podem trobar al nostre voltant aplicacions informàtiques que capten dades a través de càmeres (en aparcaments, seguiment de vehicles, control de qualitat, reconeixement de senyals de limitació de velocitat, etc.). S'intueix un camp fascinant, ple d'oportunitats i és en aquest punt quan s'obren més camins per recórrer en aquesta matèria; és per això que vaig decidir escollir un d'aquests camins per realitzar el meu projecte de final de carrera.

La proposta del projecte es centra en un sistema automàtic de diagnòstic per a fisioterapeutes. En aquest cas, la tècnica, per mitjà de la visió per computador, ajudarà a la ciència mèdica a realitzar diagnòstics més acurats i de forma més senzilla. Aquest punt és molt important per a mi, ja que en la recta final dels estudis cursats en enginyeria informàtica, aplicaré tots els coneixements adquirits, i els que adquiriré, en una aplicació que un dia pot arribar a ser determinant en la fisioteràpia i per extensió en els pacients que necessitin d'un diagnòstic fiable, ràpid i acurat.

1 Dedicatòries i agraïments

Fa una mica més de deu anys vaig començar a traçar el mapa de la meva vida, amb el suport incondicional del meu pare Julián, la meva mare Lydia i el meu germà Héctor. Vaig començar a caminar. La primera estació em va portar a les portes del món de la programació. Tot era nou, tot per descobrir, la passió pel estudi em va embriagar i gràcies als magnífics professors de l'Escola del Treball vaig aprendre gran part del que avui sé. L'esponja del coneixement – tant seca en altres temps – volia més, necessitava més. Era l'hora de seguir caminant. Les passes em van portar a Vilanova. Tenia un peu a la Universitat i tot un univers per descobrir. El camí no va ser fàcil; volia estudiar i volia treballar, volia aplicar el que sabia i aprendre del que feia. Llargues nits! Gràcies Neus per confiar en mi. On ets caminant? Somiant no, però sí pensant en què podia fer jo, d'útil per la societat; un navegador per invidents. Dos anys de treball i al final *El Navegador Lleuger, amb finalitat Acadèmica, per a Persones Invidents* va veure la llum. Oscar, no has arribat fins aquí per aturar-te, i un altre cop vaig començar a caminar. La Universitat, des del ciberespai, em tornava a donar la benvinguda amb nous reptes, més coneixement i més dubtes. Gràcies Sergio per oferir-me aquest projecte perquè el dia que el vaig veure, ja sabia que gaudiria molt aprenent a "posar ulls" a un programa. On sóc avui? No ho sé, potser encara estic al començament del camí, però no m'importa perquè no estic sol caminant; Gràcies Ángela per estar sempre amb mi, per les hores corregint aquesta memòria al meu costat, per deixar l'escriptori ple de *Post-its* amb paraules d'ànims i d'amor... Mil gracies per tot!

"Caminante no hay camino, se hace camino al andar" Antonio Machado.

2 Resum

El projecte "*Sistema automàtic per a fisioterapeutes*" neix a proposta del consultor de l'àrea de Visió per Computador. El projecte té com a finalitat dissenyar i implementar un sistema de reconeixement automàtic en una imatge d'indicadors situats específicament en el cos d'una persona, mitjançant tècniques de visió per computador. Un cop identificats els indicadors d'una determinada posició corporal, anomenada vista, cal realitzar càlculs de distàncies i angles que seràn el resultat del càlcul automàtic.

Per a la realització del projecte s'ha fet servir el llenguatge de programació C++, la llibreria OpenCV 2.1 i el *framework* Qt en la seva versió 4.6 per a la implementació de la interfície gràfica. Així, l'aplicació resultant pot ser modificada, compilada i executada en diferents sistemes operatius. Concretament, el desenvolupament de l'aplicació s'ha realitzat indistintament en un entorn Microsoft Windows XP i en un entorn Mac OS X (*Snow Leopard*) d'Apple.

The project "*Automatic system for physiotherapists*" started as a proposal of the consultant in the Computer Vision area and aims the design and implementation of a system to recognize automatically in an image, using computer vision techniques, a set of marks specifically located in the body of a person. Once the marks of a specific body shape, called view, are recognized has to be calculated distances and angles that will become the result of the automatic calculus. Once the marks of a specific body shape (called view) are recognised, the distances and angles, that will become the result of the automatic calculus, have to be calculated.

To achieve the project has been used the programming language C++, the OpenCV 2.1 library and the Qt framework, version 4.6, for the graphic interface implementation. Thus, the application can be modified, compiled and executed on different operative systems. In particular, the development of the application has been carried out using Microsoft Windows XP or Mac OS X (*Snow Leopard*) from Apple.

2.1 Paraules clau

Visió	Computador	C++	OpenCV	Qt
Diagnòstic	Automàtic	Detecció	Punts	Fisioteràpia

3 Índex

4 Introducció

4.1 Justificació

La rellevància d'aquest projecte radica en el fet de què actualment no hi ha disponible cap aplicació que realitzi la cerca de marcadors i que faci els càlculs de forma automàtica. Com a segona fase, una apliació que obté dades automàticament és una excel·lent candidata per a dotar-la d'un motor d'inferència, utilitzant tècniques d'intel·ligència artificial, amb la finalitat de realitzar auto-diagnòstics. Per tant, aquest projecte fa la primera passa en la direcció de creació d'una eïna que pot donar un tomb en el diagnòstic clínic en l'àmbit fisioterapeutic.

4.2 Objectius

L'objectiu del projecte és donar solució a una problemàtica concreta com és l'adquisició automàtica de dades sobre diferents punts ubicats en el cos humà, per l'ajut al diagnòstic de malalties i dolències, en l'àmbit fisioterapeutic.

Actualment es treballa sobre una imatge d'un pacient, a qui prèviament se li han col·locat marcadors de colors. Posteriorment, amb un software especialitzat es mesuren certs angles i certes distàncies. L'objectiu del projecte és automatitzar aquesta tasca fent servir tècniques de visió per computador.

4.3 Metodologia

La metodologia seguida pel desenvolupament d'aquest projecte es pot resumir tot indicant les següents fases de treball:

1. La primera fase del projecte va consistir en la instal·lació de l'entorn de treball.

   La gestió del projecte s'ha realitzat amb Unfuddle[L01], companyia que ofereix una solució allotjada per la gestió de projectes per equips de desenvolupament de programari, amb la posibilitat de crear repositoris Subversion[L02] o Git. El nom del projecte creat a Unfuddle és: ADiBAS (***A**utomatic **Di**gital **B**iometry **A**nalysis **S**ystem*).

2. A la segona fase del projecte es va crear un repositori – ADiBAS[L03] – per a realitzar les diferents proves amb OpenCV. És en aquesta fase quan neix el disseny i la implementació de les classes encarregades de l'encapsulació dels mètodes aplicats amb OpenCV i les classes auxiliars a aquestes.

3. La fase final del projecte inclou la creació del repositori – QtADiBAS[L04] – a fi de portar a terme, d'una banda, la integració de les classes del punt anterior i, d'altra banda, el disseny i desenvolupament de l'interfície gràfica d'usuari amb Qt. En aquest punt també s'inclouen les proves funcionals de l'aplicació, la documentació (manual d'usuari, manual d'implementació, etc.) i la redacció d'aquesta memòria.

4.4 Planificació

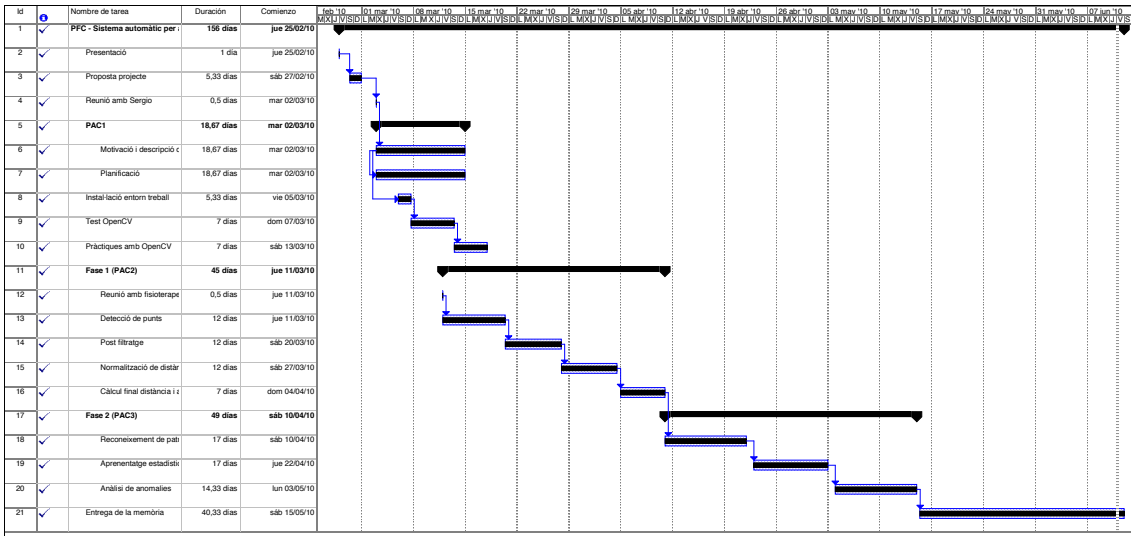


Diagrama 1

4.5 Entorn de treball

Com personalment jo treballo amb Mac OS X (*Snow Leopard*) i els fisioterapeutes (els usuaris) treballen amb Microsoft Windows, es va decidir, de forma conjunta amb el coordinador, treballar amb C++, OpenCV 2.1[1] i per la interfície gràfica d'usuari, es va optar per Qt 4.6[2]. Com a IDE de treball, sobre Mac OS X, s'ha fet servir *Eclipse* i *Designer*. A Windows s'ha fet servir *Creator*[3]. En ambdós entorns s'han utilitat eines com: *Linguist*, *Assistant*, etc. que formen part del SDK de Qt.

[1] OpenCV 2.1 és una llibrería que está disponible per Windows, Linux y Mac OS.
[2] Qt 4.6 *framework* multiplataforma pel desenvolupament de programari.
[3] *Creator* per a Mac OS X (*Snow Leopard*) – 64bits – no funciona.

4.6 Productes obtinguts

4.6.1 Disseny

Disseny d'un conjunt de classes per tractar la segmentació de color per a l'obtenció de marques, la seva gestió de marques vistes com a punts, i la gestió de punts vistos com a vista. També s'inclou el disseny de les classes que realitzaran la interoperabilitat entre OpenCV i Qt.

4.6.2 Aplicació

Implementació de les classes citades en el punt anterior i la utilització d'aquestes a fi d'implementar l'aplicació, amb nom, QtADiBAS.

5 Projecte

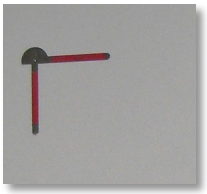
5.1 Anàlisi

5.1.1 Adquisició de dades

L'adquisició de les dades es fa sempre a partir d'una imatge del subjecte a diagnosticar en una determinada posició, anomenada vista. A aquesta persona se li hauran col·locat, de manera determinada, uns marcadors de colors. Cada vista té un nombre determinat de marques que s'han d'etiquetar pel seu tractament posterior (càlcul de distància, angle respecte la vertical, etc.). A la imatge de la dreta es pot veure la vista posterior d'una pacient amb els onze marcadors. Els marcadors solen ser de color verd perquè és un color fàcil de detectar (mitjançant la segmentació de colors) sobre la pell humana, de totes formes hi ha casos que es combinen aquests amb altres de diferents colors per diferenciar parts a tractar; però aquest cas no està inclós dins els objectius d'aquest projecte.

Imatge 1

Per tal de realitzar el càlcul correcte de les distàncies és imprescindible extrapolar les distàncies reals amb les distàncies que hi ha a la imatge. Per això les imatges incorporen un element del qual coneixem la seva distància real. Aquest element pot ser una escuadra en una cantonada de la imatge, una línia en la base on el subjecte està col·locat, etc. Ara per ara no s'ha trobat l'element més adient.



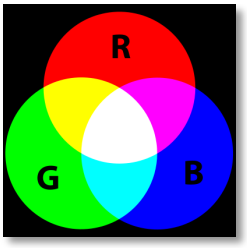
Imatge 2

Els marcadors col·locats a la pared són captats amb lleugeres deformacions per aberracions a les lents de la càmera o per la perspectiva (una imatge és el reflex en dues dimensions del món físic que té tres dimensions físiques). A la imatge superior es pot veure, a simple vista, com l'angle recte del marcador no s'ha captat correctament.

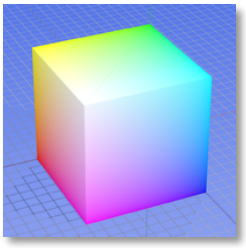
En conseqüència, atesos els problemes detectats amb els marcadors de paret, finalment es va concloure utilitzar una línia en la base del subjecte, malauradament aquesta decisió es va adoptar fa una setmana, moment en el que el projecte estava en la fase de proves; per tant aquesta funcionalitat es deixa com a possible millora del producte.

5.1.2 Espai de colors RGB

El model de color RGB és un model de color additiu a on el vermell, el verd i el blau es barregen de diverses formes per reproduir un ampli ventall de colors. El principal objectiu del model RGB és la detecció, representació i visualització d'imatges en sistemes electrònics. A la imatge 1 es pot veure la representació additiva, i a la imatge 2 es pot veure la representació geomètrica del model.



Imatge 3 [L05]



Imatge 4 [L06]

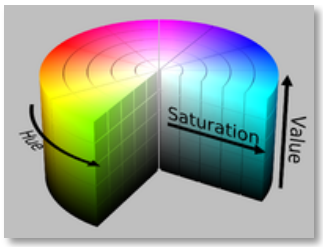
Un color, en el model RGB [L07], es representa numèricament amb un triplet:

rgb(vermell, verd, blau)

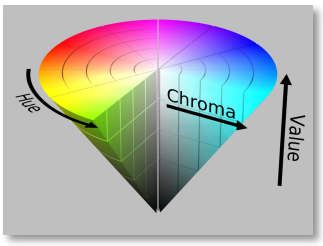
a on cada component indica la quantitat de color des de zero fins a un valor màxim. En computació és habitual representar cada canal amb 1 byte (8 bits), per tant el rang numèric a utilitzar serà de 0 fins a 255.

5.1.3 Model HSV

El model HSV, i també el HSL, són dues representacions dels punts RGB en coordenades cilíndriques en un intent de obtenir una representació més rellevant que la cartesiana. HSV significa *hue*, *saturation* i *value* (tonalitat, saturació i valor) i a la imatge 3 es pot veure la seva representació. La definició de saturació del model cilíndric entra en conflicte amb la noció intuïtiva de la puresa de color (on colors molt foscos, en HSV i HSL [L08], o massa clars en HSL són considerats totalment saturats). Per aquesta raó, és molt habitual substituir la dimensió saturació per una radial, en aquest cas anomenada *chroma*, donant lloc a una representació cònica, o bicònica en el cas de HSL (imatge 4).



Imatge 5 [L09]



Imatge 6 [L10]

La representació de la dimensió angular de la tonalitat (*hue*) comença al vermell pur en **0 graus**, passa pel verd pur en **120 graus**, el blau pur es troba a **240 graus** i d'aquí torna al vermell en **360 graus**.

La saturació es pot definir com la quantitat de gris i normalment s'expressa en termes de tant per cent, o dins el rang 0 – 1.

El valor es definiria com la quantitat de brillantor del color i varia amb la saturació. S'expressa en termes de tant per cent, o dins el rang 0 – 1 (quan el valor és 0 l'espai de color és totalment negre).

5.2 Casos d'ús

Després de rebre els requeriments d'usuari i analitzar les diferents possibilitats de disseny, el resultat és el següent diagrama de casos d'ús. Aquest diagrama és la base de tot el disseny posterior.

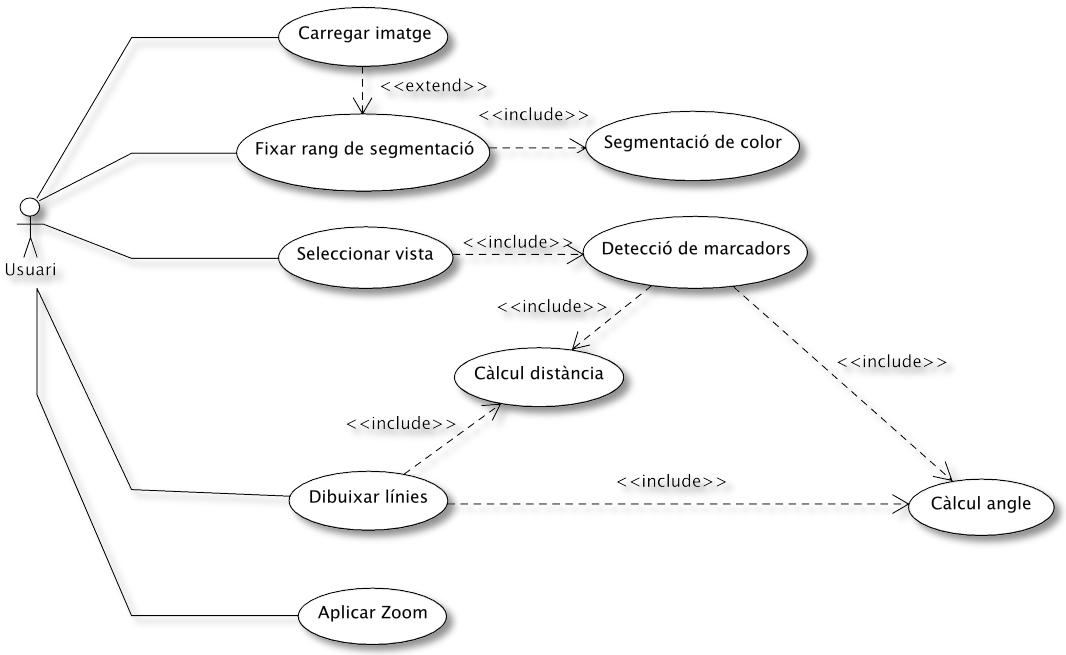


Diagrama 2

5.3 Disseny

5.3.1 Detecció de punts

Seguint les recomanacions d'en Sergio Escalera els esforços es varen centrar en la segmentació per color. Malauradament els esforços per aconseguir una bona segmentació en l'espai RGB van començar bé, però amb la segona tanda d'imatges de proves ("molt vermelles" per les condicions de llum) es va poder veure que el mètode no acabava de funcionar bé. Finalment es decideix continuar per la via de la segmentació de color, però aquest aquest cop a l'espai de color HSV.

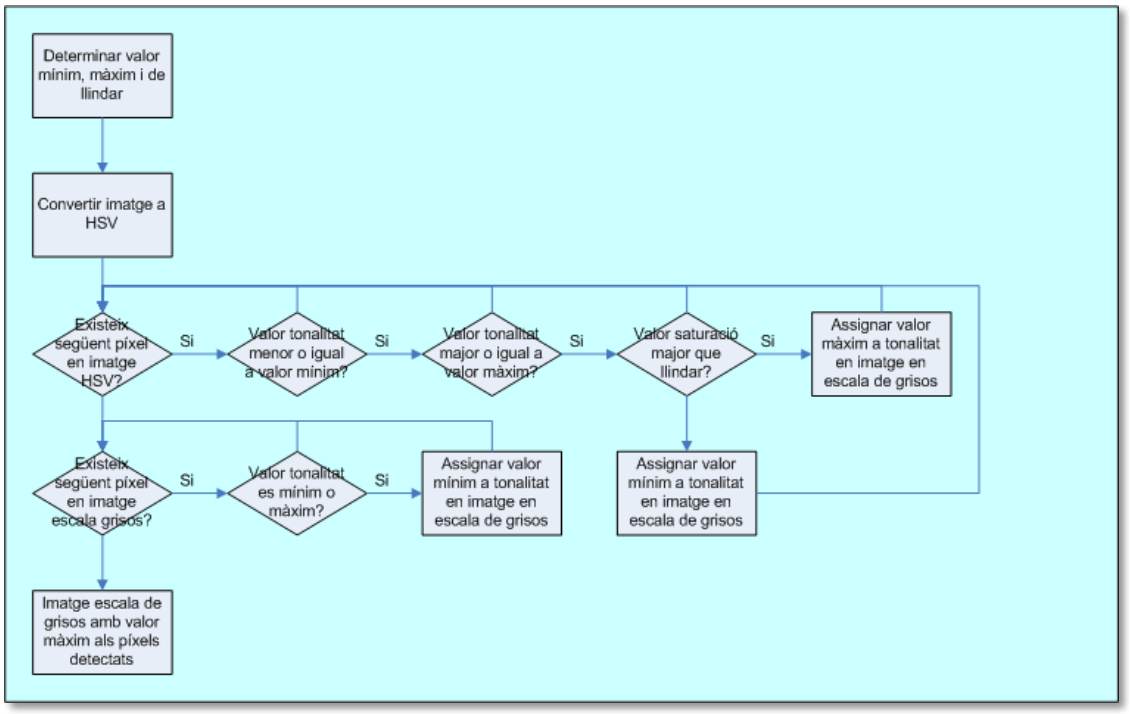


Diagrama 3

5.3.2 Classes

A continuació es pot veure el disseny de les classes que serviran de suport al tractament d'imatges (*Image*), marcadors (*Mark*), línia (*Line*), conjunt de marcadors (*Markers*) i vistes (*View*).

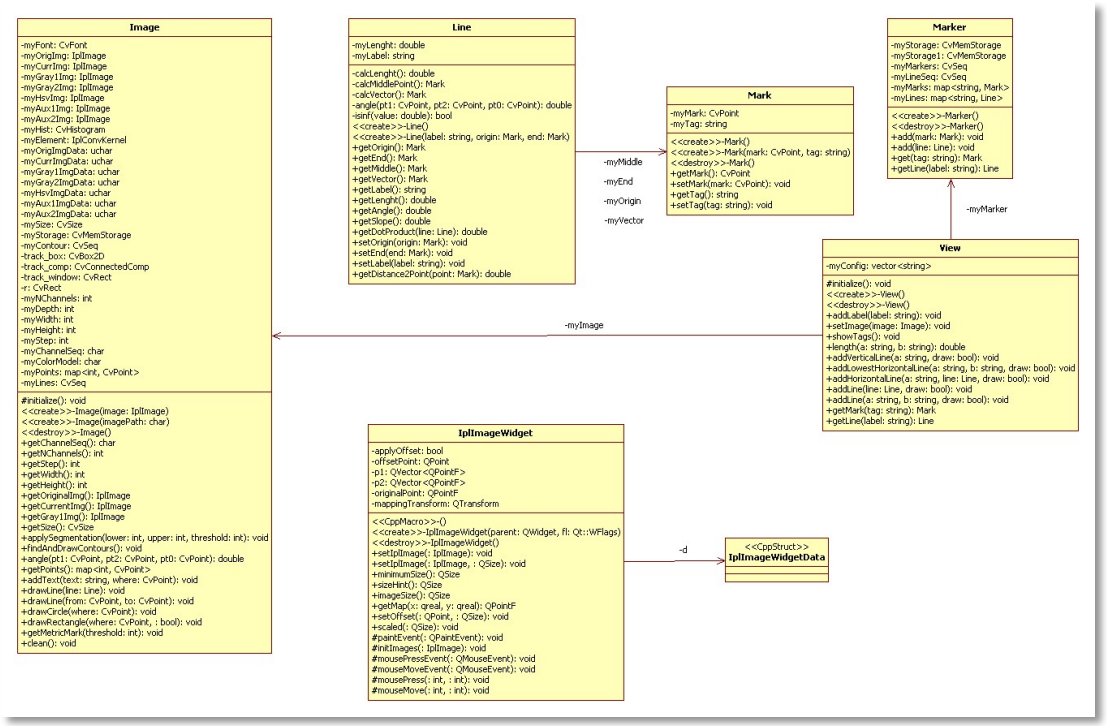


Diagrama 4

La classe *Mark* (Marca) representa un punt bidimensional i etiquetat en un plà cartesià, de dos eixos, que coincideix en tamany al tamany de la imatge original a processar, i té l'origen (0, 0) en la cantonada superior-esquerra.

La classe *Line* (Línia) representa una línia, en un espai bidimensional, etiquetada i sobre la qual es poden realitzar les següents operacions: càlcul de la distància, càlcul del pendent, l'angle respecte l'eix d'abscisses i l'angle respecte l'eix d'ordenades.

La classe *Marker* (Marcador) representa un conjunt de marques etiquetades i un conjunt de línies etiquetades.

La classe *View* (Vista) representa el marcador d'una imatge sobre la qual es poden realitzar les següents operacions: calcular la distància sobre dos etiquetes

(relacionades amb dos marques), dibuixar el conjunt d'etiquetes (de les marques) sobre l'imatge, dibuixar una línia vertical cap avall amb l'origen en una etiqueta (relacionada amb una marca).

La classe *Image* (Imatge) representa una imatge sobre la qual es poden aplicar els seguents mètodes: aplicar segmentació (utilitzant l'espai de color HSV), trobar i pintar contorns (s'aplica sempre després de realitzar la segmentació), calcular l'angle a la intersecció de dues línies (per realitzar el càlcul s'utilitzen tres punts), afegir un text, dibuixar una línia, dibuixar un cercle, dibuixar un rectangle i detectar la referència mètrica (en la cantonada superior esquerra).

La classe *IplImageWidget*[4] encapsula un *widget* Qt que transforma una instància *IplImage* (OpenCV) en una instància *QImage* (Qt). Aquesta classe està disponible per descarregar en el grup oficial OpenCV hospedatjat a *Yahoo!(R)Groups* [L11]. La classe original ha estat modificada per afegir mètodes per escalar la imatge, aplicar un *offset* a la imatge (s'utilitza per desplaçar la coordenada d'inici de la imatge) i per "emetre" els events de pitjar els botons del ratolí i moure el ratolí.

Totes les classes abans descrites disposen de mètodes auxiliars, com *getters*/*setters*, que no s'han comentat perquè no aporten informació sobre les operacions que faciliten aquestes classes.

[4] La autoria de la classe *IplImageWidget* recau en la persona que hi ha sota el *nick* de Sebojolais [L11].

5.3.3 Interfície gràfica

L'aplicació esdevindrà d'un disseny SDI[5] de la interfície gràfica. En tot moment ha prevalgut la usabilitat i la simplicitat visual, principi KISS[6], per aquesta raó des d'un primer moment es va concebre com una única finestra amb quatre parts ben diferenciades.

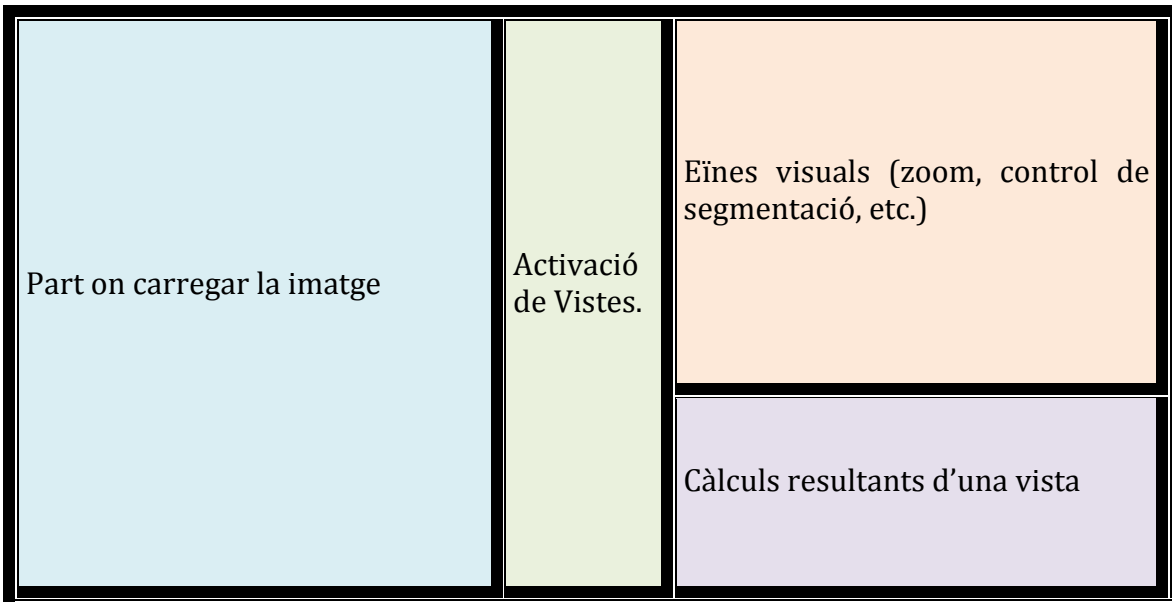


Diagrama 5

Amb aquest disseny l'usuari final tindrà tot el control de l'aplicació en una única pantalla, el que permetrà que en cap moment perdi cap element del camp.

Un altre aspecte de la interfície que cal plantejar durant la fase de disseny és la internacionalització (i18n) de l'aplicació final. Avui en dia atès que vivim en un societat globalitzada, aquest aspecte és cada cop més present i els actuals paquets de desenvolupament, com els *frameworks* que hi ha darrera, ofereixen solucions molt bones i fiables per implementar una aplicació multi-idioma.

Aquest és el cas d'aquest projecte que mitjançant les eïnes – i mètodes – que proporciona Qt oferirà al usuari una interfície en Castellà, Català i Anglés.

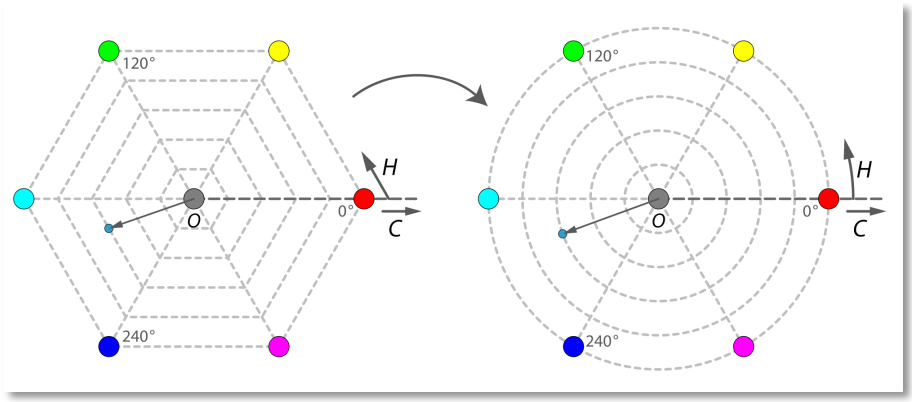
[5] *Simple Document Interface* o interfície de document únic.
[6] *Keep it simple and short!*

5.4 Implementació

5.4.1 Detecció de punts mitjançant OpenCV

Amb la primera tanda d'imatges ja es va constatar que les condicions en que es prenien les fotografies no eren les més òptimes. D'altra banda, les diferents proves utilitzant el mètode *cvHoughCircles* no van ser productives, raó per la qual es va descartar la utilització d'aquest mètode.

A l'hora de treballar l'espai de color HSV amb OpenCV, és molt important tenir en compte que un enter d'un byte (8 bits) només permet la representació de valors en el rang 0 – 255 i que per poder manipular una dimensió angular, com és la tonalitat, necessitem una representació que permeti el rang 0 – 360. Per aquesta raó OpenCV comprimeix el cercle a la meitat [L12] i obliga a dividir els valors, a tractar, a la meitat.



Imatge 7 [L13]

e.g.: Detecció de marcadors verds.

Com es pot observar a la imatge 5, el color verd té un matís – *hue* – amb valor de 120 graus. Tenint en compte que a una imatge del món real, és molt poc probable trobar tots els píxels amb aquest valor, el correcte és agafar un rang, per exemple, entre 100 graus i 140 graus. A l'hora d' aplicar l'algorisme de segmentació dins d'aquest rang, cal dividir a la meitat els valors; per tant, el rang sobre el que s'aplicarà l'algorisme és: (50, 70). D'altra banda, a més de realitzar la segmentació sobre un rang, és aconsellable fixar un llindar – *threshold* – pel component de saturació de l'espai HSV.

La segmentació i posterior detecció de punts s'ha dissenyat i implementat seguint el següent diagrama de flux:

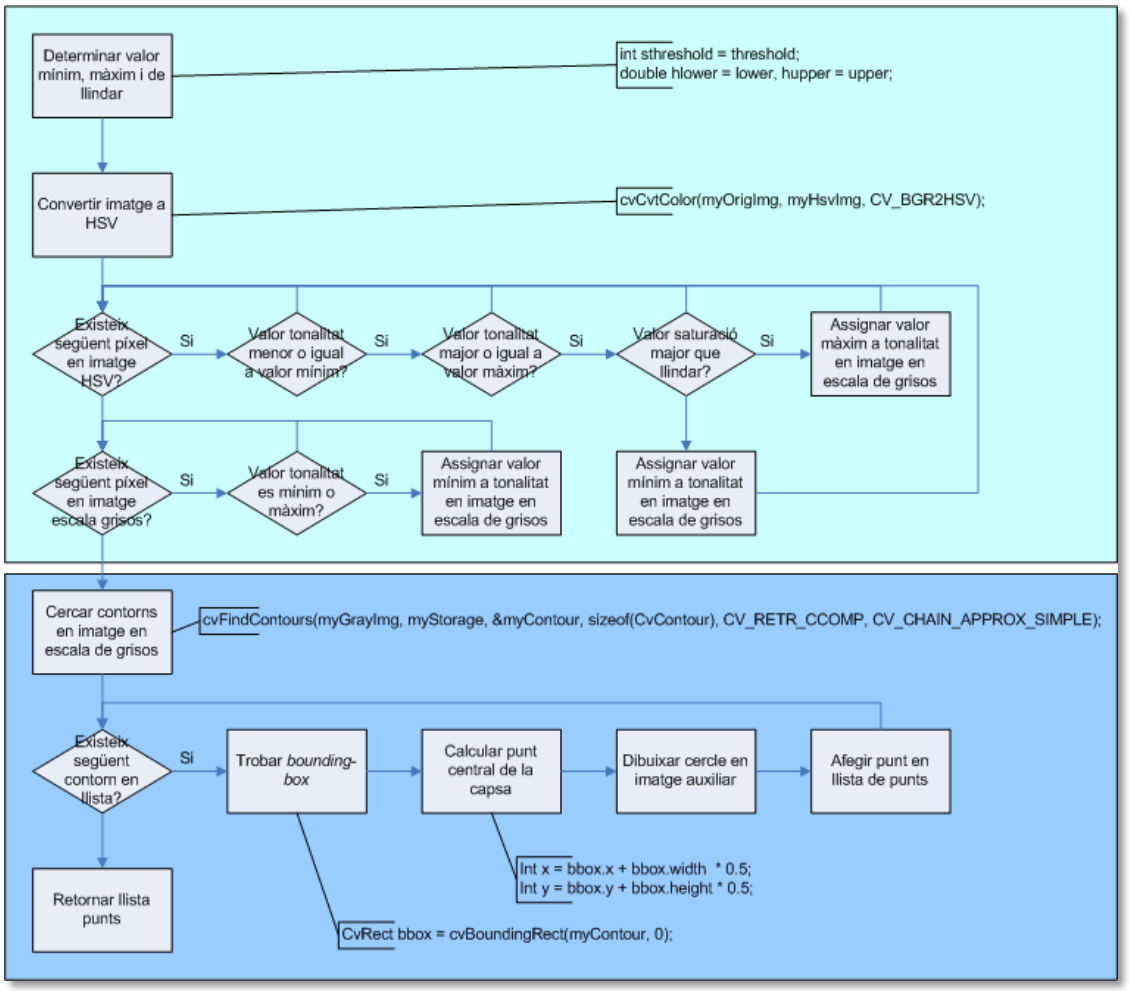


Diagrama 6

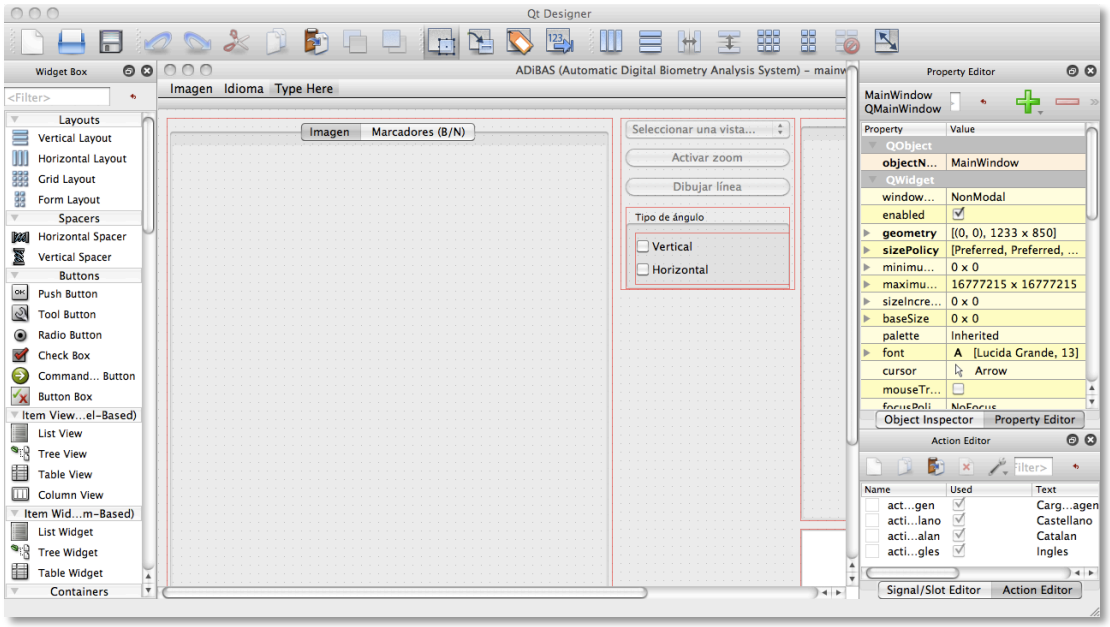
La implementació de l'algorisme de segmentació s'ha dividit en dos mètodes: *applySegmentation(..)* i *findAndDrawContours()* – blau clar i fosc, respectivament, en el diagrama 2 –. El primer mètode cerca a la imagte actual els píxels que compleixen amb el rang de segmentació del matís i el llindar de la saturació (píxel vàlid). Per cada píxel vàlid es dibuixa, en la mateixa posició, un píxel amb el valor màxim que pot tenir en una imatge en escala de grisos, on previament s'han fixat tots els pìxels a zero (resultat una imatge negre). El segon mètode cerca, amb *cvFindContours(..)*, sobre una imatge en escala de grisos, els contorns que hi puguin haver; d'una agrupació de píxels es pot treure un contorn. Un cop trobats tots els contorns, un per un, es tracten per cercar el rectangle que pot contenir el contorn, amb *cvBoundingRect(..)*, i d'aquest rectangle s'extreu el punt central que s'afegirà a la llista de punts trobats.

El perquè separar els dos mètodes, ha estat una decisió d'implementació per deixar oberta la possibilitat d'aplicar segmentació sobre una imatge sense haver de cercar contorn després, i viceversa.

5.4.2 Implementació de la interfície d'usuari mitjançant Qt

La interfície d'usuari s'ha desenvolupat amb Qt 4. Però quins avantatges té utilitzar Qt? El primer avantatge, i un dels més importants, és la qualitat i quantitat d'informació que hi ha disponible de forma oficial a través de la web o de l'aplicació *Assistant*, inclosa en el SDK. Aquest punt facilita moltíssim el disseny i la implementació i redueix les possibilitats d'acabar fent *spaghetti-code*[7] o el possible risc de fustració. D'altra banda, amb Qt no cal modificar el codi, o omplir-lo de directives de compilador, per compilar i executar el codi en diferents sistemes – "*Write once, compile anywhere*" –. I el tercer punt a tenir en compte és la facilitat amb la que es pot traduïr una aplicació amb les eïnes que proporciona el SDK.

Seguint el disseny de la interfície gràfica i amb l'aplicació *Designer* de Qt (inclosa al SDK) s'ha realitzat la implementació de la pantalla principal.

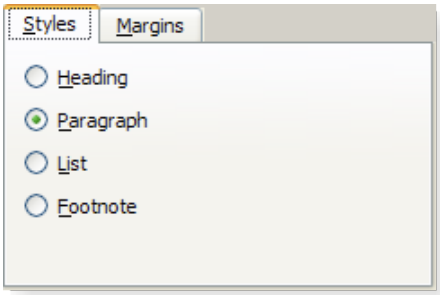


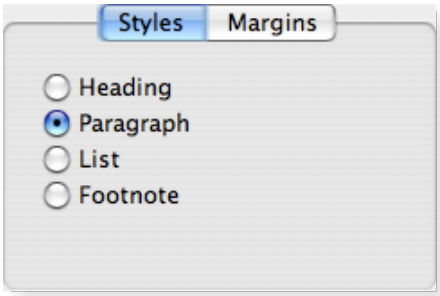
Imatge 8

[7] "*We don't know where to GOTO if we don't know where we've COME FROM*" [L14]

Una altra característica de Qt és que es poden crear *widgets*[8] nous estenent la classe QWidget (i aplicant la macro Q_OBJECT). Qualsevol nou *widget* es pot utilitar a l'aplicació Designer; per fer-ho existeixen dues tècniques: *promotoin approach* i *plugin approach*. La més senzilla és la primera, *promotion approach*, que consisteix en utilitzar la finestra de *widgets* personalitzats, en l'aplicació Designer, per donar d'alta la nova classe. A partir d'aquí qualsevol *QWidget* afegit al formulari podrà promoure cap al nou *widget* personalitzat. Gràcies al *widget* creat per Sebojolais, *IplImageWidget*, s'han pogut promoure cap aquest alguns dels *widgets* oferts per Qt.

Per exemple *QTabWidget* implementa un component visual que proporciona una pila – *stack* – de *widgets* amb pestanyes.





Imatge 9 i Imatge 10

8 "*In Qt and Unix terminoly, a widget is a visual element in a user interface.*" [B01 p.3]

Al promoure una pestanya (un *QWidget* originalment) cap a *IplImageWidget* s'obté el següent: la pestanya promoguda mostra una imatge que prèviament ha estat carregada amb OpenCV i *IplImageWidget* ha "transformat" en una instància de QImage.

Imatge 11

A més a més del treball portat a terme per l'autor del *widget*, els requeriments de l'aplicació feien necesaria una modificació del mateix per afegir la captura d'events provinents del ratolí, que són totalment necesaris quan es requereix interactivitat amb l'usuari perquè aquest faci certes operacions sobre una imatge; dibuixar una línia o aplicar zoom a una àrea concreta de la imatge.

Concretament, el que s'ha fet és definir un senyal – *signal* en terminología de Qt – per a cada event que es vol re-transmetre. *IplImageWidget* pel fet d'estendre *QWidget* pot sobreescriure[9] qualsevol mètode públic o protegit (*protected*) d'aquest. En aquest cas particular, s'han sobreescrit els mètodes *mousePressEvent(QMouseEvent*)* i *mouseMoveEvent(QMouseEvent*)* perquè cada mètode faci d'emissor d'un senyal – un event per Qt – per ser recollit per qualsevol altre classe mitjançant un *slot*.

Declaració dels senyals en l'arxiu de capçalera:

```
signals:
      void mousePress(int, int);
      void mouseMove(int, int);
```

Codi 1

[9] En programació, orientació a objectes és la capacitat que té una subclasse per oferir una implementació diferent d'un mètode públic o protegit, que ofereix la classe pare.

Sobreescriptura dels mètodes i retransmissió dels events:

```
void IplImageWidget::mousePressEvent(QMouseEvent *event)
{
      if (event->buttons() & Qt::LeftButton) {
            emit mousePress(event->x(), event->y());
      }
}

void IplImageWidget::mouseMoveEvent(QMouseEvent *event)
{
      if (event->buttons() & Qt::RightButton) {
            emit mouseMove(event->x(), event->y());
      }
}
```

Codi 2

Per entendre correctament com rebre events d'altres classes, és necessari fer cinc cèntims de la gestió d'events en Qt. Com es pot intuir del comentat fins ara, per Qt un event és un senyal – *signal* – que s'envia – *emit* – i pot ser rebut per un *slot*. Qt ofereix dues vies per connectar senyals i *slots*: de forma automàtica o manual. Utilitzant el mètode:

QObject::connect(emissor, SIGNAL(senyal), receptor, SLOT(slot))[QTBOOK]

És la forma de realitzar la connexió manualment. En aquest projecte s'han connectat tots els events de forma automàtica seguint la següent convenció a l'hora de declarar l'*slot*:

void on_<nom del *widget***>_<**nom del senyal**>(<**paràmetres del senyal **>);** [L15]

El codi següent és l'utilitzat per declarar dos *slots*, en la classe *MainWindow*, connectats automàticament als senyals *mousePress* i *mouseMove*, que ha emès el *widget* page_1.

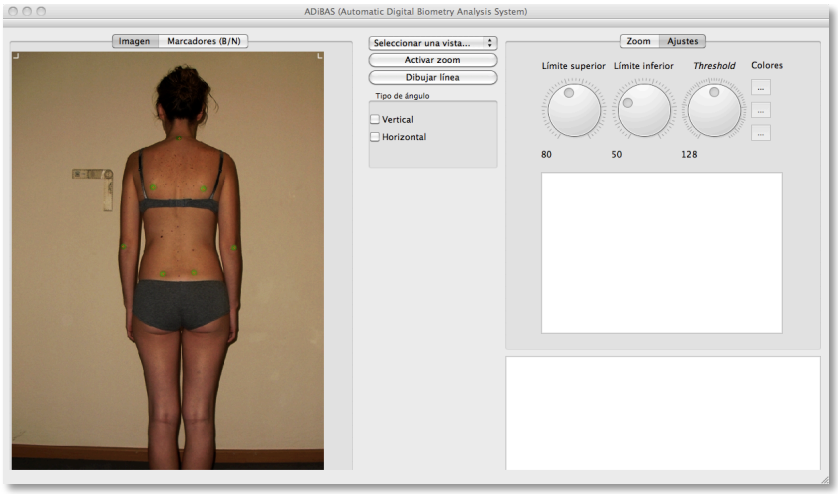
Declaració i vinculació automàtica de dos *slots* amb els senyals de IplImageWidget:

```
public slots:
    void on_page_1_mousePress(int x, int y);
    void on_page_1_mouseMove(int x, int y);
```

Codi 3

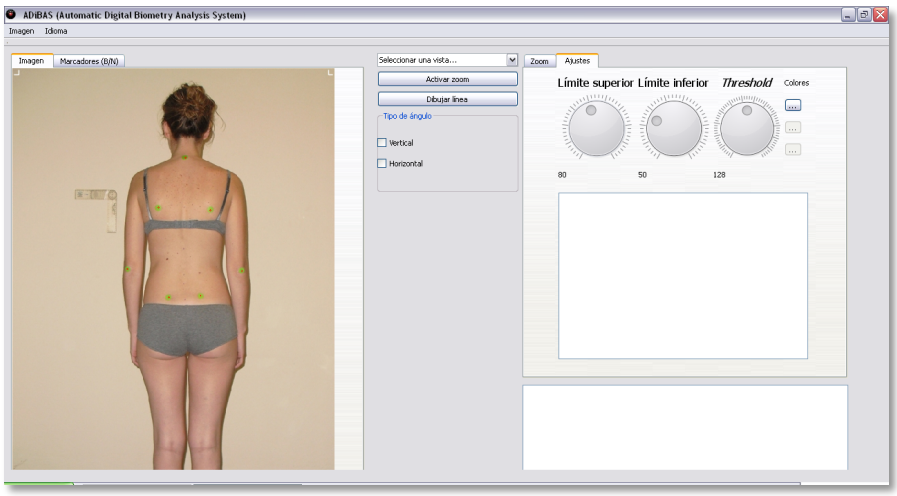
5.4.3 Interfície gràfica

Com ja s'ha esmentat, l'aplicació resultant té la particularitat de poder-la executar en multiples plataformes. Per mostrar aquest aventatge, a la següent imatge es pot veure l'aplicació compilada i executada sobre Mac OS X (Snow Leopard).



Imatge 12

D'altra banda, en aquesta imatge, es pot veure el resultat de compilar i executar, la mateixa revisió de codi, sobre Windows XP – en aquest cas particular Windows XP s'executava sobre Parallels Desktop[10] –.

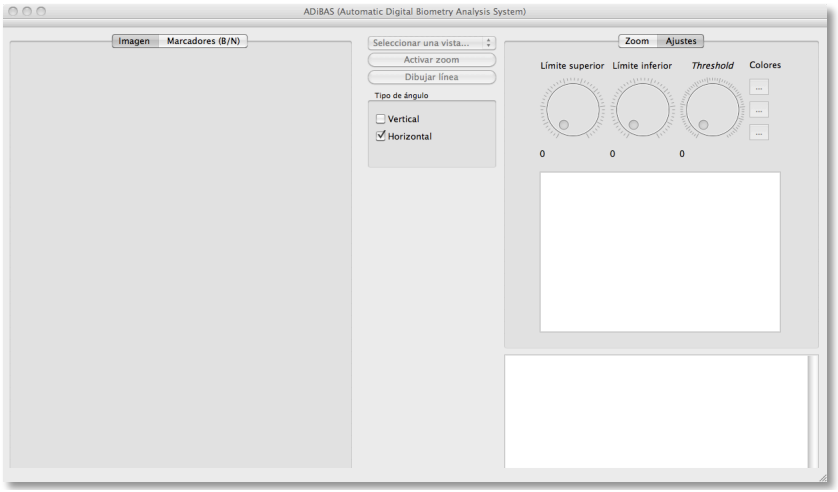


Imatge 13

[10] Aplicació de virtualització de maquinari per a Mac OS X.

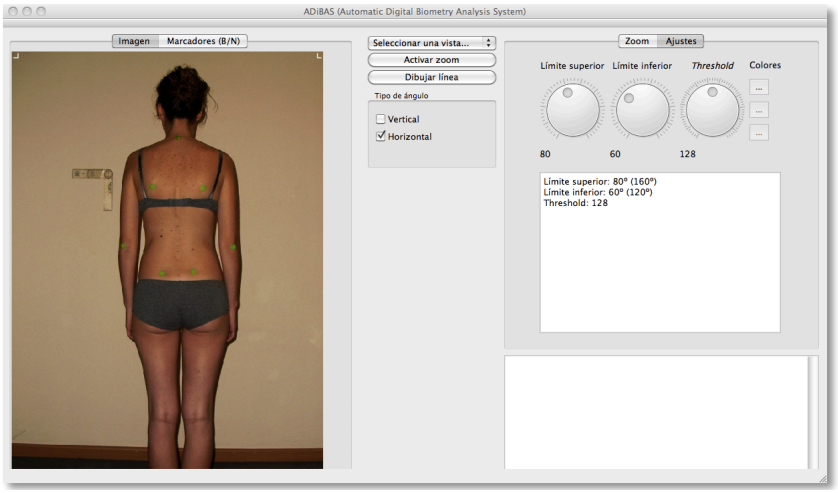
5.4.3.1 Interfície gràfica durant l'execució de l'aplicació

Quan s'executa l'aplicació, la primera pantalla que es veu es la següent. Tots els controls de la pantalla estan deshabilitats excepte l'opció de carregar una imatge.



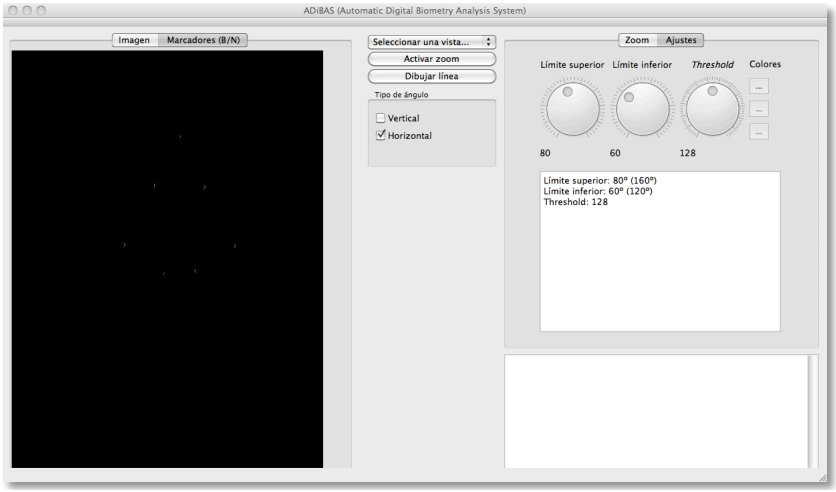
Imatge 14

Un cop carregada una imatge es carreguen uns valors per defecte per aplicar la segmentació, després s'executa la detecció de punts i finalment es mostra la imatge.



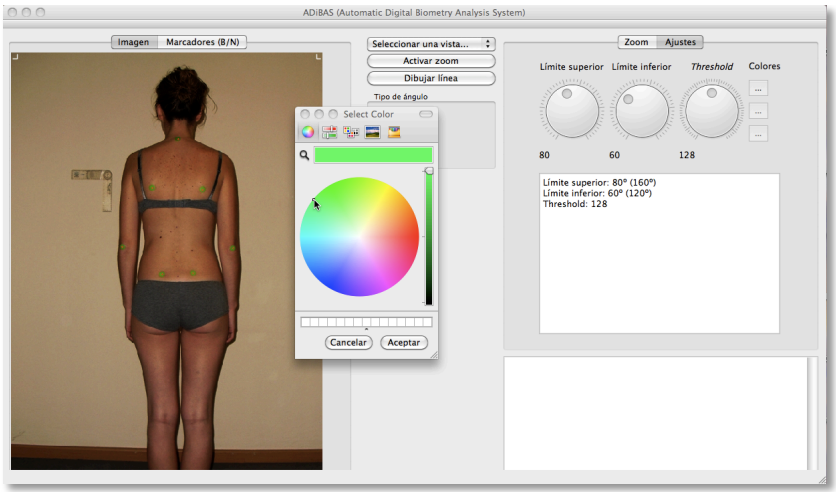
Imatge 15

En la pestanya – Marcadors (B/N) – es visualitza una imatge negra amb els contorns ressaltats. Aquesta pestanya serveix per comprovar directament la imatge amb la que treballa el mètode utilitzat per cercar contorns.



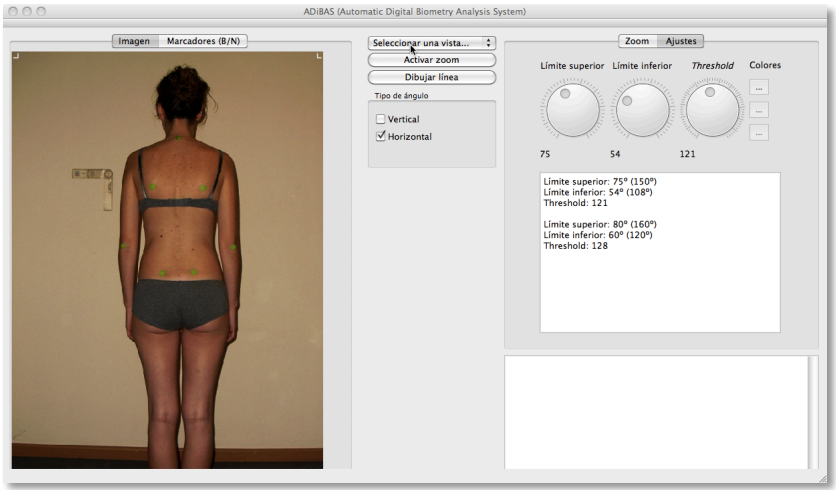
Imatge 16

En cas de voler canviar els límits de la segmentació hi ha dos possibilitats: moure els dials de cada limit o utilitzar el selector de color per seleccionar un color i obtenir els seus valors. El resultat de canviar un valor varia en temps real la imatge principal i la de control de contorns (imatge en blanc i negre).



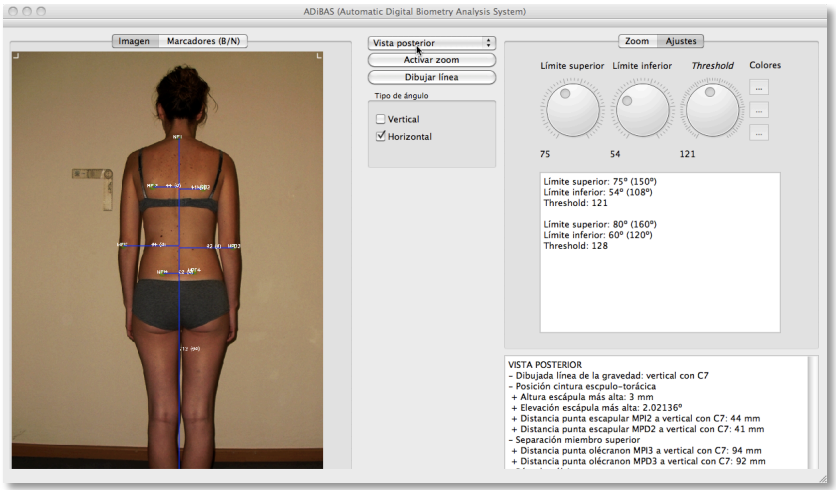
Imatge 17

A la pestanya – Ajustaments – hi ha un registre de les accions que s'han dut a terme.



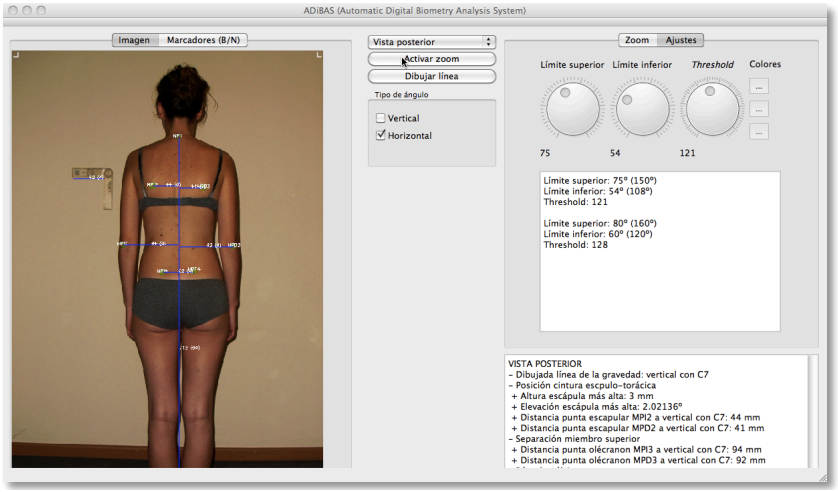
Imatge 18

Quan es selecciona una vista, automàticament es realitzen els càlculs definits i es mostren els resultats: de forma gràfica en la imatge superior i en text en l'àrea de resultats.



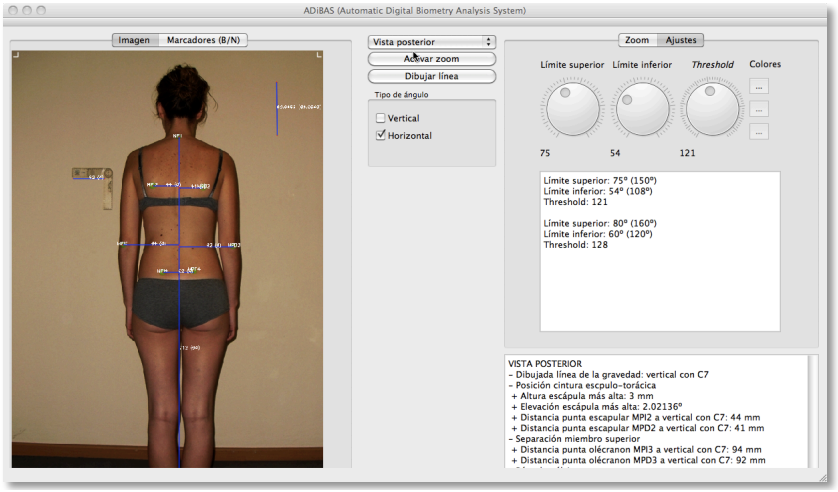
Imatge 19

Existeix la possibilitat de dibuixar línies de forma lliure. Per activar l'opció cal prémer el botó – Dibuixar línia – i marcar amb el ratolí (botó esquerra) dos punts fent click.



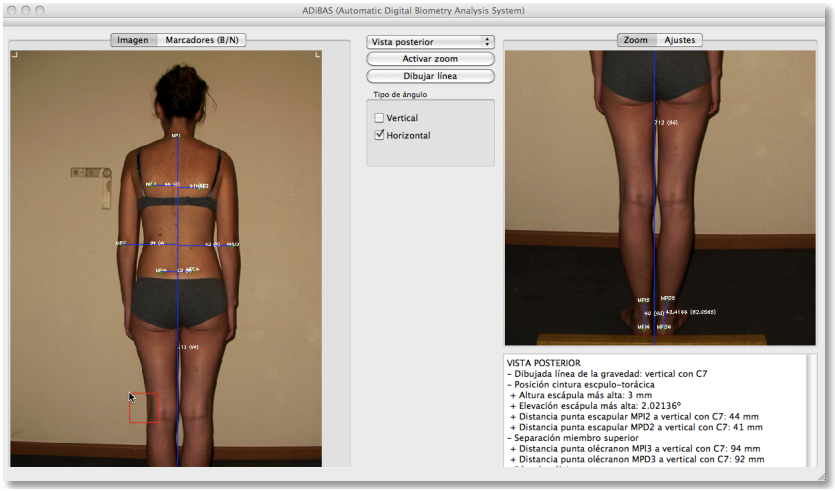
Imatge 20

La nova línia dibuixada a la imatge mostrarà la longitud i l'angle, que pot ser respecte l'eix horitzontal o respecte l'eix vertical. L'opció – Tipus d'angle – és la que determina quin angle mostrar, per defecte es calculen els angles respecte l'eix horitzontal.



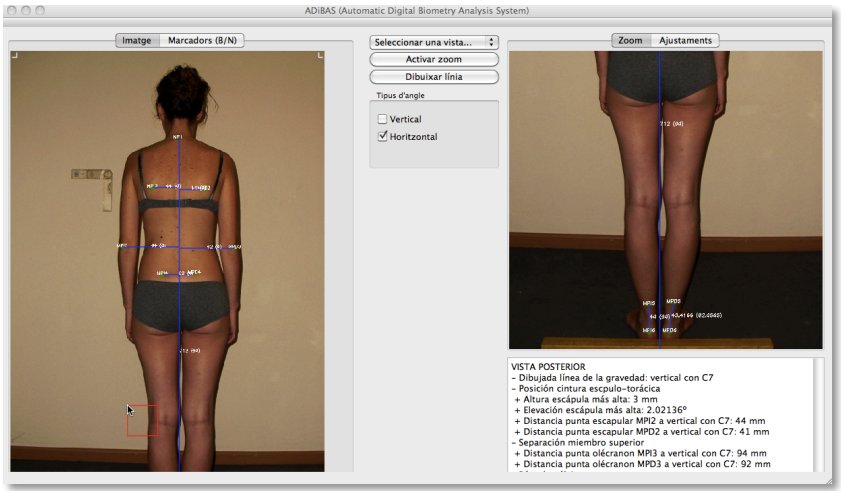
Imatge 21

Pitjant el botó – Activar zoom – s'ha bilita la visualització de la imatge original, amb la mida original[11]. La pestanya – Zoom – mostra un "finestra" amb la imatge original. Per desplaçar la imatge original i veure per la "finestra" l'àrea desitjada, només cal moure el ratolí per l'imatge principal (part esquerra) pitjant el botó dret. Un requadre vermell indicarà on comença l'àrea aproximada de visualització de la "finestra".



Imatge 22

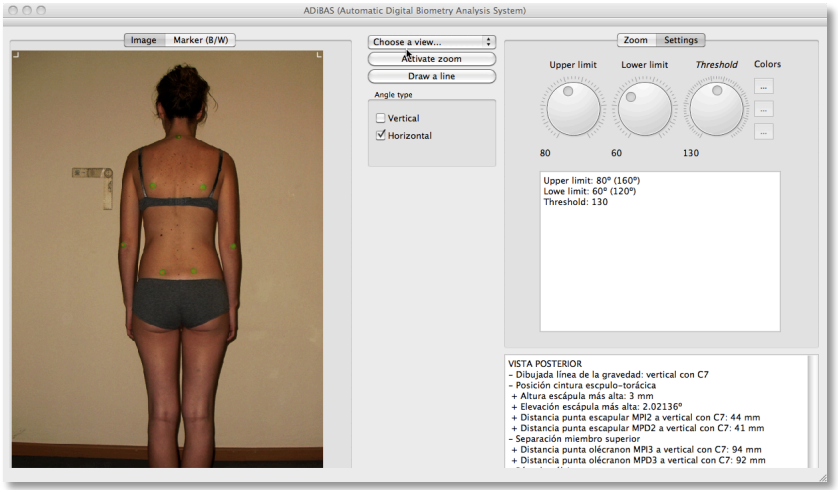
Com a mostra de la capacitat de internacionalització de l'aplicació, en la següent imatge es pot veure com dinàmicament s'han traduït tots els literals visibles[12], a la llengua Catalana amb només seleccionar l'idioma en la barra de menús.



Imatge 23

[11] Les imatges carregades sempre s'escalen perquè s'ajustin a la mida de la pestanya que conté el visualitzador.
[12] Excepte el text ja escrit en el visualitzador de resultats.

En la següent imatge es pot veure com dinàmicament s'han traduït tots els literals visibles[13], a la llengua Anglesa amb només seleccionar l'idioma en la barra de menús.



Imatge 24

[13] Excepte el text ja escrit en el visualitzador de resultats.

5.4.4 Implementació de vistes

La implementació d'una vista s'ha de definir a la classe *MainWindow* (mainwindow.h) i consta de dos passos. El primer pas és declarar i definir un objecte *View*, d'àmbit privat, i usant la següent convenció:

```
View myV<quatre primeres lletres del nom de la vista>;
```

El segon pas és declarar un mètode, d'àmbit privat, que seguirà la següent convenció:

```
void vista<nom de la vista>();
```

A l'hora de definir el mètode que implementarà la vista el primer que hi ha que fer és habilitar tots els marcadors a la instància de la classe *View* encarregada de manipular l'esmentada vista. Cal tenir en compte que els marcadors s'han de definir en el mateix ordre que la classe *Image* troba els contorns; d'abaix a dalt i de dreta a esquerra.

Per habilitar un marcador s'utilitzarà el mètode del objecte *View*:

```
addLabel(string label);
```

Un objecte *View* ofereix cinc mètodes per relacionar marcadors:

`addLine(Line, bool = true);`
Afegeix una línia a la vista i per defecte la dibuixa en la imatge.

`addLine(string, string, bool = true);`
Afegeix una línia recta entre dos marcadors. Per defecte dibuixa la línia en la imatge. Els dos marcadors han d'existir en la vista.

`addLowestHorizontal(string, string, bool = true);`
Afegeix una línia horitzontal amb l'origen en el marcador situat més avall i final en el punt d'intersecció amb la vertical que passa pel marcador situat més amunt. Per defecte dibuixa la línia en la imatge. Els dos marcadors han d'existir en la vista.

`addVerticalLine(string, bool = true);`
Afegeix una línia vertical amb l'origen en el marcador. Per defecte dibuixa la línia en la imatge. Els marcador ha d'existir en la vista.

`lenght(string, string);`
Distància entre dos marcadors. Els dos marcadors han d'existir en la vista.

Un cop habilitats tots els marcadors, el següent pas és dibuixar-los a la imatge. Per fer-ho cal executar el mètode del objecte *View*:

```
showTags();
```

És molt important assegurar que l'objecte *View* té assignada una instància, no nul·la, de la classe *Image*; sinó no es podrà visualitzar la vista ni realitzar els càlculs corresponents.

El motiu d'utilitzar en tot moment etiquetes per realitzar les operacions, és facilitar l'abstracció del esquema d'una vista a qualsevol profà en la fisioteràpia que tingi com a tasca implementar o modificar una vista. Aquesta simplicitat amb que es pot definir i modificar una vista és la gran aportació de la classe *View*, que dinamitza el manteniment de l'aplicació i facilita qualsevol evolotiu o re-enginyeria futura.

5.4.4.1 Implementació de la vista posterior del pla frontal

La implementació de la vista posterior del pla frontal s'ha fet seguint la documentació lliurada per l'usuari. Aquesta documentació consisteix en uns diagrames detallats dels marcadors ubicats en el pacient i una descripció concisa dels càlculs a fer i com fer-los.

Abans de continuar voldría remarcar la gran qualitat dels requeriments aportats. Als meus anys d'experiencia com a desenvolupador és la segona vegada que un usuari – o grup d'usuaris – aporta una documentació tan detallada i ben explicada.

Segons els requeriments lliurats, les variables a calcular es detallen a continuació, junt amb el codi implementat per obtenir els resultats.

LGP (Línia de la gravetat)	Línia perperdicular al terra amb origen en el marcador MP1 (setena vertebra cervical)
`myVPost->addVerticalLine(string("MP1"));`	

Codi 4

CETP (Cintura escapular-toràcica)	Marcadors MPD2 i MPI2 ubicats en la punta inferior de la escàpula
Conèixer l'altura del marcador més alt respecte les línies horitzontals que passen pels marcadors.	

`myVPost->addLowestHorizontalLine(string("MPI2"), string("MPD2"), false);`
`Line l = myVPost->getLine(string("LHMPI2MPD2"));`
`double d1 = l.getDistance2Point( myVPost->getMark(string("MPI2")) );`
`double d2 = l.getDistance2Point( myVPost->getMark(string("MPD2")) );`
`double d = (d1 > d2) ? d1 : d2;`	

Conèixer l'elevació de d'escàpula més alta.	

`myVPost->addLine(string("MPI2"), string("MPD2"), false);`
`Line l = myVPost->getLine(string("MPI2MPD2"));`
`double a = l.getAngle();`	

Distància de cada punta escapular a la línia de la gravetat.	

`myVPost->addHorizontalLine(string("MPI2"), myVPost->getLine(string("VMP1")));`
`myVPost->addHorizontalLine(string("MPD2"), myVPost->getLine(string("VMP1")));`
`Line l = myVPost->getLine(string("MPI2VMP1"));`
`double d1 = l.getLenght();`
`l = myVPost->getLine(string("MPD2VMP1"));`
`double d2 = l.getLenght();`	

Codi 5

PSMS (Separació membre superior)	Marcadors MPD3 i MPI3 ubicats en olècranon
Distància de cada punta olècranon a la línia de la gravetat.	

```
myVPost->addHorizontalLine(string("MPI3"), myVPost->getLine(string("VMP1")));
myVPost->addHorizontalLine(string("MPD3"), myVPost->getLine(string("VMP1")));
Line l     = myVPost->getLine(string("MPI3VMP1"));
double d1 = l.getLenght();
l          = myVPost->getLine(string("MPD3VMP1"));
double d2 = l.getLenght();
```

Codi 6

BPP (Bàscula pèlvica)	Marcadors MPD4 i MPI4 ubicats en la espina ilíaca posterior superior
Conèixer l'altura del marcador més alt respecte les línies horitzontals que passen pels marcadors.	

```
myVPost->addLowestHorizontalLine(string("MPI4"), string("MPD4"), false);
Line l     = myVPost->getLine(string("LHMPI4MPD4"));
double d1 = l.getDistance2Point( myVPost->getMark(string("MPI4")) );
double d2 = l.getDistance2Point( myVPost->getMark(string("MPD4")) );
double d  = (d1 > d2) ? d1 : d2;
```

Distància entre els dos marcadors

```
myVPost->addLine(string("MPI4"), string("MPD2"), false);
Line l      = myVPost->getLine(string("MPI4MPD4"));
double len = l.getLenght();
```

Codi 7

RP (Retropeu)	Marcadors ubicats en punta zona mitjana del tendó d'Aquil·les (MPD5 i MPI5), tubercle calcani (MPD6 i MPI6) i zona mitja calcani en contacte amb el terra
Angle format per la intersecció de marcadors 5 i 6 amb 6 i 7.	

```
myVPost->addLine(string("MPI5"), string("MPI6"), false);
myVPost->addLine(string("MPI6"), string("MPI7"), false);
Line l     = myVPost->getLine(string("MPI5MPI6"));
double a1 = l.getAngle();
l          = myVPost->getLine(string("MPI6MPI7"));
double a2 = l.getAngle();
double a  = abs(a1 - a2);

myVPost->addLine(string("MPD5"), string("MPD6"), false);
myVPost->addLine(string("MPD6"), string("MPD7"), false);
l  = myVPost->getLine(string("MPD5MPD6"));
a1 = l.getAngle();
l  = myVPost->getLine(string("MPD6MPD7"));
a2 = l.getAngle();
a  = abs(a1 - a2);
```

Codi 8

5.5 Conclusions

Aquest projecte m'ha donat la possibilitat de treballar en un àmbit totalment desconegut per a mi, amb eines que mai havia fet servir i amb resultats que no deixen de sorprendre'm. OpenCV, en la seva versió 2.1, és una llibreria molt madura amb possibilitats gairebé infinites. D'altra banda està Qt, un SDK que funciona a la perfecció amb una documentació que fa enveja i una simplicitat molt gran. Aquest binomi que he descobert, OpenCV + Qt, té una potencia realment extraordinària. Poc més puc dir, perquè realment fa molt poc temps que *passejo amb aquest parell*, i amb aquestes línies segurament estic posant punt i seguit a aquesta aventura. O no!

D'altra banda, no hi ha dubte que part del futur al camp de les noves tecnologies passa per la visió per computador. Cada cop prenen més presencia eïnes com la realitat augmentada, reconeixement automàtic d'objectes, etc. I sense dubte ADiBAS és el primer esglaó del que pot ser una eina revolucionaria en el cap de la fisioteràpia. I per això estic molt orgullós d'haver participat, d'haver aportat la meva tasseta de sorra i en definitiva d'haver començat aquest projecte.

Aquest projecte de final de carrera deixa algunes coses al tinter. Per començar és fonamental definir quin marcador utilitzar per extrapolar mesures. També és interessant dotar d'un motor d'inferència a l'aplicació, d'aquesta manera es convertiria en una eina de suport al diagnòstic. Donar d'alta noves vistes, etc. son altres tasques pel futur del projecte.

6 Glossari

Hough, transformada	Algorisme emprat en el reconeixement de patróns (línies, cercles, etc.) en imatges.
HSV/HSL	*Hue Saturation Value / Hue Saturation Lightness*. Models de colors amb coordenades cilíndriques o cóniques.
Marcador	Qualsevol cosa (un "gomet", un troç de paper, etc.) que serveix per marcar alguna cosa (un punt, una distància, etc.).
RGB	*Red Green Blue*. Model de color on es defineix un color en refèrencia a l'intensitat dels colors primaris. Normalment es codifica amb un *byte*.
SDK	*Software Development Kit*. Conjunt d'aplicacions, llibreries i documentació que permet desenvolupar una tecnologia.
Signal	És un event segons Qt.
Slot	Mètode que rep un event segons Qt
Virtualització	Abstracció dels recursos d'un computador.
Vista	Conjunt de marcadors, ubicats de forma coneguda en una determinada posició que pot ser analitzada i reconeguda.
Widget	En Mac i Linux és qualsevol component gràfic.

7 Bibliografia

Els llibres que s'han consultat per a realitzar aquest projecte de fnal de carrera són:

B01	Bradski, G. & Kaehler, A. *Learning OpenCV, Computer vision with the OpenCV library*, 1ª Edició, 2008, O'Reilly.
B02	Stroustrup, B. *Programming Principles and Practice Using C++*, 1ª Edició, 2009, Addison Wesley.
B03	Blanchette, J. & Summerfield, M. *C++ GUI programming with Qt 4, 5ª Edició,* 2010, Prentice Hall.
B04	Spiegel M., Liu J. & Abellanas, L. *Fórmulas y Tablas de Matemática Aplicada*, 2ª Edició, 2000, McGraw Hill (Serie Schaum)
B05	Gamma E., Helm R., Johnson R. & Vlissides J. Design Patterns. *Elements of Reusable Object-Oriented Software*, 31ª Edició, 2004, Addison Wesley (Professional Computing Series).
B06	Ambler, S. *The Elements Of UML 2.0 Style*, 1ª Edició, 2005, Cambridge University Press.

Els enllaços web consultats per a realitzar aquest projecte de fnal de carrera són:

L01	https://osanchezmon.unfuddle.com
L02	http://subversion.apache.org/
L03	http://osanchezmon.unfuddle.com/svn/osanchezmon_adibas/
L04	http://osanchezmon.unfuddle.com/svn/osanchezmon_qtadibas/
L05	http://en.wikipedia.org/wiki/File:AdditiveColor.svg
L06	http://en.wikipedia.org/wiki/File:RGB_color_solid_cube.png
L07	http://en.wikipedia.org/wiki/RGB_color_model
L08	http://en.wikipedia.org/wiki/HSL_and_HSV
L09	http://en.wikipedia.org/wiki/File:HSV_color_solid_cylinder_alpha_lowgamma.png
L10	http://en.wikipedia.org/wiki/File:HSV_color_solid_cone_chroma_gray.png
L11	http://tech.groups.yahoo.com/group/OpenCV/files/Qt%20IplImage%20widget/
L12	http://myopencv.wordpress.com/2008/09/28/color-detection-by-using-color-space-basics/
L13	http://en.wikipedia.org/wiki/File:Hsv-hexagons-to-circles.svg
L14	http://www.fortran.com/fortran/come_from.html
L15	http://doc.trolltech.com/4.4/designer-using-a-component.html#a-dialog-without-auto-connect

Tots els enllaços web estan actius a dia de l'entrega d'aquesta memòria.

8 Annexes

8.1 Annex 1 – Contingut del CD Adjunt

1	Última revisió estable del codi font de l'aplicació QtADiBAS.
2	Instal·lador de l'aplicació QtADiBAS per a Mac i Windows.
3	Memòria en format PDF.
4	Requeriments d'usuari en format PDF.
5	Documentació generada automàticament a partir del codi font, amb l'aplicació *Doxygen*.
6	Manual d'usuari de l'aplicació QtADiBAS.
7	Manual d'implementació de l'aplicació QtADiBAS.
8	Joc d'imatges per fer proves.

8.2 Annex 2 – Requeriments d'usuari

Els requeriments d'usuari han estat elaborats per José Ramírez que en aquest projecte ha tingut el rol d'usuari sènior[14].

A la memòria només s'inclouen els requeriments de les vistes dissenyades i implementades.

8.2.1 Vista posterior, pla frontal

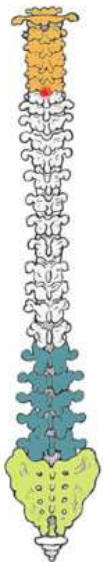
[14] *Sènior user* és un rol definit a la metodologia Prince2.


Biometría digital
ubicación marcadores
PLANO FRONTAL
Visión posterior

Visión posterior

Variable: Línea de la gravedad (LGP)

Detalle anatómico



Visión posterior

Variable: Línea de la gravedad

Marcador ubicado en espinosa de la séptima vertebra cervical (MP1). Línea perpendicular al suelo

Objetivo

Conocer el desplazamiento del cuerpo en relación a esta línea

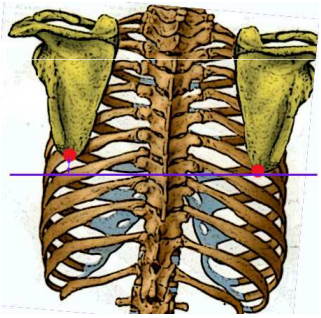
Parámetro a medir

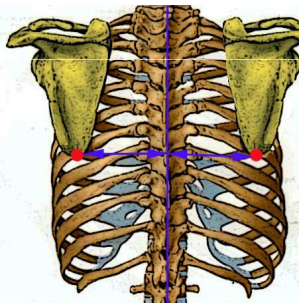
Distancia en mm. Desplazamiento a la derecha o izquierda según centro de referencia del suelo

Visión posterior

Variable: Posición cintura escápulo-torácica (CETP)

Detalle anatómico





Visión posterior

Variable: cintura escápulo-torácica

Marcadores ubicados en punta inferior de la escápula (MPD2 / MPI2)

Objetivo

Conocer la simetría de la CET respecto a la horizontal

Parámetro a medir (2)

1. En caso de diferencias de altura de los hombros: respecto a la horizontal como punto de referencia el marcador de la escápula más baja, conocer la altura en mm de esa línea a la escápula más alta (conocer elevación /descenso)
2. Distancia en mm de cada punta escapular a la vertical de referencia que pasa por C7 (conocer abd / add)

Visión posterior

Variable: separación miembro superior (PSMS)

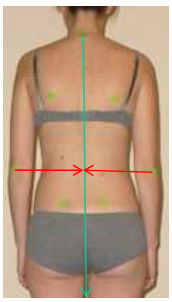
Marcadores ubicados olécranon (MPD3 / MPI3)

Objetivo

Conocer actitud del miembro superior respecto al tronco

Parámetro a medir

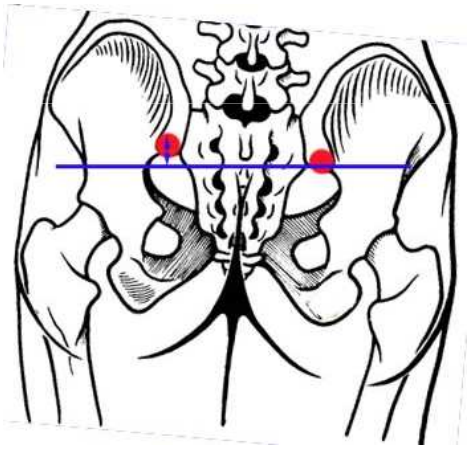
1. Distancia en mm de cada punta olécranon a la vertical de referencia que pasa por C7



Visión posterior

Variable: Báscula pélvica (BPP)

Detalle anatómico



Visión posterior

Variable: Báscula pélvica

Marcadores ubicados espina iliaca postero superior (MPD4 / MPI4)

Objetivo

Conocer la alineación de la pelvis

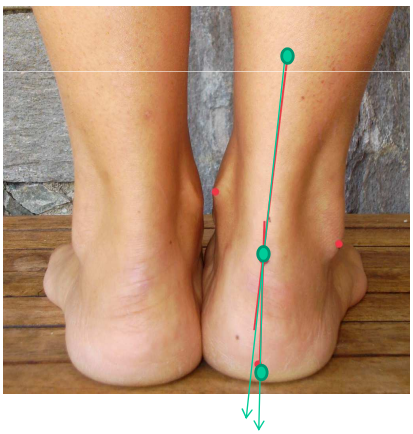
Parámetro a medir (2)

1. En caso de diferencias de altura de las EIPS: respecto a la horizontal como punto de referencia el marcador de la EIPS más bajo, conocer la altura en mm de esa linea a la EIPS más alta
2. Distancia en mm entre ambas EIPS

Visión posterior

Variable: Retropie (RP)

Detalle anatómico



Visión posterior

Variable: Retropie

Marcadores ubicados en punta zona media tendón de aquiles (MPD5 / MPI5), tuberculo calcáneo (MPD6 / MPI6) y zona media calcáneo en contacto con suelo (MPD7 / MPI7)

Objetivo

Conocer el varo / valgo del retropie

Parámetro a medir

Angulo formado por la intersección de marcadores 5 y 6 con el 6 y 7