Universitat Autònoma de Barcelona

escola d'enginyeria

5176-1: INTERFÍCIE GRÀFICA PER AL PROCESSAMENT D'IMATGES D'ECOGRAFIA INTRACORONÀRIA

Memòria del Projecte Fi de Carrera
d'Enginyeria en Informàtica
realitzat per
Laura Blanch Osset
i dirigit per
Aura Hernández Sabaté
Bellaterra, 19 de Juny de 2013

Universitat Autònoma de Barcelona

etse
Escola Tècnica Superior d'Enginyeria

El sotasignat, **Aura Hernández Sabaté**

Professor/a de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per na

Laura Blanch Osset

I per tal que consti firma la present.

Signat: Aura Hernández Sabaté

Bellaterra, 19 de Juny de 2013

Taula de continguts

1. Introducció

A dia d'avui, les malalties cardiovasculars (malalties del cor i de les artèries) són el principal motiu de mort a nivell mundial [1]. D'entre elles, les que afecten les artèries (coronàries majoritàriament) són especialment rellevants, ja que, entre d'altres coses, són molt difícils de detectar durant les primeres fases. Una de les malalties de més prevalença és l'ateroesclerosi, on el flux sanguini es veu reduït (i fins i tot pot arribar a bloquejar-lo del tot) degut a l'acumulació de placa (una barreja de sang, greixos, cèl·lules i colesterol) a l'interior del vas. L'Ecografia IntracoronàriaIntravascular (EICV o, en anglès, IVUS) és una tècnica força estesa pel diagnòstic i seguiment d'aquestes malalties. Es basa en l'ús d'un catèter especialment dissenyat amb una sonda d'ultrasò unida a un dels extrems. Així, es reconstrueixen imatges de l'artèria en estudi a partir de la reflexió d'aquesta senyal en les parets i teixits del vas, mostrant la completa morfologia d'aquest i permetent l'estudi de les seves propietats biomecàniques.

1.1. Processament de les imatges IVUS

Per tal de poder extreure la informació necessària cal inspeccionar les imatges obtingudes mitjançant l'Ecografia Intracoronària amb detall i definir els límits del vas, així com identificar els diferents elements que poden ajudar a establir el risc de malaltia (Figura 1).

Executar aquest procés de forma manual és una tasca llarga i feixuga, motiu pel qual hi ha un alt risc d'errors, a banda de generar resultats que poden variar segons els observadors, ja que és un estudi fortament lligat a la visió objectiva de qui el du a terme.

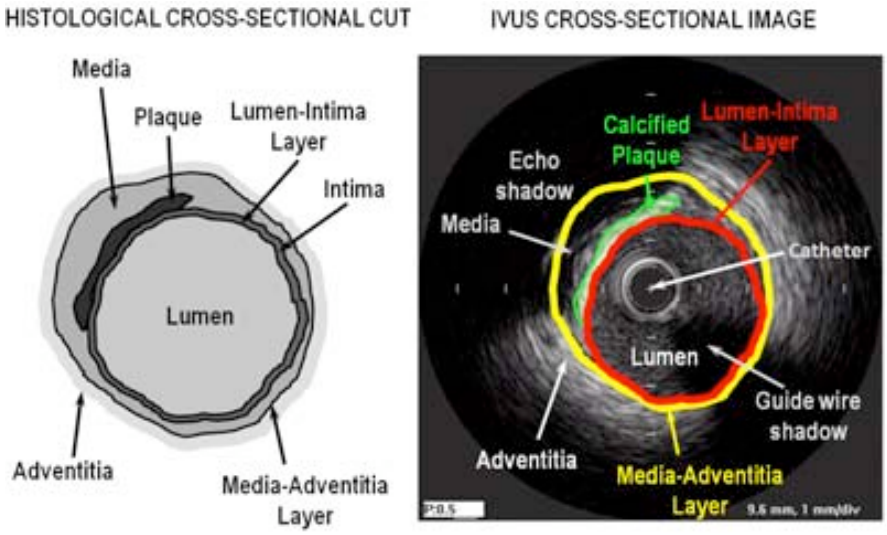


Figura 1: A l'esquerra, tall transversal d'una arteria, amb els elements que la composen. A la dreta, la corresponent imatge IVUS.

Apareix doncs, la necessitat d'eines que facilitin el processament de les imatges, automatitzant el procés el més possible per obtenir resultats més fiables i amb més rapidesa i poder així centrar els esforços en detectar i controlar la malaltia.

La recerca en aquest àmbit és activa, i ja es poden trobar diversos mètodes que adrecen alguns dels principals problemes del processament manual de les imatges. A la majoria, però, els manca una implementació del mètode enfocada a l'usuari final. Es tracta, doncs d'estudis que aporten

gran valor a nivell acadèmic, però la seva forma encara és massa inaccessible per persones fora de l'àmbit de la recerca. A més, molts d'ells encara depenen de la intervenció manual i no ofereixen una solució suficientment complerta per la diagnosi de la malaltia, ja que estan més enfocats a la detecció de les diferents estructures existents en un vas.

Pel que fa a l'oferta d'aplicacions amb interfície gràfica, el ventall no és gaire ampli. Volcano [2] i Boston Scientific [3] són les dues grans empreses tant de software com de hardware sobre Ecografia Intracoronària. Ambdues ofereixen el conjunt de màquina més programari, adaptat per l'ús mèdic (Fig. 2). Les aplicacions permeten processar imatges IVUS, calculant els contorns i la mida del vas i del lumen. A més, disposen d'un codi de colors per diferenciar els diferents tipus de teixits del vas. Els seus mètodes no són, però, del tot automàtics i, malauradament, aquestes interfícies són exclusives de la companyia que les ofereix.

Existeixen altres aplicacions que, mitjançant la norma DICOM (estàndard per manipular, emmagatzemar i transmetre informació d'imatges mèdiques), poden manipular les imatges generades per les aplicacions de Volcano i Boston Scientific. Es tracta de la COMPACS Workstation de Medimatic [4] o de echoPlaque de INDEC Medical Systems [5]. Ambdues aplicacions permeten detectar i editar contorns de forma similar a les anteriors aplicacions. Disposen, a més, de tot un seguit d'eines més administratives, on poder gestionar les dades del pacients, fer-ne gràfiques o exportar-ho en excel. En el cas de COMPACS, l'estació de treball disposa d'un servidor, que permet treballar online des d'una altra màquina fent servir les mateixes dades.

Totes les propostes vistes fins ara són de pagament i no ofereixen detalls dels algorismes o mètodes usats per la detecció d'estructures.

D'altra banda, 3D IVUS-ANGIO [6] (Fig. 2) es tracta d'una aplicació gratuïta que fusiona l'angiografia (estudi dels vasos circulatoris basat en practicar una radiografia després d'una injecció de contrast als raigs X.) amb IVUS. Donades les imatges de l'angiografia i l'ecografia intracoronària, permet detectar i editar contorns en imatges IVUS, a més d'oferir eines per analitzar les imatges angiogràfiques. En la seva darrera actualització ofereixen a més, la reconstrucció en 3D del vas. L'aplicació, però, és senzilla i presenta força errors i bugs: els creadors animen a tothom que la prova a notificar els errors i les possibles millores. Cal tenir en compte, a més, que sempre necessitem comptar amb els dos tipus d'imatge.

En definitiva, el mercat ofereix un ventall força limitat de programari de processament d'imatges IVUS i, en la majoria de casos, accedir a les aplicacions no és senzill.

La Dra. Aura Hernández [7][8] proposa vàries eines per explorar la dinàmica de les artèries i les seves estructures. Per una banda presenta un model que permet estabilitzar les imatges al llarg de la seqüència, corregint els desplaçaments no desitjats del catèter, tant a nivell transversal com a nivell longitudinal. Per altra banda, presenta un mètode de detecció automàtica de les parets arterials i proposa l'estabilització de les imatges per accelerar el procés de segmentació.

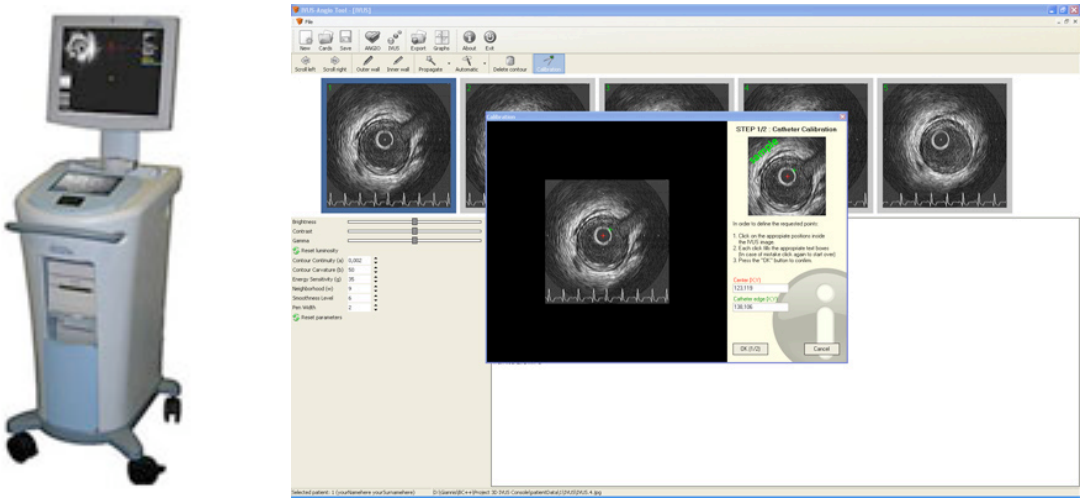


Fig. 2: iLab® Ultrasound Imaging System de Boston Scientific.(esquerra) i 3D IVUS-ANGIO (dreta).

Aquesta nova aproximació, que encara no s'ha implementat en cap aplicació amb interfície gràfica, és totalment automàtica i representa, gràcies a l'estabilització del moviment del vas, una millora substancial a l'hora de detectar els contorns. A més, el temps d'execució és manté baix, tot i la seva complexitat.

1.2. La nostra contribució

Disposem d'un mètode de processament d'imatges d'ecografia intracoronària, implementat per la Dra. Aura Hernández [7], que ofereix una solució automatitzada als principals problemes que presenta l'estudi manual de les imatges IVUS.

Amb l'aplicació **IvusGUI** proposem un programa que ofereix aquest mètode en un context amigable per clínics, metges i professionals de la medicina, mitjançant una interfície gràfica simple i una aplicació de fàcil accés i que no requereix de hardware especial.

2. Mètode de processament d'imatges IVUS

A l'hora d'estudiar l'acumulació de placa i de trobar els límits del vas, s'utilitzen dos tipus d'imatges: el tall transversal del vas, és a dir, la imatge que s'obté directament de l'ecografia intracoronària, i el tall longitudinal del vas, resultat de creuar un cert nombre d'imatges transversals sobre el mateix pla en el mateix angle (Figura 3).

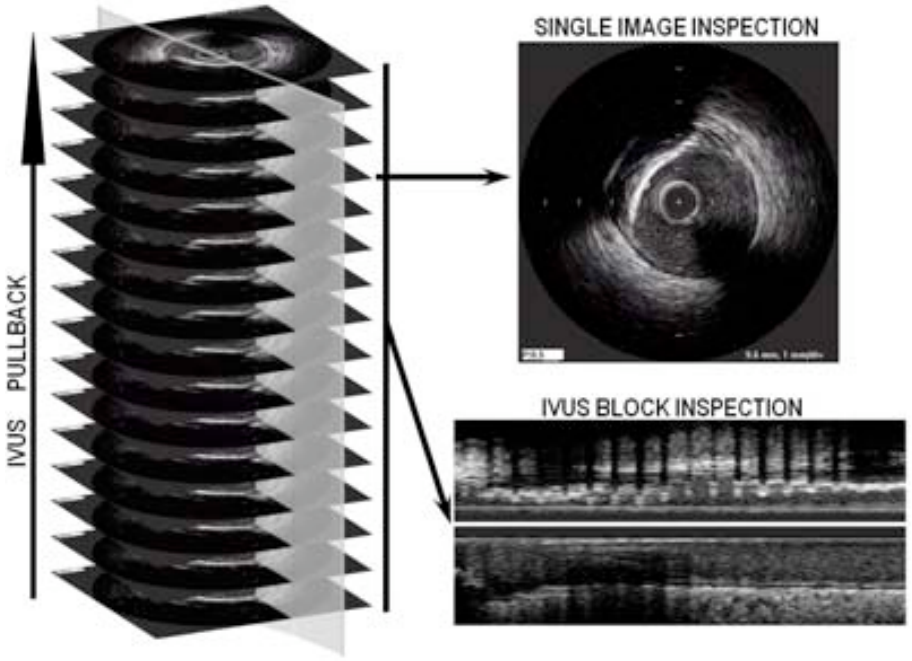


Figura 3: Imatges obtingudes de l'Ecografia Intracoronària. A l'esquerra, bloc de 16 imatges obtingudes del pas del catèter per l'arteria. A dalt a la dreta, detall d'una d'aquestes imatges, que mostra el tall transversal del vas. A baix a la dreta, tall longitudinal generat al creuar 401 frames amb el pla mostrat en gris, al mateix angle d'inclinació.

Els moviments produïts per la dinàmica cardíaca (el batec del cor, entre d'altres), afecten la visualització real del vas i fa que la seva anàlisi sigui més difícil. Aquests moviments es caracteritzen per una rotació seguida d'una translació a més d'un moviment longitudinal. Per aquest motiu, les imatges transversals no estan alineades i als talls longitudinals, les imatges no queden ordenades segons el moment de la fase cardíaca en que es troben.

L'algorisme en que es basa la nostra aplicació, creat per la Dra. Aura Hernández, fa ús d'aquesta dinàmica cardíaca per obtenir una seqüència d'imatges estabilitzada, un tall longitudinal amb compensació del moviment arterial i millors resultats a l'hora de trobar l'adventitia, una de les capes del vas més difícils de detectar degut a la seva aparició, molt dèbil, a les imatges IVUS.

2.1. Potencial de la dinàmica cardíaca per al processament d'imatges IVUS

L'algorisme es basa en l'estimació del moviment d'un cos rígid en el pla per integrar la correcció del moviment del vas amb la segmentació de les parets arterials i l'extracció de la fase cardíaca.

Per dur a terme aquesta estimació es té en compte que el moviment del vas segueix un ritme periòdic, produït pel batec del cor. Com a resultat obtenim que el moviment a corregir es pot definir per una translació i una rotació (Figura 4).

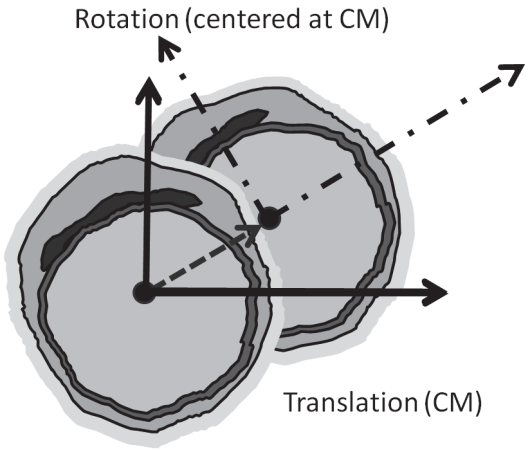


Figura 4: Aproximació al moviment del sòlid rígid

Segons la dinàmica de cossos, el punt que descriu la resposta del cos a una força externa és el centre de masses. En el nostre cas, calculem el centre de masses del vas (VCM) com una combinació del centre de masses donat per la intensitat de la imatge (ICM) i el centre de masses donat pel centre geomètric del vas (GCM). La translació a aplicar, doncs, serà la diferència entre aquest punt i l'origen de coordenades.

Pel que fa a la rotació, aquesta es transforma en una translació horitzontal al passar les imatges a coordenades polars, amb centre de coordenades al punt prèviament calculat com a VCM. Aquesta translació horitzontal s'estima calculant la fase de la diferència de les transformades de Fourier entre dos frames consecutius.

Arribats en aquest punt, tenim ja la base per accedir als tres passos principals del nostre mètode de processament d'imatges IVUS, com es mostra a la figura 5.

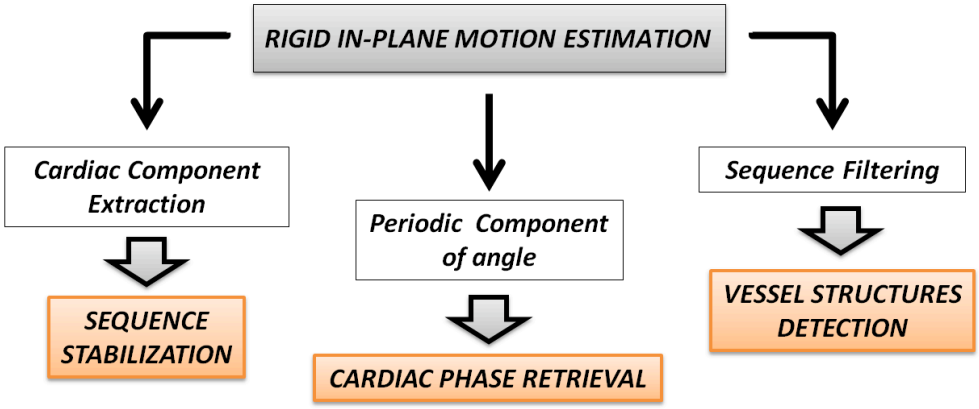


Figura 5: Principals passos que segueix el mètode

Corregint els termes dinàmics de la translació i la rotació anteriorment calculada per cada imatge de la seqüència ens permet obtenir una de les primeres solucions que el mètode ofereix: la **seqüència estabilitzada** d'imatges IVUS.

D'altra banda, podem extreure el senyal que reflexa el moviment cardíac de l'angle de rotació calculat en l'aproximació del moviment del cos rígid en el pla. Filtrant aquest senyal i extraient-ne

els punts màxims i mínims podem conèixer la **fase cardíaca** en que es troba cada imatge de la seqüència.

Això ens permet descartar els frames corresponents al moviment involuntari del catèter per tal d'obtenir un **tall longitudinal** amb la selecció d'imatges corregida.

Finalment, l'estudi de la dinàmica cardíaca ens ha permès millorar també la **detecció de les estructures del vas.** Ara, gràcies a que les imatges obtingudes a l'aplicar la compensació calculada per l'aproximació del moviment del cos rígid en el pla es troben alineades, podem reduir dràsticament el temps de pre-processament de les imatges, el que fins ara era el pas més lent i costós de l'algoritme (Figura 6).

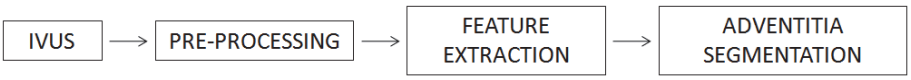


Figura 6: Diagrama de flux per l'obtenció de la capa adventitia del vas

2.2. Implementació del mètode

La implementació pràctica d'aquest mètode es troba feta en l'entorn MATLAB, per ser una eina amb forta base matemàtica i que facilita treballar amb matrius i per extensió, amb imatges.

El codi es troba dividit en múltiples funcions, entre elles les funcions principals d'entrada per les tres eines principals, ben diferenciades: "Estabilització de la seqüència", "Extracció de la fase cardíaca" i "Detecció de les estructures del vas". Comprèn, a més, la generació de les imatges del tall longitudinal del vas. Els únics paràmetres bàsics que requereixen aquestes funcions són el directori on es troben les imatges i, assumint que cada imatge té un nom comú i està enumerada segons l'ordre d'extracció, el nom comú, el primer número de la seqüència a analitzar i el darrer.

Les imatges resultants es generen automàticament dins del mateix directori on les imatges originals es troben, seguint sempre la mateixa convenció d'un nom comú seguit de la numeració corresponent.

A mode de funcionalitat extra, disposem també de dues funcions de pre-processament per netejar les imatges IVUS originals segons s'obtenen de l'aparell que processa l'Ecografia Intracoronària. Generalment aquests aparells afegeixen a les imatges informació escrita o marques per sobre del vas. És necessari pel correcte funcionament del mètode exposat, disposar d'imatges on únicament es presenti el vas, ja que qualsevol element extra afecta la distribució d'intensitats dins la imatge i pot distorsionar negativament els càlculs que es basen en aquesta (Figura 7).

Ens trobem, doncs, davant una implementació completa del mètode, si bé en aquest punt, encara no hem solucionat el principal problema: la manca d'un entorn amigable i de fàcil accés per qui ha de ser l'usuari final d'aquestes eines.

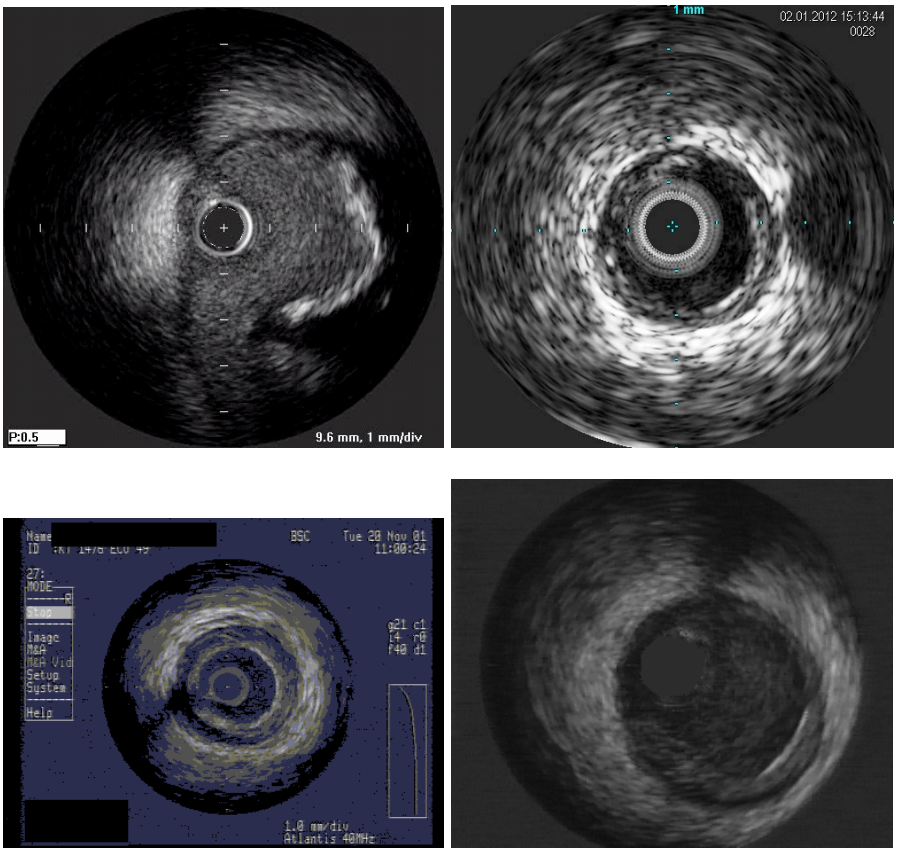


Figura 7: A dalt a l'esquerra, imatge generada per un aparell Boston Galaxy, a dalt a la dreta, imatge generada per un aparell Volcano, a baix a l'esquerra una imatge generada per un aparell Boston Clear i a baix a la dreta, exemple d'imatge neta després de ser pre-processada.

2.3. Necessitat d'una interfície gràfica.

MATLAB és a l'hora un llenguatge de programació d'alt nivell i un entorn pel càlcul numèric, la visualització i la programació. Es tracta d'una eina molt potent i molt utilitzada en universitats i centres d'investigació i desenvolupament. El seu ús es basa en l'execució de comandes via un terminal i la programació d'scripts i funcions.

En el nostre cas, i aprofitant el fet que totes les funcionalitats principals reben similars paràmetres d'entrada, hem creat un punt d'inici únic per totes les eines que el mètode ofereix, de manera que, un cop definits els paràmetres comuns (directori amb les imatges originals, nom de els imatges i numeració) es mostra un senzill menú d'opcions on podem executar, d'una manera un xic més fàcil, les eines que vulguem (Figura 8).

Tot i així, la seva execució encara requereix uns coneixements bàsics de programació i d'utilització de la plataforma MATLAB, coneixements que gran part dels usuaris finals d'aquest mètode, concretament metges, clínics i professionals del sector mèdic, no disposen.

Es fa palès, de nou, que és necessari, proporcionar una interfície gràfica que doni accés a totes les eines i funcionalitats que el mètode proposa, de forma fàcil, entenedora i senzilla.

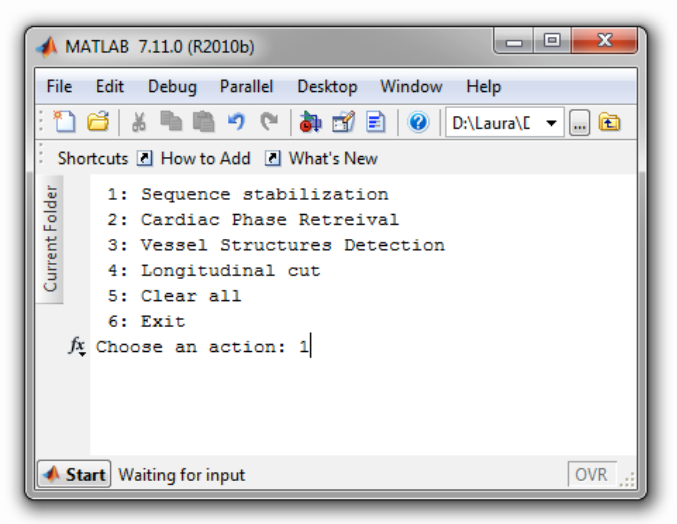


Figura 8: Senzill menú per agilitzar l'execució de les diferent eines que el mètode proposa

3. Disseny de l'aplicació

La interacció entre humans i ordenadors és un concepte relativament nou, però ràpidament ha guanyat importància degut a la gran quantitat d'aparells electrònics que, de fa uns anys en endavant, formen part del nostre dia a dia.

Oferir una interfície atractiva i agradable a la vista a la vegada que intuïtiva i de senzill ús s'ha convertit en un dels punts més importants en del disseny d'aplicacions d'avui en dia. A més, l'usuari mitjà actual disposa de temps limitat per aprendre a fer servir noves eines: és vital que les noves aplicacions tinguin el màxim de coses en comú amb elements que sabem que l'usuari coneix i fa servir sovint, com per exemple el sistema operatiu, per tal que el consumidor pugui prendre els primers passos sense ajuda i es senti en un entorn familiar.

Una nova disciplina, la de **la interacció humà-ordinador** (Human–computer interaction o HCI en anglès), ha sorgit per estudiar la planificació, disseny i implementació de les interfícies de comunicació entre persones i ordenadors. La naturalesa multi-disciplinar d'aquest camp atrau tant a estudiosos d'àrees humanístiques i socials com dissenyadors, psicòlegs i estudiosos del comportament, com perfils més tècnics de l'entorn de la enginyeria o la informàtica.

Podem definir interfície d'usuari com aquell espai que permet la interacció entre humans i màquines. En el cas concret dels ordenadors, anomenem interfície gràfica d'usuari (en anglès Graphical User Interface o GUI) aquella interfície que permet als usuaris interaccionar amb una computadora mitjançant l'ús d'imatges, símbols, metàfores visuals o dispositius apuntadors, entre altres. A dia d'avui, les interfícies gràfiques han substituït, en la majoria dels casos, la interacció a través de comandes de text.

3.1. Especificacions de l'aplicació

Disposem de tres eines de processament d'imatges IVUS que volem fer arribar, de forma fàcilment distribuïble, a la major quantitat de plataformes possibles mitjançant una interfície senzilla que qualsevol persona amb coneixements bàsics d'ús d'un ordenador pugui fer servir.

3.1.1 Perfil d'usuari

L'aplicació esta especialment dirigida a clínics, metges i professionals de la medicina que tenen accés als resultats de les ecografies intracoronàries i necessiten analitzar-les per poder diagnosticar una possible malaltia o controlar-ne l'estat. El perfil d'usuari al que ens adrecem és molt divers, avarca un gran rang d'edat (dels 25 anys als 60 anys aproximadament), té caràcter internacional i pot trobar-se en diferents nivells pel que fa a l'habilitat per utilitzar un ordenador. El punt comú és un profund coneixement de les imatges d'ecografia intracoronària.

Ens hem centrat, doncs, en poder entregar una aplicació que sigui suficientment intuïtiva i senzilla com per que un usuari amb un nivell baix de coneixements informàtics la pugui fer servir sense necessitat d'ajuda externa.

3.1.2 Requeriments de l'aplicació

La funcionalitat bàsica és la de poder visualitzar les imatges obtingudes mitjançant l'ecografia intracoronària de forma ràpida i amb la possibilitat de fer ús de les millores que el mètode de processament de les imatges ens ofereix (seqüència estabilitzada, tall longitudinal i detecció de contorns). Degut a que aquest processament és costós i lent (especialment per un gran volum d'imatges), volem que l'usuari només hagi de dur-lo a terme un cop per cada seqüència d'imatges IVUS.

Així, hem decidit fer servir el terme "**Cas IVUS**" per fer referència a la unitat bàsica amb la que l'aplicació treballa, de la mateixa manera que Microsoft Word treballa amb "Documents" i Microsoft Excel treballa amb "Fulls de càlcul". Un Cas IVUS conté no només la seqüència d'imatges a processar, sinó també un seguit d'informació relacionada que, d'una banda, pot ajudar a identificar la seqüència més tard i, per l'altra, por aportar significat extra a l'estudi de les imatges.

Els requeriments bàsics de l'aplicació són, doncs, els següents (Figura 9):

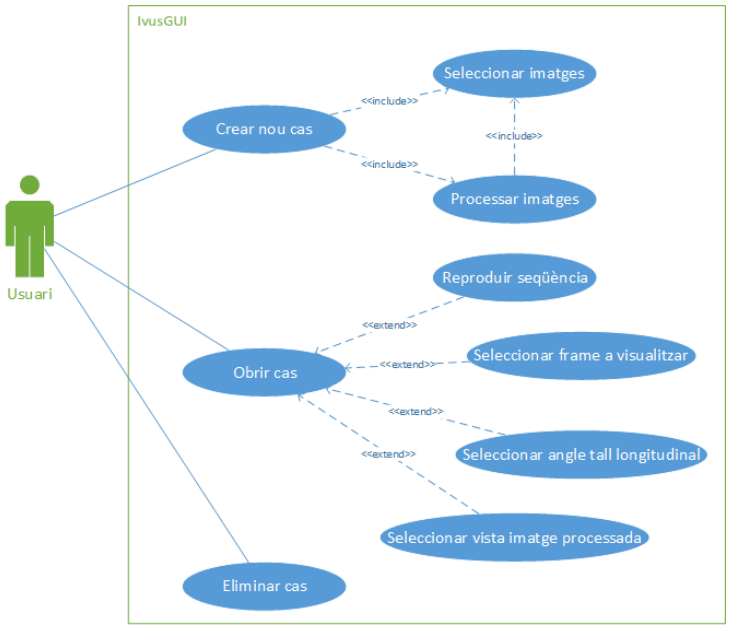


Figura 9: Diagrama de casos d'ús

- **Crear nou cas IVUS:** Es guiarà a l'usuari durant el procés de creació del nou cas, que es compondrà dels següents passos:
    - Introduir informació relacionada: títol i data del cas juntament amb sexe i edat del pacient
    - Seleccionar les imatges a processar
- **Carregar cas IVUS:** Es mostrarà un llistat de tots els casos prèviament creats i l'usuari podrà decidir quin carregar. Un cop carregat, es visualitzaran les imatges i es mostraran les següents opcions:
    - Reproducció de la seqüència d'imatges, tant original com processada, del primer a l'últim frame.

- Selecció de la imatge processada a visualitzar: a elegir entre la imatge estabilitzada, la imatge original amb detecció de contorn o la imatge estabilitzada amb detecció de contorn.

- Selecció de l'angle del tall longitudinal

- Selecció de quina imatge, dins de la seqüència, volem visualitzar.

- **Eliminar cas IVUS:** Es mostrarà un llistat de tots els casos prèviament creats i l'usuari podrà decidir quin eliminar.

S'afegeixen també, durant el procés de disseny de l'aplicació, una sèrie de funcionalitats opcionals:

- **Suport per arxius DICOM:** L'estàndard "Digital Imaging and Communications in Medicine" (DICOM) s'utilitza per guardar i transmetre informació relacionada amb la imatge mèdica. Es tracta d'un estàndard molt usat pels aparells d'Ecografia intracoronària, que d'aquesta manera poden exportar les imatges juntament amb informació del pacient en un sol arxiu. Amb aquesta funcionalitat, al crear un nou cas, l'usuari tindrà l'opció d'extreure les imatges IVUS a processar d'un arxiu DICOM.

- **Canvi de l'idioma de l'aplicació:** L'usuari podrà triar amb quin idioma vol fer servir l'aplicació

- **Neteja de les imatges IVUS:** L'usuari podrà decidir quin mètode de pre-processament es necessari aplicar a la seva seqüència d'imatges.

- **Recol·locació manual del contorn del vas:** El contorn automàticament detectat per l'aplicació podrà ser modificat per l'usuari a posteriori, si aquest ho creu necessari.

- **Editar cas IVUS:** Es mostrarà un llistat de tots els casos prèviament creats i l'usuari podrà decidir quin editar. Es mostrarà tota la informació on l'usuari pot aplicar canvis, juntament amb l'opció d'actualitzar el cas amb la nova informació.

Per clarificar les funcionalitats de l'aplicació i obtenir una primera idea de l'estructura gràfica d'aquesta, s'han creat una sèrie de maquetes interactives, que han ajudat a definir l'organització dels diferents elements de l'aplicació (Figura 10), així com detectar errors d'usabilitat o trobar noves funcionalitats. L'última versió de la maqueta es pot trobar a l'ANNEX I.

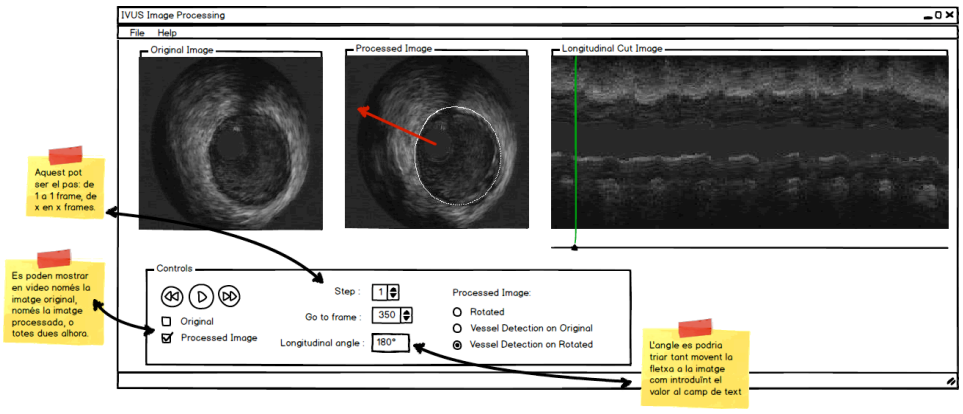


Figura 10: Maqueta de la pantalla principal de l'aplicació, amb un cas carregat

4. Detalls tècnics de la implementació

Pel que fa als requeriments tècnics, volem, d'una banda, arribar a la major varietat de plataformes possibles i, de l'altra, crear una aplicació fàcilment distribuïble. Busquem doncs, fer servir un plegat d'eines gratuïtes i cross-platform.

Volem crear una aplicació d'escriptori, que s'integri amb el Sistema Operatiu hoste el millor possible: l'aplicació ha d'oferir el mateix look and feel que l'entorn en que es troba per tal de fer el seu ús més còmode i familiar a l'usuari.

La plataforma original amb la que programarem l'aplicació i generarem la primera versió serà Windows, per ser un dels sistemes operatius més usats, especialment pel tipus d'usuari amb coneixements baixos/mitjans d'informàtica al que ens dirigim.

4.1. Eines

Per la implementació decidim fer servir **C++** com a llenguatge de programació, per ser un llenguatge amb el que hem treballat prèviament i que compleix els nostres requisits. És tracta d'un llenguatge molt popular i que es troba implementat en totes les plataformes a les que volem arribar, cosa que ens facilita la futura portabilitat de l'aplicació. Ens ofereix també, en aquest cas concret, la possibilitat d'interaccionar amb MATLAB, on tenim ja el mètode de processament d'imatges implementat, ja que aquest facilita la integració amb aplicacions C/C++ ja sigui mitjançant la traducció semi-automàtica de codi MATLAB a C/C++ o l'exportació d'aquest com a llibreria externa.

Respecte a la creació de la interfície gràfica, cada plataforma ofereix les llibreries o caixa d'eines necessàries per programar-hi, concretament MFC per Windows, Cocoa o Carbon per MacOS X i GTK+ per Linux. És recomanable, en general, fer ús d'aquestes eines ofertes pels mateixos sistemes operatius ja que assegura la continuïtat visual i minimitza els errors. En el nostre cas, però, busquem la forma de programar la interfície un sol cop de manera que el port a altres plataformes sigui possible fent servir el mateix codi i minimitzant els canvis en la programació. Afortunadament, existeixen frameworks que faciliten aquesta tasca, oferint un conjunt únic d'eines i llibreries que a l'executar-se, s'adapten a l'entorn de la plataforma on es troben. Les principals i més populars opcions en C++ són el GT Project [9] i wxWidgets [10]. Tots dos frameworks es troben en constant desenvolupament, disposen d'una documentació amplia i detallada, un bon catàleg d'exemples i una gran comunitat d'usuaris i desenvolupadors al darrera. Per oferir una millor experiència cross-platform i ser un xic més senzill de cara a programadors novells en interfícies gràfiques, ens hem decantat finalment per l'ús de **wxWidgets** en la seva última versió estable, la 2.8.12. També, per recomanació del mateix framework i per ser una eina ja coneguda hem triat **Microsoft Visual Studio 2008** com a entorn integrat de desenvolupament (IDE).

Pel que fa a la programació del mètode de processament d'imatges, com hem explicat al capítol 2, ja esta totalment implementada en **MATLAB** i és completament funcional. Implementar-la de nou en C++ no és opció ja que, per una banda, es tracta d'un mètode complex: tornar-lo implementar de zero en un altre llenguatge seria un projecte en si mateix. Per l'altra, MATLAB és un entorn optimitzat pel càlcul de tot tipus d'expressions matemàtiques i pel processament d'imatges, per tant té sentit mantenir el mètode en la seva implementació original ja que difícilment en podríem millorar el resultat. Així doncs, i gràcies a que s'ofereixen eines per la

integració amb C/C++, el mòdul de processament d'imatges de la nostra aplicació es manté en MATLAB.

Finalment, i ja que segons els requeriments és necessari guardar de forma permanent informació respecte als casos IVUS, necessitem un sistema d'emmagatzemament que treballi de forma local, i que sense ser computacionalment pesat ens ofereixi suficient flexibilitat com per guardar i recuperar informació fàcilment. Necessitem, per tant, una versió "light" del que generalment es coneix per base de dades, i l'eina **SQLite** ens ho ofereix. Es tracta d'un sistema de gestió de bases de dades relacional, que implementa la majoria de l'estàndard SQL i no necessita cap tipus d'instal·lació o configuració prèvia. A diferencia d'altres gestors de bases de dades, s'executa de forma local a la mateixa màquina que la nostra aplicació, i guarda la informació en un sol arxiu d'extensió .db. Es presenta en forma de llibreria en C i per tant pot ser compilada en pràcticament qualsevol plataforma, pel que compleix el nostre requisit de ser cross-platform.

4.2. Integració de MATLAB i C++

MATLAB ofereix dos eines generals per integrar el seu sistema amb el llenguatges de programació C i C++: MATLAB Compiler i MATLAB Coder (Figura 11).

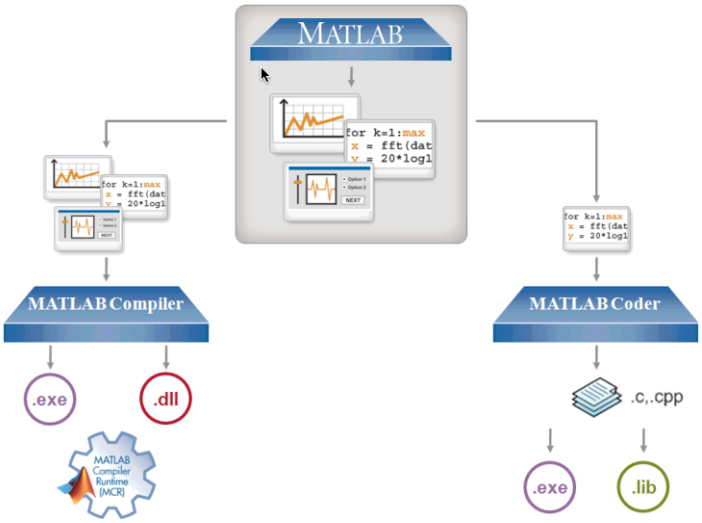


Figura 11: Integració de MATLAB amb C/C++

D'una banda, MATLAB Compiler és capaç d'exportar qualsevol programa implementat en l'entorn MATLAB com una aplicació independent o com una llibreria externa. Es tracta d'un procés totalment automatitzat en que l'usuari només ha de proporcionar la funció a exportar i, en cas de ser necessari, els paràmetres per executar-la. MATLAB Compiler s'encarrega de localitzar i incloure totes les funcions aniuades dins la funció original. Aquesta opció, però té un requeriment especial: necessita la presència del MATLAB Compiler Runtime (MCR) a la màquina on s'executa la funció. Aquells ordenadors amb l'aplicació MATLAB instal·lada ja tenen aquesta funcionalitat inclosa, però en el cas contrari, és necessària la instal·lació expressa del MCR. Afortunadament, MathWorks (l'empresa propietària) distribueix instal·ladors de MCR amb cada còpia de MATLAB i, a més, en permet la lliure distribució.

D'altra banda, MATLAB Coder genera codi C o C++ a partir de codi MATLAB. El codi generat és totalment independent de la plataforma MATLAB, de manera que pot ser inclòs en qualsevol projecte C/C++ existent o pot ser compilat directament en una aplicació o en una llibreria externa. Degut a les importants diferències entre els llenguatges de programació MATLAB i

C/C++, MATLAB Coder necessita, primerament, que durant la programació de l'aplicació original es tinguin en compte aquestes diferencies (Figura 12) i en segon lloc, que durant el procés de generació del codi l'usuari defineixi el tipus dels paràmetres d'entrada de les funcions. És imperatiu, doncs, tenir en compte que es farà servir aquesta opció des del començament de la creació de l'aplicació MATLAB original.

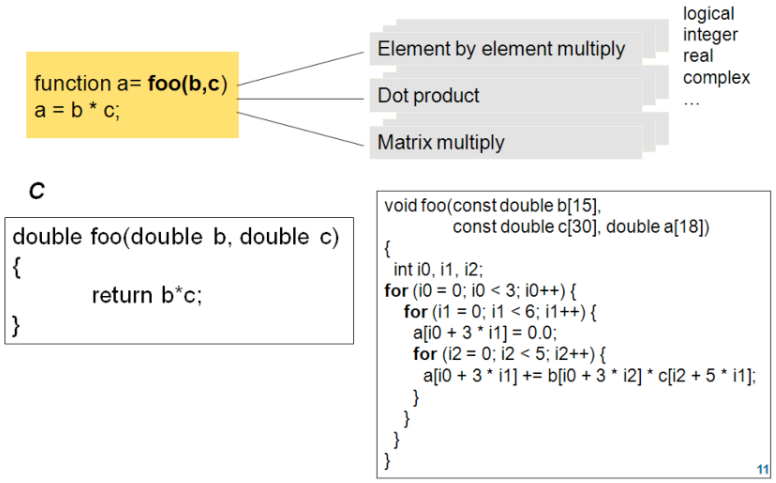


Figura 12: Una simple funció de MATLAB (quadre groc), gràcies al polimorfisme del llenguatge, pot acceptar diferents tipus de paràmetres i modificar el seu comportament en funció d'aquests. En C/C++ (quadres blancs), en canvi, és necessària una funció diferent per cada tipus i comportament específic.

És per aquesta última raó, que MATLAB Coder deixa de ser una opció factible en el nostre cas: la implementació MATLAB del mètode de processament d'imatges IVUS ja esta totalment finalitzada, i comprèn un gran nombre diferent de funcions, a més d'aprofitar totes aquelles funcionalitats que MATLAB ofereix però que C/C++ no suporta.

Així doncs, farem ús de **MATLAB Compiler** per generar llibreries externes de les principals funcionalitats que farem servir i que més endavant enllaçarem amb la nostra aplicació C++.

4.3. wxWidgets

wxWidgets [11] és basa en la programació orientada a objectes, i ofereix un gran ventall de classes, cadascuna d'elles amb un objectiu i unes funcionalitats molt clares. En general, podem dividir-les en tres grans grups, segons el seu comportament: les classes helper, els elements gràfics i els events.

Definim com una classe helper aquella que proporciona una funcionalitat extra, però que no esta relacionada directament amb l'objectiu principal de l'aplicació; en el nostre cas, totes aquelles que no tenen una funcionalitat específica dins de la interfície gràfica. És tracta de classes com wxString, que engloba el string o cadena de caràcters amb un conjunt de funcionalitats típiques relacionades amb aquest o wxArray que representa un array d'objectes i que inclou procediments per afegir, eliminar, cercar, ordenar, etc. Oferir aquest tipus de classes helper és molt comú en frameworks de programari, però, en el cas que ens ocupa, es presenta un motiu de més: el suport cross-platform. Degut a l'alta portabilitat a varies plataformes que wxWidgets ofereix (Figura 13), és necessari que controli i adapti, no només la part gràfica de l'aplicació, sinó també tota la resta. Així, mentre el programador instancia la classe wxString per treballar amb cadenes de caràcters, es possible que, interiorment, wxWidgets hagi de canviar la implementació o la llibreria en que es basa aquesta classe en funció de la plataforma on s'està executant. És recomanat, i en moltes

ocasions fins i tot obligatori, l'ús d'aquestes classes helper; no només per que ofereixen funcionalitats que, d'altra manera s'haurien d'implementar de zero, sinó per que asseguren la compatibilitat cross-platform.

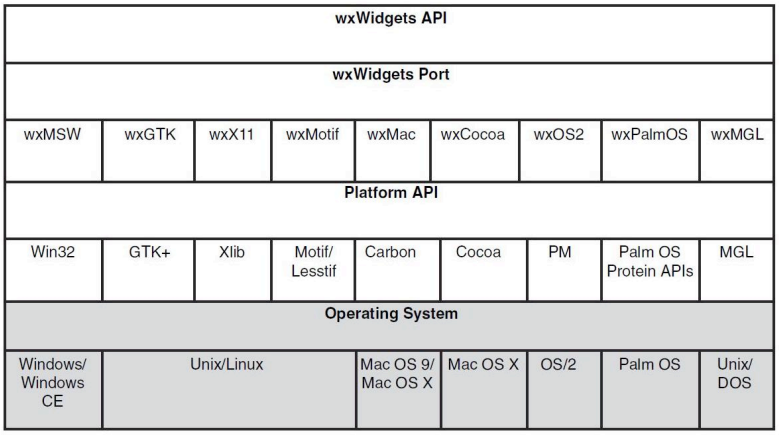


Figura 13: Taula que mostra les quatre capes conceptuals que integren wxWidgets: la API pública, és a dir, el punt d'entrada del programador, els principals ports, la plataforma API usada pel corresponent port i finalment, el sistema operatiu.

Pel que fa als elements gràfics, els podem separar en dos grans grups: finestres (o windows) i controls. Visualment, i de forma general, podem definir control com un element que ens permet fer efectuar una acció directa amb l'aplicació, com ara un botó o un selector i finestra com una àrea rectangular, de mida variable, que fa de contenidor d'altres elements com controls o d'altres finestres. El concepte de finestra, però, pot generar confusió ja que, dins del context del framework, no representa exactament el mateix concepte al que nosaltres estem acostumats. wxWidgets defineix wxWindow com la classe base de qualsevol element que sigui visible: tant pot ser un diàleg, un botó, una barra de menús... Així, a nivell d'implementació, un control és una finestra, ja que la classe wxControl deriva de wxWindow, però a nivell visual encara podem diferenciar entre una finestra i un control com elements no relacionats i amb objectius diferents.

Explicat de forma esquemàtica, doncs, la programació a nivell gràfic de l'aplicació es basa en incloure, dins d'una finestra, el seguit de controls o altres finestres que siguin necessaris per tal de dur a terme una funció específica (Figura 14). Per controlar la disposició espacial d'aquests elements dins la finestra es fan servir les anomenades classes "sizers", que ajuden a definir la posició d'un element dins de a finestra i respecte a la resta.

Pel que fa als events, wxWidgets es basa, com la majoria de GUIs, en un bucle, anomenat event loop, que bàsicament espera a que l'usuari interaccioni amb els elements gràfics que s'ofereixen, gestionant, quan això passa, l'event i redirigint-lo a les parts responsables de dur a terme les accions requerides. A la pràctica, els events no només es generen per l'usuari, també n'hi han d'autogenerats per l'aplicació, com el refresc d'una finestra cada cert interval, i generats per elements externs, com per exemple d'una connexió via socket.

Es trobem davant del que s'anomena una "single-threaded GUI", és a dir, una interfície gràfica que disposa tant sols d'un fil d'execució pel bucle principal i, per tant, serveix els events rebuts d'un a un. La creació d'altres threads és possible, sempre i quant no es relacionin amb cap element de la GUI. Evidentment, en una interfície gràfica, pràcticament totes les funcions actuen, en un moment o altre, sobre alguna peça amb representació gràfica, de manera que els threads només es poden acabar fent servir en un nombre molt reduït d'ocasions.

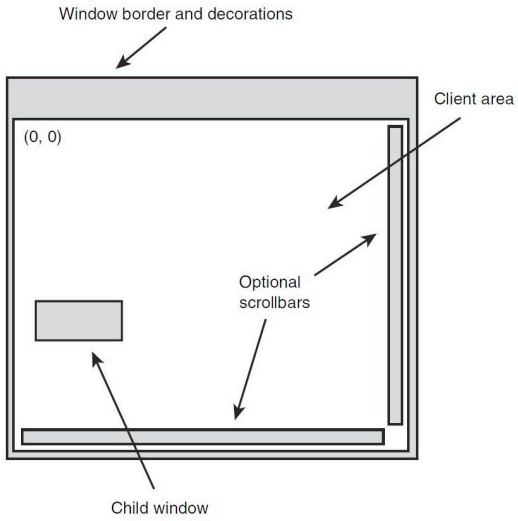


Figura 14: Elements d'una finestra, o window.

Per tant, els events són la base d'una aplicació wxWidgets, i el framework ofereix al programador varies formes en que aquest els pot generar o intervenir.

La més comú és fer servir la anomenada "taula estàtica d'events" per crear un vincle entre un event i la funció que volem executar quan aquest s'activi. La "taula" no és més que una macro (fragment de codi als que se li ha assignat un nom, de manera que cada cop que es fa servir el nom, aquest es substituït pel fragment de codi) dissenyada de forma que simuli l'estructura visual d'una taula al ser definida. A l'exemple següent podem veure una taula d'events d'un frame (un tipus de finestra) on s'intervenen els events generats per dos opcions d'un menú (Exit i About), l'event generat al canviar la mida de la finestra i l'event generat al prémer el botó OK.

```
BEGIN_EVENT_TABLE(MyFrame, wxFrame)
  EVT_MENU (wxID_ABOUT, MyFrame::OnAbout)
  EVT_MENU (wxID_EXIT, MyFrame::OnQuit)
  EVT_SIZE ( MyFrame::OnSize)
  EVT_BUTTON (wxID_OK, MyFrame::OnButtonOK)
END_EVENT_TABLE()
```

5. La nostra aplicació: IvusGUI

Al final del procés d'implementació hem obtingut l'aplicació final, que hem anomenat IvusGUI. A continuació especifiquem tots els detalls i funcionalitats d'aquesta.

5.1. Emmagatzemament de la informació

Un dels requeriments essencials del programa és poder guardar la informació relativa als casos IVUS, així com les imatges ja processades, per tal que l'usuari pugui tornar a revisar el cas fàcil i ràpidament. Donat que, en aquesta primera versió de l'aplicació ens centrem en el processament de les imatges, només guardem un conjunt mínim de camps: nom o títol el cas, data i edat i sexe del pacient. Cada cas tindrà, a més, un identificador numèric únic.

Pel que fa a les imatges, ens hem decantat per guardar-les en una carpeta anomenada IVUS dins del directori de documents de l'usuari que per defecte la majoria de sistemes operatius tenen. Per tal d'identificar a quin cas pertanyen, al crear un nou cas, generem una nova carpeta anomenada de la forma "<id>_<títol del cas>" on hi guardem la seqüència d'imatges original i, més endavant durant el processament, hi disposem la resta d'imatges processades, entre d'altres (Figura 14). D'aquesta manera, encara que l'usuari perdi la seva seqüència d'imatges original, el cas encara serà completament visible i funcional en la nostra aplicació.

Tal i com hem indicat al capítol anterior, hem seleccionat una base de dades SQLite per emmagatzemar la informació. De moment, donat el reduït nombre de camps, es guarden totes les dades en una sola taula anomenada "Cases" (Figura 15). Cada cop que necessitem interaccionar amb la BD i rebem un error de connexió, creem una base de dades i una taula nova. Així, si l'arxiu generat per SQLite s'esborra per accident o es perd per qualsevol motiu l'aplicació pot continuar funcionant correctament, tot i haver perdut la informació anterior.

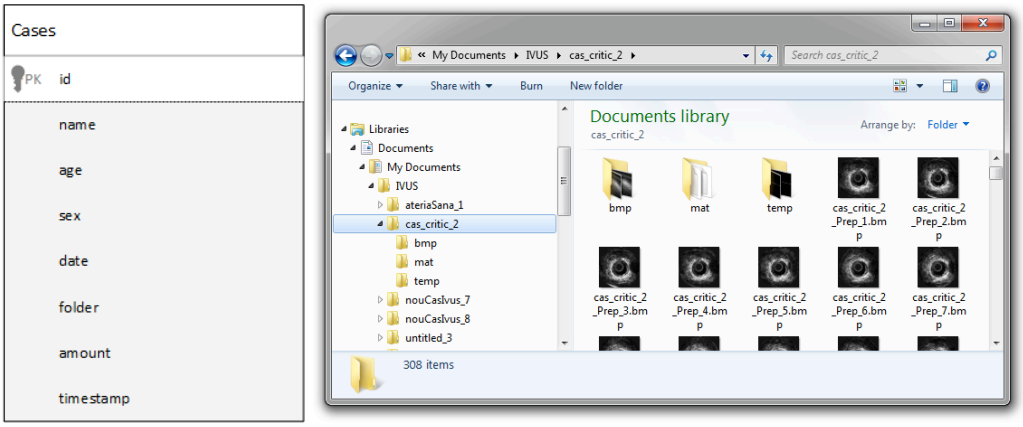


Figura 15: Diagrama de la base de dades (esquerra) i vista de l'organització de les carpetes on es guarden les imatges i altres arxius generats pel processament de la seqüència.

5.2. Distribució del programa

IvusGui es presenta en la forma d'un arxiu executable: **IvusGUI.exe**. Acompanyant-lo dins del mateix directori es troben els següents arxius:

- **seqStab.dll, vesselDet.dll, cardiacPhase.dll, longSect.dll, volcano.dll, boston.dll:** Llibreries MATLAB generades amb MATLAB Compiler.

- **sample.xpm:** arxiu d'imatge de tipus X Pixmap utilitzat com icona de l'aplicació

- **splash.bmp:** imatge que apareix just a l'iniciar l'aplicació.

- **help.pdf:** Document .pdf que conté l'ajuda del programa.

- **MCRInstaller.exe:** Instal·lador del MATLAB Compiler Runtime (MCR)

- **instruccions.pdf:** Document .pdf que conté les instruccions de com executar l'aplicació i, en cas que sigui necessari, com instal·lar el MCR.

Un cop executada l'aplicació per primer cop, apareixeran dos arxius més:

- **IVUSCase.db:** Arxiu generat per la base de dades (es crea a partir del primer cas creat amb èxit)

- **error.txt:** arxiu de text on l'aplicació grava, en cas d'error, els corresponent missatge explicatiu. Es buida a cada execució.

5.3. Funcionalitats de l'aplicació

A l'executar IvusGUI.exe el primer que ens apareix és una splash-screen amb una imatge de l'aplicació. Aquesta pantalla es manté durant uns quants segons fins que desapareix i dona a pas a la nostra pantalla principal. Les splash-screen es solen fer servir per mantenir a l'usuari en espera mentre l'aplicació s'inicialitza; en el nostre cas, si bé l'aplicació per si mateixa no necessita cap inicialització, hem aprofitat per arrancar l'entorn MATLAB i les corresponents llibreries.

Mitjançant la crida a la funció **mclInitializeApplication(const char **options, int count)** posem en marxa l'entorn MCR, en el nostre cas sense cap opció especial. De manera similar, i en cas que la crida anterior hagi tingut èxit, hem de crear una instància del MCR per cada llibreria externa generada amb el MATLAB Compiler mitjançant la crida a la funció **<nomdelallibreria>Initialize()** (en el nostre cas: seqStabInitialize(), cardiacPhaseInitialize(), etc...). El motiu pel que duem a terme aquestes inicialitzacions tant aviat, fins i tot abans de que hagi sorgit la necessitat real de fer cap processament, és que el MCR només es pot inicialitzar un cop per execució de l'aplicació. Tenint en compte que la crida a les funcions MATLAB és, probablement, la part més lenta i costosa de tota l'aplicació, hem volgut evitar haver de sumar-hi, a més, la gestió de les inicialitzacions.

Les llibreries i l'entorn MCR estan sempre preparades, doncs, es facin servir o no. Durant la destrucció de l'aplicació aprofitem per cridar als corresponents mètodes **<nomdelallibreria>Terminate()** per cada llibreria i, finalment, a **mclTerminateApplication()**.

5.3.1 Pantalla principal: inici

De bon començament, la pantalla inicial es presenta buida, a excepció d'una barra d'estat (a la part inferior) i una barra de menú (a la part superior) amb dos opcions: **File** i **Help** (Figura 16).

Aquest és el frame principal de l'aplicació: si el tanquem, finalitza també el programa. Es tracta d'una "top level window" ja que no té pare i estarà, per tant, sempre present visualment, ja sigui en primer o segon pla. Serà, també, l'element de referència de la barra de tasques del sistema operatiu.

Des d'aquí podem accedir a totes les funcionalitats de l'aplicació, que s'aniran desplegant en diferents finestres i diàlegs, sempre mantenint aquest frame com a pare (veure diagrama de pares-fills a l'ANNEX II).

El frame conté, a més, un gestor de finestres tipus notebook (finestres que s'agrupen mitjançant pestanyes o tabs). No es pot veure tot just al arrancar l'aplicació per que no disposa de cap pestanya per mostrar, però a mesura que anem creant i obrint casos IVUS el podrem fer servir per navegar entre les diferents finestres.

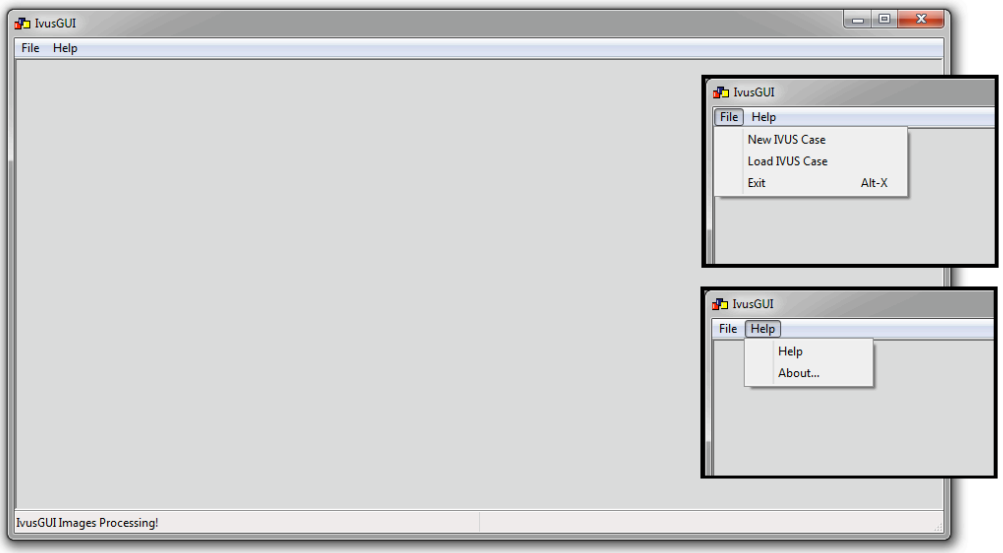


Figura 16: Pantalla principal i menús

5.3.2 Crear nou cas IVUS

Seleccionant l'opció "New IVUS Case" del menú "File" s'obrirà un diàleg en forma modal, és a dir, que no podrem tornar a interaccionar amb cap altre element de l'aplicació fins que no haguem acabat de treballar amb ell (Figura 17). Aquest diàleg ens presentarà un formulari que haurem d'omplir amb la informació relativa al nou cas que volem crear. Hem fet ús de quatre tipus de controls diferents intentant oferir sempre el més adient pel tipus de dades a introduir:

- **Camp de text** pel títol
- **Spin Control** per l'edat
- **Llista desplegable** per seleccionar el sexe
- **Calendari** per seleccionar una data

D'aquesta manera, en tres dels quatre casos no ens hem de preocupar de validar l'input de l'usuari: tant amb la llista desplegable com amb el calendari no donen marge a modificació i l'Spin

Control es defineix amb uns valors mínims i màxims de manera que ignorarà qualsevol valor fora del rang.

Ens queda, però, validar el camp més important: el títol. En general, voldrem que el títol sigui una cadena de caràcters només alfanumèrics, acceptant com a excepció els espais en blanc per la separació de paraules. No podem acceptar cap altre tipus de caràcter ja que fem servir el títol a l'hora d'anomenar el directori con guardem els resultats del processament. wxWidgets disposa d'una bona selecció de validadors de text, però cap d'ells compleix els requisits: o bé accepten massa caràcters o bé no accepten l'espai en blanc. Finalment, doncs, hem implementat el nostre propi validador, on comprovem que el text compleixi els requisits.

Un cop omplert correctament el formulari, hem de seleccionar la carpeta on es troba la seqüència d'imatges IVUS (Figura 17) mitjançant un diàleg estàndard de cerca de carpeta, que s'obre per damunt de l'actual de forma modal.

Fem servir un selector de carpeta enlloc d'un selector d'arxius per que, generalment, el software que extrau les imatges d'una ecografia intracoronària, les desa totes, de forma numerada segons l'ordre d'extracció, en una mateixa carpeta. En el següent pas tindrem l'oportunitat de seleccionar quines imatges, d'entre tot el conjunt, volem processar.

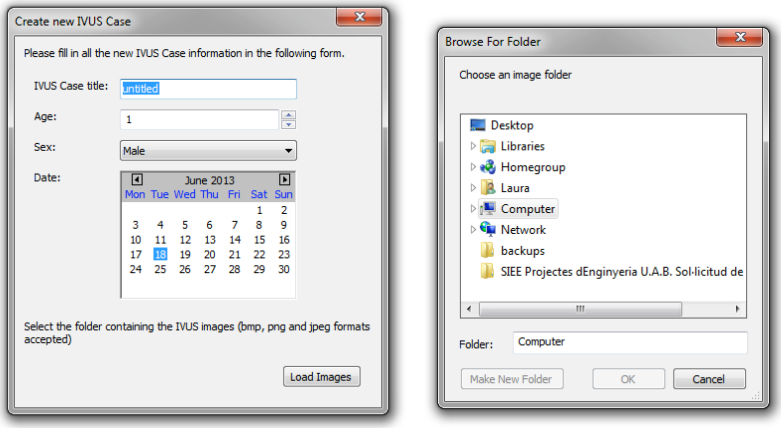


Figura 17: Diàleg per introduir la informació del cas (esquerra) i diàleg per seleccionar la carpeta on tenim la seqüència d'imatges a processar (dreta).

5.3.3 Seleccionar imatges a processar

Un cop triada la carpeta contenidora de les imatges, i en el cas que contingui els tipus de fitxers que suportem (bmp, png i jpeg), tanquem els dos diàlegs anteriors i n'obrim un de nou: el selector d'imatges (Figura 18).

Es disposen totes les imatges trobades al directori seleccionat en horitzontal, amb l'ajuda d'una barra de scroll en el cas que sigui necessari, ordenades mitjançant un algorisme d'ordenació natural (veure punt 5.4.2). Es recomana seleccionar seqüències d'imatges consecutives; és per això que hem definit una galeria d'imatges horitzontal, que emfatitza la idea de que hi ha una correlació entre les imatges. L'aplicació no ofereix cap limitació d'ordre o veïnatge a l'hora de seleccionar les imatges, per tant l'usuari disposa de total llibertat per elegir.

Les imatges es pinten dintre d'un rectangle que comprèn la imatge i el nom d'arxiu, i es redimensionen a la mida thumbnail definida pel rectangle. El color de fons del rectangle indica si

la imatge esta, o no seleccionada: fons blanc (igual que el fons de la finestra) per imatges no seleccionades, fons gris per les que si que ho estan.

Per defecte totes les imatges es troben sense seleccionar. L'usuari pot seleccionar-les via els botons de "Select All" i "Deselect All" i amb l'ús del ratolí. En aquest últim cas, hem implementat el mateix mecanisme que el que ofereixen la majoria de sistemes operatius en els seus d'exploradors de fitxers: un click per seleccionar/deseleccionar una imatge, click amb la tecla shift pressionada per seleccionar un rang i click més la tecla ctrl pressionada per afegir una imatge individual a la selecció (veure punt 5.4.3).

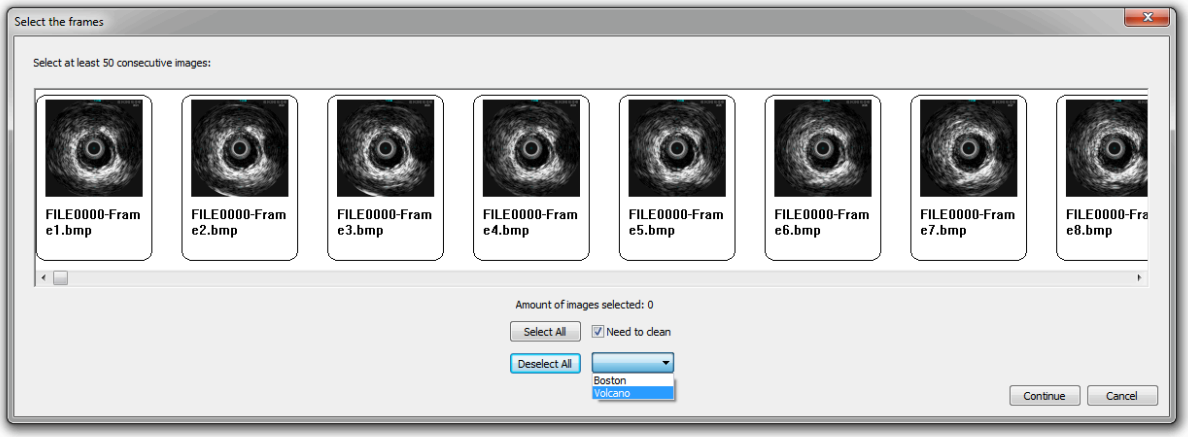


Figura 18: Diàleg selector d'imatges

Un cop triades les imatges, si considerem que és necessari netejar-les abans d'iniciar el processament, podem triar un dels dos mètodes de pre-processament que s'ofereixen: Volcano o Boston.

Si en qualsevol moment premem el botó de cancel·lar, o tanquem el diàleg, tornarem al frame inicial, o pantalla principal.

Finalment, premem el botó "Continue" per començar el processament de les imatges. L'aplicació fa, però, un parell de comprovacions abans: en primer lloc s'assegura que el nombre d'imatges seleccionades sigui igual o major que 50; de no ser així mostra un missatge informant de l'error. Aquesta comprovació es duu a terme per garantir uns mínims bons resultats del mètode de processament. Tenint en compte que normalment les quantitats d'imatges extretes d'una ecografia intracoronària ronden les 500 i 1000 unitats, no creiem que sigui una comprovació que salti molt sovint. En segon lloc, comprovem si la quantitat és múltiple de 10. En cas negatiu, el programa demanarà permís a l'usuari per reduir la quantitat d'imatges fins al múltiple de 10 més proper per sota. Aquesta comprovació és necessària pel correcte funcionament de procés de detecció del vas, ja que per magnificar els elements del vas es fa una mitjana cada 10 imatges de la seqüència estabilitzada.

Quan les dues comprovacions es superen satisfactòriament, es mostra un petit diàleg modal amb una barra de progrés i es passa a executar el mètode de pre-processament, si n'hi ha un de seleccionat. En cas contrari, copiem les imatges directament al nostre directori de destí.

Mantenint sempre la barra de progrés en modal, invoquem el mètodes d'estabilització de la seqüència, d'obtenció de la fase cardíaca i de generació del tall longitudinal. Si tot va bé, guardem el nou cas a la base de dades i tanquem primer el diàleg amb la barra de progrés i després el selector d'imatges. Per una visió més clara del procés, veure ANNEX 3.

Tornem a trobar-nos, doncs, amb la finestra principal, i aquest cop tenim un cas obert per visualitzar.

Aquest ha estat, amb diferencia, el punt més crític de la implementació: es tracta del veritable coll d'ampolla del programa ja que és l'únic pas que dona accés a l'execució de l'algorisme de processament. En el punt 5.4.1, 5.4.2 i 5.4.3 expliquem en detall quins problemes s'han presentat en aquest punt, i com els hem solucionat.

5.3.4 Pantalla principal: estudi d'un cas IVUS

La finestra principal és, doncs, el nostre espai de treball, on podem visualitzar els casos oberts, i treballar amb ells. Per facilitar la interacció amb els resultats del processament de les imatges, s'ofereixen les següents eines: (Figura 19)

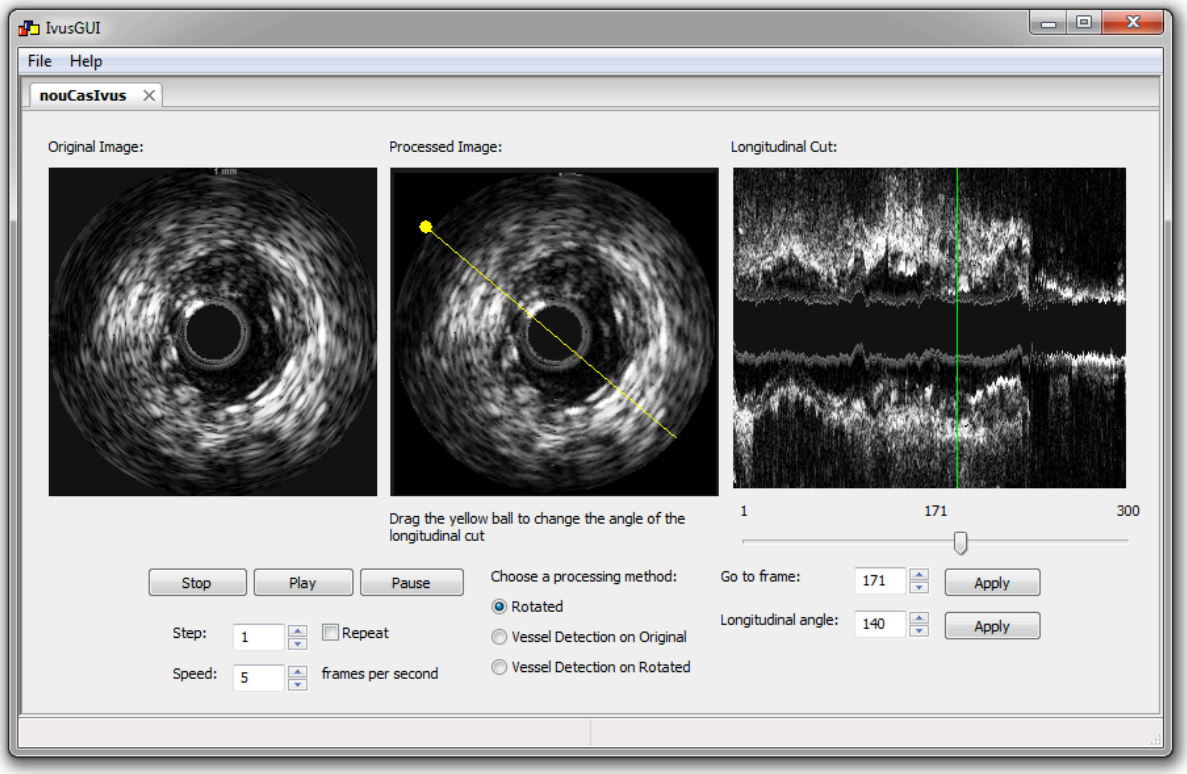


Figura 19: Finestra principal amb un cas carregat

- **Reproductor de seqüència:** Mitjançant els botons "Stop", "Play" i "Pause" podem aturar i tornar al primer frame, reproduir, i aturar la seqüència d'imatges, respectivament.

  Mentre s'està reproduint la seqüència totes les eines de l'espai de treball es desactiven per evitar conflictes.

  La resta de controls ajuden a configurar la reproducció

  - **Step:** defineix la separació, en frames, d'una imatge a la següent durant la reproducció.

- **Speed (frames per segon):** defineix el nombre d'imatges que es veuen en un segon.

- **Repeat:** Si esta marcat, la reproducció torna a començar a l'arribar a l'últim frame. Si no, s'atura.

La implementació d'aquest reproductor ha sigut molt senzilla gràcies a l'ús de la classe wxTimer (oferta pel mateix framework wxWidgets), a la que al assignar-li un període de temps en microsegons, llença un event cada cop que és compleix l'interval. Intervenint aquest event i pintant les noves imatges cada cop que s'executa obtenim un reproductor de vídeo amb baix cost computacional i sense necessitat d'implementar un thread.

- **Canvi del mètode de processat:** Ens permet decidir quin tipus d'imatge processada volem veure: imatge rotada (la que es mostra per defecte al obrir el cas), imatge original amb detecció del vas o imatge rotada amb detecció del vas. Les últimes dues opcions necessiten de l'execució de l'últim pas del processament d'imatges: la detecció del vas. Aquest pas només s'executa al seleccionar una d'aquestes dues opcions, i només cal fer-ho un cop. Mentre s'estan processant les imatges es mostra de nou un diàleg modal amb una barra de progres.

- **Canvi de frame:** Es pot canviar el frame que actualment s'està visualitzant de tres maneres diferents:

  - arrossegant el slider situat sota el tall longitudinal

  - introduint un valor manualment a la casella "Go to frame" i prement el botó "Apply"

  - Augmentat o disminuint el valor del frame fent clic a les fletxes de l'spin control

- **Canvi de l'angle del tall longitudinal:** Es pot canviar l'angle del tall longitudinal de dues maneres:

  - Arrossegant la bola groga per damunt del vas i deixant-la anar a l'arribar a la inclinació desitjada

  - Modificant el valor de la casella "Longitudinal angle" i prement el botó "Apply". El valor es pot modificar manualment o a través de l'ús de les fletxes incrementals.

El màxim valor de l'angle és 180º ja que el tall longitudinal comprèn tot el diàmetre del vas i per tant, pels angles compresos entre 181º i 360º el tall longitudinal és el mateix que pels angles que van de 1º a 180º. Tot i així, les probabilitats de que l'usuari obtingui el tall longitudinal per tots els angles entre 1º i 180º són molt baixes. El que fem, doncs, és calcular inicialment només el tall longitudinal per un angle de 1º, que és el que es mostra per defecte al carregar un cas l'espai de treball, i un cop l'usuari ha seleccionat un altre angle diferent, cridem a la funció MATLAB corresponent per generar aquesta nova vista.

5.3.5 Càrrega d'un cas IVUS

Seleccionant l'opció "Load IVUS Case" del menú "File" s'obre un nou diàleg de forma modal, on es llisten en una taula tots els casos IVUS que hem creat fins al moment (Figura 20). La taula mostra, també, un resum de la informació que hem guardat per cada cas. Fent clic a la taula, podem seleccionar una o més files i dur a terme una de les accions següents:

- **Eliminar cas IVUS:** Elimina el directori corresponent al cas seleccionat on s'emmagatzemen tots els fitxers generats pel processament de les imatges alhora que elimina l'entrada de la base de dades.

- **Carregar cas IVUS:** Tanca la finestra i obre tots els casos seleccionats a la finestra principal, situant cadascun d'ells en una pestanya diferent.

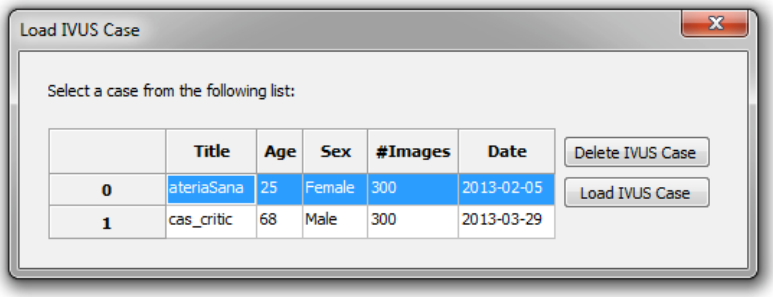


Figura 20: Diàleg de selecció de casos IVUS tant per eliminar-los com per obrir-los.

5.3.6 Ajuda i Sobre l'aplicació.

Per acabar, seleccionant l'opció "Help" del menú "Help" s'obre l'ajuda de l'aplicació en PDF, i seleccionant l'opció "About..." rebrem un breu missatge sobre el programa i la màquina on s'executa (Figura 21)

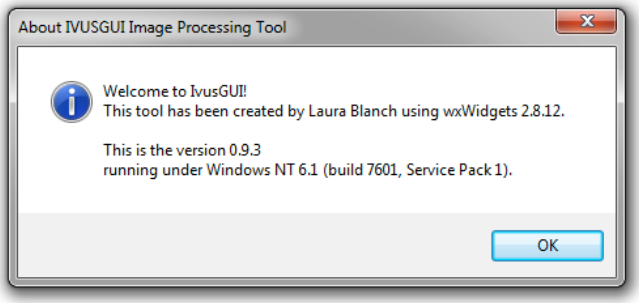


Figura 21: Diàleg de selecció de casos IVUS tant per eliminar-los com per obrir-los.

5.4. Punts interessants de la implementació

Durant el procés de programació de l'aplicació han sorgit punts especialment interessants que han requerit d'una solució o implementació fora del normal. A continuació se'n llisten alguns.

5.4.1 Galeria d'imatges amb pintat dinàmic

Un els passos en el procés de creació d'un cas IVUS és la selecció, per part de l'usuari, de les imatges a processar. Necessitem, doncs, mostrar totes les imatges incloses en la carpeta prèviament seleccionada juntament amb el nom d'arxiu.

Si bé wxWidgets no disposa de cap implementació d'una galeria d'imatges, la documentació els programes de mostra que s'inclouen en el paquet de distribució ens permet fer-nos una idea dels elements i events que haurem de fer servir en el nostre cas. Descobrim d'una banda, que l'única forma de mostrar una imatge és pintar-la manualment mitjançant un grup de classes anomenades "Device Context (DC)" que implementen el pintat de gràfics directament a l'aparell de sortida (en el nostre cas, la pantalla) i d'altra banda, que tota finestra rep, automàticament, l'event de pintat cada cop que és necessari, i el propaga a aquells elements fills (o child) que necessiten ser redibuixats. En definitiva, hem d'implementar una funció que, fent servir el device context corresponent, pinti la imatge i vincular-la a l'event de pintar de la finestra.

En primer lloc necessitem decidir quina serà la nostra classe contenidora de les imatges. Sabem que volem una distribució de la galeria totalment horitzontal per tal de facilitar la visualització de les imatges com una seqüència. Per això, i sabent també que per norma general el nombre d'imatges total serà massa gran com per poder mostrar-les totes de cop, la millor opció es fer servir la classe wxScrolledWindow, que implementa una finestra amb barres de desplaçament.

Arribats a aquest punt, se'ns presenten dues possibles implementacions:

1. Pintar les imatges i el nom d'arxiu corresponent directament en la finestra contenidora, calculant les distàncies en píxels entre les imatges manualment i controlant quines imatges són visibles a cada moment.

2. Crear una classe nova que representi una sola imatge i afegir-la, tantes vegades com imatges tinguem, a la finestra contenidora, mitjançant un sizer. D'aquesta manera, l'aplicació s'encarrega quasi automàticament del procés de pintat i del posicionament de les imatges.

Un cop implementats totes dues aproximacions, descobrim, però, que per quantitats d'imatges molt grans (a partir de les 300 aproximadament) el segon mètode dóna problemes, molt probablement, per que el sizer no pot gestionar tants elements a la vegada. Així doncs, l'aplicació final implementa el primer mètode, que funciona sense problema també amb grans quantitats d'imatges.

5.4.2 Algoritme d'ordenació natural (Natural sorting algorithm)

Com hem comentat en punts anteriors, hem considerat important donar la sensació de seqüència a l'hora de mostrar les imatges a seleccionar pel processament durant la creació d'un nou cas (punt 5.3.3). D'altra banda, és vital que les imatges es mostrin ordenades segons el nom d'arxiu, ja que aquest és l'únic element que ens dona una pista de quin va ser l'ordre original en que van extreure durant l'ecografia intracoronària.

Un cop l'usuari selecciona la carpeta on hi ha les imatges a processar, i per tal de poder accedir a elles més tard, guardem els noms de fitxer en un objecte tipus wxArrayString: un array de cadenes de caràcters que ofereix, a més, la funcionalitat (entre altres) d'ordenar els strings dins de l'array.

La funció d'ordenació per defecte no ens interessa, però, ja que en els casos en que el string conté números a més de lletres (i aquest serà el nostre cas el 99% de les vegades) no té en compte l'ordre numèric i ordena segons el valor ASCII de cada caràcter, produint resultats com els següents:

Algorisme d'ordenació per defecte	**Algoritme d'ordenació natural**
z1.bmp	z1.bmp
z10.bmp	z2.bmp
z100.bmp	z3.bmp
z101.bmp	z4.bmp
z102.bmp	z5.bmp
z11.bmp	z6.bmp
z12.bmp	z7.bmp
z13.bmp	z8.bmp
z14.bmp	z9.bmp
z15.bmp	z10.bmp
z16.bmp	z11.bmp
z17.bmp	z12.bmp
z18.bmp	z13.bmp
z19.bmp	z14.bmp
z2.bmp	z15.bmp
z20.bmp	z16.bmp
z3.bmp	z17.bmp
z4.bmp	z18.bmp
z5.bmp	z19.bmp
z6.bmp	z20.bmp
z7.bmp	z100.bmp
z8.bmp	z101.bmp
z9.bmp	z102.bmp

Com veiem doncs, l'ordre que necessitem ve donat per l'anomenat algorisme d'ordenació natural, que té en compte el valor total dels nombres que apareix a l'string i els ordena segons el seu ordre natural, mentre que es manté l'ordre ASCII pels caràcters alfabètics.

Així, només hem de configurar la funció d'ordenació amb una funció de comparació que faci servir aquest algoritme natural. Com que es tracta d'un problema molt comú, és fàcil trobar-ne implementacions en C/C++ de lliure distribució. En el nostre cas, hem fet servir la implementació "Natural Order String Comparison" per part del programador Martin Pool [12].

5.4.3 Selecció de fitxers segons l'estàndard de Windows Explorer

Durant la creació d'un cas IVUS nou, l'usuari ha de seleccionar quines imatges, del total trobat en la carpeta indicada, vol processar. Pot fer-ho mitjançant els botons de seleccionar / deseleccionar tot o pot fer servir el ratolí. En aquest últim cas, és necessari implementar un algoritme de selecció que doni a l'usuari la opció, no només de seleccionar imatges d'una en una, sino també de seleccionar rangs.

En una primera aproximació, l'usuari podia canviar l'estat de selecció d'una imatge fent clic al damunt sense canviar l'estat de la resta. Si es pressionava la tecla SHIFT al mateix temps que es feia clic, es canviava l'estat de selecció de totes les imatges que es trobaven entre l'última seleccionada i la actual. Després de varies proves vam concloure, però, que aquest

comportament, diferent al de la majoria de navegadors de fitxers existents, es feia estrany i poc intuïtiu.

Així, hem decidit finalment implementar el mateix sistema de selecció que trobem en l'explorador de fitxers del sistema operatiu Microsoft Windows (entre altres sistemes operatius). Ara, doncs, al fer clic en una imatge, deseleccionem totes les imatges excepte l'actual i canviem l'estat de selecció d'aquesta. Si es pressiona la tecla SHIFT al mateix temps que es fa clic, deseleccionem totes les imatges excepte l'actual, que queda seleccionada, i seleccionem totes les imatges que es troben entre l'actual i la clicada prèviament (sense apretar SHIFT). Si es pressiona la tecla CTRL al mateix temps que es fa clic, mantenim l'estat de selecció de la resta d'imatges i canviem l'estat de l'actual.

5.4.4 Disseny del layout: boxsizers i gridsizers

Com hem comentat al punt 4.3, wxWidgets proporciona un conjunt de classes anomenades "Sizers" que ajuden a situar els diferents elements dins d'una finestra. Si no els fem servir, hem d'especificar el punt, en píxels, on volem situar l'element en qüestió, així com la seva mida, obtenint una disposició totalment estàtica i que comporta tenir un control de les mides de tots els elements molt acurada.

Els sizers, en canvi, ofereixen diferents estructures d'organització per posicionar els nostres elements, i s'encarreguen de la seva disposició segons l'ordre en que els afegim i la mida. A més, permeten configurar les distàncies entre ells i la proporció amb la resta d'elements dins del sizer. Els sizer poden contenir altres sizer, per tal de construir estructures més complexes.

Finalment, totes les finestres disposen d'un mètode anomenat "SetSizer" que, donat el sizer que conté tots els elements, ajusta la mida de la finestra per tal que siguin tots visibles.

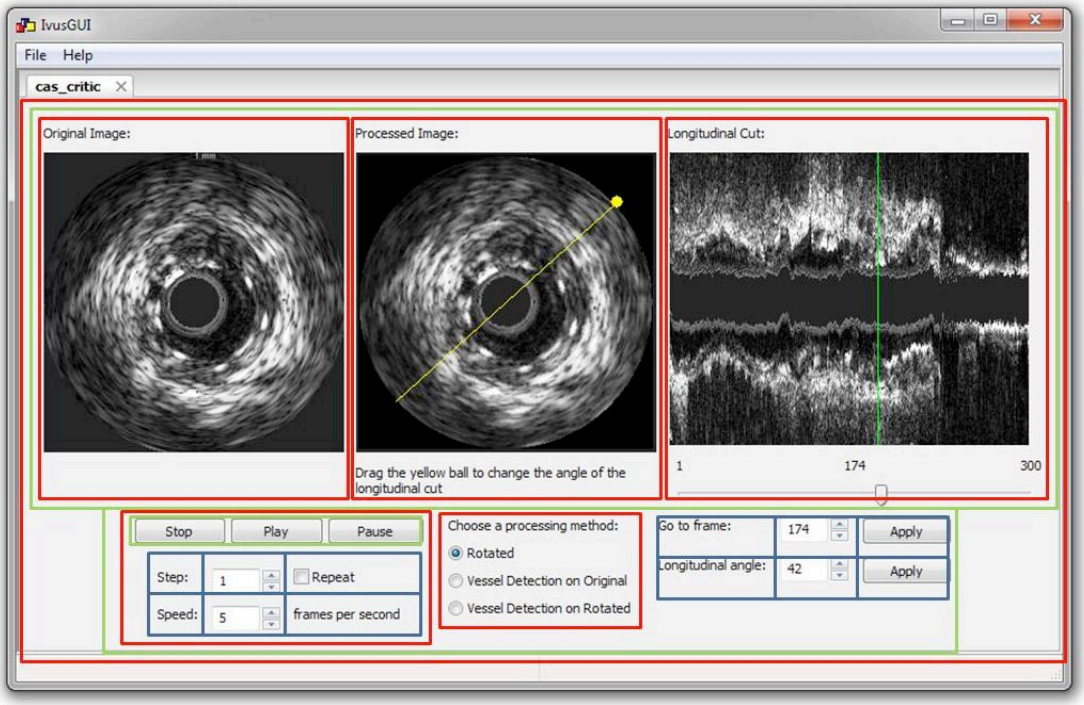


Figura 22: Diagrama dels diferents sizers fets servir per disposar els elements de l'entorn de treball: en vermell els wxBoxSizer verticals, en verd els wxBoxSizer horitzontals i en blau els wxFlexGridSizer.

En la nostra aplicació hem fet servir els sizers de tipus wxBoxSizer, que disposa els elements apilats segons la direcció en que el configurem i de tipus wxFlexGridSizer, que crea una graella de tantes columnes i files com definim i col·loca un element en cada casella. A la Figura 22 es pot observar un exemple d'una pantalla que fa servir els tres tipus de sizer, aniuant-los, per obtenir el layout desitjat.

5.4.5 Barra de progrés i Threads

Per tal d'executar el mètode de processament d'imatges IVUS, hem cridar les funcions MATLAB exportades, mitjançant MATLAB Compiler, en llibreries externes. L'execució d'aquestes funcions consumeix, per unes més que per d'altres, força temps, fet que ens presenta dos problemes: d'una banda hem d'avisar a l'usuari que el processament s'està executant i prevenir-lo de fer cap altra cosa mentrestant; de l'altra, ens trobem amb que al llarg de la duració de la crida a les funcions MATLAB, que en alguns casos pot arribar als minuts, l'aplicació es queda bloquejada i no respon a cap interacció, fent l'efecte que l'aplicació .

Per tal d'adreçar aquestes qüestions, hem decidit, en primer lloc, fer ús de la classe wxProgressBar, que implementa una barra de progrés estàndard i en segon lloc executar totes les funcions MATLAB en un fil d'execució paral·lel, per deixar lliure el thread principal .

Encara queda per resoldre, però, una última dificultat: com actualitzar la barra de progrés si és impossible saber quant temps tarda la funció en executar-se? Arribats a aquest punt, hem creat una classe anomenada "ProgressBar" que, fent ús de l'abans mencionada wxProgressBar i essent l'encarregada de llençar els threads que executen les funcions MATLAB, incrementa la barra de progrés fent ús d'un timer, que cada mig segon incrementa la barra una unitat.

6. Conclusions

Durant el procés d'implementació d'aquesta aplicació, hem fet ús d'eines totalment noves per a nosaltres, com ara el framework wxWidgets o MATLAB Compiler, que hem hagut d'aprendre a fer servir de forma autònoma. Si bé en alguns moments s'han presentat dificultats i en alguns punts l'avanç ha sigut una mica lent, considerem hem adquirit un nivell de coneixements molt satisfactori.

Sobre la planificació, hem patit alguns retards, causats per un càlcul del temps no gaire encertat, hem pogut aprendre d'aquesta experiència i estem segurs que, de cara al pròxim projecte ho haurem corregit, com a mínim, en part.

Un fet que valorem molt positivament, és el d'haver creat, totalment de zero, una aplicació que ha complert les expectatives. Des de la fase de disseny, a la redacció d'aquesta memòria, hem passat per múltiples fases, i de totes ens n'hem endut valuoses lliçons.

Pel que fa a l'aplicació resultant, creiem que compleix els objectius proposats en un inici: és tracta d'una aplicació molt senzilla, que permet processar qualsevol seqüència d'imatges IVUS fent servir el mètode de processament implementat per Aura Hernández. El programa és totalment funcional, i compleix tots els requisits necessaris que es s'han llistat durant la fase de disseny. Si bé no s'han compilat versions per d'altres plataformes, durant la implementació s'ha tingut molt present el concepte de cross-platform, evitant usar llibreries i eines que no suportessin aquesta condició.

6.1. Ampliacions futures

Si bé l'aplicació esta llesta per ser usada per l'usuari final, hi ha una sèrie de millores i funcionalitats extra que creiem que, un cos incloses al programa final, ens acostarien, encara més, al nostre objectiu, així com millorarien l'experiència de l'usuari al fer-la servir. Són les següents:

- Implementació dels requisits opcionals restants: suport per arxius DICOM, canvi d'idioma per l'aplicació, recol·locació manual del contorn del vas i editar cas IVUS (veure punt 3.1)

- Millores visuals: canviar el text d'alguns botons (per exemple, els de play, stop, pause) per icones descriptives, redistribuir alguns controls per millor usabilitat... En general, fer un estudi més profund del disseny gràfic de l'aplicació per millorar-ne l'aspecte i la usabilitat.

- Implementació d'una ajuda interactiva, integrada dins de la mateixa aplicació.

7. Bibliografia

[1] World Health Organization. "Cardiovascular diseases (CVDs)"*Fact sheets*. (Maig 2013). Juny 2013 < http://www.who.int/mediacentre/factsheets/fs317/en/index.html >.

[2] Volcano. "IVUS IMAGING: VH® IVUS Imaging System". Juny 2013 <http://eu.volcanocorp.com/products/ivus-imaging/vh-ivus.php>.

[3] Boston Scientific. "iLab® Ultrasound Imaging System". Juny 2013 <http://www.bostonscientific.com/interventional-cardiology/products.html?>.

[4] Medimatic. "ComPACS Workstation". Juny 2013 <http://www.medimatic.com/Schema4.aspx?Item=18>.

[5] INDEC. "echoPlaque Analysis Software". Juny 2013 <http://www.indecmedical.com/IMed_Internal/InternalechoPlaque.asp>

[6] CERTH/ITI Centre for Research and Technology Hellas/Information Technologies Institute, C.E.A Laboratory of the 1st Dpt of Cardiology at AHEPA University Hospital. "3D IVUS-ANGIO Tool". Juny 2013 <http://mklab.iti.gr/ivus/>

[7] Aura Hernàndez-Sabaté and Debora Gil (2012). The Benefits of IVUS Dynamics for Retrieving Stable Models of Arteries, Intravascular Ultrasound, Dr. Yasuhiro Honda (Ed.), ISBN: 978-953-307-900-4, InTech, Available from: http://www.intechopen.com/books/intravascular-ultrasound/the-benefits-of-ivus- dynamics-for-retrievingstable-models-of-arteries

[8] Aura Hernàndez-Sabaté. "Exploring Arterial Dynamics and Structures in IntraVascular UltraSound Sequences." *Ph.D. dissertation*, Universitat Autònoma de Barcelona, 2009.

[9] Qt Project Hosting. "Qt Project". (2013) Juny 2013 <https://qt-project.org/>

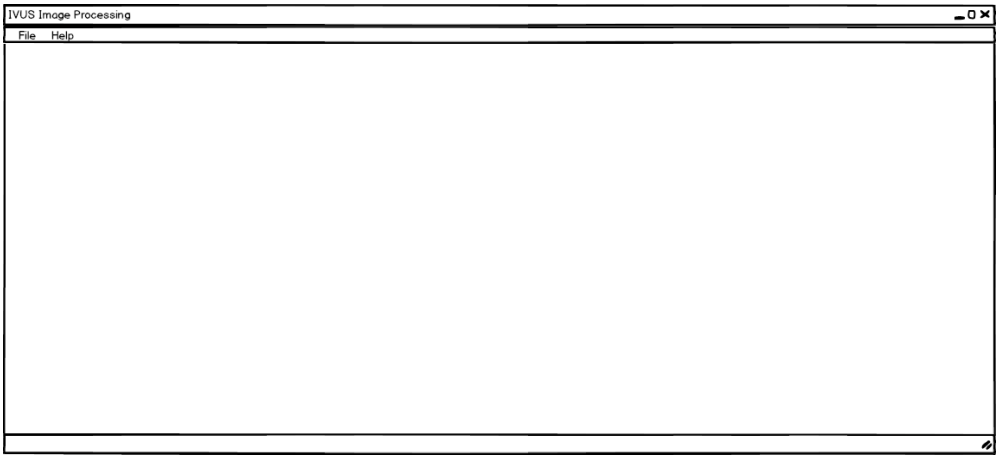
[10] wxWidgets. "wxWidgets". Juny 2013 <http://www.wxwidgets.org/>

[11] Julian Smart and Kevin Hock. "Cross-Platform GUI Programming with wxWidgets". United States of America: Pearson Education,Inc., 2006.

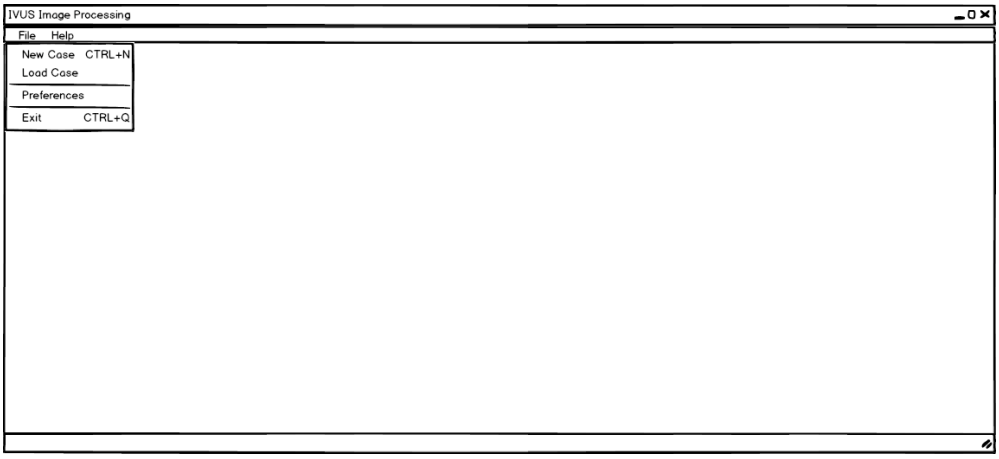
[12] Martin Pool. "Natural Order String Comparison" Juny 2013 <http://sourcefrog.net/projects/natsort/>

ANNEX 1: MAQUETA DE L'APLICACIÓ

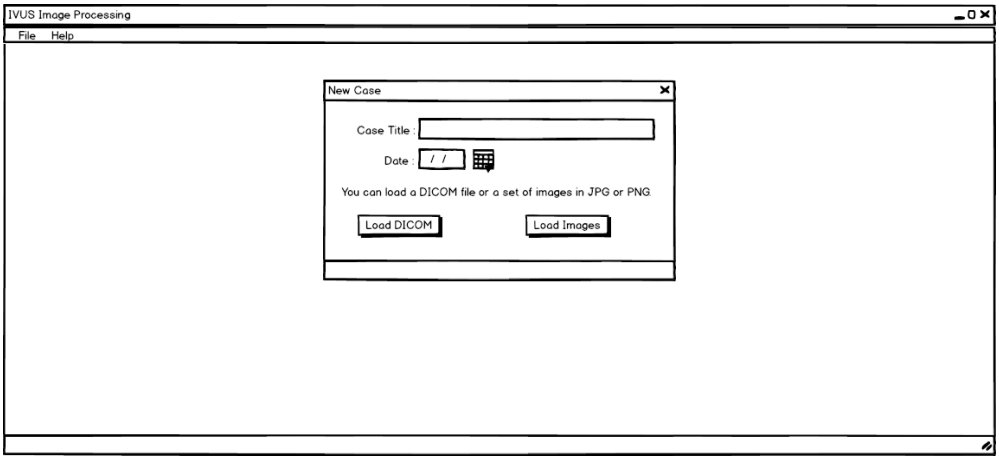
Pantalla inicial:



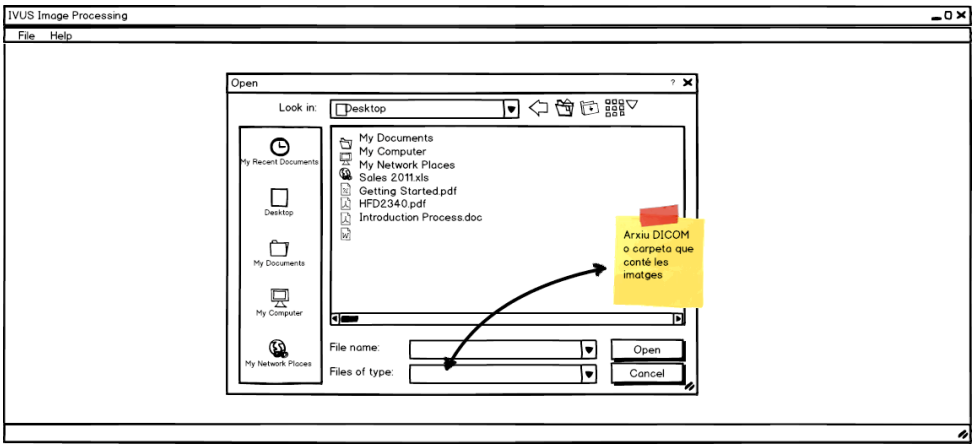
Pantalla inicial (amb menú):



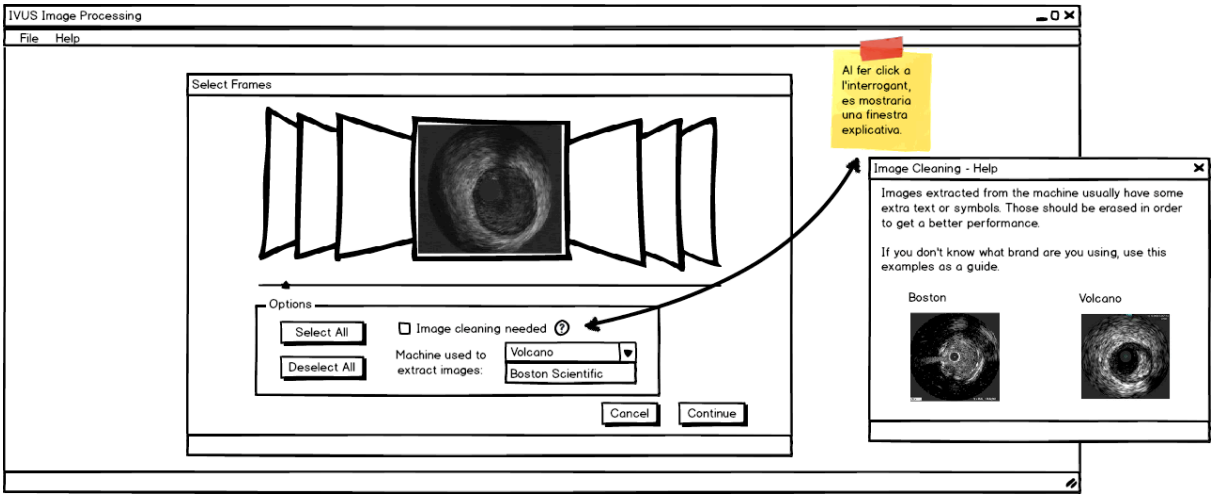
Creació de nou cas IVUS



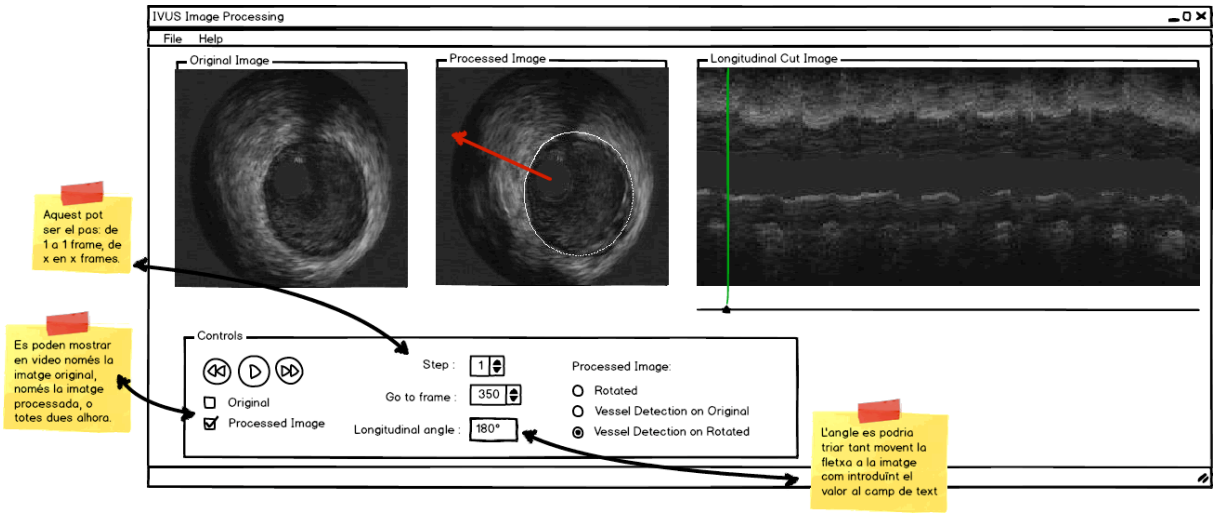
Selecció de la carpeta on es troben les imatges a processar:



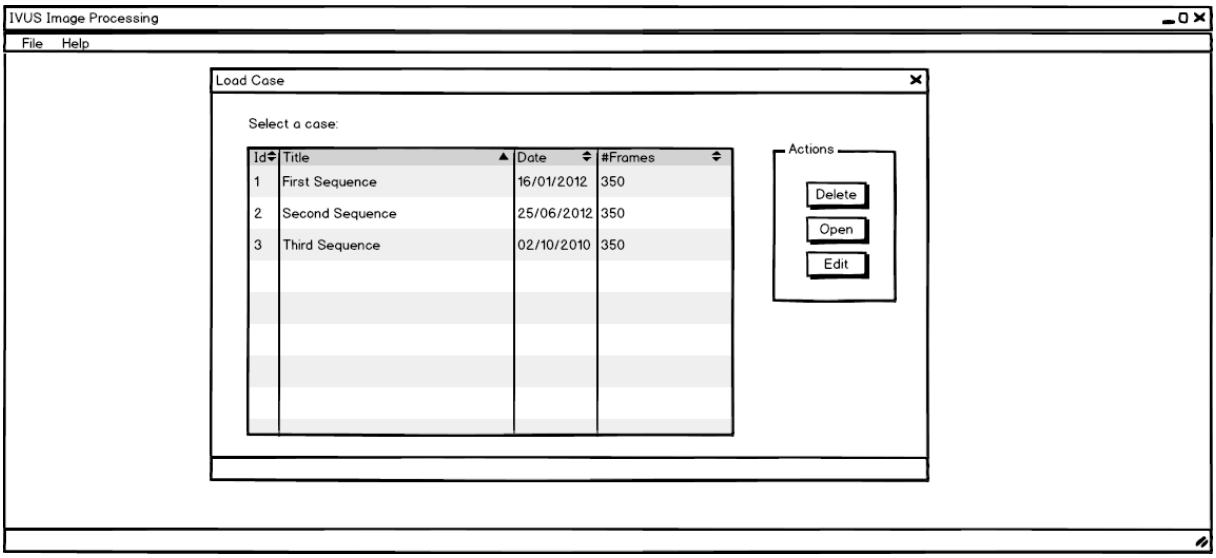
Selecció de les imatges a processar:



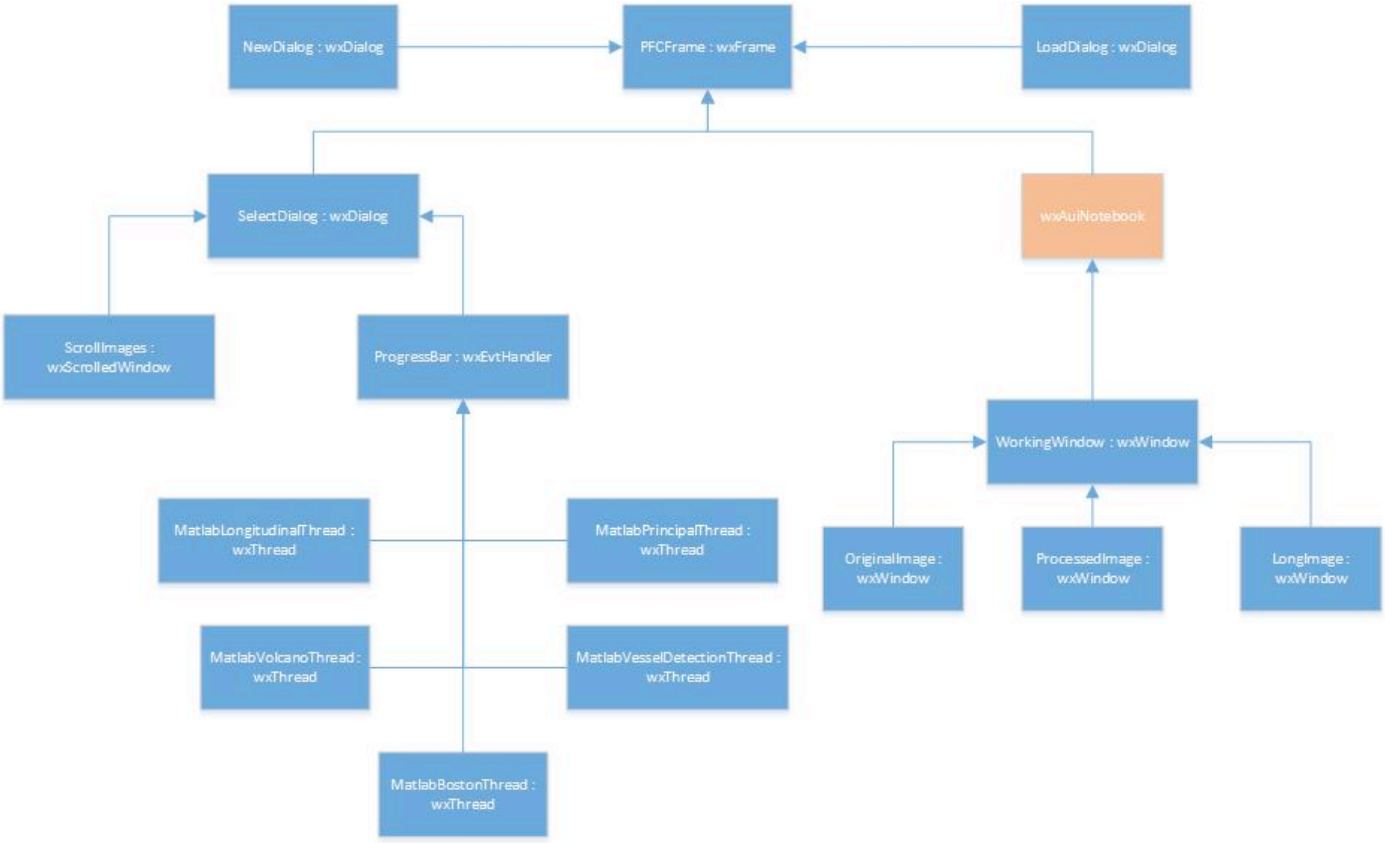
Pantalla principal amb cas carregat:



Llistat de casos existents:



ANNEX 2: DIAGRAMA DE CLASSES GRÀFIQUES IMPLEMENTADES



ANNEX 3: DIAGRAMA DE FLUXE