Universitat de Girona
Escola Politècnica Superior

Treball final de grau

Estudi: Grau en Enginyeria Informàtica

Títol: Interfície d'usuari per a un programari de construcció de mapes acústics submarins

Document: Memòria

Alumne: Luis Ribés Nebot

Tutor: Pere Ridao Rodriguez, Natàlia Hurtós, Narcís Palomeras
Departament: Arquitectura i Tecnologia de Computadors
Àrea: Arquitectura i Tecnologia de Computadors

Convocatòria (mes/any): Setembre/2016

Índex

Glossari

AUV: *Autonomous Underwater Robot*, Robot submarí autònom.

CIRS: Centre d'Investigació en Robòtica Submarina.

CSV: *Comma Separated Values*, Valors separats per comes.

FFT: *Fast Fourier Transform*, Transformada ràpida de Fourier.

FLS: Forward-Looking Sonar, Sonar de visió frontal.

GUI: *Graphical User Interface*, Interfície gràfica d'usuari.

MVC: Model-Vista-Controlador.

ROV: *Remoted Operated Vehicle*, Vehicle de control remot.

UVL: Laboratori de visió submarina.

VICOROB: Grup de Visió per Computador i Robòtica.

1. Introducció

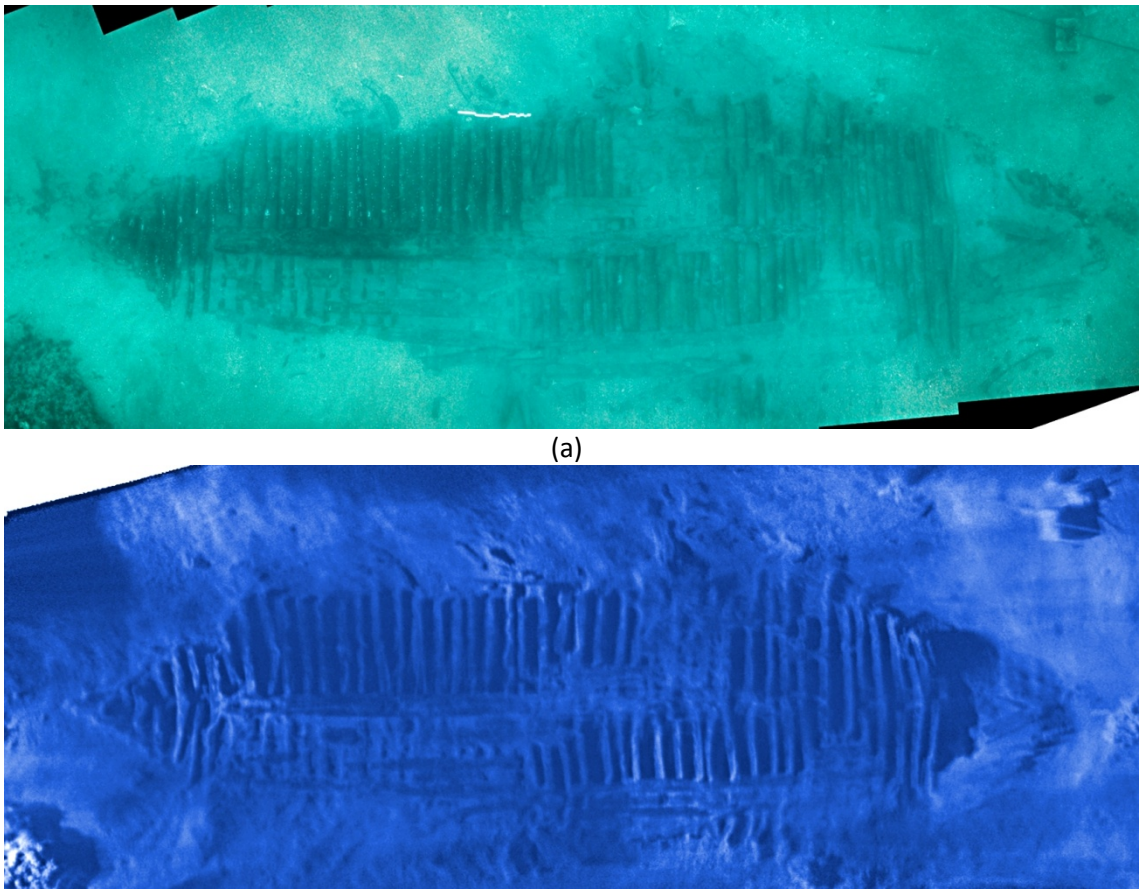
La necessitat d'explorar i de conèixer tot allò que ens envolta, ha estat present en l'ésser humà des de l'antiguitat. L'instint de curiositat dels humans, ha fet que el món sigui tal i com el veiem ara. Hem avançat molt com a societat en els camps de les matemàtiques, la medicina o l'astronomia. Però tot hi això, hi ha camps on encara ens queda molt per descobrir.

Un d'aquests camps, és l'oceanografia, és a dir , la ciència que estudia tots els processos físics, químics i biològics que es donen al mar i als oceans. Actualment, els mapes més detallats que existeixen, només cobreixen el 10%-15% dels oceans (Copley, 2014), quan més profund anem, major és el desconeixement. És per això, que cal avançar més en aquest àmbit i desenvolupar tecnologia que puguin fer servir els científics(biòlegs, geòlegs, etc) per poder explorar aquest entorn submarí. D'altra banda, una gran part dels recursos que utilitzem els humans s'obtenen, provenen o estan relacionats amb l'entorn submarí i requereixen la instal·lació d'infraestructures sota l'aigua. Per exemple, plataformes petrolíferes, observatoris submarins, cables transoceànics, granges d'aqüicultura, etc. Per tant també és necessari continuar investigant en tecnologia que ens permeti inspeccionar, mantenir i analitzar l'impacte mediambiental d'aquestes infraestructures submergides.

Tradicionalment les exploracions sota l'aigua s'han realitzat amb submarinistes o, quan la profunditat no ho permetia, amb submarins tripulats. No obstant, per reduir el risc que comporta pels humans endinsar-se en aquest entorn, fa dècades que s'han començat a utilitzar vehicles per a fer aquests tipus d'exploració. Existeixen bàsicament dos tipus de vehicles submarins, els teleoperats (també anomenats ROVs de l'anglès *Remotely Operated Vehicle*) i els autònoms (anomenats AUVs, de l'anglès *Autonomous Underwater Vehicle*). Mentre que els primers són conduits des de la superfície per un humà a través d'un cable que permet enviar les senyals de control i l'alimentació, els segons estan proveïts de bateries i la sensorització i computació necessària per executar missions autònomament.

A la Universitat de Girona (UdG), el Grup de Visió per Computador i Robòtica (VICOROB) fa recerca en el camp de la robòtica submarina, a través del seu Centre d'Investigació en Robòtica Submarina (CIRS). A banda de la construcció de AUVs, una de les principals línies de recerca del laboratori és l'elaboració de mapes del fons marí i d'estructures submergides. Això es pot fer capturant imatges amb càmeres òptiques mentre els vehicles realitzen trajectòries submarines. No obstant, un dels grans problemes quan es treballa sota l'aigua és la visibilitat. Els sensors òptics tenen un rang de percepció molt limitat quan l'aigua és tèrbola. En els últims anys han aparegut al mercat una nova generació de sonars de visió frontal (*Forward-Looking Sonars*, FLS), d'alta freqüència i amb la capacitat de generar imatges acústiques a un alt ritme de refresc, convertint-se així en una alternativa a les càmeres òptiques.

En l'actualitat, el CIRS disposa d'un conjunt d'algoritmes que permeten construir mosaics acústics a partir d'imatges d'un sonar de visió frontal i així obtenir una representació de l'entorn submarí independentment de la visibilitat de l'aigua (veure Figura 1.1).



(a)

(b)

Figura 1.1: (a) Mosaic òptic. (b) Mosaic acústic. Exemples de mosaics elaborats amb dades agafades al mateix lloc, en aquest cas amb bona visibilitat. Sinó hi hagués visibilitat, el mosaic acústic s'obtindria igualment.

Aquests algoritmes de construcció de mosaics estan implementats com un conjunt d'eines executades a través de la línia de comandes, cosa que dificulta el seu ús a nivell d'usuari i per tant dificulta també l'expansió d'aquesta tecnologia. Més d'una desena d'empreses i institucions s'han interessat durant els últims anys davant la possibilitat d'adquirir un software que permeti generar mosaics acústics utilitzant les tècniques desenvolupades al CIRS. No obstant, tot i disposar dels algorismes, la falta d'una interfície d'usuari impedeix poder oferir un producte a potencials usuaris.

És en aquest moment on sorgeix la idea de desenvolupar aquest projecte, és a dir, incorporar una interfície gràfica, per poder interactuar amb aquest conjunt d'eines d'una manera més simplificada i interactiva, i així, donar accés a usuaris no experts en informàtica com poden ser biòlegs o arqueòlegs. A més a més, l'elaboració de mosaics acústics és una tasca que des del CIRS també es realitza puntualment, ja sigui per a projectes de recerca, demostracions o col·laboracions amb altres institucions. Així doncs aquesta interfície també facilitarà que personal del CIRS no expert en el tema dels mosaics acústics pugui generar mosaics d'una manera més fàcil i intuïtiva.

1.1. Antecedents

Aquest projecte s'ha realitzat al laboratori de Robòtica Submarina del Grup de Visió per Computador i Robòtica de la Universitat de Girona. El CIRS realitza investigació en robòtica submarina des de l'any 1992, finançada a partir de diversos programes de recerca espanyols i europeus (CIRS, 2016). Des del laboratori s'han desenvolupat múltiples prototips d'AUVs: en GARBÍ, un vehicle originalment dissenyat com a ROV i després convertit a AUV (Amat et al.1999); l'URIS un AUV petit i lleuger (Batlle et al.2005); l'ICTINEU que va guanyar el primer concurs europeu de robòtica submarina per estudiants (SAUC-E) el 2006 (Ribas et al.2007), l'Sparus que va guanyar el mateix concurs el 2010 i recentment s'ha redissenyat en versió comercial (Carreras et al.2013) i el Girona 500 , un AUV rconfigurable que té una gran capacitat per carregar sensors (Ribas et al.2012).

Els dos prototips que estan en funcionament actualment (l'Sparus II i el Girona 500) tenen una àrea dedicada per muntar sensors específics per a cada missió. Els dos són capaços d'integrar el sonar ARIS Explorer 3000 (Soundmetrics, 2016) que és el FLS existent al CIRS, i del qual parteixen gran part de les dades que s'han utilitzat per testejar la interfície desenvolupada en aquest projecte.

Al laboratori es fa i s'ha fet recerca sobre múltiples temes relacionats amb els AUVs, col·laborant també amb el laboratori de visió per computador (UVL*, Underwater Vision Lab*) del mateix grup de recerca. S'ha fet recerca sobre arquitectures de control (Ridao et al.2002) (Palomeras et al.2012), aprenentatge automàtic, (Carreras et al.2001, El-Fakdi & Carreras, 2013), control de missió (Palomeras et al.2006), intervenció autònoma (Prats et al.2012b, Ribas et al.2012), mapeig i localització (Ribas et al.2008, Mallios et al.2014), planificació de camins (Hernández et al.2011, Galceran Yebenes 2014), mosaics òptics (Ferrer et al.2007), tècniques d'alineament global per l'elaboració de mapes òptics (Elibol et al.2011a), fusionat d'imatges (Prados et al.2014), reconstruccions 3D (Nicosevici et al.2009) i classificació d'imatges (Shihavuddin et al.2013). El treball conjunt entre els dos laboratoris ha donat lloc a experiments satisfactoris en el context d'aplicacions reals com ara inspecció de preses (Ridao et al.2010), inspeccions arqueologiques (Gracias et al.2013), o mapeig i intervenció en ports (Prats et al.2012a).

El 2014 es va començar una línia de recerca centrada en l'elaboració de mapes d'imatge acústica, dotant els vehicles del CIRS de la possibilitat de generar mapes d'entorns submarins independentment de la visibilitat de l'aigua. Aquesta és la línia de recerca en la qual ha contribuït aquest TFG.

1.2. Motivació

El fet de tenir un Centre d'Investigació en Robòtica Submarina a la Universitat de Girona dóna lloc a que molts projectes que es proposin o es plantegin als estudiants en alguns dels graus d'enginyeria (informàtica, electrònica, mecànica, industrial) girin al voltant d'aquest àmbit. Aquest pas és molt important per a que futurs professionals s'interessin per aquest món de la robòtica submarina i en vulguin formar part. L'informàtica és un dels camps científics que permet aplicar el seu coneixement sobre un altre camp per millorar la seva investigació. Avui en dia, en qualsevol camp científic podem trobar diferent tecnologia que ajuda en la investigació d'aquest, però a vegades aquestes eines poden ser difícils d'utilitzar si no es té un

mínim coneixement informàtic. D'aquí sorgeix la idea de crear una interfície gràfica, que faciliti l'ús d'una eina de construcció de mapes acústics submarins que es disposa actualment en el CIRS.

1.3. Objectius

L'objectiu principal d'aquest projecte és:

> *"Dissenyar i implementar una interfície gràfica d'usuari, per al programari de construcció de mapes acústics submarins existent al Centre de Investigació en Robòtica Submarina. La interfície ha de permetre que el procés de construir un mosaic acústic sigui senzill per l'usuari, que li permeti modificar els paràmetres necessaris i visualitzar les representacions de les diferents dades que intervenen durant el procés de creació de mapes."*

Podem dividir el propòsit d'aquesta interfície en objectius més específics:

- **Intuïtiva i fàcil d'utilitzar:** La interfície ha de ser utilitzable per usuaris no experts en informàtica i amb mínims coneixements sobre el procés de construcció de mosaics.
- **Robusta i fiable:** Ha de respondre de manera satisfactòria a les peticions de l'usuari i ser tolerant a fallades.
- **Extensible**: Donat que el *software* sobre el que es construeix es troba en un context de recerca i en continu desenvolupament ha de ser possible afegir noves funcionalitats de manera fàcil.
- **Modular:** La interfície ha de conviure amb el *software* original, de manera que es pugui utilitzar l'aplicació a través de la línia de comanda o de la interfície.

A banda del desenvolupament de la interfície un objectiu paral·lel és contribuir en la **millora i la implementació de noves funcionalitats al *software* original**.

1.4. Estructura de la memòria

L'estructura d'aquest document segueix, en essència, la *Guia dels Projectes/Treballs de Final de Carrera de les Enginyeries Informàtiques.* Alguns apartats que no eren aplicables al cas del projecte en qüestió s'han suprimit i d'altres s'han fusionat en un de sol per tal de conservar la rellevància i la continuïtat dels conceptes presentats en el text.

A continuació es presenta una breu descripció de cada capítol d'aquesta memòria:

Planificació: Defineix l'estratègia seguida per arribar als objectius plantejats. Es descriurà tant la metodologia utilitzada com la temporització de cada bloc dins del projecte.

Marc de treball i conceptes previs: Per tal de situar el lector en la temàtica que engloba el treball, en aquest capítol es descriuran els diversos aspectes relacionats amb el desenvolupament general del projecte que permetran entendre molt millor els següent capítols.

Anàlisi: S'identificarà tant els requeriments del sistema com les seves funcions.

Disseny: Es proposarà una solució detallada que compleixi les necessitats especificades tant en el l'apartat d'objectius com en el capítol d'anàlisi.

Implementació: Es detallaran els problemes i les solucions aparegudes en la implementació.

Implantació, proves i resultats: Es descriurà el procés d'implantació de la interfície desenvolupada al CIRS, així com es descriuran els resultats de les diferents proves que s'han fet per avaluar la interfície.

Conclusions i treballs futurs: Es recullen les conclusions i valoracions a que s'ha arribat al final del projecte, a més de comentar possibles línies de millora de cares a un futur.

Bibliografia: Recull de totes les referències utilitzades per desenvolupar el projecte

2. Planificació

2.1. Metodologia

Un dels requeriments del projecte era que la interfície gràfica s'havia d'anar fent en paral·lel, de manera que no interferís en el desenvolupament del software existent. Calia una metodologia que donés resultats des del primer moment, per tant, es van descartar les metodologies clàssiques *pesades* de desenvolupament de software, que necessiten un gran avanç del projecte per poder començar a obtenir resultats.

Calia una metodologia que donés resultats des del primer moment, per tant, es van descartar les metodologies clàssiques *pesades* de desenvolupament de software, que necessiten un gran avanç del projecte per poder començar a obtenir resultats.

Per això, es va fer recerca sobre les metodologies àgils que actualment més es fan servir. Després de veure les característiques de cada una, tot i tenir molts punts en comú, per realitzar el projecte es va decidir usar la metodologia *eXtreme Programming* (XP). Es va escollir aquesta metodologia, perquè un dels punts importants és que el projecte sigui susceptible a canvis o a nous requeriments de l'aplicació.

Podem trobar tota la informació d'aquesta metodologia dins del llibre *Extreme Programming Explained: Embrace Change [Beck et al., 1999].* És una metodologia de les més destacades, dins de les metodologies àgils.

Aquest tipus de programació es diferencia de les tradicionals degut a que prioritza la adaptabilitat. És a dir, considera que els canvis en els requisits de l'aplicació durant tot el desenvolupament són un aspecte natural, inevitable i en molts casos, desitjable.

Aquesta metodologia consta de 5 principis bàsics:

- **Simplicitat:** és la base d'aquesta metodologia. Es simplifica el disseny per poder agilitzar el desenvolupament i facilitar el manteniment.

- **Comunicació:** si el codi es manté senzill, permetrà una millor comunicació. Una bona comunicació amb el client (en aquest cas el tutor del projecte) és vital per poder adaptar el codi a les característiques d'aquest, i també perquè pugui solucionar els dubtes que puguin aparèixer.

- **Retroalimentació (feedback):** com que el client forma part del projecte, es pot saber en tot moment que és el que pensa i quina és la prioritat. Això ajuda a realitzar el projecte en petits cicles y no haver d'esperar a programar gran quantitat de codi per ensenyar resultats.

- **Coratge:** s'ha de ser capaç de modificar part del codi quan faci falta, encara que això impliqui més feina. S'ha de ser persistent i involucrar-se al màxim en el projecte per poder resoldre els problemes que es van trobant el més aviat possible.

- **Respecte:** cada membre del projecte ha de valorar la feina de la resta i respectar-la a l'hora de fer la seva part. Per tant, s'ha de tenir cura de no perjudicar als companys i sobretot treballar tan com la resta.

2.2. Temporització

El projecte va començar el febrer de 2016 amb la intenció de ser finalitzat i entregat al setembre de 2016. Es van planificar les tasques seguint la metodologia comentada i dividint-les en iteracions de unes dos setmanes cada una. A continuació en la figura 2.1 podem veure el diagrama de Gantt que mostra la divisió de tasques del projecte juntament amb la seva durada.

Una vegada finalitzat el projecte s'han calculat les hores totals que s'ha dedicat a l'elaboració de la interfície i a la documentació (excloent la redacció de la memòria) per poder tenir una estimació del cost que haurà suposat aquesta implementació. El nombre d'hores dedicades es mostren a través de la següent taula:

Tasques	Hores
Estudi d'eines	50 h.
Disseny classes	50 h.
Implementació	300 h.
Proves	200 h.
Documentació	20 h.
Total:	620 h.

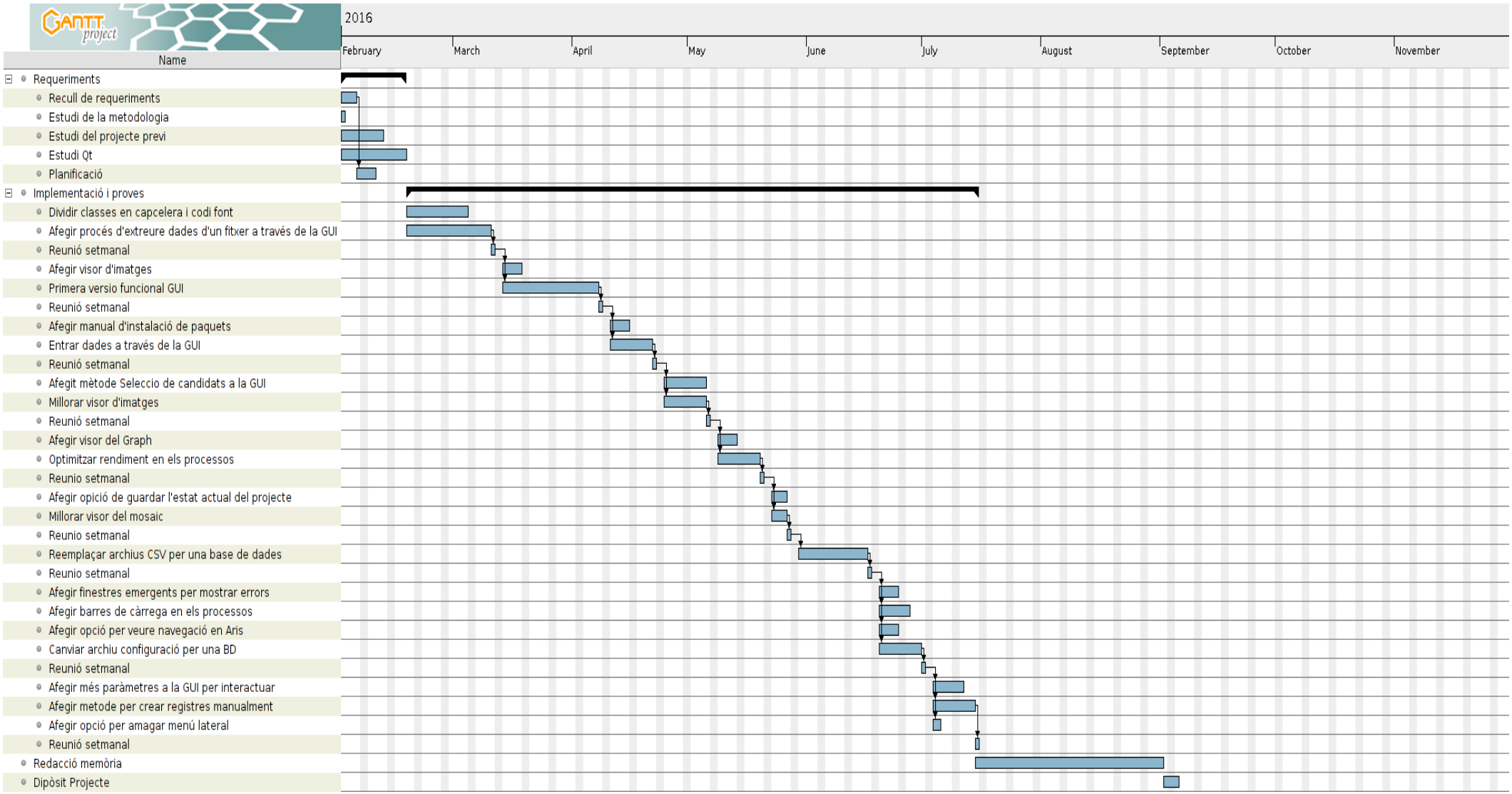


Figura 2.1: Diagrama de Gantt

3. Marc de treball i conceptes previs

En aquest capítol es descriu el context i l'entorn en el qual s'ha desenvolupat el treball i s'introdueixen alguns conceptes que cal conèixer abans de procedir amb una explicació més detallada del que s'ha dut a terme en aquest projecte.

3.1. Marc de treball

Com ja s'ha comentat a la introducció, el projecte forma part de la continuació de la recerca desenvolupada al CIRS, part integrant del grup de recerca VICOROB.

El CIRS(fig. 3.1) es troba situat al Parc Tecnològic de Girona.

Figura 3.1: CIRS

En l'actualitat hi ha varies línies de recerca obertes:

- L'estudi d'arquitectures de control per a vehicles autònoms.
- El disseny i desenvolupament de simuladors de robots, missions i entorns.
- La fusió d'informació de diferents sensors per a la navegació i localització.
- La construcció de mapes visuals d'estructures submergides o del fons marí, en coordinació amb el grup de Visió Submarina.

Aquest TFG es troba emmarcat dins aquesta última línia de recerca, la creació de mapes sota l'aigua, en aquest cas utilitzant dades de sonar. El projecte original, on es van desenvolupar les tècniques que permeten el registre i la construcció dels mosaics acústics es va dur a terme dins una tesis doctoral realitzada al CIRS, que va finalitzar l'any 2014 (Hurtos, 2014). Des de llavors, s'han refinat alguns dels algoritmes i se n'ha millorat la seva implementació. Al inici d'aquest projecte s'ha partit d'un conjunt de comandes, cada una amb el seu conjunt de paràmetres, que es poden executar des d'un terminal d'un sistema operatiu UNIX. Aquest conjunt de llibreries és l'element bàsic del marc de treball inicial.

3.2. Conceptes previs

A continuació s'expliquen diversos conceptes necessaris per entendre l'objectiu i la funcionalitat de la interfície que s'ha desenvolupat. Per entendre tot el que la interfície ha de fer és necessari conèixer primer com són les dades dels sonars amb què es treballarà i en què consisteix el procés de construcció de mosaics. Aquests temes son recurrents al llarg de tot el treball i poden fer-se servir com a material de consulta en la lectura dels capítols següents.

3.2.1. La tecnologia sonar

Les càmeres òptiques són molt útils i estan presents en molts escenaris, però quan parlem del fons marí, treballar amb aquest tipus de càmeres pot portar moltes dificultats. . L'atenuació que la llum pateix dins l'aigua dificulta la visió a mesura que augmentem la profunditat. A més a més, existeix la possibilitat de que l'aigua sigui tèrbola i tingui partícules en suspensió.

Figura 3.2: Diferents exemples d'imatges òptiques afectades per la retrodispersió i/o aigua tèrbola.

Degut a les limitacions de les càmeres òptiques, en l'entorn submarí s'acostuma a utilitzar la tecnologia sonar, basada en el rebot de les ones acústiques. Les ones acústiques es veuen menys afectades per l'atenuació de l'aigua, facilitant així la detecció d'objectes en condicions difícils. Tot i això, la utilització de sonars, també fa que les dades que recollim siguin més difícils d'interpretar i processar.

Existeixen diferents tipus de sonars que es diferencien per la manera en la que recullen les dades. A la figura 3.3 es poden veure els diferents tipus de dispositius sonar existents avui en dia, juntament amb una il·lustració de com insonifiquen l'entorn i quin tipus de dades poden donar.

Les ecosondes i les sondes multifeix donen rang (distància) a partir de la mesura del retorn d'un (o múltiples) feixos acústics. S'han fet servir amb èxit amb la finalitat d'evitar obstacles, navegar i localitzar-se i sobretot per crear cartes batimètriques del fons marí.

Els sonars d'imatge com els sonars d'escàner mecànic o els sonars d'escàner lateral, , a banda de retornar el rang mesurat també són capaços de representar les intensitats acústiques de retorn d'una àrea insonificada la qual cosa els ha fet útils per a construir mapes, i detectar objectes o obstacles. No obstant, el tipus de dades que retornen requereixen molt de processament i tot i que comprèn rangs molt grans, la seva resolució és bastant pobre.

Recentment, ha aparegut una nova generació de sonars d'imatge anomenats *two-dimensional Forward-Looking Sonars* o sonars de visió frontal. Són sonars de molta alta freqüència i que per

tant són capaços de donar dades de molta resolució a rangs curts. Estan emergent com una alternativa per ambients on la visió és molt reduïda, gràcies a les seves capacitats de lliurar imatges 2D d'alta qualitat gairebé a la velocitat de vídeo.

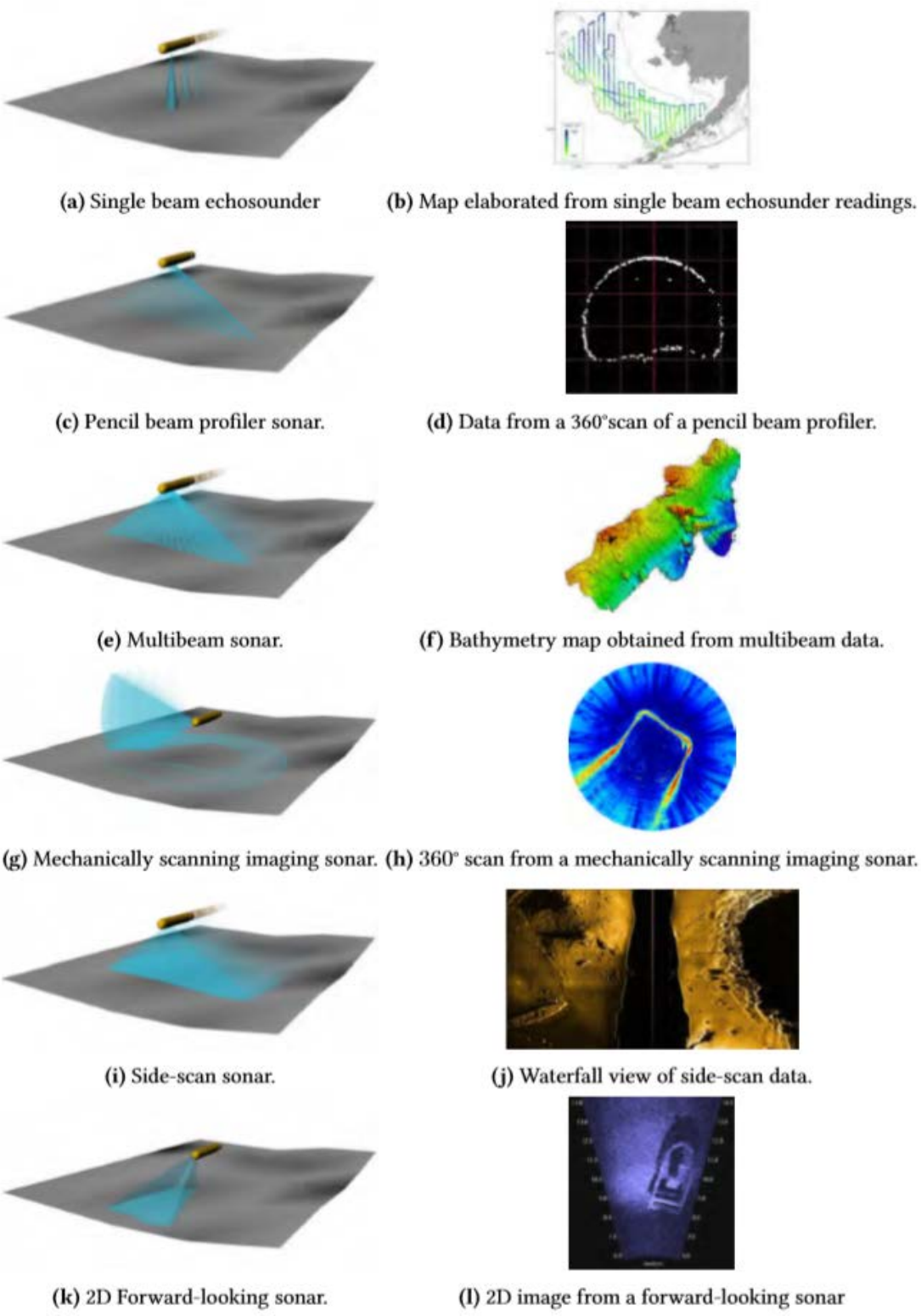


Figura 3.3: Diferents tipus de sonars submarins i els seus resultats

3.2.2. Forward-Looking Sonar (FLS)

Com s'ha dit en l'apartat anterior, els FLS són una alternativa pels entorns amb visibilitat reduïda. A diferència d'altres sonars d'imatge, els FLS, generen directament imatges acústiques 2D, donant així una visió més pròxima al que veu l'ull humà, per tant redueix el nivell de processament i interpretació que es necessita.

Els *FLS 2D,* anomenats a vegades càmeres acústiques, són una nova categoria de sonars que ofereixen imatges acústiques d'alta definició a una freqüència de refresc ràpida. Tot i algunes especificacions relatives com la freqüència de funcionament, l'amplada del feix acústic, la velocitat dels fotogrames i la tecnologia interna de formació del feix depenen del model i del fabricant de cada sonar, el principi de funcionament és igual per tots. El sonar insonifica l'escena amb una ona acústica, que s'expandeix en el seu camp de visió en les direccions d'azimut(θ) i d'elevació (Φ).Tot seguit, la intensitat del retorn acústic és mostrejada per una sèrie de transductors en funció del rang i la orientació (fig. 3.4). Degut a aquesta geometria, no és possible determinar l'angle d'elevació del retorn acústic que s'origina en una distància i orientació particular. En altres paraules, el ressò reflectit podria tenir el seu origen en qualsevol punt al llarg de l'arc d'elevació corresponent. Per tant, la informació 3D es perd quedant una imatge 2D (Hurtos, 2014).

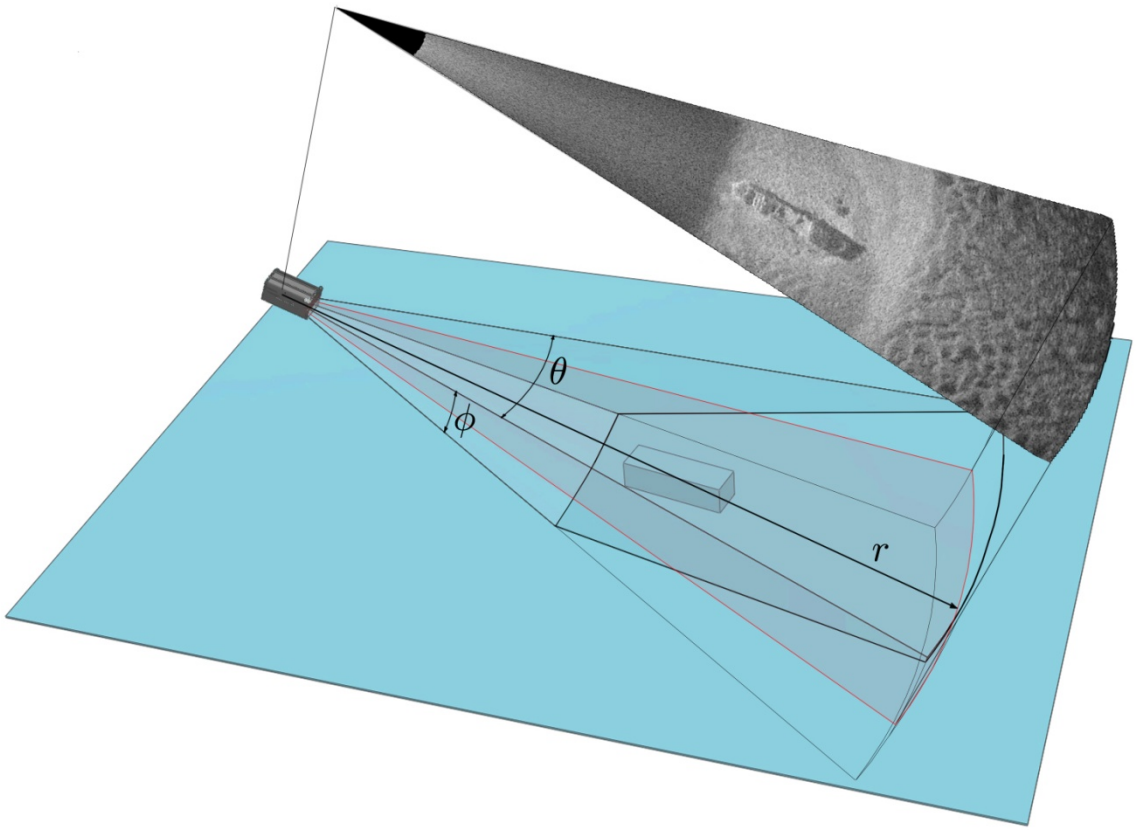


Figura 3.4: Operació del FLS. . El sonar emet una ona acústica que comprèn l'ample del feix en les direccions d'Azimut(θ) i d'elevació(Φ). L'energia del so retorna i es mostra en funció del rang i l'orientació (r,θ).

Com s'ha dit en l'apartat 3.2.1, les càmeres acústiques ens permeten generar imatges, en llocs on les càmeres òptiques no tenen visibilitat, a base de treballar amb un tipus de dades més difícil de tractar. Tanmateix els FLS no estan exempts de dificultats. A continuació s'exposen els problemes més rellevants que es poden trobar en treballar amb dades FLS:

- **Baixa resolució:** Encara que es considerin sonars d'alta resolució, la resolució d'imatge dels 2D FLS encara està molt lluny de la resolució de les càmeres estàndard actuals que fan ús de sensors 2D amb milions de píxels. La resolució d'una imatge FLS sol ser d'uns pocs centenars de píxels. Per altra banda, la naturalesa polar del sensor, fa que quan es representen les imatges en un espai cartesià, les mesures siguin més disperses a mesura que augmenta el rang. Per exemple, per al sonar ARIS configurat a 3 MHz en que el rang mínim és 1 m i el màxim 10 m, la diferència entre el píxel més gran i el més petit pot ser de 10, el que significa que un píxel de la imatge polar pot ocupar d'1 a 10 píxels de la imatge cartesiana. Per tant, això es tradueix a una resolució no uniforme que contribueix a degradar l'aparença visual de la imatge.

- **Baixa relació senyal/soroll:** Igual que amb altres sistemes com ara el radar o les imatges per ultrasons, els 2D FLS pateixen de baixa relació senyal/soroll. En el procés de construcció de mosaics, aquesta relació senyal/soroll es pot millorar mitjançant el registre i fent una mitjana de múltiples passades de la mateixa escena.

- **Insonificació no homogènia:** Els dispositius FLS normalment inclouen un mecanisme TVG (*Time Varying Gain)* que compensa la pèrdua de transmissió. D'aquesta manera, objectes similars que es troben en diferents rangs es perceben amb intensitats similars. No obstant això, un canvi en l'angle d'incidència i/o en la inclinació de la superfície, pot introduir variacions en la il·luminació de la imatge, ja sigui dins d'una seqüència o dins d'una sola imatge (Figura 3.5). Aquestes intensitats no homogènies poden afectar l'etapa de registre d'imatges, i per suposat, tenir un impacte en l'etapa de la fusió imatges.

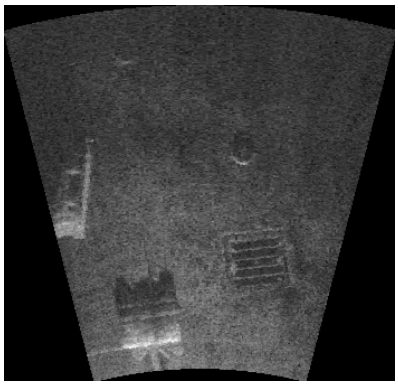


Figura 3.5: Exemple : Exemple d'una imatge capturada d'un FLS amb insonificació no homogènia. El terreny és pla però la il·luminació no és constant a causa de l'angle d'inclinació del sonar.

- **Canvis de punt de vista:** Imatges de la mateixa escena des de dos punts de vista diferents poden fer moure les ombres dels objectes presents o causar oclusions en les

imatges, és a dir, generen alteracions significatives en l'aparença visual del contingut, que compliquen el procés de registre. Per minimitzar aquests efectes, és preferible obtenir les imatges sempre des del mateix punt de vista, encara que a vegades dificulti l'obtenció de dades.

3.2.3. Dades d'entrada

És important entendre les dades d'entrada que l'aplicació ha de manegar. Per això és necessari conèixer que aquests tipus de sonars es poden utilitzar de diferents maneres: pot ser portat per un submarinista, muntat en un suport des d'una embarcació o instal·lat en un robot submarí. El processat necessita, com a mínim, les imatges que s'han capturat al llarg d'una trajectòria i, com es veurà en la següent secció, a partir d'aquestes imatges es trobaran les seves posicions per tal que encaixin de manera consistent en un mosaic. No obstant, si el sonar ha estat muntat en una embarcació o un vehicle pot ser que es tinguin dades de navegació de la trajectòria (ja sigui capturades a través d'un sensor GPS o a través dels sensors de navegació del vehicle en qüestió). Si aquestes dades són presents, poden ajudar a determinar la posició de les imatges en el mapa.

Per tant, treballarem en dos possibles casos de dades d'entrada:

- **Només imatges:** En aquest cas, el més general, només tindrem fitxers provinents del dispositiu sonar. Seran fitxers en format propietari del fabricant del dispositiu. Per el cas del sonar del CIRS, són fitxers en format ´.aris´. Els formats propietaris també podrien tenir navegació adjunta, però rarament és així i per tant assumirem que només contenen imatges.

- **Imatges i navegació:** En aquest cas comptem amb informació de posició (com a mínim x,y, theta) de la trajectòria que s'ha seguit. Aquesta navegació podria estar en molts formats, però treballarem amb el format ´.bag´ de ROS donat que és el format amb què els vehicles dels CIRS guarden tota la informació d'una missió. Així doncs un fitxer '.bag' conté tant les imatges com les dades de navegació.
-

3.2.4. *Pipeline* de construcció de mosaics acústics

És important entendre bé el sistema de creació de mosaics, per el qual es vol crear la interfície d'usuari. A nivell conceptual, el procediment a seguir per crear un mosaic consta essencialment de tres fases:

1. El registre de parelles d'imatges.
2. L'alineament global de tot el conjunt d'imatges.
3. Renderitzat del mosaic.

La figura 3.4 ofereix un resum visual dels passos de la construcció de mosaic i tot seguit es descriu cadascun d'ells amb més detall.

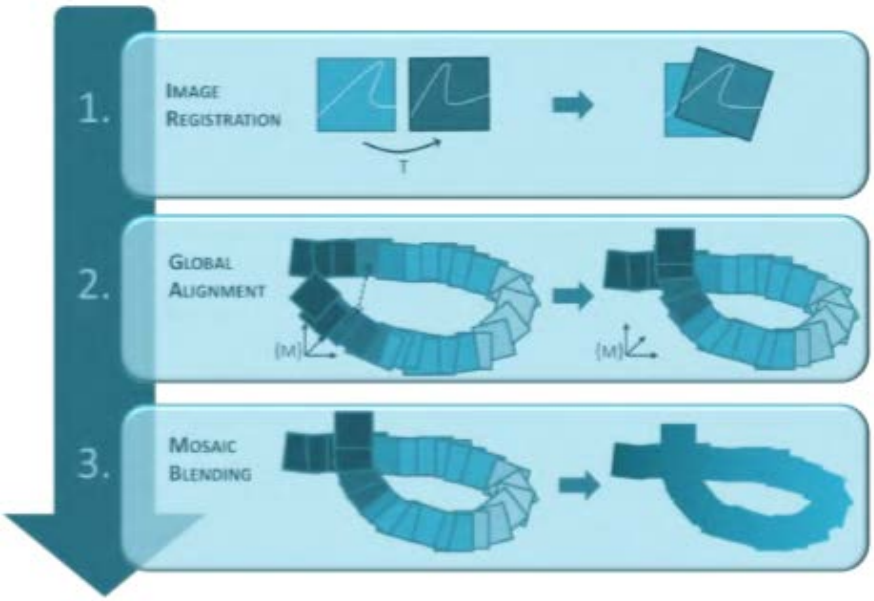


Figura 3.4: Pipeline de la construcción de mosaics acústics

Registre d'imatges

El registre d'imatges sonar és un pas clau en el sistema de construcció de mosaics, consisteix en trobar la transformació (en aquest cas rotació i translació) que existeix entre una parella d'imatges per tal de poder-les situar en el mateix sistema de coordenades. Les característiques de les imatges dels sonars de visió frontal, com ara la baixa resolució, la baixa relació senyal-soroll, o les variacions d'intensitat a causa dels canvis de punt de vista, causen moltes complicacions a les tècniques que típicament s'utilitzen per registrar imatges òptiques. Els algoritmes desenvolupats al CIRS treballen en el domini freqüencial, el qual té en compte tot el contingut de les imatges en el procés de registre, aconseguint així més robustesa davant del soroll i dels diferents artefactes associats amb la formació de la imatge acústica. El mètode es basa en el principi de la correlació de fase per computar els desplaçaments de la imatge i està adaptat per fer front a les múltiples fonts de soroll que poden influenciar el registre. Donat que es treballa en el domini freqüencial, i les vores de les imatges són un component d'alta freqüència, el mètode de registre requereix emmascarar aquestes vores per tal que no influeixin i només es tingui en compte el contingut real de les imatges per dur a terme l'alineació de les imatges. A més a més, del mateix mètode de registre es deriva una mesura d'incertesa que ens dóna una idea de "com de bo" és el registre computat. Comparacions quantitatives demostren que aquest mètode té un rendiment superior a altres tècniques de registre de l'estat de l'art basades en punts d'interès i ofereix, al mateix temps, la possibilitat de ser implementat eficientment.

Alineament Global

L'alineament global del mosaic consisteix en imposar consistència entre totes les parelles d'imatges, ja siguin consecutives o no consecutives. Es tracta d'optimitzar un graf on els nodes són les posicions de les imatges i les restriccions espacials entre ells provenen del registre de les diferents parelles o de les dades de navegació, en cas de disposar d'aquestes. L'estratègia a seguir per determinar quines restriccions s'han d'incorporar al graf es fa d'acord amb una estimació inicial de la trajectòria del vehicle i una selecció de les parelles d'imatges que poden tenir solapament. Per tal de donar el pes adequat a cada restricció dins de la optimització, s'utilitza la mesura d'incertesa que es deriva del mètode de registre. És com un sistema de molles on, segons les posicions que s'han computat entre cada parella d'imatges, s'intenta arribar a un estat d'equilibri que determina les posicions finals de cada imatge.

Fusió de les imatges

L'últim pas, és la fusió de les imatges acústiques en un únic mosaic d'aparença nítida i informativa, aconseguint al mateix temps una relació senyal-soroll i una resolució millorades respecte les de les imatges individuals. Bàsicament aquest procés consisteix en fer una mitjana dels valors dels píxels de totes les imatges que tenen solapament a una mateixa posició.

3.3. Eines utilitzades

A continuació s'expliquen les eines que s'han fet servir en el desenvolupament d'aquest projecte i que formen part del marc de treball que s'ha establert.

3.3.1. C++

Figura 4.5: Logotip C++

El C++ és un llenguatge de programació que suporta la programació orientada a objectes. Bjarne Stroustrup, el seu creador, deia que la intenció era crear un llenguatge que fos una extensió del llenguatge de programació C.

Aquest llenguatge està pensat per a donar suport directe i extensivament a múltiples estils de programació (*procedural programming,* abstracció de dades, programació orientada a objectes

i programació genèrica. A més a més, també està dissenyat per donar sempre al programador l'oportunitat d'escollir, fins i tot si el programador escull erròniament (C++, 2016).

El C++ és un llenguatge flexible, que permet fer les coses de moltes maneres diferents, com més opcions hi ha, més possibilitats existeixen de millorar un fragment de codi. Al ser un llenguatge tan versàtil, els programes que s'executen tenen una millora de velocitat respecte altres llenguatges, com pot ser el Java.

La interfície gràfica s'ha desenvolupant en C++, degut a que el projecte original com les seves llibreries, estaven escrites en aquest llenguatge. A més a més, per desenvolupar la GUI, s'han fet servir les llibreries Qt, que estan escrites també en C++, fet que impulsa a seguir escrivint en aquest llenguatge.

3.3.2. Qt

Figura 4.6: Logotip Qt

Qt és un *framework* multiplataforma per a la creació de programes amb interfície gràfica d'usuari o GUI. Les seves llibreries estan escrites en C++. Qt es desenvolupa com un *software* lliure i de codi obert a través de Qt Project.

Les llibreries Qt també poden ser utilitzades per altres llenguatges de programació, com el Python, Ruby, C, Perl o Pascal , a través dels *bindings.*

L'entorn de desenvolupament Qt s'utilitza en molts programes coneguts com poden ser: Autodesk, Adobe Photoshop Elements, Skype, VLC media player i VirtualBox.

S'han escollit les llibreries Qt per al projecte, degut a que estan escrites en C++, fet que ajuda a integrar-les dins dels *software* original d'una manera més senzilla. A més a més, hi ha l'opció d'adquirir les llibreries sota llicència GPL o sota ús comercial. Qt té un gran suport per la comunitat que l'envolta, a més d'estar molt ben documentat en la seva web, això ajuda en el desenvolupament inicial, on es desconeix l'ús de les llibreries. L'aplicació està pensada per treballar sobre UNIX, però les llibreries Qt al ser multiplataforma, permeten convertir el codi fàcilment a altres plataformes.

3.3.3. Git

Git és el *software* de control de versions que s'utilitza al grup de recerca. Va ser dissenyat per Linus Torvalds i està pensat per a l'eficiència i facilitat de manteniment de versions d'aplicacions amb una gran quantitat d'arxius de codi font. Destaquen els següents aspectes:

- Desenvolupament distribuït.
- Eficient a l'hora de mantenir projectes grans.
- Xifrat de l'historial de versions.
- Existència de *plug-ins* per a la integració amb *Qt Creator.*

L'ús d'aquest tipus de *software* és essencial en un projecte com aquest, permet treballar de manera local i fer tots els canvis en el propi sistema. Quan s'ha acabat de programar i de provar les funcions que s'estan desenvolupant, després es pot penjar al núvol i compartir amb la resta de l'equip.

3.3.4. SQLite

Figura 4.7: Logotip SQLite

SQLite és un projecte de domini públic creat per D. Richard Hipp que implementa una petita biblioteca programada en C, que funciona com un sistema de gestió de base de dades relacionals (Cabrero & Maldonado, 2007).

A diferencia dels motors de base de dades convencionals com l'arquitectura *client-servidor,* SQLite es independent, ja que no es comunica com un motor de base de dades, sinó que les llibreries passen a integrar l'aplicació. La mateixa aplicació utilitza les funcionalitats d'SQLite a través de crides simples a subrutines i funcions.

Per tant, SQLite són unes llibreries molt útils en el nostre projecte, ja que el que intentem, és ser el màxim eficients i donar un temps de resposta petit per cada funció. Al tenir les llibreries integrades dins del projecte millora considerablement el temps de resposta, reduint la latència en els accessos a la base de dades, degut a que les funcions són molt més eficients que la comunicació típica entre processos. A més a més, el conjunt de la base de dades es guarda com un sol fitxer estàndard, en la màquina local.

4. Anàlisi

En l'anàlisi s'ajuda a identificar tant els requeriments del sistema com les seves funcions. En aquest apartat es detallen els requisits del sistema (secció 4.1) i l'anàlisi de casos d'ús (secció 4.2).

4.1. Requisits del sistema

En aquest apartat es descriu els requisits del sistema, tant funcionals com no funcionals, que expliquen a grans trets els objectius de l'aplicació.

4.1.1. Requeriments funcionals

En enginyeria del *software*, els requeriments funcionals defineixen quins són els serveis que oferirà l'aplicació. En el nostre cas, els requeriments funcionals que s'han definit són els següents:

- La interfície ha de permetre crear un projecte a partir dels fitxers que enregistra el sonar.
- La interfície ha de permetre configurar els processos que intervenen en la construcció de mosaics.
- La interfície ha de permetre l'execució simultània dels processos de construcció de mosaics, sense que això afecti al rendiment d'aquesta.
- La interfície ha de permetre visualitzar gràficament la trajectòria realitzada, ja sigui a partir de les dades de navegació del projecte o de la navegació obtinguda un cop creat el mosaic.
- La interfície ha de permetre visualitzar el conjunt d'imatges que conté un projecte de manera eficient independentment del nombre d'imatges que aquest contingui.
- La interfície ha de permetre visualitzar el graf optimitzat que es generà en el procés de construcció de mosaics.
- La interfície ha de permetre visualitzar el mosaic final que s'obté un cop finalitzats tots els processos.
- La interfície ha de permetre guardar i carregar l'estat en el que es troba un projecte.
- La interfície ha de permetre a l'usuari crear registres de forma manual a partir de dues imatges. Aquest registres s'han de poder combinar amb els registres creats de forma automàtica.

4.1.2. Requeriments no funcionals

Els requeriments no funcionals ens informen de les restriccions globals en un sistema de *software,* imposades normalment pel client, com per exemple: Restriccions de costos de

funcionament, de rendiment, de desenvolupament, de fiabilitat... En el nostre cas, els requeriments no funcionals que s'han definit són els següents:

- La interfície ha de ser intuïtiva. Els processos han d'estar ben definits i a la vista i s'ha d'evitar col·locar elements que confonguin l'usuari. S'ha de tenir en compte que aquesta interfície ha de ser utilitzable per personal no tècnic.
- La interfície ha de carregar ràpidament la representació de dades gràfiques com les visualitzacions de navegació, grafs, mosaics... sense alentir el procés de creació de mosaics.
- La interfície ha de ser fàcil de mantenir. Al realitzar-se en un grup de recerca, ha de ser fàcil d'adaptar el codi a nous canvis per part d'altres membres de l'equip.
- La interfície ha de ser extensible. S'ha de permetre estendre la funcionalitat d'una manera fàcil.
- La interfície ha de ser modular. Ha de conviure amb el software original sense generar conflictes.

4.2. Anàlisi de casos d'ús

Els diagrames de casos d'ús, donen una visió general simple dels diferents casos d'ús que es poden trobar dins del *software.* La figura 4.1 mostra el diagrama de casos d'ús de context del sistema, on es poden identificar els diferents casos d'ús i l'actor que intervé, que serà l'usuari de l'aplicació.

4.2.1. Visió general

A partir de l'anàlisi dels requeriments podem entendre que el sistema haurà de constar de les següents funcions:

- Crear un nou projecte.
- Carregar un projecte existent.
- Guardar l'estat del projecte.
- Visualitzar la navegació (en cas que n'hi hagi).
- Visualitzar les imatges a través d'un visor d'imatges.
- Visualitzar el graf.
- Veure el mosaic resultant.
- Crear nous registres manualment.
- Editar els paràmetres de configuració de manera interactiva.
- Executar els processos de construcció de mosaics amb els paràmetres definits.

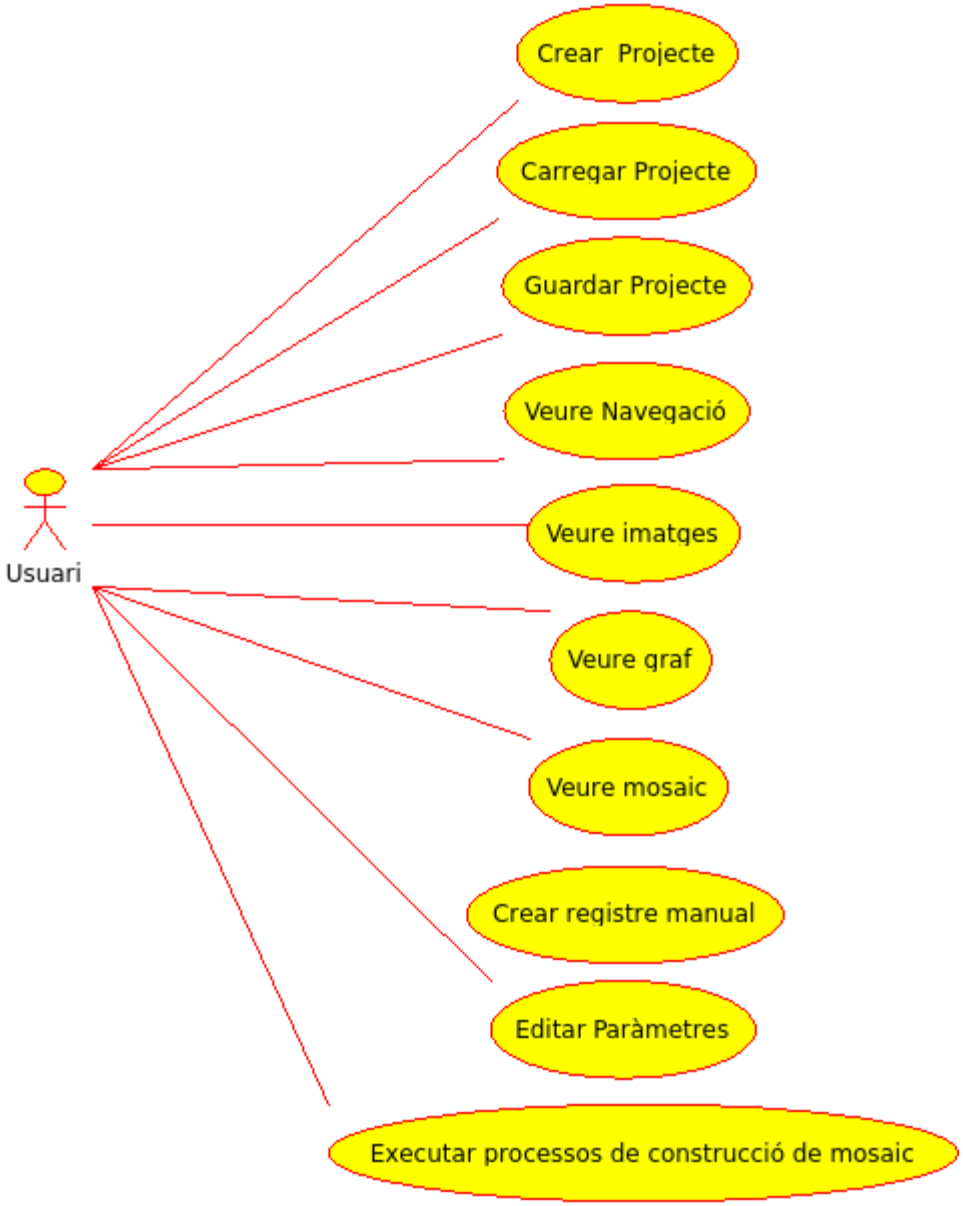


Figura 4.1: Diagrama de context del sistema

4.2.2. Funcions del Sistema

Un cop descrita una visió general del que seran les funcions del sistema , a continuació es detalla cada funció punt a punt, per tal de determinar una primera versió de cada cas d'ús.

Crear un nou projecte

- Ha de permetre crear un nou directori on es desaran les dades.

- Ha de crear una taula en una base de dades, aquesta taula contindrà totes les configuracions del projecte.
- Seleccionar un arxiu que contingui les dades obtingudes del sonar i/o de la navegació, identificant si és un arxiu amb extensió '.aris' o '.bag', i extreure les dades necessàries en el directori seleccionat.

Carregar un projecte existent

- S'ha de poder seleccionar un directori que contingui un projecte, identificar-lo i carregar les dades que contingui dins de la interfície per tal de restablir l'estat del projecte.

Guardar un projecte

- Ha de permetre guardar l'estat del projecte actualitzant el valor de tots els paràmetres de configuració per l'estat actual indicat a la interfície.

Visualitzar la Navegació

- En cas que el fitxer d'entrada contingui dades de navegació, s'ha de poder visualitzar aquesta navegació en un gràfic 2D. En cas que no hi hagi navegació en els arxius originals, un cop generats els registres, s'ha de poder visualitzar la navegació obtinguda a partir de la construcció del mosaic.

Visualitzar les imatges

- Ha de permetre carregar les imatges dins de l'aplicació.
- Les imatges s'han de mostrar en un visor d'imatges de manera eficient, de tal manera que només es mantinguin aquelles que es veuen dins de la interfície, una gran quantitat d'imatges carregades al mateix temps podria consumir tots els recursos del sistema.
- S'ha de poder seleccionar la imatge en el visor i que aquesta es mostri més ampliada a la finestra.
- S'ha de poder moure amb les tecles de direcció (esquerra i dreta) a través del visor d'imatges, de manera que sigui fàcil el desplaçament entre imatges.

Visualitzar el graf

- Ha de permetre generar un graf de forma gràfica que mostri la relació entre els registres de les imatges.
- Ha de ser possible visualitzar el graf dins de la interfície.
- S'ha de poder fer zoom i desplaçar-se a través del graf d'una manera senzilla.
- La incertesa en els registres s'ha de representar a partir del color de les arestes que uneixen els nodes, d'aquesta manera l'usuari pot veure ràpidament quins registres són menys fiables i ajustar el llindar per a tenir un millor mosaic.

Executar els processos de construcció del mosaic

- Un cop les dades han estat extretes i els paràmetres de configuració han estat introduïts per part de l'usuari, aquest ha de poder executar el procés de construcció de mosaics.
- Els processos de construcció de mapes, s'han d'executar en paral·lel amb la interfície, per tal que aquesta no és "congeli".
- A mesura que els processos van generant resultats, aquest s'han de desar de forma permanent. EL procés de creació d'un mosaic és llarg i és important que si alguna cosa falla no s'hagi de tornar a començar de zero.

Veure el mosaic resultant

- Un cop el procés de construcció del mosaic ha finalitzat i es té una estimació de la posició de cada imatge sonar, ha de ser possible renderitzar el mosaic resultant mesclant, en una única imatge, totes les imatges individuals.
- El mosaic s'ha de mostrar de manera eficient, i ha de ser possible ampliar-lo o reduir-lo i desplaçar-se a través d'ell.

Crear nous registres manualment

- L'usuari ha de ser capaç de seleccionar dues imatges i superposar-les de tal manera que coincideixin creant d'aquesta manera un nou registre.
- Per tal que les imatges coincideixin, l'usuari ha de poder desplaçar una imatge sobre de l'altre utilitzant el teclat de l'ordinador, o a través d'uns botons en la interfície.
- El registre s'ha de guardar a la mateixa base de dades on hi ha els registres obtinguts de forma automàtica.
- Les imatges que es seleccionen per fer el registre, s'han de poder escollir a través d'un visor d'imatges que anirà mostrant les imatges que existeixen en el projecte.

Editar els paràmetres

- S'han de poder editar els següents paràmetres dins de la interfície:

  - **Step:** El paràmetre Step determinarà el salt entre imatges. Per exemple, un Step de 2, farà que s'utilitzi 1 imatge de cada 2. Aquest paràmetre és útil per si les dades originals s'han capturat amb molts *frames* per segon i volem reduir el número d'imatges que intervindran en la construcció del mosaic.

  - **Force Rotation to 0:** Si aquesta casella està activada, la rotació sempre valdrà zero, en cas que estigui desactivada, es s'estimarà una rotació per a cada registre. Aquest paràmetre és útil quan l'usuari sap que la tarjectoria s'ha realitzat sense rotacions i vol evitar que el mosaic es deformi com a conseqüència d'acumular petites rotacions que l'algoritme de registre pot estimar entre dues imatges.

  - **From Image Center:** Si està activada es computarà la rotació entre imatges des del centre d'aquestes, enlloc de computar la rotació respecte l'origen del sonar.

  - **Navigation:** Si aquesta casella està activada, en el procés d'alineament global, s'utilitzarà la navegació resultant de la concatenació de les imatges enlloc de la navegació original del dataset provinent de sensors externs de navegació.

  - **Offset: :** Aquest camp conté el desplaçament i la rotació entre el centre el sistema de coordenades que s'ha definit com a centre de la navegació i el sistema de coordenades del sensor sonar.

  - **Candidate selection:** Permetrà escollir com volem emparellar els registres per intentar registrar-los entre sí.
    - **Sequence Window:** Es registra cada imatge amb les anteriors amb un màxim d'imatges definit per l'usuari.
    - **Time Window (seg):** Es registra cada imatge amb les anteriors sempre i quan el temps entre l'actual i l'anterior sigui inferior a un valor definit per l'usuari.
    - **Distance Window (m):** Es registra cada imatge amb totes les anteriors sempre i quan la distància entre elles sigui inferior a un valor definit per l'usuari. Per aplicar aquest mode és necessari disposar de navegació.
    - **All with All:** Es registren totes les imatges entre elles, és a dir, cada imatge es registra amb totes les altres.

  - **Goodness Threshold:** Determina el llindar de confiança per tal d'acceptar o descartar un registre durant el procés d'alineament global. .

- Els paràmetres han de ser fàcils d'identificar i s'han de poder editar en qualsevol moment.
- Els paràmetres han de tenir una configuració robusta i fiable per evitar valors que el sistema no toleraria.

5. Disseny

Un cop explicat de manera detallada les necessitats del sistema en l'apartat anterior (Anàlisi), en aquest apartat de Disseny, es proposa una solució que compleix les necessitats especificades. Donat que s'ha programat en C++ i que les tècniques d'orientació a objectes són la millor manera actualment de desenvolupar grans projectes, aquest disseny estarà orientat a objectes.

Aquest projecte es basa en crear una interfície gràfica, per aquest motiu la classe principal de l'aplicació serà la *GUI*, a partir de la qual és gestionaran tots els processos. Com a patró de disseny, s'ha agafat el Model-Vista-Controlador (*Model-View-Controller*). Aquest patró separa l'aplicació en tres components diferents que estan interconnectats (CakePHP, 2016):

- **Model:** És la representació de la informació que fa servir l'aplicació. És a dir, s'encarrega de la gestió de les dades que intervenen en l'aplicació, d'acord amb les comandes del controlador i proporcionant-les a la vista quan és necessari.
- **Vista:** fa una presentació de les dades del model, és a dir, és el component encarregat de mostrar a l'usuari la representació de dades en un format adequat per tal de que aquest hi pugui interactuar.
- **Controlador:** gestiona les peticions d'usuari en l'aplicació, processant-les i convertint-les a comandes que actualitzaran o bé el model o bé la vista.

Tal i com s'ha comentat, aquest projecte ha partit d'una sèrie de processos existents, que conformen essencialment el component de controlador. Així doncs aquests processos només s'han hagut d'adaptar lleugerament i convertir en classes, mentre que els components de Model i Vista han significat la part principal desenvolupada en aquest projecte.

5.1. Diagrama de classes

La Figura 5.1 presenta el diagrama de classes que s'ha dissenyat per tal d'abordar totes les funcions de l'aplicació. En els següents apartats es detallaran les diferents classes, dividint-les d'acord amb el patró Model-Vista-Controlador que s'ha adoptat.

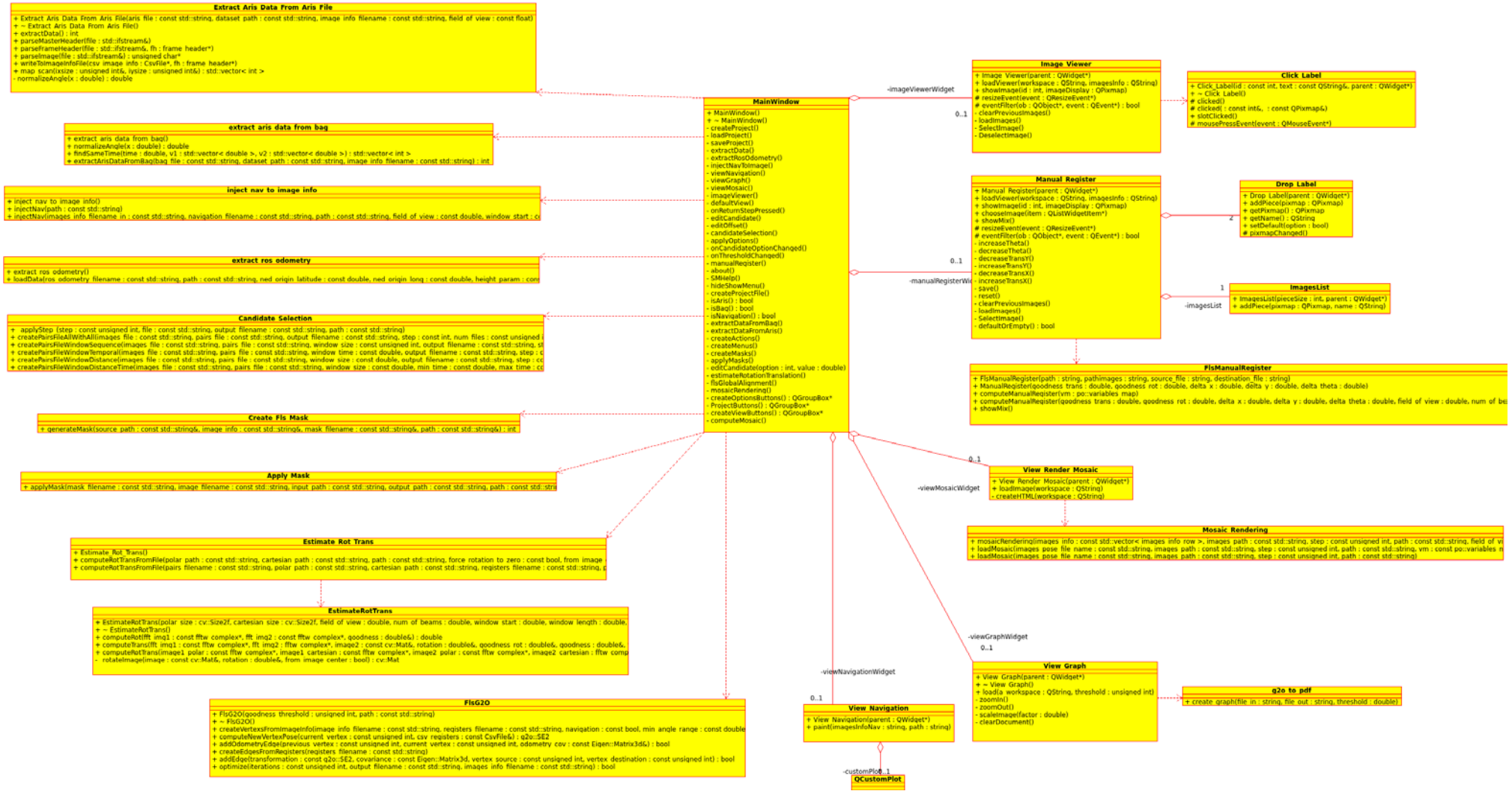


Figura 5.1: Diagrama de classes de disseny de la interfície

5.2. Model

El model representa aquella part del sistema que s'encarrega de gestionar les dades i convertir-les en un concepte útil per a l'aplicació. En el nostre cas el model està format per les imatges de cada *dataset* així com totes les dades que es guarden i/o computen al llarg del procés de creació d'un mosaic (fitxers de màscares, registres de parelles d'imatges, incerteses, etc).

5.2.1. Disseny de la base de dades

Al començament de l'aplicació, abans que es decidís afegir la funcionalitat d'una interfície, l'aplicació funcionava amb un conjunt de fitxers CSV (*comma separated values)* que s'utilitzaven com a base de dades. Degut al fet de treballar en fitxers, el temps de resposta i la latència eren bastant grans quan s'accedia a la base de dades.

És per això que es va decidir utilitzar la tecnologia SQLite en el projecte. Aquestes llibreries són molt més eficients i donen un temps de resposta menor per cada acció.

Les llibreries SQLite permeten utilitzar una base de dades dins de l'aplicació sense la necessitat de disposar d'un servidor SQL instal·lat i configurat en el mateix sistema. Amb aquestes llibreries, es pretén millorar la resposta i fiabilitat de l'aplicació, ja que molts algorismes accedeixen constantment a la base de dades per consultar o modificar valors.

Així doncs, per tal de gestionar totes les dades necessàries en l'aplicació, s'ha dissenyat una base de dades que consta de 5 tipus de taules diferents: la taula que ens permet guardar els paràmetres de configuració de l'aplicació, la taula d'informació de les imatges, la taula de parelles d'imatges, la taula de navegació i la taula de registres. A continuació es presenta una breu descripció de cadascuna d'elles.

Taula de Configuració

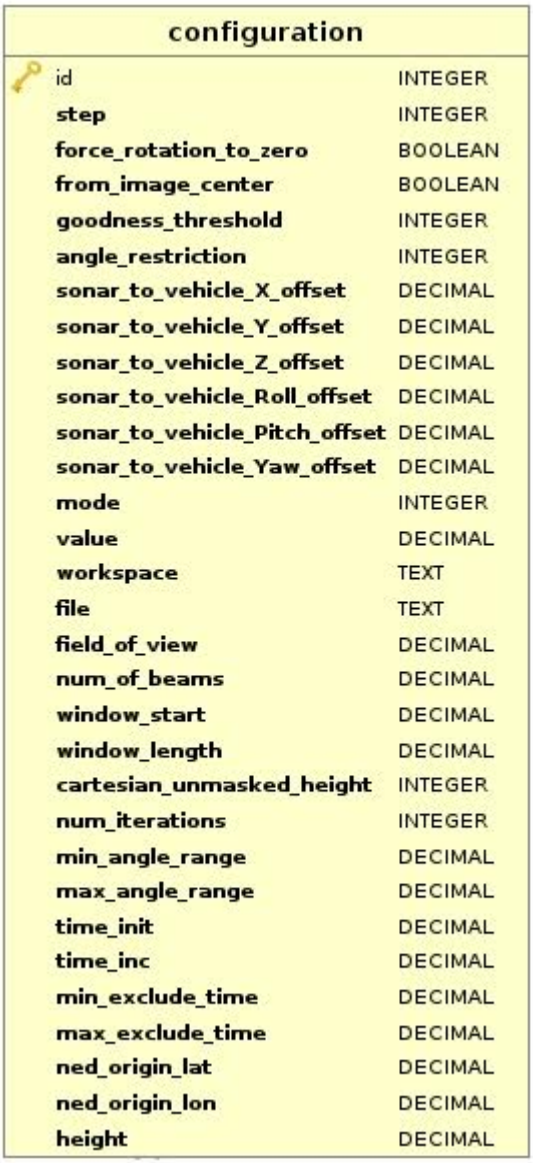


Figura 5.2: Taula de configuració

La taula de configuració (fig. 5.2) tal i com el seu nom indica, guarda les dades de configuració del projecte i està composada per els següents atributs:

- *Id* identifica quin conjunt de paràmetres de configuració està actiu, actualment només n'hi ha un, però en el futur és possible que n'hi hagi varis dins d'un mateix projecte.
- *Step:* configura el salt entre imatges per determinar les que es tindran en compte per processar el dataset.
- *Force_rotation_to_zero:* valor booleà per indicar si s'utilitza la rotació o no a l'hora de computar els registres.
- *From_image_center:* si és cert, computarà la rotació entre imatges des del centre d'aquestes en lloc de computar la rotació respecte l'origen del sonar.
- *Goodness_threshold:* determina el llindar de confiança per tal d'acceptar o descartar un registre durant el procés d'alineament global.

- *Angle_restriction:* determina si hi ha restricció en l'angle a la hora de calcular les rotacions.
- *Sonar_to_vehicle_offset:* conté el desplaçament i rotació entre el centre del sistema de coordenades en (X, Y, Z, Roll, Pitch, Yaw).
- *Mode:* determina quin mètode de selecció de parelles candidates s'ha escollit: *Sequence Window(0), Time Window(1), Distance Window(2)* o *All with all(3)* .
- *Value:* conté el valor de la opció que hi hagi en el paràmetre *Mode* (número d'imatges en el cas de *Sequence Window*, número de segons en el cas de *Time Window*, o distància en metres per *Distance Window*)
- *Workspace:* ruta on es troba el directori del projecte.
- *File:* ruta del fitxer que s'ha utilitzat per extreure les dades.
- *Field_of_view:* camp de visió del sonar
- *Num_of_beams:* nombre de feixos que emet el sonar.
- *Window_start:* distància d'inici de la finestra del sonar.
- *Window_length:* mida de la finestra del sonar.
- *Cartesian_unmasked_height:* mida de la imatge cartesiana en píxels.
- *Num_iterations:* nombre de iteracions que es faran en el procés d'optimització.
- *Min_angle_range:* angle mínim per determinar el rang.
- *Max_angle_range:* angle màxim del rang.
- *Time_init:* temps inicial del recorregut que ha seguit el sonar.
- *Time_inc:* salt de temps entre imatges.
- *Min_exclude_time:* temps més petit d'exclusió.
- *Max_exclude_time:* temps màxim per excloure.
- *Ned_origin_lat:* origen de la latitud en la navegació NED (North Est Down).
- *Ned_origin_lon:* origen de la longitud en la navegació NED.
- *Height:* Alçada a la que s'ha realitzat la trajectòria (respecte el terra).

Taula de informació de les imatges

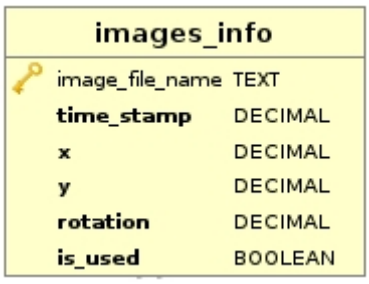


Figura 5.3: Taula de informació de les imatges

La taula *images_info* (fig. 5.3) conté la informació de les imatges capturades per el sonar. A continuació se'n detallen els atributs:

- *Image_file_name:* nom de la imatge, serà la clau principal de la imatge.
- *Time_stamp:* temps en què s'ha adquirit la imatge.
- *X:* coordenada X de la navegació, en cas que n'hi hagi, per defecte 0.
- *Y:* coordenada Y de la navegació, en cas que n'hi hagi, per defecte 0.

- *Rotation:* angle (theta) de la navegació, en cas que n'hi hagi, per defecte 0.
- *Is_used:* determina si la imatge s'utilitzarà en el procés de creació del mosaic.

Taula de parelles

Figura 5.4: Taula de parelles d'imatges

La taula *pairs* (fig. 5.4) es crea en el procés de *Candidate Selection*. Defineix totes les parelles d'imatges que posteriorment s'intentaran registrar:

- *Id:* Identificador de la parella d'imatges.
- *Image_file_name_1:* Nom de la primera imatge de la parella.
- *Image_file_name_2:* Nom de la segona imatge de la parella.

Taula de navegació

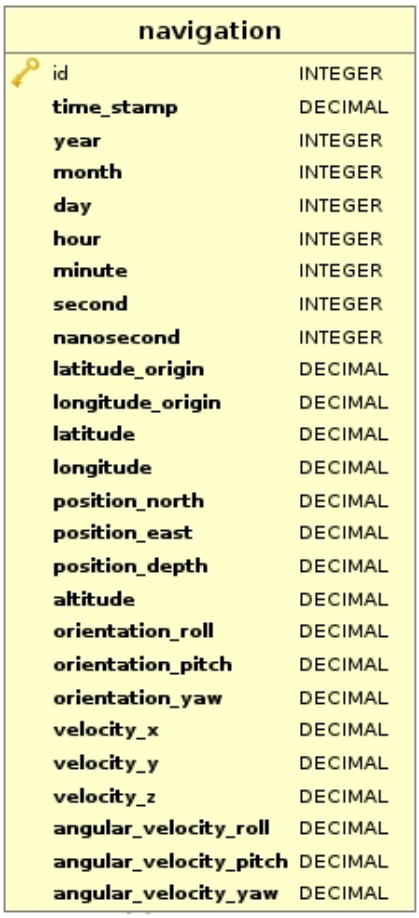


Figura 5.5: Taula de navegació

La taula de navegació es crea al extreure les dades de un fitxer en format '.bag'. Gràcies al *framework* ROS podem obtenir aquesta navegació, i a través de *sqlite* guardar-la a la base de dades.

La taula conté els següents atributs:

- *Id:* camp clau, tot i que a la pràctica s'utilitza el *time stamp*.
- *Time_stamp:* temps en què s'ha pres la mesura de navegació.
- *Year:* any.
- *Month:* mes.
- *Day:* dia.
- *Hour:* hora.
- *Minute:* minut.
- *Second:* segon.
- *Nanosecond:* nanosegon.
- *Latitude_origin:* inici de la latitud.
- *Longitude_origin:* inici de la longitud:
- *Latitude:* latitud actual.
- *Longitude:* longitud actual.
- *Position_north:* nord respecte l'origen.
- *Position_east:* respecte l'origen
- *Position_depth:* posició profunditat respecte la superfície
- *Altitude:* altitud respecte el fons
- *Orientation_roll:* orientació en *roll*.
- *Orientation_pitch:* orientació en *pitch*.
- *Orientation_Yaw:* orientació en *yaw*.
- *Velocity_x:* velocitat en X en coordenades del robot.
- *Velocity_Y:* velocitat en Y en coordenades del robot.
- *Velocity_Z:* velocitat en Z en coordenades del robot.
- *Angular_velocity_roll:* velocitat angular en roll.
- *Angular_velocity_pitch:* velocitat angular en pitch.
- *Angular_velocity_yaw:* velocitat angular en yaw.

Taula de registres

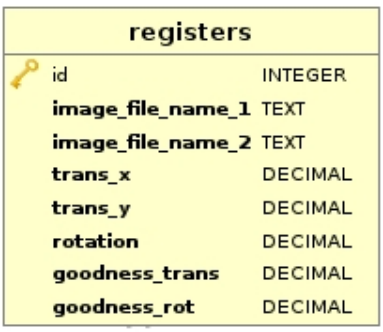


Figura 5.6: Taula de registres

La taula de registres emmagatzema els registres entre les parelles d'imatges. Cada registre conté informació sobre la translació,la rotació i la qualitat (o incertesa) del registre. Els atributs són:

- *Id:* camp clau per identificar el registre determinat.
- *Image_file_name_1:* Nom de la primera imatge.
- *Image_file_name_2:* Nom de la segona imatge.
- *Trans_X:* translació en X entre la imatge 1 i la 2.
- *Trans_Y:* translació *en Y entre la imatge 1 i la 2.*
- *Rotation:* rotació entre la imatge 1 i la 2.
- *Goodness_trans:* determina com de bona és la translació que s'ha estimat entre les dues imatges.
- *Goodness_rot:* determina com de bona és la rotació que s'ha estimat entre les dues imatges.

5.3. Vista

Formen part del component Vista totes les classes que mostren a l'usuari la representació de les dades i permeten interactuar-hi. Segons el diagrama de classe dissenyat (Fig. 5.1), l'aplicació consta de varies classes que s'encarreguen de mostrar o representar les dades. A continuació s'explicarà el funcionament de cada una.

5.3.1. Finestra principal de la interfície (*MainWindow*)

Com a interfície gràfica que es tracta, s'ha decidit optar per tenir una classe principal que hereda de *QMainWindow* (biblioteca especifica de Qt que determina la finestra principal), a partir de la qual es generaran la resta de processos. Aquesta classe té la seva pròpia UI (User *Interface*) com es pot veure a la figura 5.7, a més a més gestiona la resta de classes que tinguin també UI.

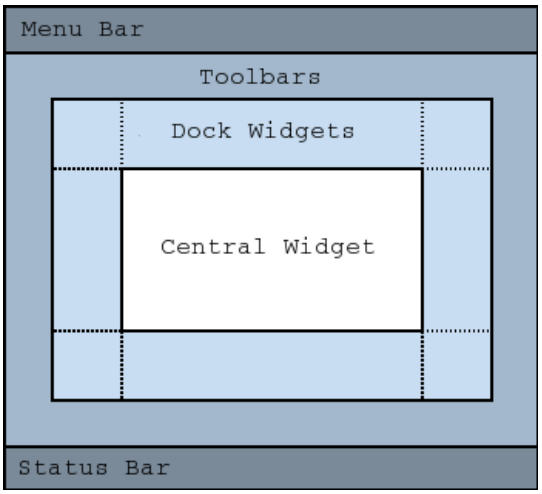


Figura 5.7: Composició de la classe MainWindow.

Per poder gestionar les diferents parts de la interfície, les llibreries Qt estan dotades de sistemes de *signals and slots*, que s'activen quan hi ha un *trigger* o un *event.* És a dir, una senyal (*signal)* és una notificació que un objecte emet quan té lloc algun canvi., ja que aquest canvi pot incidir sobre un altre objecte. Un *slot* és la funció que s'activa quan s'emet una determinada senyal. Per exemple, podem tenir un *signal* que sigui un botó per fer *zoom*, llavors quan aquest botó es prem, s'activarà un *slot* que serà la funció per ampliar la imatge.

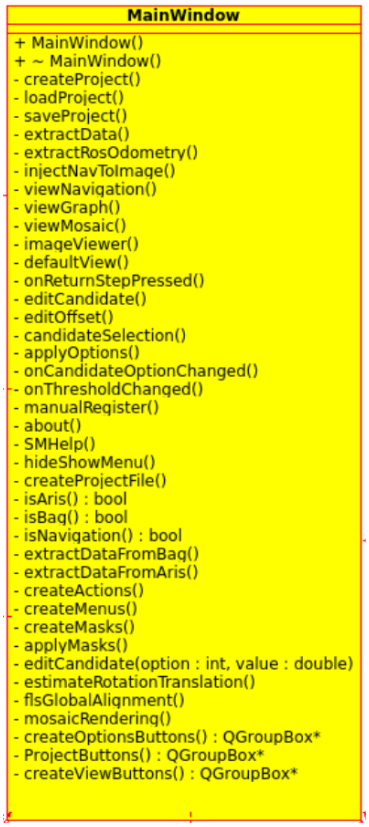


Figura 5.8: Classe *MainWindow*

La classe *MainWindow*(fig. 5.8) és la classe de més alt nivell de l'aplicació i consta de les següents operacions:

- *CreateProject:* crea un nou projecte a partir del directori i del fitxer que entrarà l'usuari.
- *LoadProject:* carrega un projecte existent en el sistema.
- *SaveProject:* guarda l'estat actual del projecte.
- *extractData: :* extreur les dades del fitxer entrat, reconeixent si és un fitxer de format '.aris' o un '.bag'.
- *extractRosOdometry:* en el cas que sigui un fitxer en format '.bag', extreu la odometria de la trajectòria guardada amb ROS.
- *injectNavToImage:* en les cas que sigui un fitxer en format '.bag', injecta la navegació a la taula d'informació de les imatges.
- *ViewNavigation:* mostra la navegació.
- *ViewGraph:* mostra el graf d'optimització.
- *ImageViewer:* mostra un visor d'imatges amb totes les imatges del projecte.

- *DefaultView*: mostra una pantalla per defecte que dona la benvinguda a l'aplicació.
- *OnReturnStepPressed:* actualitza la base de dades amb els nous paràmetres de configuració. Aquesta funció s'activa quan es prem la tecla *Enter* al teclat sobre la casella del paràmetre *Step*.
- *EditCandidate:* obre una nova finestra per seleccionar el paràmetre de *Candidate Selection.*
- *EditOffset:* obre una nova finestra per seleccionar el paràmetre de *Candidate Selection.*
- *CandidateSelection:* executa la funció de *Candidate Selection* per tal de determinar com s'emparellaran les imatges.
- *ComputeMosaic:* activa la seqüència dels processos: *Candidate Selection, Estimate Rotation and Translation* i *fls Global Alignment.*
- M*anualRegister:* permet crear registres manualment a partir de dues imatges que s'hagin escollit.
- *About:* mostra un missatge d'informació sobre l'aplicació.
- *SMHelp:* mostra l'ajuda.
- *hideShowMenu:* mostra o amaga el menú d'opcions que es troba a la banda lateral esquerra.
- *CreateProjectFile:* crea la base de dades amb la taula de configuració.
- *isAris:* retorna cert si el fitxer és de format '.aris'.
- *isBag:* retorna cert si el fitxer és de format '.bag'.
- *isNavigation:* retorna cert si el fitxer conté navegació.
- *extractDataFromBag:* extreu les dades d'un fitxer en format '.bag' (les imatges cartesianes, les polars i la navegació).
- *extractDataFromAris:* extreu les dades d'un fitxer en format '.aris' (les imatges cartesianes, les polars i la navegació en cas que hi sigui).
- *createActions:* inicialitza les QActions per poder utilitzar els mètodes del menú.
- *createMenus:* inicialitza els menús que es troben a la part superior de la interfície.
- *createMasks:* crea les màscares de les imatges en forma cartesiana i polar.
- *applyMasks:* aplica les màscares sobre totes les imatges, tan en forma cartesiana com en polar, i genera els directoris que contenen les imatges resultants .
- *estimateRotationTranslation:* executa el procés per a calcular els registres de les parelles d'imatges.
- *flsGlobalAlignment:* executa el procés per a calcular l'alineament global.
- *mosaicRendering:* mostra el renderitzat del mosaic resultant.
- *createOptionsButtons:* crea tots els botons que hi ha a la interfície.

5.3.2. Visor d'imatges (*Image_Viewer*)

El visor d'imatges és un *Widget* que pot mostrar dades, rebre informació de l'usuari i contenir altres grups de *Widgets.*

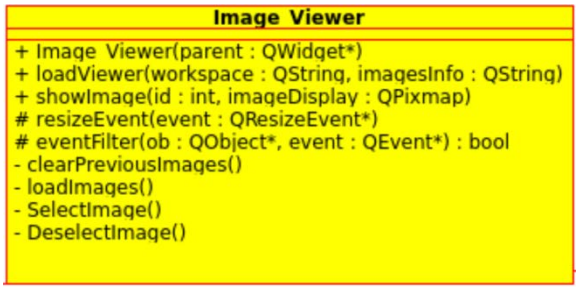


Figura 5.9: Classe Image_Viewer

Per tant, el visor d'imatges anomenat *imageViewerWidget*(fig. 5.9), és un *Widget* que hereta de *QWidget,* es crida des de la classe principal i llavors es mostra per pantalla dins de la interfície. Els seus mètodes són:

- *loadViewer:* carrega les imatges a la classe.
- *showImage:* posa la imatge seleccionada en el centre del visor.
- *resizeEvent:* funció heretada, quan la pantalla es redimensiona, reescala el visor d'imatges.
- *eventFilter:* funció heretada, captura els *events* que hi pot haver. En aquest cas s'ha fet que quan es premin les tecles de direcció esquerra i dreta, l'usuari pugui desplaçar-se pel visor seleccionant una imatge determinada.
- *clearPreviousImages:* permet eliminar les imatges carregades en el visor.
- *loadImages:* permet carregar les noves imatges sobre el visor.
- *SelectImage:* permet posar color al marge de la imatge, per identificar-la i saber quina tenim seleccionada.
- *DeselectImage:* elimina el color al marge de la imatge, per determinar que ja no la tenim seleccionada.

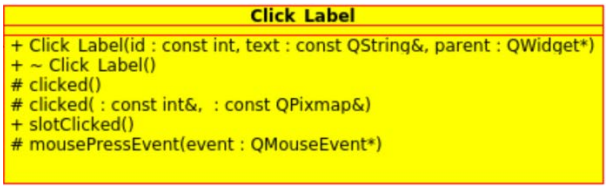


Figura 5.10: Classe Click_Label

Les imatges estan carregades sobre una classe anomenada *Click_Label*(fig. 5.10) aquesta classe hereta de QLabel, però s'ha editat per tal que l'ítem es pugui clicar. D'aquesta manera es permet que les imatges es puguin seleccionar en el visor.

5.3.3. Visualitzar la navegació (*View Navigation*)

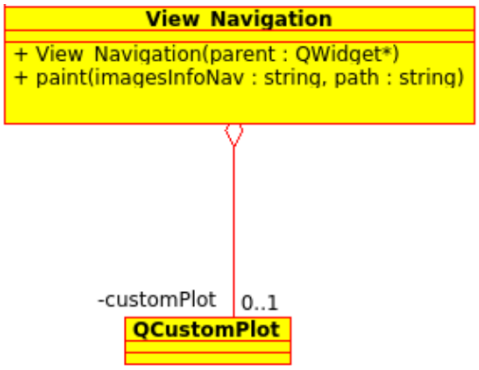


Figura 5.11: Classe View_Navigation i QCustomPlot

La classe *View_Navigation* permet visualitzar les dades de navegació a través d'un *Widget* extern anomenat *QCustomPlot.* Aquesta biblioteca esta pensada per mostrar figures 2D de gran qualitat, grafs i gràfics.

El mètode d'aquesta classe és:

- *Paint:* llegeix les posicions de les imatges i, mostra a través d'un gràfic 2D com estan disposades espaialment per tal que l'usuari pugiu fer-se un idea de quina trajectòria s'ha seguit alhora d'adquirir les imatges.

5.3.4. Visualitzar el graf (*View Graph*)

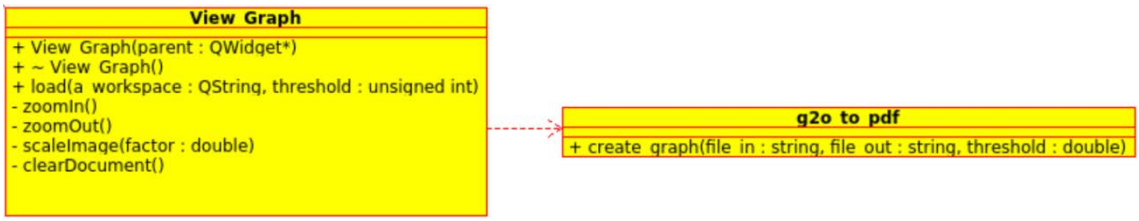


Figura 5.12: Classe View_Graph i g2o_to_pdf

Per tal de poder visualitzar el graf que es construeix en el procés de l'alineament global s'ha creat un *widget* anomenat *viewGraphWidget* que mostrarà de manera interactiva un *pdf* que contindrà el graf. Aquest *pdf* es crea a partir del fitxer g2o que utilitzen les llibreries d'optimització global i que es genera un cop computats els registres.

Els mètodes d'aquesta classe son els següents:

- *Load:* donat el directori del projecte i un *threshold* establert dins de la interfície, es cridarà a un mètode de la classe *g2o_to_pdf* anomenat *create_graph* que crearà el pdf a partir de *Load:* donat el directori del projecte i un llindar d'incertesa establert a partir de la interfície, es crida a un mètode de la classe *g2o_to_pdf* anomenat *create_graph* que crea el pdf a partir de l'arxiu g2o. La incertesa en els registres es representa utilitzant el color de les arestes que connecten els nodes (posició de les

imatges) del graf. Un cop creat el graf en format *pdf*, gràcies a les llibreries *Poppler,* es carrega el *pdf* dins de la interfície.

- g2o. La incertesa en els registres es representarà a partir del color de les arestes que connecten els nodes Un cop creat, gràcies a les llibreries *Poppler,* es carregarà el pdf dins de la interfície.
- *ZoomIn* i *ZoomOut:* permeten ampliar i reduir la visualització del graf.
- *scaleImage:* segons el factor que se li passi, escalarà la imatge per fer-la més gran o més petita.
- *clearDocument:* eliminarà el graf de la interfície.

5.3.5. Visualitzar el Mosaic Renderitzat (*View Render Mosaic*)

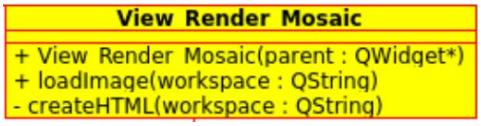


Figura 5.13: Classe View_Render_Mosaic

Un cop acabats tots els processos de construcció de mapes i renderitzat el mosaic, el mosaic s'ha de poder visualitzar. Per fer-ho, s'ha dissenyat un *widget* que fa servir la tecnologia *QtWebKit* per poder mostrar el mosaic en un visor web. S'ha adoptat aquesta solució ja que la imatge final del mosaic pot ser de gran dimensions, s'utilitzen les llibreries *Leaflet* per tal de carregar i visualitzar dinàmicament una estructura de "*tiles*" o sub-images a diferents resolucions.

El *Widget* permet fer les següents operacions:

- *CreateHTML:* crea un fitxer HTML que permet mostrar la imatge del mosaic utilitzant a la tecnologia *Leaflet.*
- *LoadImage:* carrega l'HTML que s'ha creat prèviament.

5.4. Controlador

La part que forma el component del Controlador, són totes aquelles classes que fan els càlculs necessàris a mesura que l'usuari ho demana a través dels components Vista i envien peticions al Model per sol·licitar o guardar algun tipus de dades.

5.4.1. Extract Aris Data

Hi ha dues classes que poden intervenir en el procés d'extracció de dades:
Extract_Aris_Data_From_Aris_File i *Extract_aris_data_from_bag.*

Extract Aris Data From Aris File

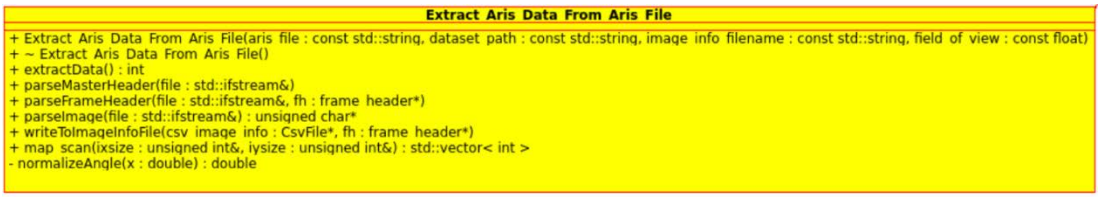


Figura 5.14: Classe Extract_Aris_Data_From_Aris_File

En el cas que sigui un fitxer amb format '.aris' es cridarà *Extract_Aris_Data_From_Aris_File* (fig. 5.14) , que està composada per les següents operacions:

- *extractData:* funció que permet extreure les dades d'un fitxer en format '.aris'.
- *parseMasterHeader:* analitza la capçalera principal (*Master Header)* d'un fitxer aris.
- *parseFrameHeader:* analitza el *Frame Header* , actualment només s'analitza la informació necessària per el procés de construcció de mapes.
- *parseImage:* funció per analitzar les imatges cartesianes d'un fitxer en format aris.
- *Map_scan:* funció per computar l'altura de la imatge cartesiana, així com un mapa vectorial per transformar una imatge polar a una cartesiana.
- *normalizeAngle:* funció per normalitzar l'angle.

Extract Aris Data From Bag

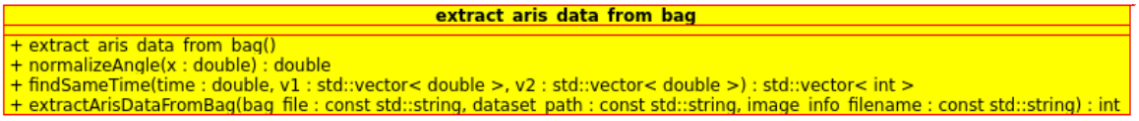


Figura 5.15: Classe Extract_aris_data_from_bag

Si s'ha de treballar amb un fitxer en format '.bag', *MainWindow* cridarà a aquesta classe(fig 5.15) per extreure les dades. Els mètodes que conté són els següents:

- *extractArisDataFromBag:* extreu les dades d'un fitxer en format '.bag'.
- *normalizeAngle:* funció que permet normalitzar l'angle entre –pi i pi.
- *findSameTime:* sincronitza les imatges polars i cartesianes així com la navegació a partir del temps d'adquisició.

5.4.2. Crear les màscares (*Create FLS Mask*)

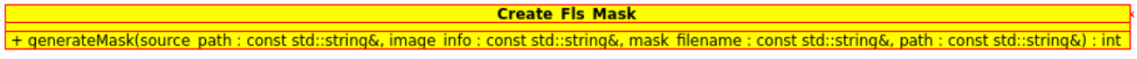


Figura 5.16: Classe Create_Fls_Mask

Tal i com s'ha explicat al capítol 4, és necessari crear dues màscares, una per les imatges cartesianes i l'altre per les imatges polars per tal que les vores de imatges no causin problemes durant el procés de registre. Això es pot fer gràcies a la classe *Create_Fls_Mask*(fig. 5.16) a través del mètode:

- *generateMask:* donat un directori amb imatges, permet crear una màscara per tractar les imatges.

5.4.3. Aplicar les màscares (*Apply Mask*)

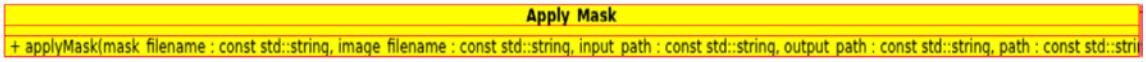


Figura 5.17: Classe Apply_Mask

Un cop creades les màscares, el següent pas és aplicar-les sobre el conjunt d'imatges cartesianes i polars, cada màscara al seu corresponent directori. La classe *Apply_Mask*(fig 5.17) permet aplicar aquestes màscares amb el següent mètode:

- *applyMask:* donat una màscara i un directori que contingui imatges, aplica la màscara al conjunt d'imatges i guarda les imatges resultants en un nou directori.

5.4.4. Extreure l'odometria ROS (*Extract ROS Odometry*)

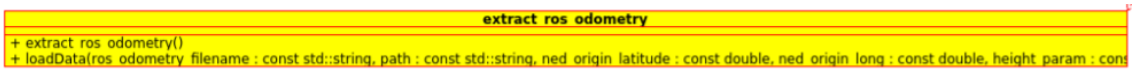


Figura 5.18: Classe extract_ros_odometry

Gràcies al *framework* ROS, es pot extreure la navegació que ha seguit el sonar si el fitxer que s'analitza està en format '.bag'. Aquesta navegació es pot extreure a través de la classe *Extract_ROS_Odometry*(fig. 5.18) amb el mètode:

- *loadData:* llegeix la odometria d'un fitxer ROS i carrega aquestes dades a la taula de la base de dades anomenada *navigation.*

5.4.5. Afegir navegació (*Inject Nav To Image Info*)

Figura 5.19: Classe inject_nav_to_image_info

Quan s'extreuen les dades d'un fitxer en format '.bag', també s'extreu la navegació. La classe *Inject_Nav_To_Image_info*(fig. 5.19) afegeix aquesta navegació a la base de dades. El mètode que permet afegir la navegació és:

- injectNav: afegeix la navegació extreta d'un fitxer en format'.bag' a la taula de la base de dades que conté la informació de les imatges.

5.4.6. Elecció de parelles candidates (*Candidate Selection*)

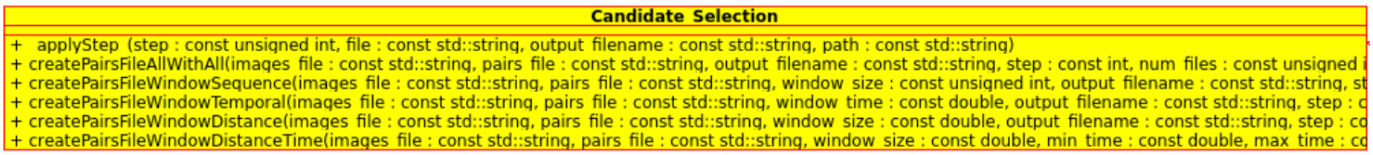


Figura 5.20: Classe Candidate_Selection

Un cop escollit com volem emparellar les imatges a partir dels paràmetres en la interfície, la classe *Candidate_Selection*(fig 5.20) s'encarregar de generar una taula a la base de dades anomenada *pairs* que guarda la relació de les parelles que posteriorment s'intentaran registrar. Per dur a terme aquest procés, la classe consta de les següents operacions:

- *applyStep:* aplica el salt entre imatges a la base de dades.
- *createPairsFileAllWithAll:* emparella cada imatge amb totes les altres.
- *createPairsFileWindowSequence:* emparella cada imatge amb una finestra d'imatges anteriors, amb un màxim definit per l'usuari.
- *createPairsFileWindowTemporal:* emparella cada imatge amb les anteriors, sempre que el temps entre l'anterior i l'actual sigui menor que el valor definit.
- *createPairsFileWindowDistanceTime:* emparella cada imatges amb totes les anteriors, sempre i quan la distància entre elles sigui inferior a un valor definit per l'usuari. Requereix navegació.

5.4.7. Crear els registres (*Estimate Rot Trans i EstimateRotTrans*)

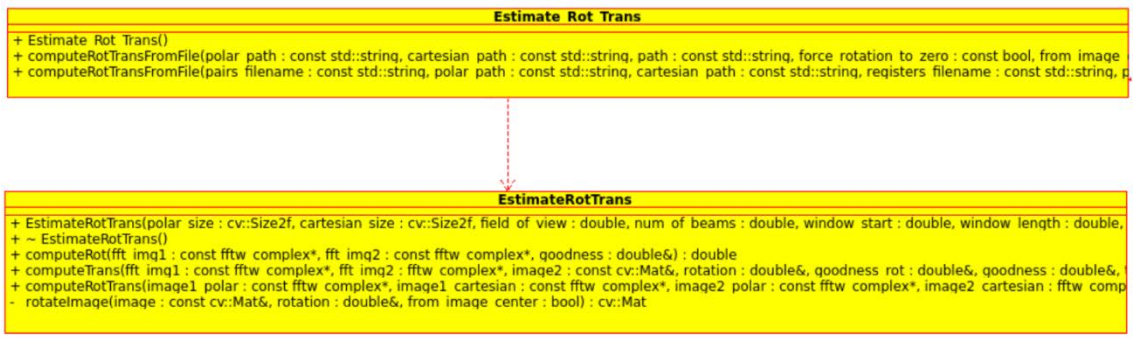


Figura 5.21: Classes Estimate_Rot_Trans i EstimateRotTrans

Un cop establertes les parelles candidates que volem registrar, el següent pas es calcular aquests registres. Hi ha dues classes que formen part del procés de creació de registres: *Estimate_Rot_Trans* i *EstimateRotTrans*(fig. 5.21).

Els mètode principal de la classe *Estimate_Rot_Trans* és:

- *computeRotTransFromFile:* mètode principal que permet calcular els registres i desar-los en una taula (*registers)* a la base de dades.

L'altra classe *EstimateRotTrans* complementa l'anterior amb les següents operacions:

- *computeRot:* Donada la FFT (*Fast Fourier Transform)* de dues imatges, calcula la rotació relativa en radians.
- *computeTrans:* Donada una imatge cartesiana, la FFT d'una altra imatge cartesiana i la rotació entre elles, primer s'aplica la inversa d'aquesta rotació a la imatge i llavors es calcula la translació entre elles.
- *computeRotTrans:* Donada la FFT de dues imatges polars i la FFT de dues imatges cartesianes, es calcula la rotació i la translació per mitjà del *cross power spectrum.*

5.4.8. Fer un registre manual (*Manual Register*)

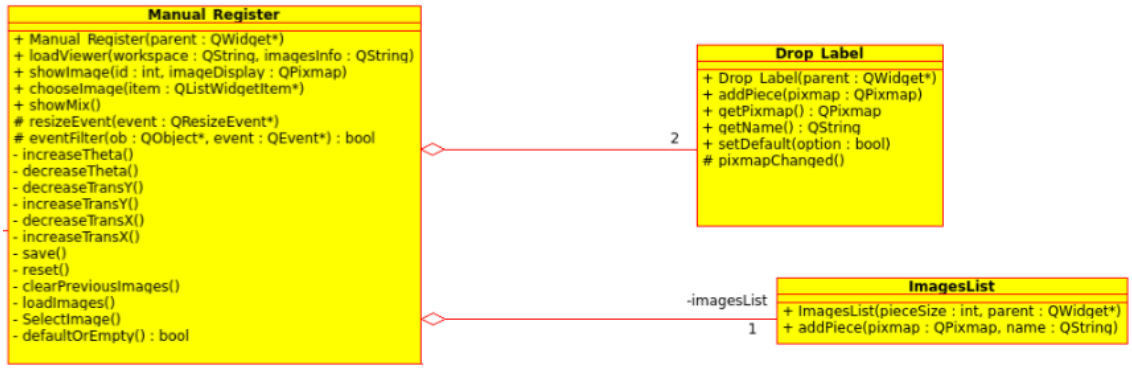


Figura 5.22: Classes Manual_Register, Drop_Label i ImagesList

Es poden generar registres de manera manual, a través de la classe *Manual_Register*(fig. 5.22). Aquesta classe és un *Widget* que a partir de dues imatges seleccionades d'un visor, permet a l'usuari ajustar interactivament la translació i rotació d'una de les imatges per tal de que coincideixi amb l'altra i així generar un registre que s'afegirà a la base de dades dels registres.

Aquesta classe té els següents mètodes:

- *loadViewer:* carrega les imatges del projecte en un visor d'imatges.
- *showImage:* a partir d'un *QPixmap,* mostra de manera més ampliada la imatge seleccionada i la estableix com a actual.
- *chooseImage:* semblant al mètode *showImage,* mostra de manera ampliada la imatge seleccionada, però en aquest cas, a partir d'un *QListWidgetItem, i també l'estableix com a actual.*

- *showMix:* mostra dues imatges superposades, una a sobre de l'altra, amb diferents mapes de colors per tal de visualitzar fàcilment el seu encaix.
- *resizeEvent:* funció heretada que s'ha editat per poder redimensionar les caselles a on es troben les imatges.
- *eventFilter:* funció heretada que s'ha editat per poder rebre comandes des de teclat, es poden anar passant imatges al visor o editant la superposició de les imatges seleccionades.
- *increaseTheta:* augmenta l'angle Theta en la superposició.
- *decreaseTheta:* redueix l'angle Theta en al superposició.
- *increaseTransY:* augmenta el valor de Y en la superposició.
- *decreaseTransY:* redueix el valor de Y en la superposició.
- *increaseTransX:* augmenta el valor de X en la superposició.
- *decreaseTransX:* redueix el valor de X en la superposició.
- *save:* permet guardar el registre generat en la base de dades conjuntament amb els registres generats automàticament.
- *reset:* restableix a 0 les coordenades X i Y i també l'angle Theta.
- *clearPreviousImages:* elimina les imatges del visor les imatges.
- *loadImages:* carrega les noves imatges al visor.
- *SelectImage:* determina quina és la imatge que es té seleccionada.
- *defaultorEmpty:* funció booleana que determina si la casella conté la imatge per defecte o està buida.

Aquesta classe, consta de dues classes més: *Drop_Label* i *ImageList.*

Drop_Label és una classe que hereta de *QLabel* però amb la característica afegida que s'hi poden arrossegar imatges a sobre. *ImageList* és una classe que actua com a visor d'imatges, permetent guardar imatges que es podràn arrossegar amb el ratolí, cap a les posicions on hi hagi un *Widget* de *Drop_Label.*

5.4.9. Renderitzar el mosaic (*Mosaic Rendering*)

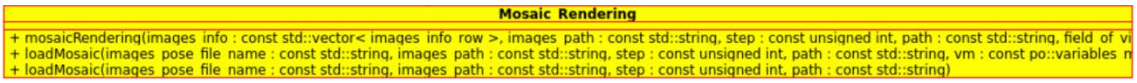


Figura 5.23: Classe Mosaic_Rendering

Una vegada s'ha completat tot el procés de construcció de mapes, l'últim pas per acabar tota la seqüència és renderitzar el mosaic. La classe *Mosaic_Rendering* és l'encarregada de completar aquest procés amb els mètodes:

- *mosaicRendering:* fusiona el conjunt d'imatges acústiques en un únic mosaic, aconseguint una relació senyal-soroll i una resolució millorada respecte les imatges individuals.
- *loadMosaic:* prepara les dades per tal que el mètode *mosaicRendering* es pugui executar correctament.

6. Implementació

En aquest apartat es descriuen les biblioteques externes que s'han utilitzat per a la implementació així com també els detalls i solucions adoptades per implementar les diferents funcionalitats de l'aplicació.

6.1. Biblioteques externes

Tant important és fer un bon codi, com saber quan és possible aprofitar llibreries de tercers. En enginyeria del software és fonamental simplificar i aprofitar el codi per tal de millorar l'aplicació.

A continuació s'explicaran algunes llibreries addicionals que s'han utilitzat durant la implementació tant en el projecte original com en el desenvolupament de la interfície:

1. **OpenCV:** són unes biblioteques que permeten processar imatges amb molts algorismes i operacions de visió per computador com poden ser segmentacions o transformacions. Aquestes biblioteques han sigut una eina essencial en el projecte original degut a la capacitat que tenen de processar les imatges en temps real.
   Les llibreries OpenCV estan desenvolupades sota llicència BSD, fet que permet desenvolupar l'aplicació en total llibertat, fins i tot per ús comercial.

2. **Eigen:** és una biblioteca per a operar de manera algebraica en C++ . Els càlculs de matrius i vectors numèrics s'han implementat amb Eigen. També està desenvolupat sota llicència BSD.

3. **G2O:** és una biblioteca escrita en C++ de codi obert dedicada a la la optimització de funcions no lineals en forma de grafs. G2O també està sota llicència BSD. S'ha utilitzat per poder generar el graf de posicions a partir del conjunt de registres d'imatges i llavors optimitzar-lo.

4. **FFTW3:** és una biblioteca que calcula la transformada de Fourier. Calcular la transformada de Fourier de les imatges és necessari dins el mètode de registre, ja que la translació i la rotació entre les imatges es calcula en el domini freqüencial. FFTW3 està desenvolupat sota llicència GPL.

5. **ROS:** és un *framework* per al desenvolupament de software per robots. ROS és software lliure sota llicència BSD. Donat que l'arquitectura software dels vehicles del CIRS es basa en ROS, moltes vegades el fitxers d'entrada per a fer un mosaic seran fitxers en format ´.bag´ on es guarden tant les imatges com la navegació del vehicle. En aquest cas, s'utilitzaran les funcions de ROS per extreure la navegació.

6. **Poppler:** és una biblioteca de software lliure que permet interactuar amb fitxers pdf. S'ha utilitzat per poder mostrar el graf dels registres, ja que es guarda en un fitxer pdf. Les biblioteques *Poppler* estan desenvolupades sota llicència GPL.

6.2. Detalls d'implementació

En aquesta secció es detallaran els problemes més rellevants en la fase d'implementació i la decisió que s'ha pres per solucionar-los.

6.2.1. Estructura de la GUI

Com s'ha comentat en la secció 3.3.2, es va escollir dissenyar la interfície amb Qt degut a la gran acollida que ha tingut per els usuaris i al potencial que ofereixen les seves llibreries.

La primera decisió que s'ha de prendre quan es vol dissenyar una interfície gràfica és escollir com es vol estructurar per tal de ser el màxim intuïtiva possible i utilitzar eines que facilitin la seva creació.

El disseny de la GUI partia de zero, és per això que es va decidir consultar els exemples de la pàgina web de Qt (Qt, 2016) per tal de fer-se una idea i poder estructurar el codi de la millor manera possible.

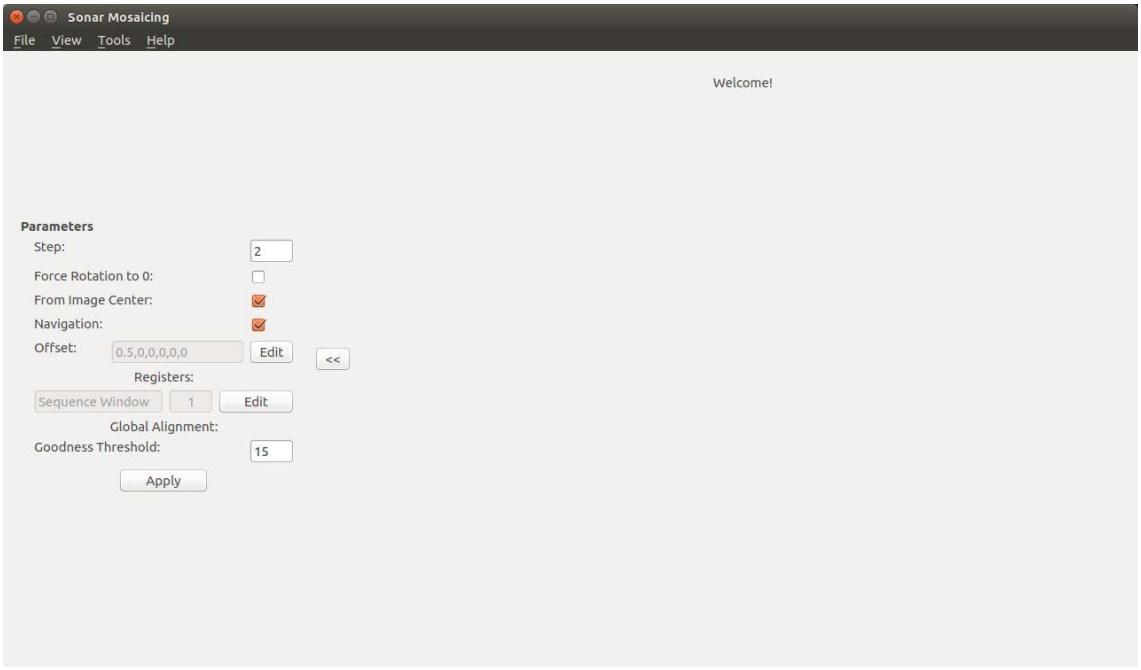


Figura 6.1: Versió final de la GUI

Una decisió que es va prendre va ser simplificar les opcions del menú fins a deixar només els *Parameters.* El que es volia aconseguir amb aquest canvi, era simplificar el menú lateral per tal

que les diferents opcions com: Crear un projecte, Carregar un projecte o Veure la navegació, que són opcions que només requereixen prémer una vegada el botó, estiguessin només en el menú superior, per tal de reduir espai en el *centralWidget.* A més a més s'ha introduït un botó que permet mostrar/amagar aquest menú lateral, donant més espai a les visualitzacions de dades.

Els *Widgets* de Qt permeten crear interfícies completament independents entre elles, però amb la capacitat de poder mostrar-se en el mateix espai o en la mateixa finestra. Aquest pas, és el que es va plantejar a la hora de mostrar les diferents visualitzacions que es poden trobar a l'aplicació. Per fer-ho, es van desenvolupar diferents *Widgets*, un per cada opció de visualització(visor d'imatges,veure la navegació, veure el graf, veure el mosaic resultant...) i es va afegir cada un d'ells en un tipus de dades anomenat *QStackedWidget*. Aquest *Widget* permet agrupar diferents *Widgets* de tota mena en un sol vector. Segons la opció seleccionada, se'n mostrarà un o un altre.

Per tant, la versió final de la interfície (Fig. 6.1) simplifica les opcions que es poden veure a primera vista.

6.2.2. Ús de quadres de diàleg

Moltes aplicacions recorren a l'ús de finestres emergents per ampliar les funcionalitats de la interfície. En el cas d'aquesta GUI es va decidir afegir quadres de diàleg per tal d'interactuar amb l'usuari en moments que requereixen la seva interacció.

Per exemple un dels moments que es requereix de l'ús de diàlegs, és a l'hora de crear un projecte (Fig. 6.2). En aquest diàleg es demanarà la ruta del directori on es vol treballar i el fitxer de dades que es farà servir. Un cop introduït i acceptat, es passarà aquesta informació a la interfície principal, que s'encarregarà de cridar els mètodes d'extracció de dades i de inicialitzar el projecte.

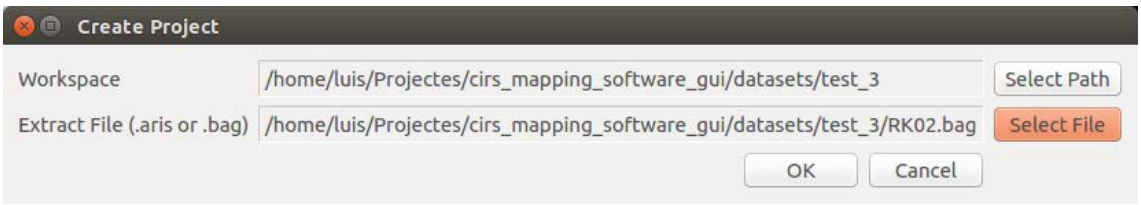


Figura 6.2: Diàleg per crear un projecte

Un altre moment que requereix un diàleg, és a l'hora d'escollir els paràmetres del *Candidate_Selection* (Fig. 6.3). Dins del quadre de diàleg podrem escollir com volem emparellar els registres en el procés de construcció de mapes.

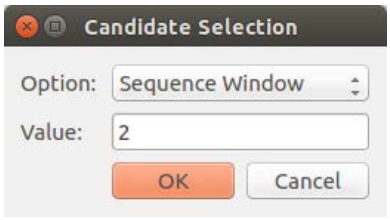


Figura 6.3: Diàleg per seleccionar el candidate

Un altre exemple de diàleg seria el que s'utilitza al guardar un registre generat manualment (Fig. 6.4). Donat el nom de les imatges, la translació i la rotació definits en el *widget* per crear registres manualment, dins del diàleg es poden definir els *offsets (distància i angle entre el centre d'una imatge i l'altra)* i també la covariància de la translació i la rotació.

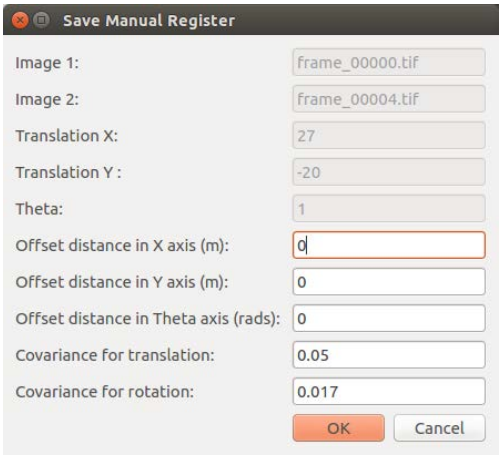


Figura 6.4: Diàleg per guardar un registre manualment

L'ús d'aquestes finestres emergents ens ajuden a l'hora de mantenir la GUI amb els menors elements possibles en pantalla, és a dir, aquests quadres de diàleg apareixen només quan es necessiten i un cop s'han fet servir desapareixen.

6.2.3. Dividir en classes els processos

Com s'ha comentat en l'apartat 1.3, un dels objectius i requeriments és que la interfície convisqui amb les funcions que es criden per línia de comanda. La llibreria original estava dividida en una sèrie de classes cadascuna amb els seus mètodes i el seu propi *main.* Donat que des de la interfície es vol tenir accés a totes aquestes classes, es va haver de fer un canvi en l'estructura. Es van adaptar els mètodes que hi havia a la classe original, perquè poguessin ser accessibles des de la interfície convertint els mètodes en objectes. Per tal de que la llibreria original pogués continuar essent utilitzada, s'ha creat, per a cada procés, un nou fitxer que implemnta l'objecte corresponent (*main).*

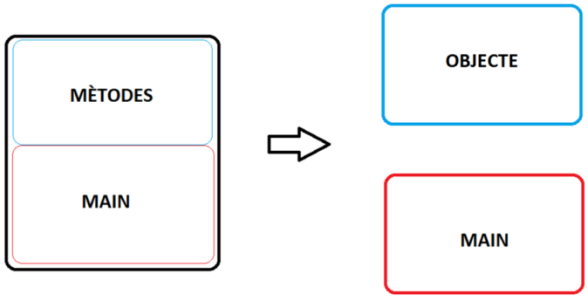


Figura 6.5: Esquema de separació de mètodes-main

6.2.4. Fils d'execució (*Threads)*

Un dels requisits del sistema (secció 4.1) és que la interfície respongui ràpidament a les peticions d'usuari. Degut a que els processos de construcció de mosaics poden necessitar molts recursos, es podria donar el cas que, la GUI deixés de respondre.

Per solucionar aquest fet, s'ha utilitzat una API de Qt, anomenada *QtConcurrent,* que permet fer *multi-threading*.

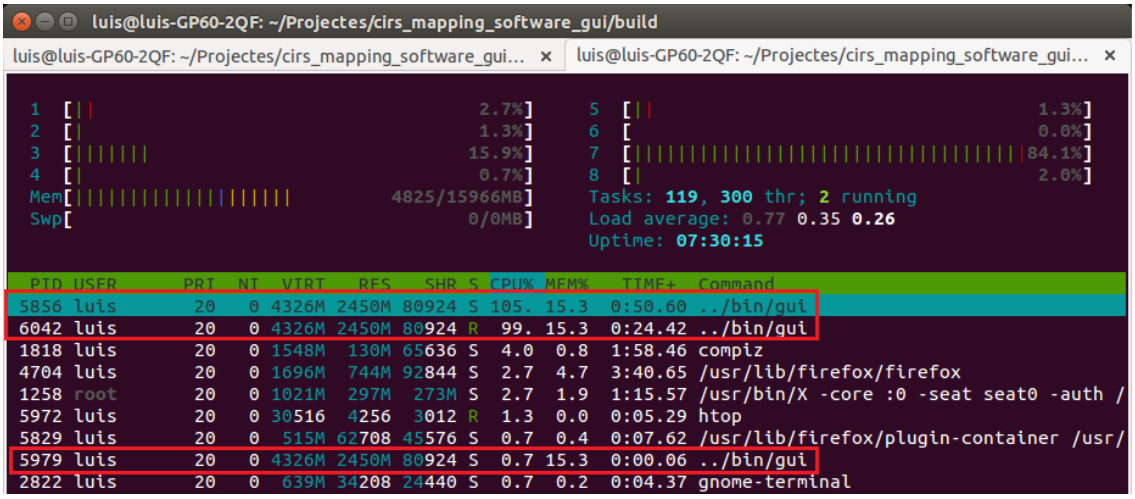


Figura 6.6: Consum de recursos del sistema

D'aquesta manera els processos que es generen en segon pla ajusten el número de *threads* automàticament segons el nombre de nuclis que tingui la màquina i les necessitats del procés, assegurant així que la interfície respon a les peticions d'usuari. La figura 6.6 mostra el consum que l'aplicació fa dels recursos del sistema (en els requadres vermells). Es pot veure com hi ha diferents processos que han estats cridats per la mateixa comanda "../bin/gui", això ens permet identificar que hi ha processos de l'aplicació que estan corrent en segon pla.

A més a més, Qt també disposa d'unes classes anomenades *QFuture* i *QFutureWatcher* que permeten saber quan s'ha acabat el procés. Gràcies a aquestes classes, podem generar unes

barres de càrrega (Fig. 6.7) dins de la interfície per mostrar a l'usuari que hi ha processos en procés de computació.

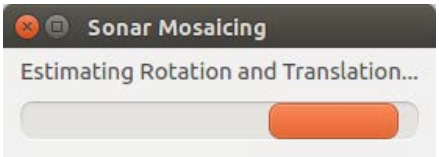


Figura 6.7: Exemple de barra de càrrega

6.2.5. Visor d'imatges

Un dels problemes dels visors d'imatges és la quantitat de recursos que pot requerir si no s'ha s'implementa correctament. És per això, que el visor que s'ha implementat (Fig. 6.8) carrega només una part de les imatges i a mesura que es va avançant per aquestes, va carregant les successives en petites quantitats.

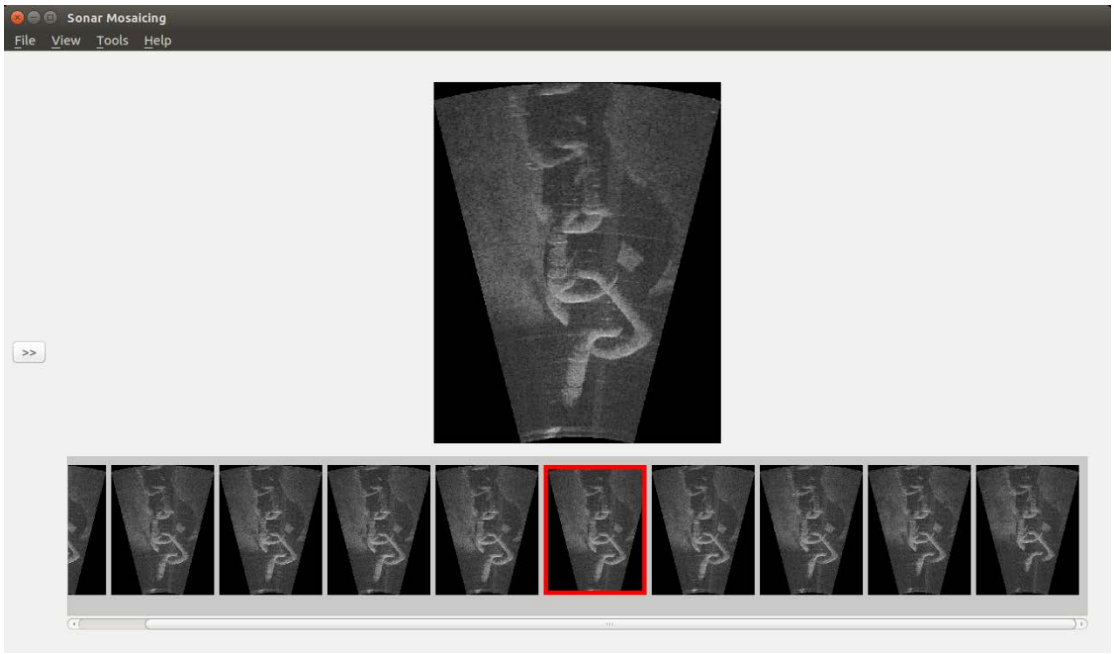


Figura 6.8: Visor d'imatges

La llista on es carreguen les imatges està situada a la part inferior de la finestra, d'aquesta manera quan es selecciona una imatge, aquesta es carrega a la part superior permetent la visualització ampliada de la imatge. S'ha establert un límit de 10 imatges carregades al mateix temps a la llista per tal d'optimitzar els recursos, en cas que s'avanci per la llista i es vulgui avançar o retrocedir més enllà d'aquestes, automàticament s'esborren les actuals i es carreguen les noves.

Per tal de poder seleccionar les imatges des de la llista, s'ha creat una classe anomenada *click_label* que hereta de *QLabel* i implementa uns nous mètodes per tal de que aquest objecte

es pugui clicar. A més a més, s'ha editat la funció heretada *resizeEvent* per tal de poder redimensionar el visor quan ajustem la pantalla.

6.2.6. Visualitzar el graf

Un dels problemes de la visualització del graf era que al ser un arxiu pdf en el moment de fer zoom es perdia la qualitat dels nodes i de les arestes. Per solucionar-ho s'ha fet que cada cop que es fa zoom s'extregui dinàmicament la imatge escalada del pdf utilitzant les llibreries Poppler i es carregui .

Per poder carregar el pdf de manera escalada en el visor (Fig. 6.9), cal extreure'l determinant els DPI (Dots per inch) desitjats. Per tant, per fer *ZoomIn*, multipliquem per 1.2 la mida actual del pdf i per fer *ZoomOut,* multipliquem per 0.8.

Figura 6.9: Exemple de la visualització d'un graf optimitzat. Els nodes són les diferents posicions de les imatges, les arestes indiquen els registres entre elles i el color de les arestes indica la incertesa del registre.

6.2.7. Visualitzar el mosaic resultant

Un cop acabats tots els processos de construcció de mapes i renderitzat el mosaic, el mosaic s'ha de poder visualitzar. En les llibreries originals, el mosaic resultant es mostrava a través d'un fitxer HTML en un navegador, per tal de poder implantar aquest sistema dins de la interfície, tal i com s'ha explicat en l'apartat 5.3.5, s'ha dissenyat un visor dins de la interfície gràcies a les llibreries *QtWebKit,* que permeten visualitzar codi HTML dins d'ella.

Seguint el mateix exemple de la cadena que s'ha vist en imatges anteriors, en la figura 6.10 es pot veure una imatge del mosaic resultant després d'aplicar la fusió de les imatges.

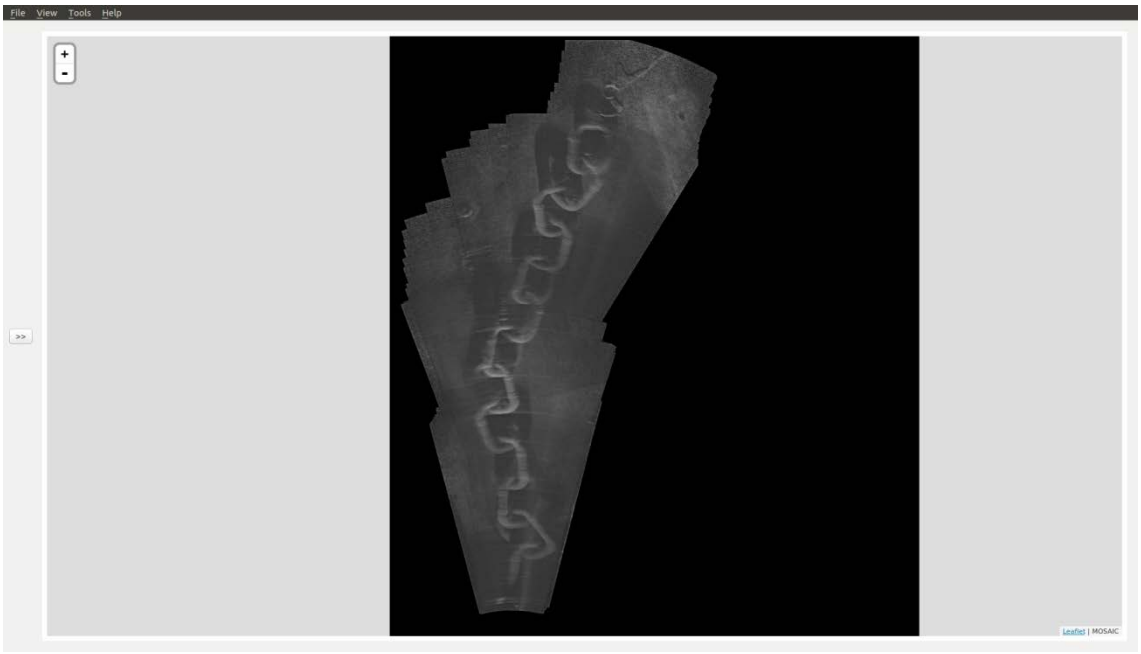


Figura 6.10: Mosaic d'una cadena, resultat d'aplicar tots els processos de construcció de mapes.

6.2.8. Crear registres manualment

Poder crear els registres manualment no és una tasca trivial, tal i com s'ha comentat en el capítol 4. Per poder crear un registre es necessari un parell d'imatges i definir la translació i la rotació, a més a més del llindar per determinar com de bo és el registre en els dos termes.

Es va partir de la idea de tenir una llista amb totes les imatges del projecte i d'aquesta manera escollir les imatges que es volguessin emparellar seguint la mateixa idea del visor d'imatges. En aquest cas però, les imatges seleccionades s'havien de poder arrossegar fins a unes caselles que les permetessin visualitzar d'una manera ampliada per tal de poder crear el registre més fàcilment.

Per tant, s'ha dissenyat una petita classe anomenada *ImagesList* que hereta de *QListWidget,* aquesta classe té la particularitat que permet arrossegar els ítems que conté. S'han modificat les funcions heretades: *dragEnterEvent, dragMoveEvent, dropEvent* i *startDrag* per tal de que

a la llista d'imatges no s'hi pugui afegir cap ítem extern ni els elements es puguin canviar de posició, és a dir, només permetre que l'element seleccionat de la llista s'arrossegui cap a un altre *widget* extern. A més a més, l'element seleccionat també es mostrarà maximitzat en un *widget* que permet mostrar imatges, tal i com es pot veure a la figura 6.11.

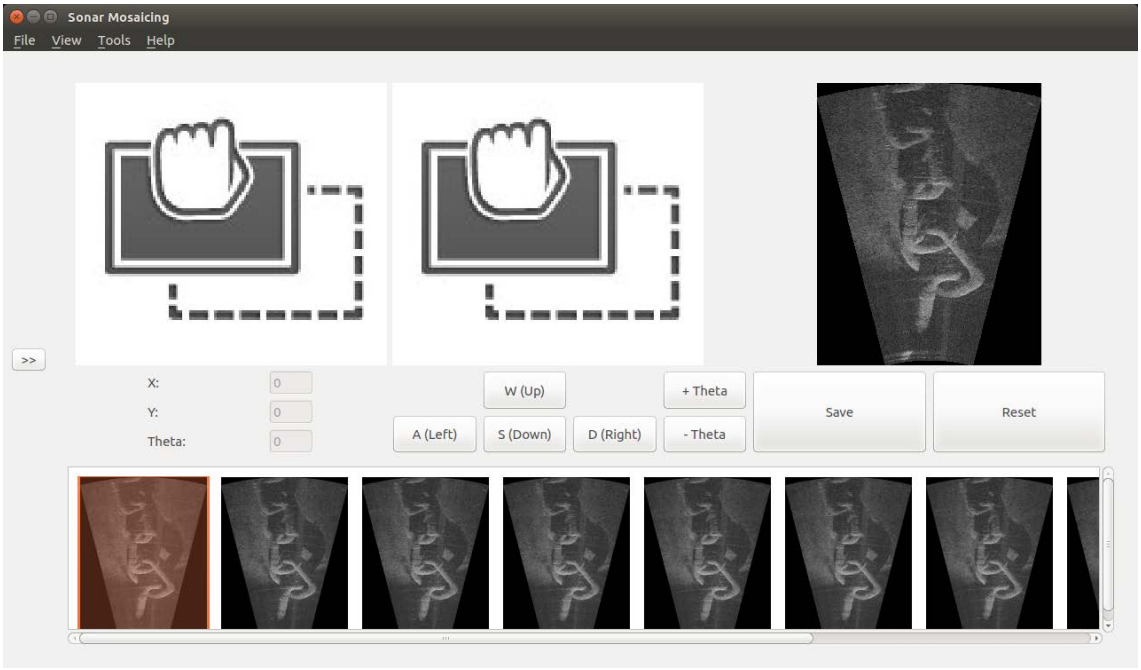


Figura 6.11: Interfície on es crea el registre manual

Un cop creada la llista, es necessitaven dues caselles, on poder deixar anar les imatges seleccionades i d'aquesta manera computar el registre. Per fer-ho s'ha creat una classe anomenada *drop_label* que permet representar les dades que conté la classe *ImageList.* S'han modificat les funciones de *drop_label* heretades de *QLabel*: *dragEnterEvent, dragMoveEvent i dropEvent* fent que només s'acceptessin imatges amb la nomenclatura *image/manual-image,* és a dir, imatges procedents de la classe *ImageList.*

L'algorisme de registre manual s'inicia quan detecta que hi ha una imatge a cada *drop_label*. Per tal d'ajudar l'usuari a realitzar el registre es mostra la superposició de les dues imatges que es vol registrar, utilitzant el mateix *widget* on es mostra la imatge seleccionada de la llista (Figura 6.12).

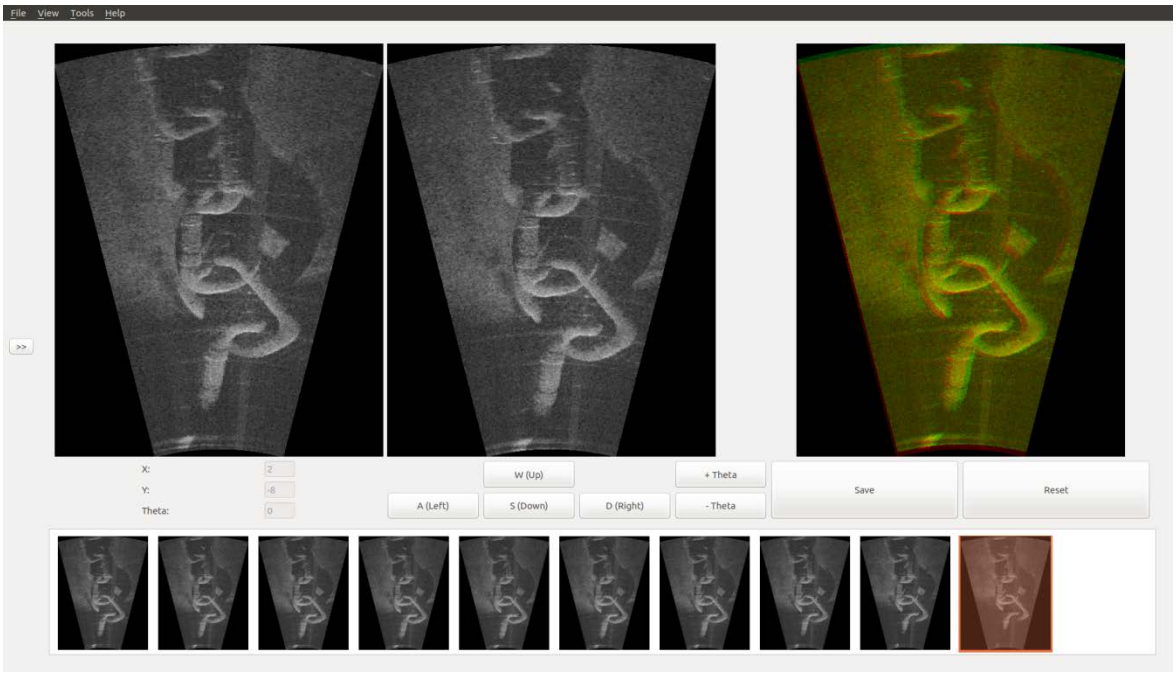


Figura 6.12: Interfície per crear registres manuals amb les imatges superposades

Per poder desplaçar i rotar la segona imatge sobre la primera s'han afegit uns botons que incrementen i disminueixen la translació i la rotació en petits intervals per tal de poder ajustar les dues imatges. També s'ha fet que les tecles: W,A,S,D,+,- controlin els mateixos moviments . Finalment, s'han afegit tres petites caselles que mostren els valors de la translació X i Y i també la rotació en Theta en tot moment.

Un cop l'usuari té les dues imatges ajustades, l'últim pas és agafar aquests valors i afegir el registre corresponent a la base de dades juntament amb tots els altres. Això es fa a través del diàleg de creació manual de registres mencionat a la secció 6.2.2 d'aquest capítol.

7. Implantació, proves i resultats

En aquest capítol es descriu el procés d'implementació al CIRS de la interfície desenvolupada, així com es descriuen els resultats de les diferents proves que s'han fet per avaluar la interfície.

A banda de potencials clients que estiguin interessats en poder generar ells mateixos els mosaics amb les seves dades, els primers a treure rendiment d'aquesta interfície són el personal del CIRS, que puntualment ha de realitzar aquest tipus de mosaics en el context de diferents projectes. En aquest sentit, la implantació de la interfície s'ha dut a terme al CIRS i s'ha testejat amb usuaris d'allà.

Per poder instal·lar i posar en marxa l'aplicació en un sistema UNIX s'ha elaborat un manual d'instal·lació. Aquest manual es pot trobar a l'apèndix A d'aquesta memòria. És important ressaltar que de moment l'aplicació només s'ha testejat en plataformes UNIX ja que és l'entorn utilitzat al CIRS. No obstant, tot i que no s'hi ha dedicat temps en el context d'aquest projecte, la majoria de llibreries utilitzades (Qt, OpenCV, etc) són multiplataforma i en un futur es pot treballar per implantar l'aplicació en altres sistemes operatius.

Per a comprovar que s'ha dissenyat una interfície gràfica per a un programari de construcció de mapes submarins recollint els objectius que s'havien plantejat, s'han anat fent periòdicament un seguit de proves.

La majoria de proves s'han portat a terme sobre un equip amb les següents característiques:

- Intel Core i7
- NVIDIA GTX 950M
- 16Gb de RAM
- Ubuntu 12.04

Però també s'ha provat l'aplicació en equips menys potents per comprovar el rendiment i analitzar que tots els processos funcionessin correctament. L'equip menys potent provat i que permet executar el *software* és:

- Intel Core 2 Quad
- NVIDIA GT610
- 4Gb de RAM
- Ubuntu 12.04

A mesura que s'han anat programant funcionalitats a la interfície, s'han anat provant utilitzant com a dades de test alguns datasets existents al CIRS, tant en format '.bag' com en format '.aris'. Ha estat important comprovar en cada pas que s'ha provat que el resultat que s'obté des de la interfície sigui el mateix que s'obté cridant les comandes manualment. Per tal de fer-se una idea de tot el procediment de creació dels mosaics a partir de la interfície, i també a mode de presentació de resultats, a continuació s'inclou un exemple detallat de l'elaboració d'un mosaic amb tots els passos des de l'inici:

- Un cop instal·lades totes les biblioteques segons el manual de l'Apèndix A, ja es pot obrir la GUI (Fig. 7.1). A primera vista hi ha un menú lateral esquerra on apareix un seguit de paràmetres per ajustar, però al no tenir cap projecte carregat, no es podran editar encara. Per tant, primer necessitem crear un projecte.

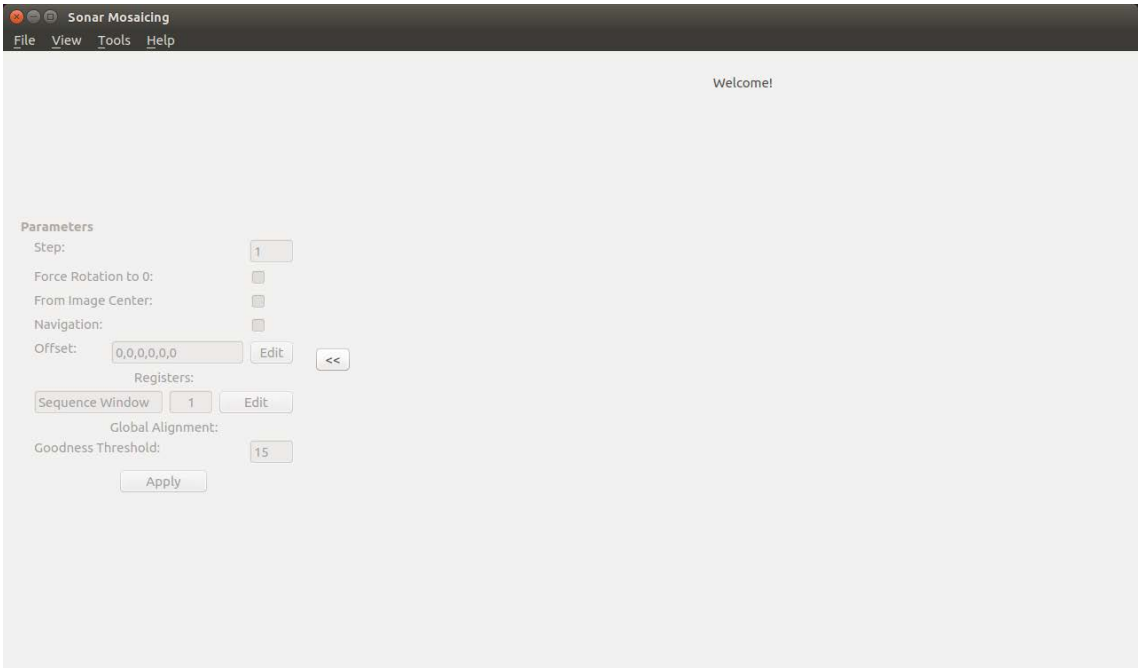


Figura 7.1: Finestra d'inici al obrir l'aplicació

- En el menú superior, trobem diferents submenús on estaran tots els processos, per tant per crear un projecte s'haurà de anar a *File -> Create Project* i se'ns obrirà un Diàleg tal i com s'ha explicat a l'apartat 7.2.2.
- En aquest diàleg s'introduirà un directori de treball per desar les dades i també es seleccionarà un fitxer per treballar. En aquest cas, s'ha seleccionat un fitxer '.bag' que conté les dades d'un experiment realitzat a un vaixell enfonsat a Cap de Vol, situat a la badia del Port de la Selva (Girona). Aquest experiment es va realitzar amb el Girona 500 i el sonar ARIS Explorer 3000 i conté 2720 imatges que es van agafar teleoperant el robot en diferents transsectes.
- Un cop comprovats les rutes del directori i del fitxer, s'obrirà una finestra de càrrega (fig. 7.2) que mostrarà quin procés s'està executant en segon pla.

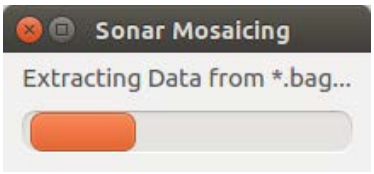


Figura 7.2: Barra de càrrega d'extracció de dades de un fitxer '.bag'

- Primer s'extrauran les dades del fitxer i es desaran al directori que s'hagi seleccionat, tot seguit s'extraurà la navegació que ha seguit el robot gràcies al *framework* ROS i tot seguit s'afegiran al projecte, després es crearan les màscares cartesianes i polars de les imatges i s'aplicaran aquestes màscares sobre les imatges cartesianes i polars

respectivament, creant un directori que contindrà les imatges resultants d'aplicar aquest procés.

- Un cop extretes les dades, se'ns permetrà editar els paràmetres des de la pantalla principal. Per aquest *dataset* amb tantes imatges, configurarem els paràmetres de la següent manera:
    - Posarem el paràmetre *step* a 2, per indicar que només volem treballar amb una imatge de cada dues, ja que el refresc del sonar era molt alt. Per ajudar a determinar aquest paràmetre l'usuari pot anar al menú *View Images* (Fig. 7.3) i moure's per les diferents imatges. Si trobem que el salt de posició entre imatges consecutives fos molt petit potser podem ampliar el paràmetre *step* i treballar encara amb menys imatges.

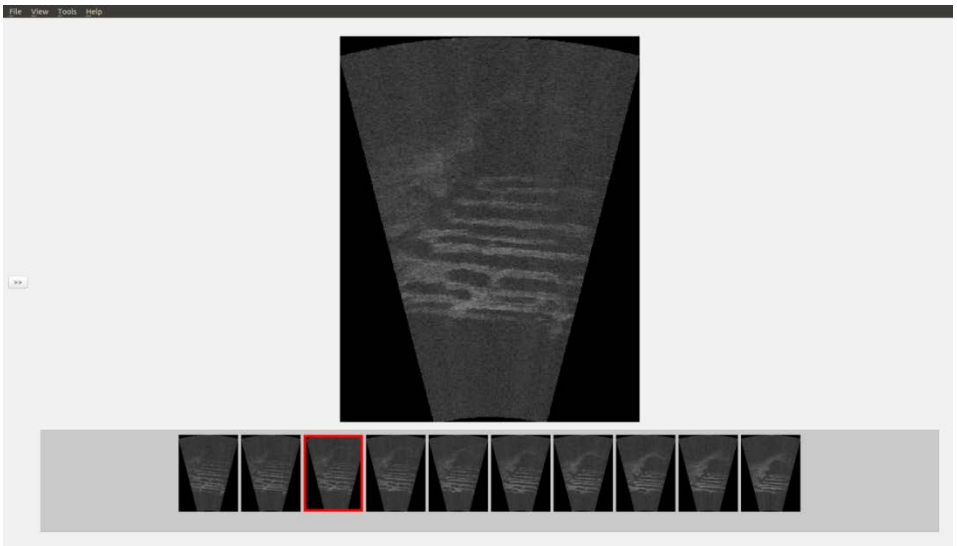


Figura 7.3: Visor d'imatges

    - Donat que sabem (o que veiem a través del visor d'imatges) que la trajectòria no comporta rotacions sinó que les imatges es van agafar en transectes rectes forçarem la rotació a 0 per tal de no acumular petits errors en el càlcul de la rotació.
    - La casella de navegació també l'activarem(en aquest cas com que el fitxer d'entrada és un '.bag' que conté navegació, el sistema l'activa automàticament) per indicar que volem partir de la trajectòria que s'ha fet amb el robot com a posicions per a les imatges.
    - A continuació s'han de triar els paràmetres que determinaran com s'emparellaran els registres (*Candidate Selection).* En aquest cas, com que hi ha navegació, si recorrem a l'opció *View-Navigation* (Fig 7.4), ens ajudarà a determinar la millor estratègia. Com que hi ha diversos transectes la millor opció és intentar establir registres amb l'opció *DistanceWindow (m),* així s'intentarà registrar cada imatge amb les imatges que estiguin a un radi de distància donat i es podran trobar registres entre els diferents transsectes. Com que els transsectes estan separats més o menys 1 metre, posarem el valor a 1.2 metres per assegurar que emparellem imatges de diferents

transsectes Si es tractés d'una trajectòria d'un sol transsecte llavors tindria més sentit utilitzar *Sequence-Window* i intentar registrar imatges només amb un conjunt de les imatges anteriors de la seqüència.

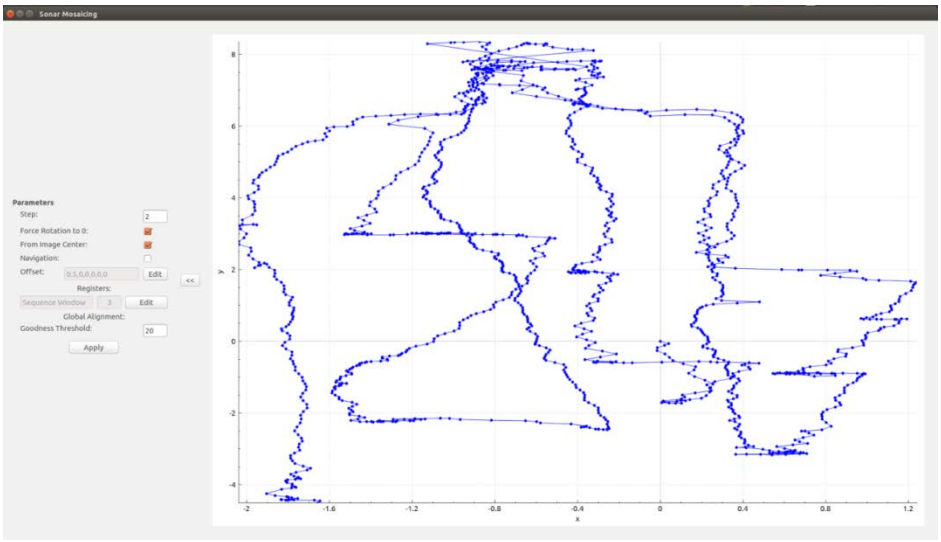


Figura 7.4: Mètode *View Navigation*

- El llindar de confiança per a descartar registres no massa fiables l'establirem a 20, aquest és un valor típic ja que per sota de 20 els registres normalment no són bons.

- Una vegada configurats els paràmetres, procedirem a aplicar-los clicant el botó "Apply" i començarà el procés de computació. Al ser un projecte tan gran, aquest pas ha trigat més de 4 hores per computar-ho tot, ja que al establir un candidat a una distància de 1.2 metres fa que cada imatge s'emparelli amb totes les imatges que es trobin dins d'aquest rang.

- Un cop acabats els processos de construcció de mosaics, l'últim pas es fusionar les imatges i mostrar el mosaic renderitzat resultant. Això es fa a través del menú *View-Mosaic* (Fig 7.5) que s'haurà habilitat quan les computacions hagin finalitzat.

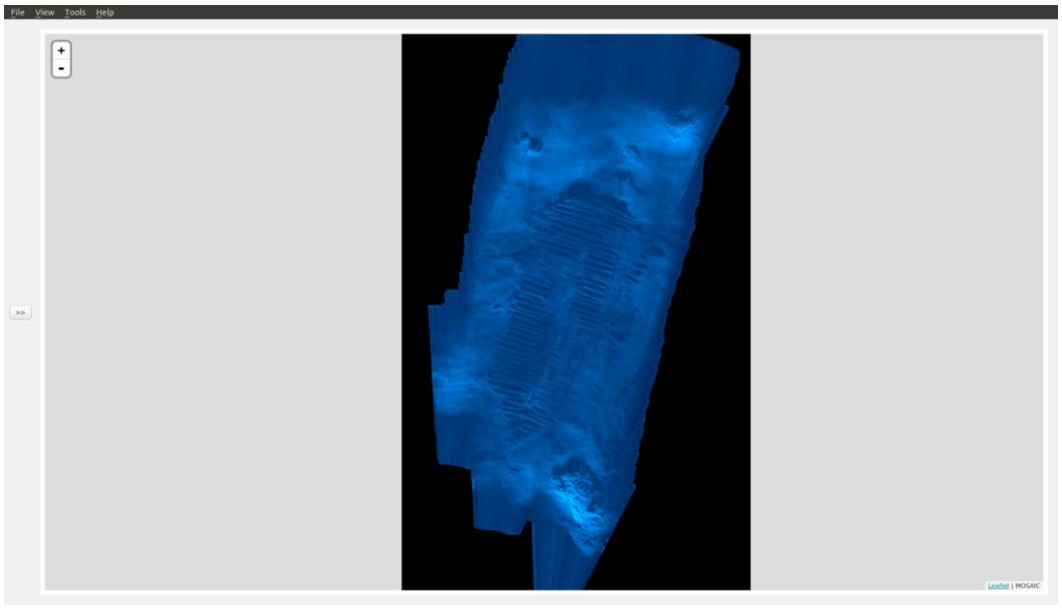


Figura 7.5: Mosaic resultant renderitzat

- Si el mosaic resultant no ens acaba de convèncer (hi ha zones borroses o es veu que en algún lloc no ha encaixat bé), es poden tocar els paràmetres del menú lateral i tornar a recomputar el mosaic. Si hi ha registres que ja s'havien computat el sistema automàticament els obviarà i computarà només aquells que faltin segons els nous paràmetres.

- A banda d'utilitzar més imatges (reduir el paràmetre *Step*) hi ha una sèrie de visualitzacions que ens poden ajudar a ajustar els paràmetres per aconseguir un millor mosaic.

- Podem veure a través del mateix menú *View-Navigation* la trajectòria de navegació però en aquest cas ja estarà corregida segons els registres que s'han computat (les posicions on va cada imatge). Això ens pot ajudar a determinar un nou mètode de *Candidate Selection* o canviar-ne el valor si veiem que els transsectes realment han quedat més separats del que indicava la navegació original.

- Anant a *View-Graph* (Fig. 7.6) podem veure el graf optimitzat que es genera un cop calculat l'alineament global. Això ens ajudarà a veure si hem de moure el llindar d'incertesa si és que veiem que apareixen moltes arestes amb colors que indiquen registres poc fiables i que puguin estar distorsionant el mosaic.

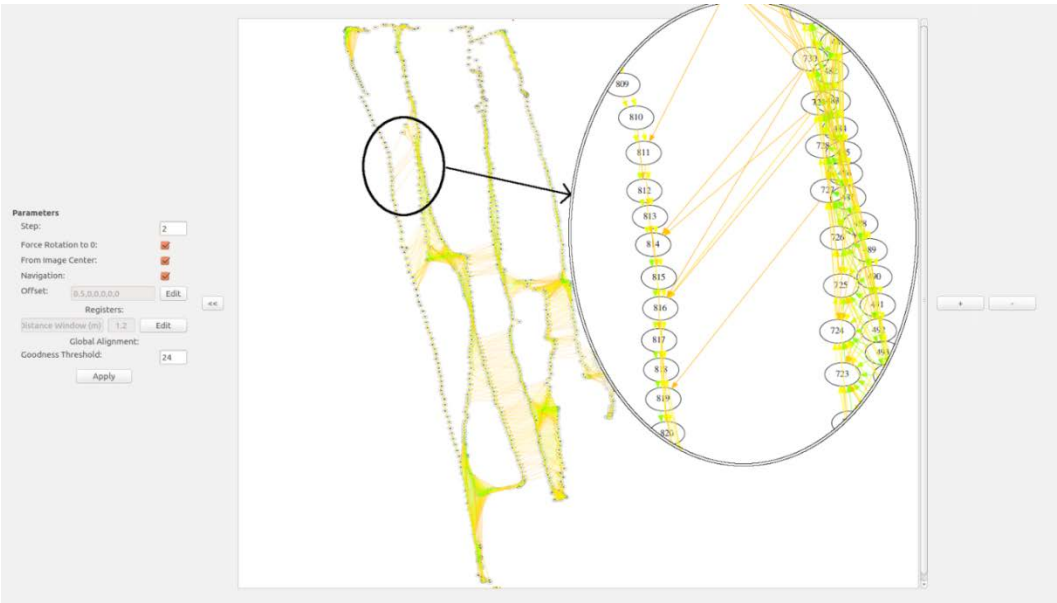


Figura 7.6: Mètode *View Graph*

- Finalment si veiéssim que el mosaic ha quedat dividit en dues o més parts tot i utilitzar el màxim d'imatges disponibles, això indicaria que en el lloc de la divisió per el motiu que sigui, les imatges no s'han pogut registrar correctament. Sota aquestes circumstàncies podem utilitzar l'eina que s'ha implementat per tal d'afegir registres manualment (Fig. 7.7). Per fer-ho, tal i com s'ha explicat en apartats anteriors, es parteix d'una llista on es seleccionaran les imatges que es volen emparellar, un cop seleccionades s'arrossegaran fins a les caselles marcades i d'aquesta manera es començarà el procés de superposició ajudant-se de les tecles o botons per traslladar o rotar una imatge fins aconseguir el millor ajust entre les dues.

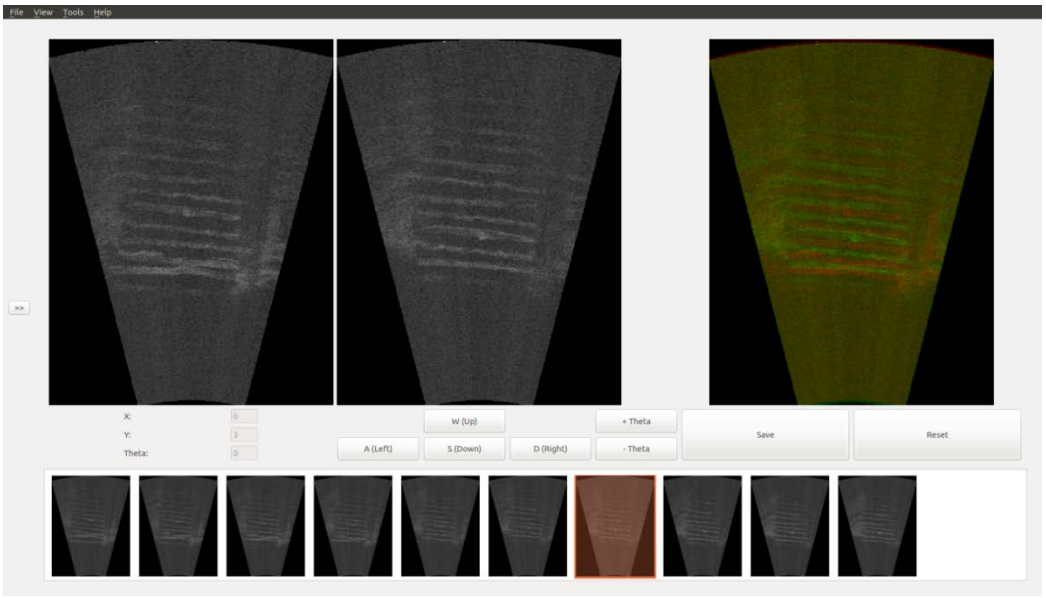


Figura 7.7: Mètode per crear registres manualment

Aquest seria un exemple complert de la creació d'un mosaic. No obstant les diferents parts d'aquest procediment s'han anat testejant per separat durant el període de desenvolupament

de la interfície fins acabar amb provant tot el procediment, amb les diferents opcions i visualitzacions al complert. Aquestes proves s'han fet primer amb els usuaris experts (els que tenen un coneixement profund del sistema de generació de mosaics i saben crear-ne a partir de les llibreries de línia de comandes) i més endavant amb usuaris no experts. Les proves amb els usuaris experts han estat fonamentals per anar adaptant la interfície, decidint quins menús havien de ser accessibles i quins no, quines opcions calia oferir a l'usuari i quines no i com s'havien d'adaptar les diferents visualitzacions perquè fossin útils. Això és important perquè les llibreries per línia de comandes existents permeten moltes opcions i en canvi des de la interfície moltes vegades aquestes opcions es poden ajustar internament depenent del resultat d'una comanda anterior i així automatitzar una mica més tot el procés i que l'usuari no hagi d'executar tantes comandes o determinar tants paràmetres.

Quan s'ha deixat provar la interfície en un usuari no expert, s'han trobat algunes dificultats en el sentit que després de fer un processat (és a dir carregar dades, seleccionar paràmetres i computar el mosaic (que internament computa registres,alineament global i renderitzat), l'usuari no sabia ben bé quins paràmetres canviar (o fins i tot a quines visualitzacions podia recórrer per ajudar-lo a determinar aquests paràmetres) per tornar a recomputar el mosaic i millorar-lo. Això fins a un cert punt és normal degut a que el procés de construcció dels mosaics no és trivial i si no es té un mínim coneixement dels passos interns que són necessaris, es vagi una mica perdut. Tot i que s'ha intentat sempre que l'usuari només pugui fer el que es permet fer en cada moment (deshabilitant opcions en la interfície per exemple), l'ajust dels diferents paràmetres no és una qüestió que es pugui guiar molt, sinó que es tracta de conèixer bé els diferents paràmetres i com afecten la generació del mosaic. En aquest sentit, i de cares a futures millores potser es pot ampliar el manual d'usuari de l'aplicació per explicar més explícitament aquests aspectes.

També és important comentar que durant la realització d'aquest TFG, el personal del CIRS implicat en l'elaboració de mosaics acústics ha atès un total de 4 persones/empreses interessades en el servei de producció de mosaics acústics a partir de dades de FLS que han adquirit. A aquestes persones se'ls hi ha fet una demostració de la capacitat de construir mosaics, generant un mosaic d'una petita part de les dades que han enviat. Aquests mosaics, que s'han tornat als interessats perquè poguessin avaluar la qualitat del resultat final, han estat tots generats a través de la interfície. En un dels casos, a partir del mosaic demostració la empresa ha resultat interessada en el processament complert de totes les dades i el CIRS ha efectuat un servei remunerat de generació del mosaic. Això és un resultat positiu d'aquest TFG, ja que gràcies a la interfície fer aquestes demostracions i petits mosaics s'ha convertit en una feina molt menys tediosa i que pot ser realitzada per altres persones del CIRS si la persona experta no està disponible. Per tant facilita també la possibilitat d'oferir el servei de processament de dades FLS des del laboratori. A més a més, cal afegir que durant el processament d'aquestes dades, agafades amb plataformes diferents als AUVs del CIRS (embarcacions, submarinistes, ROVS), també s'han anat trobant petits errors en el codi i millorant aspectes de la interfície.

8. Conclusions i treballs futurs

En aquest capítol es recullen les conclusions a què s'ha arribat al final del projecte i s'apunten possibles línies de millora de cara a un futur.

8.1. Conclusions

En aquest projecte s'ha aconseguit dissenyar una interfície gràfica que facilita l'ús de les biblioteques de creació de mapes acústics que té actualment el CIRS. Aquesta GUI permet executar els mateixos processos que es cridaven des de la línia de comanda, però ara en un entorn més amigable. A més a més de l'execució de les comandes en sí, també permet visualitzar les dades que intervenen al llarg del procés de manera gràfica, de manera que s'ajuda a l'usuari a entendre millor les seves dades i aconseguir un millor mosaic amb menys temps.

Per determinar l'assoliment del propòsit del projecte, s'analitzaran un a un els objectius que s'havien plantejat a l'inici:

- **Intuïtiva i fàcil d'utilitzar:** s'ha dissenyat una interfície que permet l'ús del programari a persones no expertes en informàtica, però amb coneixements sobre el procés de construcció de mosaics.
- **Robusta i fiable:** les diferents proves i resultats fets al llarg del projecte, han resultat essencials per desenvolupar un programari que respon a les peticions d'usuari sent tolerant a fallades.
- **Extensible:** La solució elaborada permet afegir noves funcionalitats dins de la interfície de manera fàcil, per exemple la visualització de navegació no es contemplava en les llibreries originals i s'ha fet aquesta extensió per poder veure-la dins de la interfície.
- **Modular:** La interfície gràfica complementa el programari original, tant es pot utilitzar les biblioteques a través de la línia de comanda com de la GUI.

Un objectiu paral·lel al desenvolupament de la interfície gràfica era contribuir a la millora de les biblioteques originals, en aquest cas s'ha ajudat a reestructurar el sistema de fitxers amb l'ajuda de les llibreries SQLite per treballar sobre una base de dades, la qual cosa ha beneficiat tota l'aplicació en termes d'eficiència. També s'ha inclòs l'eina gràfica per a la computació de registres manuals, que no existia originalment. És un pas que tot i que no sigui automàtic, a la pràctica pot resultar molt útil i ajudar als usuaris a crear un mosaic consistent i sense talls.

Per concloure, desenvolupar una interfície gràfica sobre un programari de construcció de mapes submarins no ha estat una tasca senzilla. Primer de tot s'ha hagut d'analitzar el projecte inicial i entendre'l per poder dissenyar quelcom que satisfaci les necessitats gràficament. Tenir uns objectius clars i segui un patró com el MVC ha ajudat a estructurar el codi, fent que futures modificacions o noves funcionalitats siguin fàcils d'afegir. Tot i les dificultats, una vegada

arribats a la conclusió d'aquest TFG la valoració és positiva ja que s'ha comprovat que el personal del CIRS ja fa un ús de la interfície de forma habitual i això incentiva a continuar-la millorant de cares a que en un futur es pugui a oferir a clients externs.

8.2. Treballs futurs

La interfície desenvolupada és un primer prototip que ha aconseguit assolir els objectius plantejats. No obstant encara hi ha marge de millora i hi ha molts aspectes que es poden perfeccionar, des de coses simples fins a més complexes.

A més a més, donat que la interfície gràfica és un complement a unes biblioteques existents el fet de desenvolupar noves funcionalitats per aquestes biblioteques també farà que la GUI s'hagi d'anar evolucionant i millorant d'acord amb aquestes.

Així doncs, diferents millores possibles, que es poden implementar de cares a un futur pròxim són:

- **Nous tipus de dades:** per tal d'arribar a més usuaris seria interessant que el programari fos capaç de treballar amb nous tipus de dades provinents d'altres fabricants de FLS.

- **Mostrar percentatge a les barres de càrrega:** en la interfície quan un procés es posa a executar en segon pla, apareix a la interfície una barra de càrrega, però que no determina el tant per cent que porta computat, una millora podria ser mostrar el percentatge en aquesta barra de càrrega per donar a l'usuari una estimació de quan queda per computar.

- **Millorar la visualització del graf:** la solució adoptada parteix de que les llibreries originals generaven el graf en un fitxer *pdf*. Tot i que la visualització actual d'aquest *pdf* dins la interfície és efectiva, es podria crear la visualització del graf d'una manera més interactiva, que permetés seleccionar un node i veure les seves arestes o ensenyar els valors del registre corresponents a una aresta seleccionada. Això podria donar més informació a l'usuari per tal de treure registres dolents o adaptar paràmetres.

- **Millorar el registre manual:** una altra millora seria afegir en la creació de registres manualment l'opció d'afegir punts d'interès sobre les imatges per indicar que el punt d'una imatges és el mateix que el de l'altra o que el punt està a X distància de l'altre, d'aquesta manera, en el cas que trobéssim oclusions, amb aquesta tècnica podríem definir més acuradament el registre per a generar el mosaic resultant.

- **Millora de la documentació per a l'usuari:** tot i que s'ha fet un esforç per documentar les funcionalitats de l'aplicació a través d'un manual d'usuari

disponible a partir del menú *Help*, potser caldria documentar millor el procediment a seguir per obtenir un bon mosaic (quines visualitzacions mirar en cada moment, com adaptar els paràmetres en diferents casos, afegir exemples, etc), per tal de guiar l'usuari que encara no té molta experiència.

- **Multiplataforma:** una millora pendent és migrar l'aplicació cap a altres sistemes operatius. Totes les llibreries que s'han utilitzat són multiplataforma, per tant només s'haurien de fer algunes modificacions per tal d'efectuar la migració.

9. Bibliografia

Amat, J., Monferrer, A., Batlle, J., & Cufí, X. (1999). Garbi: a low-cost underwater vehicle. *Microprocessors and Microsystems* , 23(2):61-67.

Batlle, J., Ridao, P., Garcia, R., Carreras, M., Cufi, X., El-Fakdi, A., et al. (2005). Uris: Underwater robotic intelligent system. *Automation for the Maritime Industries* , 177-203.

C++. (2016). *A Brief Description - C++ Information*. Consultat el febrer / 2016, a http://www.cplusplus.com/info/description/

Cabrero, G. A., & Maldonado, D. (2007). *Sqlite: Rápido, ágil, liviano y robusto*. Consultat el març / 2016, a https://es.scribd.com/doc/52882068/SQLite

CakePHP. (2016). *Entendiendo el Modelo - Vista - Controlador*. Consultat el febrer / 2016, a http://book.cakephp.org/2.0/es/cakephp-overview/understanding-model-view-controller.html

Carreras, M., Batlle, J., & Ridao, P. (2001). Hybrid coordination of reinforcement learning-based behaviors for auv control. *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems* , 1410-1415.

Carreras, M., Candela, C., Ribas, D., Mallios, A., Magí, L., Vidal, E., et al. (2013). Sparus II, design of a lightweight hovering auv. *International Workshop on Marine Technology (Martech). SARTI* .

CIRS. (2016). *CIRS - Underwater Vision and robotics*. Consultat el Març / 2016, a http://cirs.udg.edu

Copley, J. (9 / Octubre / 2014). *The Conversation*. Consultat l' agost / 2016, a https://theconversation.com/just-how-little-do-we-know-about-the-ocean-floor-32751

El-Fakdi, A., & Carreras, M. (2013). Two-step gradient-based reinforcement learning for underwater robotics behavior learning. *Robotics and Autonomous systems* , 61(3):271-282.

Elibol, A., Garcia, R., & Gracias, N. (2011). A new global alignment approach for underwater optical mapping. *Ocean Engineering* , 1207-1219.

Ferrer, J., Elibol, A., Delaunoy, O., Gracias, N., & Garcia, R. (2007). Large-area photo-mosaics using global alignment and navigation data. *MTS/IEEE OCEANS Conference, Vancouver, Canada* , 1-9.

Galceran, E., Djapic, V., Carreras, M., & Williams, D. (2012). A real-time underwater object detection algorithm for multi-beam forward looking sonar. *IFAC's workshop on Navigation, Guidance and Control of Underwater Vehicles (NGCUV)* .

Hernández, E., Carreras, M., Antich, J., Ridao, P., & Ortiz, A. (2011). A topologically guided path planner for an auv using homotopy classes. *IEEE International Conference on Robotics and Automation(ICRA)* , 2337-2343.

Hurtos, N. (2014). Forward-Looking sonar mosaicing for underwater environments. Universitat de Girona.

Mallios, A., & al., e. (2014). Sonar scan matching for simultaneous localization and mapping in confined underwater environments. *PhD thesis, Universitat de Girona* .

N, G., Ridao, P., Garcia, R., Escartin, J., L'Hour, M., Cibecchini, F., et al. (2013). Mapping the moon: Using a lightweight auv to survey the site of the 17th century ship 'la lune'. *MTS/IEEE OCEANS-Bergen* , 1-8.

Nicosevici, T., Gracias, N., Negahdaripour, S., & Garcia, R. (2009). Efficient three-dimensional scene modeling and mosaicing. *Journal of Field Robotics* , 26(10):759-788.

Palomeras, N., Carreras, M., Ridao, P., & Hernandez, E. (2006). Mission control system for dam inspection with an auv. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* , 2551-2556.

Palomeras, N., El-Fakdi, A., Carreras, M., & Ridao, P. (2012). Cola2: A control architecture for auvs. *IEEE Journal of Oceanic Engineering* , 37(4):695-716.

Prados, R., Garcia, R., & Neumann, L. (2014). Image blending techniques and their application in underwater mosaicing. *Springer* .

Prats, M., Garcia, J., Wirth, S., Ribas, D., Sanz, P., Ridao, P., et al. (2012a). Multipurpose autonomous underwater intervention: A system integration perspective. *20th Mediterranean conference on Control & Automation (MED)* , 1379-1384.

Prats, M., Ribas, D., Palomeras, N., García, J., Nannen, V., Wirth, S., et al. (2012b). Reconfigurable auv for intervention missions: a cause study on underwater object recovery. *Intelligent Service Robotics* , 5(1):19-31.

Qt. (2016). *Qt Documentation*. Consultat el febrer / 2016, a http://doc.qt.io/

Ribas, D., Palomeras, N., Ridao, P., Carreras, M., & Hernandez, E. (2007). Ictineu auv wins the first sauc-e competition. *IEEE International Conference on Robotics and Automation (ICRA)* , 151-156.

Ribas, D., Palomeras, N., Ridao, P., Carreras, M., & Mallios, A. (2012). Girona 500 AUV: From Survey to Intervention. *IEEE/ASME Transactions on Mechatronics* , 17(1):46-53.

Ribas, D., Ridao, P., Tardós, J., & Neira, J. (2008). Underwater SLAM in man made structured environments. *Journal of Field Robotics* , 25(11-12):898-921.

Ridao, P., Batlle, J., & Carreras, M. (2002). O^2CA^2 a new object oriented control architecture for autonomy: the reactive layer. *Control Engineering Practice* , 10(8):857-876.

Ridao, P., Carreras, M., Ribas, D., & Garcia, R. (2010). Visual inspection of hydroelectric dams using an autonomous underwater vehicle. *Journal of Field Robotics* , 27(6):756-778.

ROS. (2016). *ROS.org | Powerting the world's robots*. Consultat l'agost / 2016, a http://www.ros.org/

Shihavuddin, A., Gracias, N., Garcia, R., Gleason, A., & Gintert, B. (2013). Image-based coral reef classification and thematic mapping. *Remote Sensing* , 1809-1841.

Soundmetrics. (2016). *Sound Metrics*. Consultat l'agost / 2016, a http://www.soundmetrics.com/

APÈNDIX A: Manual d'instal·lació

En aquest capítol es pretén explicar com fer la posta en marxa de l'aplicació instal·lant les biblioteques necessàries per al seu funcionament. La instal·lació s'ha realitzat en un entorn UNIX, per tant aquest manual seguirà els passos per instal·lar-ho en aquesta plataforma.

Les diferents biblioteques utilitzades són multiplataforma, tot i que no s'ha desenvolupat en un altre entorn, com pot ser Windows, les llibreries permetrien la migració a un altre sistema.

Si no es té l'eina de control de versions *git*, es descarregarà amb la comanda:

- sudo apt-get install git

S'ha de descarregar el projecte en el sistema, per fer-ho escriurem la següent comanda i clonarem el repositori des de *Bitbucket*:

- git clone git@bitbucket.org:u1922744/cirs_mapping_software_gui.git

Un cop descarregat es procedirà a descarregar les llibreries externes. Primer de tot instal·larem *cmake* per poder compilar el projecte:

- sudo apt-get install cmake

Instal·larem les llibreries QT:

- sudo apt-get install libqt4-dev
- sudo apt-get install qt5-default
- sudo apt-get install qt5-image-formats-plugins
- sudo apt-get install libqglviewer-dev

També s'instal·larà Fftw3:

- sudo apt-get install libfftw3-dev

Suite Sparse:

- sudo apt-get install libsuitesparse-dev

Per instal·lar g2o s'haurà de descarregar el projecte des de github amb la comanda:

- git clone https://github.com/RainerKuemmerle/g2o.git

I llavors s'haurà de compilar el projecte g2o:

- cd g2o
- mkdir build
- cd build
- cmake ../
- make
- sudo make install

Per instal·lar les llibreries OpenCV, primer s'haurà de fer:

- sudo apt-get install build-essential
- sudo apt-get install libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev libswscale-dev

Un cop instal·lades aquestes llibreries, es descarregarà el projecte des de github:

- git clone https://github.com/Itseez/opencv.git

I es compilarà el projecte OpenCV, semblant a com s'ha fet amb el de g2o:

- cd opencv
- mkdir release
- cd release
- cmake -D CMAKE_BUILD_TYPE=RELEASE –D CMAKE_INSTALL_PREFIX=/usr/local ..
- make
- sudo make install

També instal·larem el *framework* ROS seguint el manual que es troba a la seva pàgina web oficial:

- http://wiki.ros.org/jade/Installation/Ubuntu

Per poder dividir el mosaic resultant en sub-imatges, es faran servir les llibreries gdal2tiles:

- sudo apt-get install python-gdal

Per poder treballar amb fitxers pdf dins de Qt, s'utilitzen les llibreries Poppler:

- sudo apt-get install libpoppler-qt5-dev

Un cop instal·lades totes les llibreries externes, el següent pas es compilar el projecte. Per fer-ho, ens dirigirem al repositori on es trobi i un cop a dins, farem les següents comandes:

- mkdir build
- cd build
- cmake ../
- make

A la carpeta bin/ del projecte ja tindrem la GUI a punt per ser executada.