UNIVERSITAT DE BARCELONA

Treball final de grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

GAT: Geological Analysis Tool

Autor: Iván Sevilla Ramos

Directora: Dra. Anna Puig Puig
Realitzat a: Departament de Matemàtiques i Informàtica

Barcelona, 26 de juny de 2019

Abstract

This project arises to help researchers of the Faculty of Earth Sciences to analyze geographical accidents, making an effective link from the photometric images to 3D models, both obtained by photometric devices, such as LIDAR (Laser Imaging Detection and Ranging). The 3D input data is a set of a large number of points so that its treatment tends to be slow and requires good memory management.

Therefore, in this project, we have carried out the analysis, design, implementation, and validation of an application that allows the management of projects consisting of a series of images, the calibration data of the cameras, and a point cloud (3D model). This application helps the user to digitalize 2D points based on the images, taking into account their projection on the 3D model. We focus on the usability of the program, its accuracy, and its robustness concerning the storage of the data and in the operations of digitization. Thanks to this app, researchers could analyze the fractures of geological formations in different locations as well as they could predict future landslides.

Resum

Aquest projecte sorgeix de la necessitat dels investigadors de la Facultat de Ciències de la Terra de disposar d'una eina per a facilitar l'anàlisi d'accidents geogràfics, realitzant un lligam efectiu entre les dades fotomètriques i models 3D obtinguts per dispositius de captació del terreny, com pot ser la tecnologia LIDAR (Laser Imaging Detection and Ranging). Les dades 3D estan formades per una gran quantitat de punts i el seu tractament acostuma a ser lent, requerint una bona gestió de memòria.

Així doncs, en aquest projecte s'ha realitzat l'anàlisi, disseny, implementació i validació d'una aplicació que permet la gestió dels projectes formats per un conjunt d'imatges, les dades de calibració de les càmeres i un point cloud (model 3D), per tal de realitzar digitalizacions de punts basades en les imatges tenint en compte la seva projecció en el model 3D. S'ha donat especial importància a la usabilitat del programa, la seva precisió i la robustesa en l'emmagatzemament de les dades i en les operacions de digitalització. Gràcies a aquesta aplicació, es facilitarà predir esllavissades en terrenys i l'anàlisi de fractures geològiques en diferents localitzacions.

Agraïments

Vull agrair a David Garcia i a Laura Blanco per haver confiat en mi i donar-me la oportunitat de fer aquest treball.

També vull agrair a la meva parella Maite Mateo per donar-me ànims i ajudar-me alhora de millorar la redacció d'aquest document.

Finalment vull agrair a la meva tutora, la doctora Anna Puig, pel seu inestimable ajut alhora de buscar sol·lucions i alhora de dissenyar, tant l'aplicació com aquest mateix document.

Índex

1 Introducció

1.1 Àmbit i motivació del projecte

El projecte s'emmarca en l'àmbit de la geologia. La finalitat principal es facilitar l'anàlisi de les dades obtingudes al camp (com són fotos i núvols de punts - o *point clouds* -) per apoder detectar els plegaments i fractures dels terrenys i així poder predir esllavissades o tenir coneixement del comportament intern de les formacions geològiques.

Aquest projecte va sorgir d'una petició per part d'un investigador de la Facultat de Ciències de la Terra per a disposar d'una aplicació que facilités l'anàlisi d'accidents geogràfics, realitzant un lligam efectiu entre les dades fotomètriques i models 3D obtinguts per dispositius de captació del terreny. Aquestes dades estan formades per una gran quantitat de punts i el seu tractament acostuma a ser lent, requerint una bona gestió de memòria.

Es parteix d'un primer programa, Geo150 [1], realitzat per l'investigador en Visual Basic que, tot i que permetia funcionalitats molt interessants i innovadores dins de l'àrea de recerca de la geologia, no era prou robust i necessitava d'un hardware d'altes prestacions per a poder operar amb les dades.

Així, en el present projecte es pretén millorar el programa anterior, realitzant l'anàlisi, disseny i desenvolupament d'una aplicació que permeti digitalitzar dades fotomètriques i lligar-les amb el model de *point cloud* de forma fiable i interactiva en una interfície gràfica usable per geòlegs.

Així, la motivació darrera d'aquest projecte sorgeix d'oferir una eina simple, però potent, per a facilitar la fenia del anàlisi de falles i fractures vistes o trobades als accidents geogràfics. Es busca oferir una interfície neta i clara que permeti a l'usuari trobar el que necessita i dotar-lo de la informació necessària per a agilitzar el seu treball, permeten desfer operacions incorrectes o refer alguns punts de la seva digitalització.

Durant aquest projecte, es fa ús de l'anàlisi de disseny guiat per l'usuari, l'ús de metodologies AGILE, com per exemple, l'ús del disseny iteratiu o *continuos integration*. També, en interactuar amb investigadors de Ciències de la Terra, ha calgut realitzar una presa de requeriments mitjançant l'anàlisi de casos d'ús. finalment, ha estat necessari utilitzar eines de gestió de memòria i gestió gràfica per poder mantenir visualitzacions interactives i robustes dels diferents terrenys.

1.2 Objectius generals

L'objectiu del projecte consisteix en l'anàlisi, disseny i desenvolupament d'una aplicació que permeti agilitzar l'anàlisi de dades i fer un lligam entre les dades fotomètriques i els models 3D generats pels investigadors de la Facultat de Ciències de la Terra.

1.3 Objectius específics

L'objectiu general es desglossa en objectius més concrets que es llisten a continuació:

- Anàlisi del problema de digitalització de dades geològiques: entendre el problema dels geòlegs, en quines situacions utilitzaran l'aplicació i realitzar la presa de requeriments.
- Anàlisi de plataformes existents i les seves prestacions: aquí es planteja realitzar un estudi de les principals aplicacions utilitzades pels geòlegs en aquest tipus de problema i analitzar-les en base als requeriments obtinguts en el primer subobjectiu.
- Disseny i implementació d'un editor 2D de digitalització de punts sobre les imatges: en aquest objectiu es planteja dissenyar un editor i digitalitzador de dades, que sigui eficaç i robust, assegurant la seva portabilitat i la seva utilitat.
- Disseny i implementació de la gestió de projectes geològics: amb aquest objectiu s'aconseguirà el tractament general de projectes geològics composats per imatges i *point clouds*, permetent guardar, carregar i modificar, de forma fàcil i efectiva tant les dades capturades en el camp, com les generades pel programa.
- Disseny de eines de interacció: Estereoplot. En aquest objectiu es planteja el disseny d'una eina de representació 3D utilitzada àmpliament en geologia, que mostra el lligam entre les dades 2D i les dades 3D. A més caldrà que la solució plantejada sigui robusta i eficient.
- Validació de l'aplicació desenvolupada: es realitzarà una anàlisi d'usuari en la utilització de l'aplicació, per a validar si la solució plantejada compleix el requeriments que els usuaris demanaven en la presa de requeriments.

1.4 Planificació temporal

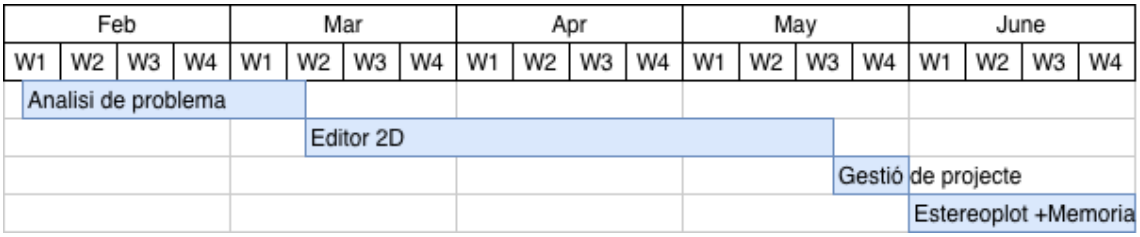


Figura 1: Diagrama de Gantt duració Iteracions

Aquest projecte es va dividir en quatre iteracions de desenvolupament.

En la primera iteració, es va planejar analitzar el problema i els antecedents dels què es disposava, realitzar el codi de la versió primigènia i altres eines que feien servir. Per a aquesta iteració es van planificar 30 dies, que van acabar sent uns 35 dies.

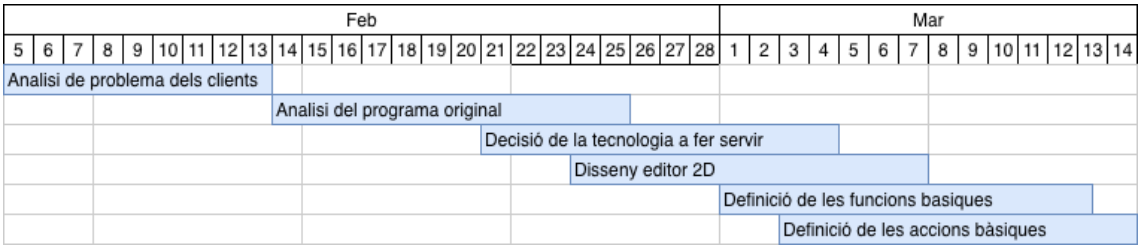


Figura 2: Diagrama de Gantt Primera iteració

En la segona iteració, es va planejar desenvolupar una base d'edició 2D, que permetés la digitalització d'una imatge, i dissenyar les estructures internes del programa, tant objectes gràfics com objectes de gestió d'events. Per a aquesta iteració es van planificar 2 mesos, que van acabar sent uns 65 dies.

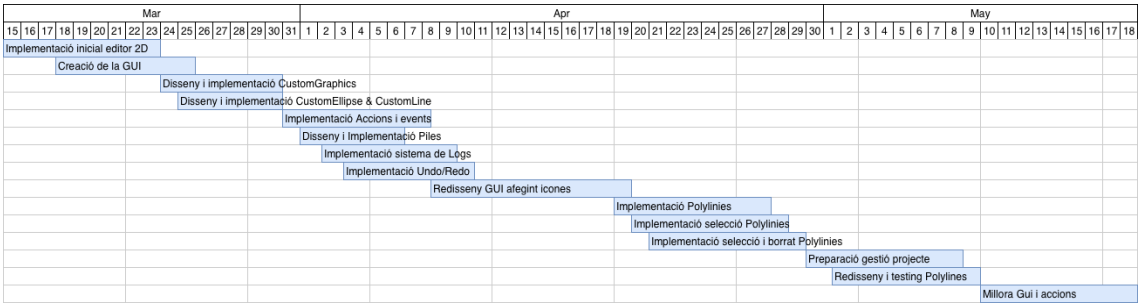


Figura 3: Diagrama de Gantt Segona iteració

En la tercera iteració, es va plantejar desenvolupar la part de gestió de projectes, el disseny d'aquests i millorar la interfície gràfica. Per a aquesta iteració es van planificar 15 dies, que van acabar sent uns 12 dies.

En la darrera etapa, es va plantejar fer el lligam del 2D amb el 3D, el desenvolupament de l'objecte estereoplot, la millora final de la interfície gràfica i la generació de la documentació escaient per al projecte. Per a aquesta iteració es van planificar 30 dies, que van acabar sent uns 26 dies.

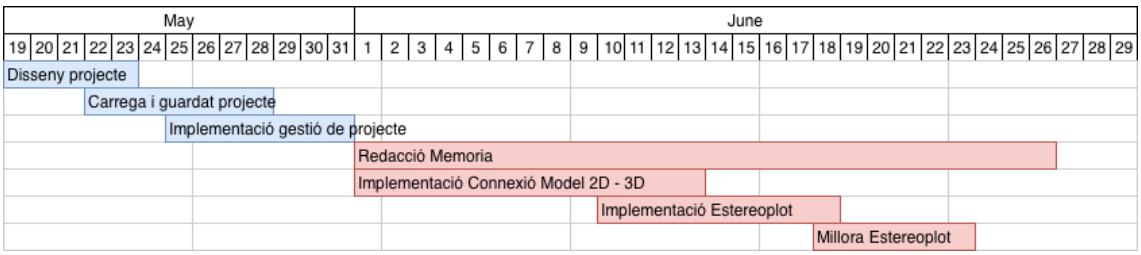


Figura 4: Diagrama de Gantt Tercera i darrera iteració

1.5 Organització de la memòria

En aquesta secció es defineix el contingut dels capítols de la memòria. La memòria està formada, a part del capítol d'introducció, per altres quatre capítols i un annex.

El capítol 2 descriu l'anàlisi del problema, de les eines i l'estudi de plataformes similars. A més, es defineixen els requeriments tant funcionals com tecnològics i el perquè de la utilització de les tecnologies escollides.

El capítol 3 explica el disseny realitzat, mostrant en detall informació sobre la interfície gràfica, el seu disseny i la seva evolució durant les etapes de desenvolupament. A més inclou una comparativa amb el disseny de l'antic programa usat pels geòlegs. Aquest capítol també detalla l'arquitectura del sistema emprada i del disseny dels elements més característics de l'aplicació, tals com l'ús d'una *pila* de comandes i l'*escena* utilitzada en la interacció i els seus elements.

El capítol 4 detalla la implementació de l'aplicació, els resultats i les simulacions fetes amb els usuaris, tals com el test d'usabilitat i/o la validació per part dels usuaris de la feina feta.

Per a finalitzar, en el capítol 5 es recullen les conclusions i les millores que es poden realitzar en l'aplicació.

En l'annex es detallen els requeriments d'instal·lació i d'ús de l'aplicació.

2 Anàlisi

2.1 Anàlisi del problema

Els clients, investigadors de la Facultad de Ciències de la Terra de la Universitat de Barcelona, analitzen diferents un llocs o accidents geogràfics. Per a realitzar aquest anàlisi, els geòlegs van físicament al lloc geogràfic a estudiar (per exemple, Montserrat) emportant-se diferents aparells de mesura. Aquests aparells permeten recol·lectar dades del lloc, com localitzacions, fotografies, núvols de punts, etc.. per analitzar-les a posteriori. Per a obtenir aquestes dades, si el accident a analitzar és molt gran, es pot utilitzar un dron que té una càmera d'alta definició i un sistema LiDAR (figura 5).

Figura 5: Imatge d'un Dron equipat amb sistema LiDAR

També es poden utilitzar càmeres d'alta definició i sistemes LiDAR a terra (figura 6) per a analitzar una paret on hi hagi un accident geològic (figura 10).

Figura 6: Sistema LiDAR terrestre

El LiDAR [2] és un radar làser que serveix per a analitzar el terreny i generar un model tridimensional del accident, permetent-lo exportar fàcilment a un model informàtic. Aquest model permet als geòlegs tenir un mapejat de les imatges a la superfície 3D que es pot analitzar a posteriori. Aquest model, gràcies al làser de la tecnologia LiDAR, els permet descartar quasi tota la vegetació, obtenint un model net amb el què treballar. A més, el model els permet analitzar l'accident de manera precisa, sense tenir que tornar al accident per a tornar a capturar les dades.

Les càmeres HD els permeten generar les dades fotomètriques de l'accident, efectuant fotografies des de diferents angles. Aquestes fotografies passaran a ser el model 2D que els investigadors analitzaran. Per a obtenir-les, els investigadors es situen al costat del LiDAR, o en el cas del dron la càmera va al costat del LiDAR, i fotografien diferents angles del què s'està estudiant.

Aquestes eines generen una gran quantitat de dades, les quals no es poden analitzar ïn situ"actualment, ja que necessiten una aplicació que els permeti lligar aquest model 2D (fotografies [fig. 7]) amb el model 3D (model digital [fig. 8] i el model digital processat 'a posteriori' [fig. 9]).

Figura 7: Fotografia aèria

Figura 8: Model virtual generat pel LiDAR

El problema principal dels investigadors és la gran quantitat de temps que triguen entre que es dirigeixen a l'accident, generen les dades, tornen a la facultat, analitzen les dades i en treuen conclusions. Fins ara, en no poder fer el lligam de manera fàcil entre els models 2D i 3D, l'anàlisi consisteix en marcar les fractures o sediments sobre la imatge impresa, buscant grups o categories de fractures o sediments, fer els càlculs necessaris i comprovar amb l'ajut del model 3D que les falles o sediments marcats a la imatge són reals i si els plans que generen les fractures s'intersequen, mostrant que hi ha perill de trencament o esllavissada.

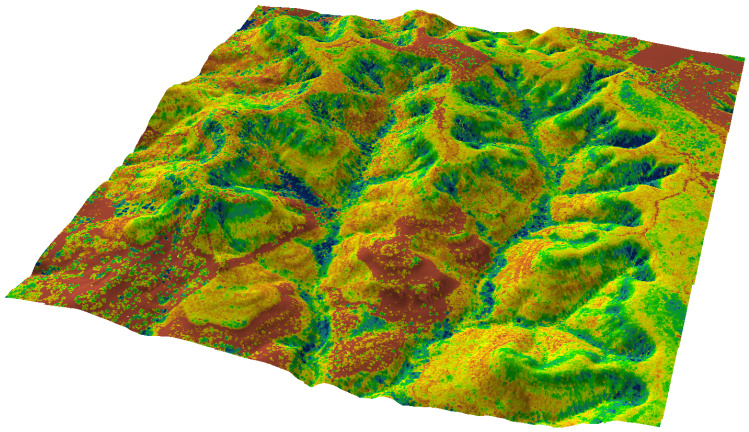


Figura 9: Model virtual processat per l'ordinador

Els programes que existeixen en l'àrea de recerca només els permeten fer la digitalització de les falles al model 2D o treballar directament amb el model 3D per a trobar les falles. És interessant accelerar aquest procés, doncs aquesta anàlisi pot predir esllavissades de terreny i permet securitzar accidents geogràfics. A més, això també permet generar un històric de falles, que permet una millor anàlisi i predicció de esfondraments de terreny, evitant així casos de esllavissades com els que es mostren en la figura 10 i en la figura 11.

Figura 10: Esfondrament a Fumanya

Figura 11: Esfondraments a Lleida

2.2 Anàlisi de l'aplicació

Es desitja crear una eina que permeti a l'usuari analitzar les fotografies d'un emplaçament. Aquestes fotografies es complementen amb un núvol de punts 3D (o *Point Cloud*). L'usuari obtindrà el vincle entre les fotografies 2D i el núvol de punts editant sobre les fotografies i veurà en temps real una visualització de la seva edició sobre el núvol de punts, permetent així saber si la seva selecció de família de fractures és correcta.

L'aplicació a desenvolupar per aquest projecte està basada amb un software anterior que permetia fer aquest lligam 2D-3D, anomenat Geo150 ([1]). Geo150 va ser desenvolupat al 2014 per tècnics geòlegs de la Facultat de Geologia de la UB. Està desenvolupat en VisualBasic 6. És un software útil pels geòlegs per a digitalitzar possibles orientacions del terreny a partir de les fotografies. Tot i així, es troba en una fase beta de desenvolupament, sobretot en la part de la digitalització 2D. En aquest software, les eines d'edició i selecció de punts 2D són bàsiques i alhora pateix errors en el manegament de memòria degut a una baixa gestió de memòria, fet que produeix errors de robustesa en el manegament de projectes grans, provocant tancaments inesperats de l'aplicació.

2.3 Requeriments funcionals

La nova aplicació a dissenyar, doncs, té els següents requeriments funcionals: els referents a la gestió de projectes i tota la informació associada, els referents a les eines de visualització de la imatge, les eines d'edició 2D i les connexions entre els models 2D i el model 3D. A continuació es descriuen en detall cadascuna d'elles.

2.3.1 Gestió de projecte

L'aplicació ha de permetre la creació i gestió d'un projecte format per un conjunt de subprojectes, els quals han de incloure les imatges i les dades associades a aquest subprojecte. Aquests projectes poden contenir dades de calibració de les imatges, un fitxer de correspondència per a cadascun dels models 2D amb el model 3D, el model 3D compartit per a totes les imatges (*Point Cloud*) i una matriu de canvi de base de les coordenades del model 3D. La gestió de projecte ha de permetre la càrrega i guardat del projecte, així com la seva modificació.

El projecte i els subprojectes han de ser fàcils d'utilitzar per l'usuari i de ràpida comprensió i alhora, han de ser flexibles per a poder ser modificats amb diferents opcions. Les direccions de les dades contingudes als projectes han de ser relatives a la carpeta d'execució de l'aplicació, permetent generar una carpeta de projectes on es puguin guardar, facilitant la importació i exportació dels projectes i tota la seva informació associada.

2.3.2 Eines de selecció i visualització de la imatge

L'aplicació ha de tenir un mode que permeti la selecció de la imatge a mostrar, a més de poder moure-la, ja sigui mitjançant barres de desplaçament o mitjançant *panning* (clicar i arrossegar la imatge). També ha de permetre mostrar la imatge completa, allunyar-se i apropar-se, a més de poder mostrar un mini-mapa de context per indicar a quina zona correspon la visualització en relació a la imatge completa.

2.3.3 Eines d'edició i digitalització 2D

L'aplicació ha de permetre dibuixar punts, segments i polilínies sobre qualsevol imatge del projecte. Els segments són la recta delimitada per dos punts i la polilínia és un conjunt connectat de segments.

En relació als punts, l'usuari pot fer les següents accions: Adició, Esborrat, Moure, Unió i Intersecció. A la figura 12 es representen cadascuna de les accions de forma gràfica.

L'**Adició** ha de poder fer-se de forma fàcil i còmoda, permetent generar un punt al lloc de la imatge on s'ha clicat. En afegir diversos punts, l'aplicació ha de generar automàticament les línies entre els punts afegits. L'usuari podrà seleccionar tant punts, segments o polilínies. A més a més, un cop l'usuari considera que ha acabat d'adició de punts en una polilínia, l'aplicació ha de permetre el tancament de les polilínies mitjançant una interacció simple i ràpida, ja sigui utilitzant un doble clic o un clic amb el botó dret del ratolí. Aquesta acció no s'ha de poder fer a una polilínia que ja hagi estat tancada prèviament.

En relació a l'**esborrat**, l'usuari ha de poder eliminar de forma àgil, aquells objectes erronis o innecessaris. S'han de poder esborrar punts de forma simple, gestionant l'esborrat de les línies de connexió i, alhora, s'ha de poder eliminar la polilínia, fent servir algun mecanisme de selecció, donant una retroacció visual, donant una opció al menú, i advertint amb un diàleg per assegurar que no s'ha efectuat l'acció de forma no deliberada.

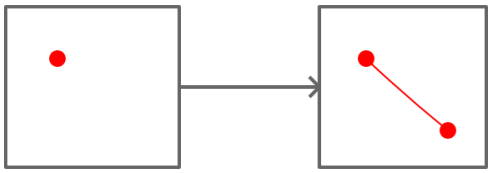
L'acció de **Moure** ha de permetre moure un punt sobre la imatge, fent clic per a seleccionar-lo i mantenint-lo premut per a desplaçar-lo Per a aquest cas, l'aplicació ha de gestionar no només el moviment conjunt del punt sinó que també el canvi en els segments que estan connectats a ell.

La **Unió** és una operació d'esborrat d'un punt que pertany a dos segments, permetent l'unió automàtica dels segments formant-ne un de nou. Tot i que és un esborrat i es podria afegir com a un cas especial dins de la acció d'esborrat, es considera una operació diferent que donarà casuístiques diferents a la del simple esborrat d'un punt aïllat o el punt final d'una polilínia.

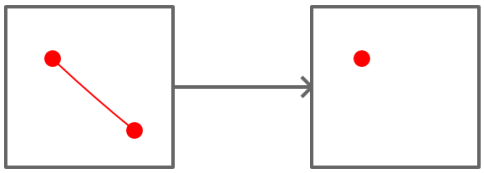
La **Intersecció** ha de permetre afegir un punt nou en un segment. L'usuari haurà de seleccionar el segment on es vol afegir el punt i la seva posició en la imatge. L'aplicació haurà de gestionar la creació de nous segments produïts per aquesta operació.

A més, l'aplicació ha d'oferir un sistema o acció que permeti seleccionar a quin grup o categoria pertanyen els punts, els segments o polilínies. El concepte de categoria és important per fer famílies de polilínies que determinen la direccionalitat del terreny. Per a ressaltar-les, s'utilitzarà un codi de color per a cada categoria

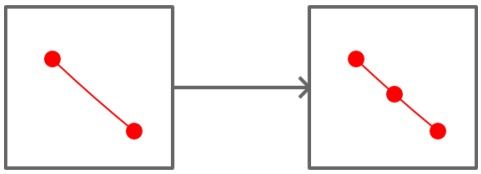
La codificació de categories o famílies de polilínies i el fet que en un mateix subprojecte es puguin tenir un gran nombre de categories implica la necessitat de amagar o fer visibles elements o conjunts d'elements. Aquesta nova acció permetrà a l'usuari canviar la visibilitat de la categoria de les polilínies, punts o segments, fent-los desaparèixer de la imatge. També aquesta mateixa acció ha de permetre fer-los aparèixer de nou a la imatge.



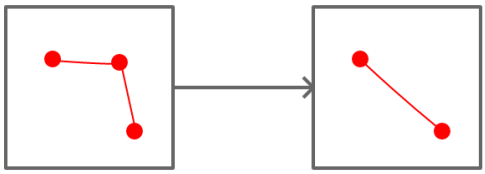
(a) Acció d'adició



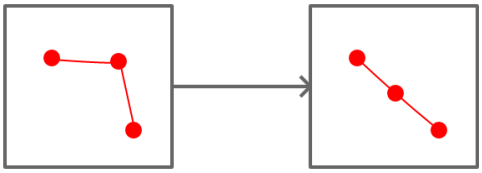
(b) Acció esborrat



(c) Acció Intersecció



(d) Acció Unió



(e) Acció de moure un punt

Figura 12: Accions sobre la imatge

2.3.4 Connexió model 2D - 3D

L'aplicació ha de permetre generar una connexió entre ambdós models: el conjunt d'imatges i el model 3D o *point cloud*.

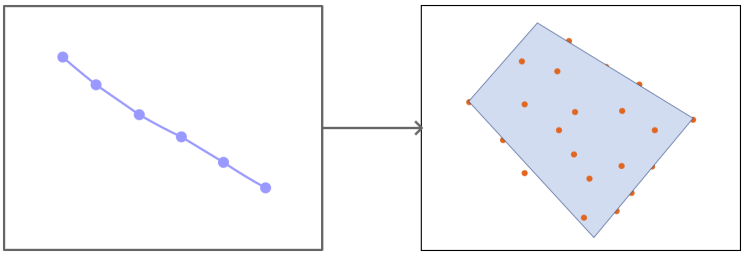


Figura 13: Representació d'una polilínia respecte al seu pla en el Point Cloud

Per a fer aquesta connexió, s'ha d'aproximar un pla vàlid dins del *Point Cloud* que contingui els punts 3D corresponents als punts 2D de la polilínia. Aquesta connexió es representarà amb un tipus especial de visualitzador de dades anomenat **estereoplot** [11]. Aquest visualitzador és una forma de veure l'orientació dels plans 3D a partir dels punts 2D. És un visualitzador àmpliament utilitzat en la comunitat de geòlegs tant per veure l'orientació del pla 3D del point cloud associat a una polilínia, com per a veure la coplanarietat de les diferents polilínies pertanyents a una categoria. De fet, l'estereoplot és un ajut a la digitalització que verifica si els punts digitalitzats són correctes en relació als plans que s'estan identificant de la mateixa categoria.

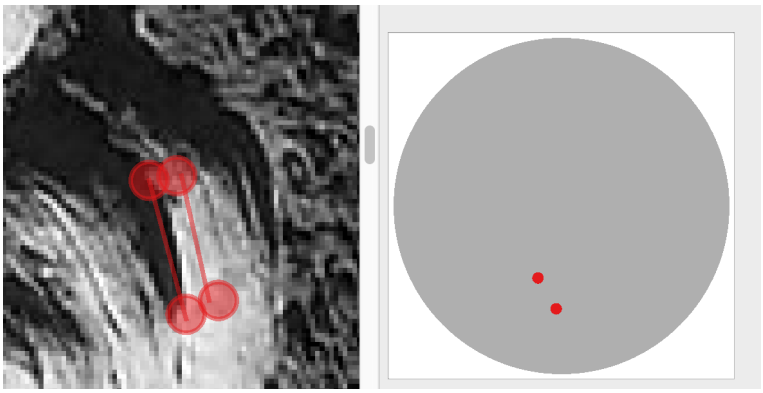


Figura 14: Exemple Estereoplot

2.4 Requeriments tecnològics

Per a desenvolupar aquesta aplicació s'han de complir els requisits de software i hardware enumerats seguidament.

Per part del Software, com a qualsevol software, el codi font de l'aplicació ha d'ésser fàcilment reutilitzable i/o fàcil de interpretar, de manera que en possibles actualitzacions, el codi permeti un fàcil manteniment o millora. En aquest projecte és especialment rellevant, ja que el codi podrà ser ampliat per altres informàtics o pels mateixos arqueòlegs. Aquest requeriment es pot aconseguir, mitjançant bones pràctiques i maneres alhora de dissenyar i codificar així com utilitzant una estructura d'enumeració de variables, que identifiqui de forma entenedora quin és el propòsit de cadascuna de les funcions o variables en joc.

Seguidament, s'han de poder suportar canvis i extensions en el codi, de forma transparent a l'usuari final de l'aplicació. Per a aconseguir aquesta fita, s'ha de generar el codi, creant classes i funcions bàsiques que emmascarin gran part de codi, fent-lo modular. D'aquesta manera, s'aconsegueix que els canvis i actualitzacions del codi passin desapercebuts.

Un altre requisit de software a tenir en compte especialment en aquesta aplicació és la robustesa. L'aplicació ha de ser capaç de recuperar-se, tant de fallades simples com d'errors crítics. Això es pot acomplir gestionant i fent que l'usuari no pugui generar l'error o gestionant les excepcions generades per les classes en ocórrer la fallada.

Com a requisit important, donat que es maneguen grans volums de punts en el model 3D (o point cloud) és l'optimització de l'aplicació en relació al consum de memòria.

Per a fer això, l'aplicació ha de ser eficient alhora de generar i carregar dades, eliminant duplicats. A més s'ha de gestionar de manera correcta i eficient l'alliberació de aquestes dades, minimitzant el risc d'abús en l'ús de memòria.

L'últim requisit, fonamental per al ús general del programa i el seu context d'aplicació, és l'ús d'aquesta en un entorn sense connexió, com per exemple, en el moment en el què els geòlegs estan en el camp, prop del terreny a analitzar. Els usuaris han de disposar d'una eina portable, que els permeti fer la digitalització, en cas de que ho vulguin, en el mateix lloc de l'anàlisi. Per això, l'aplicació no ha de dependre de llibreries o serveis lligats a la connexió via Internet. Aquesta és una de les principals raons de la importància de la robustesa i l'optimització de l'aplicació, ja que en no poder fer servir serveis externs, com per exemple una connexió client-servidor, tota la gestió de memòria, de les dades i de l'aplicació s'ha de fer en el mateix terminal on s'utilitza.

Ara bé, per part del requisits de hardware, és important destacar la necessitat d'utilitzar l'aplicació tant en ordinadors de sobretaula, com a ordinadors portàtils. A més a més, donada la diversitat dels equips utilitzats, l'aplicació ha de pensar-se per a poder-se compilar i executar en diferents sistemes operatius, facilitant la utilització i portabilitat del software.

A causa de la seva natura sense connexió, l'aplicació ha de ser dissenyada de forma que s'optimitzi l'ús de CPU alhora de fer els càlculs. Aquest requisit és fonamental per a donar un software portable i utilitzable en gran quantitat de terminals diferents, ja que hi han dispositius que s'usaran que tenen poca potència de CPU.

L'optimització de l'aplicació en quan a memòria també ve determinada pel requisit de poder executar l'aplicació en entorns amb una quantitat petita de memòria volàtil disponible. S'ha de tenir en compte el consum de memòria RAM, ja que molts ordinadors, encara no sobrepassen els 4GB de memòria i per tant, l'aplicació pot copsar a causa d'aquest factor. Això també es degut a les mancances d'alguns dispositius, amb processadors gràfics integrats a la CPU, que fan que tinguin una menor potència de càlcul.

Seguint amb el consum energètic, la gestió gràfica dels models ha de ser lleguera i alhora eficient i, per tant, ha d'utilitzar pocs recursos de targeta gràfica (o GPU).

Per a petició dels clients, s'ha afegit la possibilitat de poder utilitzar l'aplicació en entorns amb grans pantalles. Això, s'ha de complir fent que el entorn gràfic s'ajusti segons la mida i la disposició de la pantalla (entorn adaptable). Aquest requisit permet, a més, poder tenir una visió més amplia de l'escena i, alhora, poder fer que l'aplicació es redimensioni per a terminals amb pantalles petites.

2.5 Antecedents

Analitzant diferents aplicacions que tracten temes geològics relacionats amb imatges obtingudes amb LIDAR o HD, es poden trobar aquelles que maneguen només informació 2D (com ArcGis/ArcPad), només informació 3D (com "Digifract"y Openplot) o ambdós (Matlab), excepte Geo150 que ofereix el lligam 2D-3D. (veure taula 1).

A continuació es descriuen les principals contribucions en software realitzades i resumides en la taula 1.

Les eines ArcGis/ArcPad [3] serveixen per a traçar línies en les imatges en 2D dimensionada amb GPS per a treure les dimensions en el sistema mètric però sense tenir en compte la profunditat o el 3D. Aquest tractament de dades no té gestió del projecte ni un sistema que uneixi el model 2D amb el 3D generat pel LiDAR. S'utilitza per a digitalitzar les fractures i extreure propietats com l'espaiament entre fractures o la intensitat.

Per una altra banda, existeix una eina per a treballar amb MatLab anomenada "Descontinuity Analysis"que permet usar amb processos de extracció de informació 2D i 3D semi-automàtics de fractures en massissos muntanyosos, treballant sobre el núvol de punts [4]. Amb aquesta eina es pot extreure dades com l'orientació de les fractures, l'espaiament entre elles i la seva longitud. No té gestió de projecte, ni permet connectar el model 2D i 3D, però conté un sistema de cerca automàtica de plans i eines de digitalització en 2D per a traces.

La següent eina analitzada s'anomena "DigiFract"[5] i mesura l'espaiament en l'entorn Quantum GIS, el qual és un entorn Open Source de creació, edició, visualització, anàlisi i publicació de dades Geoespacials. Aquest entorn treballa en 3D digitalitzant fotografies dimensionades directament amb GPS. Aquesta eina és una continuació de les eines ArcGis/ArcPad però no permet generar un lligam entre el model generat pel LiDAR i el model fotogràfic, ja que genera el model 3D, guiant-se de coordenades GPS i això fa que sigui menys precís. Com a extensió de les eines ArcGis/ArcPad, té eines de digitalització de fractures, però no suporta gestió de projectes.

OpenPlot [6] és una eina de digitalització en 3D sobre un MDT (Model Digital de Terreny) triangulat on és entapissat per l'ortoimatge. L'edició es fa directament sobre el núvol de punts, fet que fa difícil el treball de l'anàlisi de falles. El programa conté eines d'orientació de plans, i al treballar amb un MDT triangulat, fa que aquest sigui molt lent al moure'l en 3D.

App	Fractures	Projectes	Imatges	Edició	2D	3D	2D - 3D
ArcGis/ArcPad	✓	×	✓	✓	✓	✓	×
OpenPlot	✓	×	✓	✓	×	✓	×
DigiFract	✓	×	✓	×	×	✓	×
Matlab	✓	×	✓	✓	✓	✓	×
Geo 150	✓	✓	✓	×	✓	×	✓
GAT	✓	✓	✓	✓	✓	✓	✓

Taula 1: Taula Comparativa de Aplicacions

Finalment, l'aplicació Geo150 [1] es va dissenyar com a intent de solucionar els problemes anteriorment anomenats i com a alternativa que permetés el lligam entre les dades 2D i el point cloud de referència a les aplicacions de tractament de falles, tant comercials com lliures. A la darrera línia de la taula, es mostra l'abast on vol arribar el present projecte, anomenat GAT (Geological Analysis Tool), com a millora del projecte Geo150.

A la figura 15 es veu una aplicació Geo150 en la que es basarà el present projecte GAT. Analitzant la interfície de Geo150 s'han extret molts dels requeriments d'usuari necessaris. Tot i que la interfície és complexe amb un gran nombre de dades, Geo150 permet la digitalització i la connexió entre el model 2D i el 3D, mitjançant una eina de visualització de dades anomenada estereoplot.

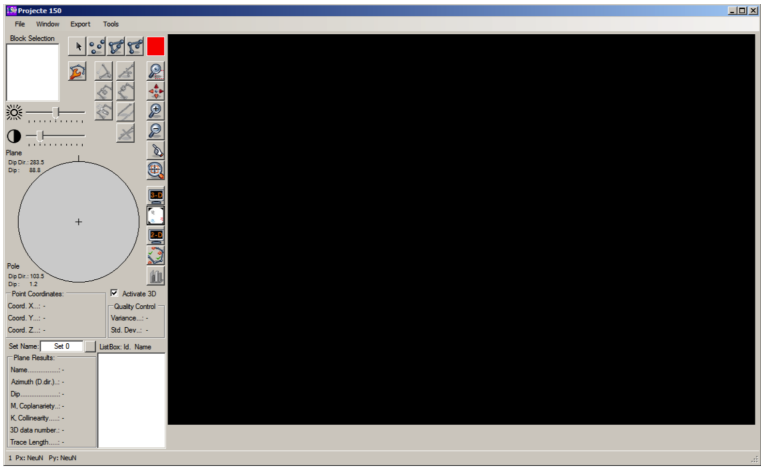


Figura 15: Interficie de la app Geo150 [1]

En l'aplicació Geo150, però, per a digitalitzar les fractures o plans 3D, es fan servir eines d'edició que permeten pintar a sobre de les imatges punts, polilínes i polígons. Aquestes eines són les que fan que Geo150 falli en algunes ocasions, sobretot quan es tracten projectes amb gran quantitat de dades.

Per altre banda, l'aplicació en sí té mancances a l'hora de poder ser utilitzada en altres sistemes operatius diferents a Windows i la seva gestió interna de memòria fa que sigui un programa molt pesat, el qual demana alts requisits de memòria.

Per la part de gestió de projecte, es gestiona el projecte sencer, fent servir un fitxer de text pla, que l'usuari ha d'omplir seguint unes directrius específiques. El fet de que el projecte estigui especificat així, fa que sigui difícil l'ús i extensió dels projectes per a usuaris no experts.

2.6 Justificació de la tecnologia utilitzada

A continuació, s'analitzen els llenguatges i entorns possibles per a desenvolupar el projecte GAT donats els requeriments esmentats en les seccions prèvies, focalitzant en la Simplesa, Portabilitat, potència gràfica, l'activitat de la comunitat i la possibilitat de treballar sense connexió. En la taula 2 es resumeix les característiques dels diferents entorns analitzats.

Llenguatge	Simplesa	Portabilitat	Grafica	Comunitat	No Connexió
Java	✓	✓	×	✓	✓
C++	×	✓	✓	✓	✓
QT + C++	✓	✓	✓	✓	✓
QT + Python	✓	✓	×	✓	✓
Webapp	×	✓	✓	✓	×

Taula 2: Taula Comparativa d'entorns de desenvolupament i Llenguatges.

La utilització de Java permet crear una aplicació executable directament a qualsevol sistema operatiu, mitjançant el fitxer .jar. També, gràcies al seu *garbage collector*, permet no tenir que preocupar-se per la memòria dinàmica a utilitzar i el fet que el codi es compili en una maquina virtual, fa que doni una capa extra de seguretat. Java permet crear aplicacions de forma ràpida i fàcil, amb i sense connexió, però la seva gestió gràfica deixa molt que desitjar. La gestió d'objectes gràfics és lenta i en fer servir una màquina virtual, els recursos que disposa no són suficients per a garantir una aplicació gràfica fluida quan hi han molts objectes a la pantalla. També, degut al *garbage collector*, es tendeix a fer classes i funcions no gaire òptimes, fent que l'aplicació baixi el seu rendiment. Finalment, Java, en ser un llenguatge popular, té una gran comunitat d'usuaris que creen llibreries i *frameworks* per a optimitzar el procés de creació, reduir la complexitat del codi i oferir ajuda als altres usuaris.

La següent opció estudiada és l'ús de C++. Aquest és un llenguatge antic, però ràpid, robust, fàcilment compilable i executable. A més, permet una boníssima gestió tant de la memòria com de la part gràfica. C++, també aporta una bona potència de càlcul i com que no s'executa en un entorn virtual ni té un *garbage collector*, tot i que actualment s'estan desenvolupant llibreries per a fer-ho, s'ha de tenir molt en compte la gestió dels objectes, tant la creació com la destrucció, per a no omplir tota la memòria. Un dels principals problemes que té C++ és la seva dificultat per a depurar el codi. Utilitzant programes externs com gbd o valgrind, es pot arribar a depurar suficientment el codi, però la bona praxis i el coneixement de les estructures de dades són requisits indispensables per a fer una aplicació robusta.

Una de les millors opcions possibles és l'entorn format per QT [7] i C++. Aquesta unió, a més d'oferir un suport molt gran de la comunitat i un editor bastant potent, ofereix bones eines de depuració del codi i les llibreries gràfiques de QT. Aquestes llibreries, permeten crear i gestionar objectes de forma fàcil. A més, les llibreries s'encarreguen de la reserva i alliberació de memòria dels seus objectes. Aquesta opció dóna totes els avantatges de C++, però permitent una gestió millor dels objectes i agilitzant la creació de aplicacions, gràcies al *framework* que ofereix la llibreria QT.

Un altre opció és l'unió entre el *framework* de QT i Python. Amb una gran potència de càlcul i una gran comunitat darrere, aquesta és una de les opcions més viables. Per definició del llenguatge, Python és molt fàcil de llegir i molt simple, tant d'aprendre com d'entendre. S'aprofita de la potència i dels objectes que ofereix QT, al igual que QT i C++. En quan a la visualització de dades i gestió de aquestes, pocs llenguatges poden fer-ho com Python, a més de la gran quantitat de paquets i d'exemples de codi que es poden trobar. La seva portabilitat, en ésser un llenguatge interpretat, és fantàstica, ja que només es necessita el intèrpret, disponible per a tots els sistemes operatius, per a poder executar l'aplicació. La seva mancança més important és la gestió gràfica de volums grans de dades en temps real, tant en models 2D interactius com en models 3D.

La darrera opció estudiada és la creació d'una Webapp fent servir l'arquitectura Client - Servidor. Aquesta opció és la més òptima en quan a gestió de recursos, la més lleugera i permet dotar als usuaris d'una aplicació molt robusta. Es poden trobar infinits exemples i *frameworks* ([8], [9], [10]) per a crear una webapp, utilitzant llenguatges com: Java, Python, JavaScript, PHP, etc. Tot i això, una aplicació creada amb aquest tipus de tecnologia, té inconvenients molt clars. Primer de tot, l'aplicació, per a poder oferir aquest servei, necessitaria connexió constant d'Internet, cosa que per a definició del problema, no es pot complir sempre. A més a més, relacionat amb el problema anterior, es podria executar el servidor a l'ordinador on es vol treballar, però això provoca que l'usuari hagi d'executar el servidor i es perdi la possibilitat d'oferir una aplicació lleugera. Finalment, per a poder dissenyar aquesta aplicació, s'haurien de crear dues aplicacions: una aplicació client, que enviï, rebi i gestioni la informació, i un Servidor, que gestioni els càlculs i les peticions de les aplicacions client. S'ha descartat la creació d'una aplicació purament web, ja que les interfícies web poden ser altament complexes, amb la dificultat que això implica.

En conclusió, la tecnologia emprada per a desenvolupar l'aplicació GAT és la combinació formada per les llibreries i *frameworks* QT amb el llenguatge C++. Aquesta opció, ofereix unes eines suficientment bones per a poder crear una aplicació que acompleixi amb els requeriments, tant tecnològics com funcionals.

3 Disseny

3.1 Disseny de la interfície gràfica guiat per l'usuari

La interfície gràfica de l'aplicació ha tingut varies etapes de disseny amb diferents entrevistes validades amb els usuaris finals, que són els investigadors de Ciències de la Terra.

Es parteix de la interfície de l'aplicació desenvolupada prèviament Geo150, que es mostra a la figura 15.

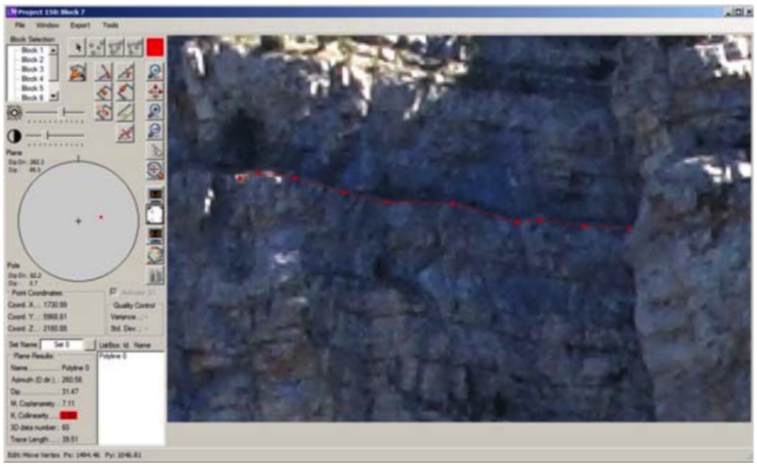


Figura 16: Interfície de la app Geo150 en funcionament

La interfície gràfica de la figura 16 té una quantitat molt gran de elements, fet que fa que l'usuari tingui una sensació de confusió, ja que el usuaris solen treballar amb una pantalla i sobretot amb ordinadors portàtils.

Es poden apreciar com els botons i els visualitzadors de dades, tant l'estereoplot com les llistes, estan tots ordenats a l'esquerra. Això denota un excés de informació, provocant la fatiga i saturació a l'usuari.

Aquest disseny peca també de excés de colors, fent que no sigui adequat per a persones amb problemes de visió.

Les icones que es poden apreciar, mostren informació excessiva, la gran quantitat de colors, a més de les formes intricades i la seva mida minúscula fa que s'hagi de forçar la vista per a entendre les dades.

Com es pot observar, la part de visualització de dades és la part més atapeïda. Els components, com les llistes o les finestres de informació, donen massa informació. S'observa una mida de lletra difícil de llegir a simple vista a més de que el color de fons, ja que el contrast del color de la lletra i del fons és menor, dificultant encara més la identificació de les dades.

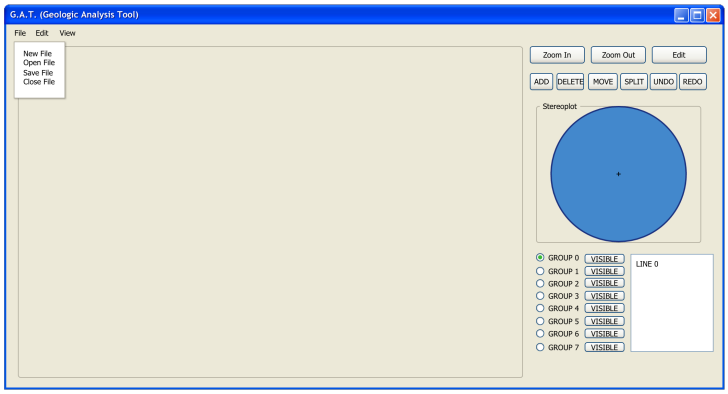
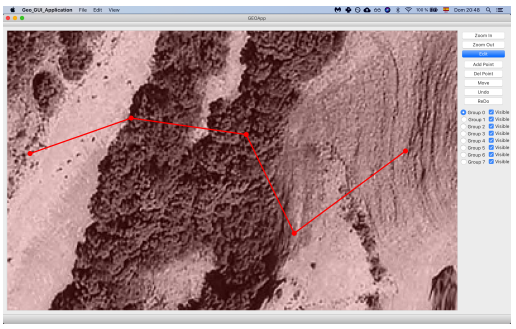
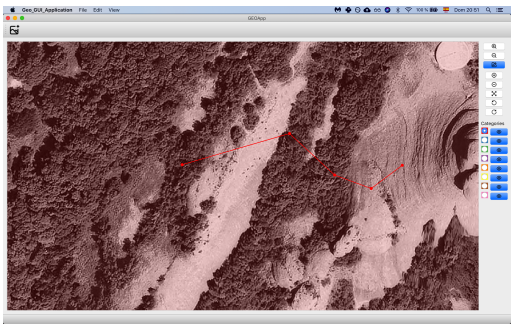


Figura 17: Disseny de interfície en Low Fidelity

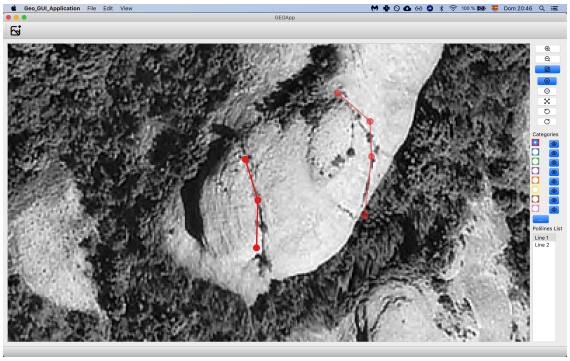
Considerant tots aquests aspectes i inconvenients, s'ha decidit fer, en una primera aproximació, una interfície més minimalista i agradable per a l'usuari (figura 17. Encara que aquest primer disseny no presenta cap icona, mostra una interfície gràfica més neta i agradable. En una segona iteració del disseny, ja durant la implementació del prototip, s'ha decidit fer una versió més simple (figura 18a), sense visualització de dades. En successives iteracions s'han definit les icones (figura 18b), els menús i elements de selecció (figura 18c) i en la iteració final (figura 19) es trasllada la idea principal amb un toc de color i icones simples.



(a)



(b)



(c)

Figura 18: Primeres Iteracions de Disseny

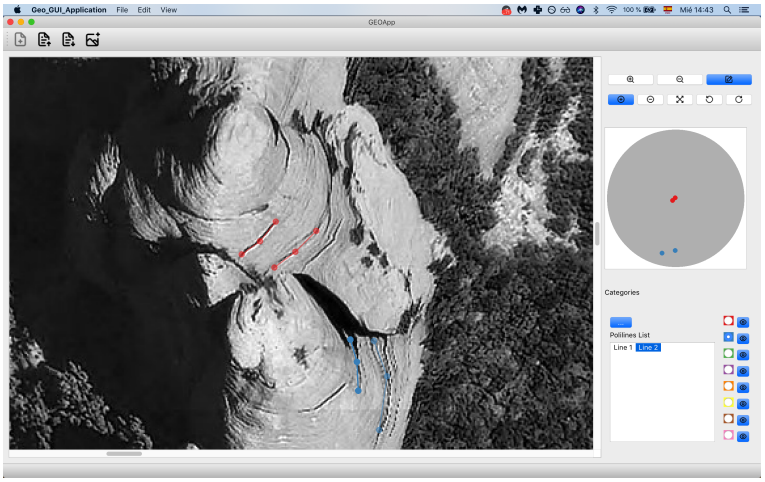


Figura 19: Iteració Final de Disseny

En el disseny bàsic mostrat a la figura 18a), validat pels usuaris, tot i que és un punt de partida, hi han errors d'alineació entre els botons i la grandària d'aquests. La idea d'aquesta iteració en el disseny és iniciar una idea bàsica del què es podrà utilitzar i que permeti identificar totes les opcions amb claredat. En el disseny de Geo150, totes les icones, botons i visualitzadors són a l'esquerra. Però, degut a que la majoria d'usuaris són destres i a la tendència dels usuaris a tenir el ratolí a la zona dreta de la pantalla, s'ha decidit moure les opcions, simplificant-les, a la dreta.

En la següent iteració de disseny (fig18b), s'ha decidit canviar els missatges en text per a icones. Aquestes acompleixen les següents característiques: són fàcilment identificables, són monocromàtiques i són simples, atorgant un toc minimalista i donant la informació necessària en cada cas.

En la tercera iteració de disseny de l'aplicació (fig 18c), es segueix l'estil de la iteració anterior, reduint la mida de botons i afegint selectors per a les polylines. Una característica notable d'aquesta iteració es la introducció d'un *highlight* o resseguit de les polilínies seleccionades, per a afavorir l'ús i facilitar de forma gràfica la identificació del objecte amb el que s'està treballant.

Les anteriors iteracions (fig 18b i 18c) segueixen el mateix estil, però en començar amb la integració del **estereoplot** (fig 19), s'ha considerat un redisseny de la secció de botons, per tal d'encabir, com en el disseny de l'aplicació primigènia (figura 16, l'estereoplot. Per això, s'ha canviat l'estructura vertical per a una horitzontal que ens permet, a més d'obtenir una alineació correcta dels botons, obtenir una amplada més que suficient per a inserir l'estereoplot.

Degut a aquest redisseny, els botons de categories s'han desplaçat al costat esquerra i a la seva dreta, s'ha generat espai per a encabir un selector de polylines més gran. Aquest espai més gran ha vingut donat a petició dels usuaris, per a permetre veure i moure's amb més facilitat entre les diferents polylines dibuixades. L'adició de més menús de fàcil accés, tals com carregar, o salvar les dades, permet una utilització més ràpida i simple de les opcions de gestió de projecte.

També, l'accés directe a aquestes accions agilitza la utilització dels menús més usats per als usuaris, a l'hora de generar els projectes. Finalment, la seva disposició a l'esquerra, denota que són accions de menor ús que les accions d'edició o visualització.

3.2 Arquitectura del sistema

Per a crear aquesta aplicació s'ha decidit fer servir l'arquitectura MVC (Model - Vista - Controlador). Aquests tres blocs estan interconnectats entre sí, com es pot veure a la figura 20.

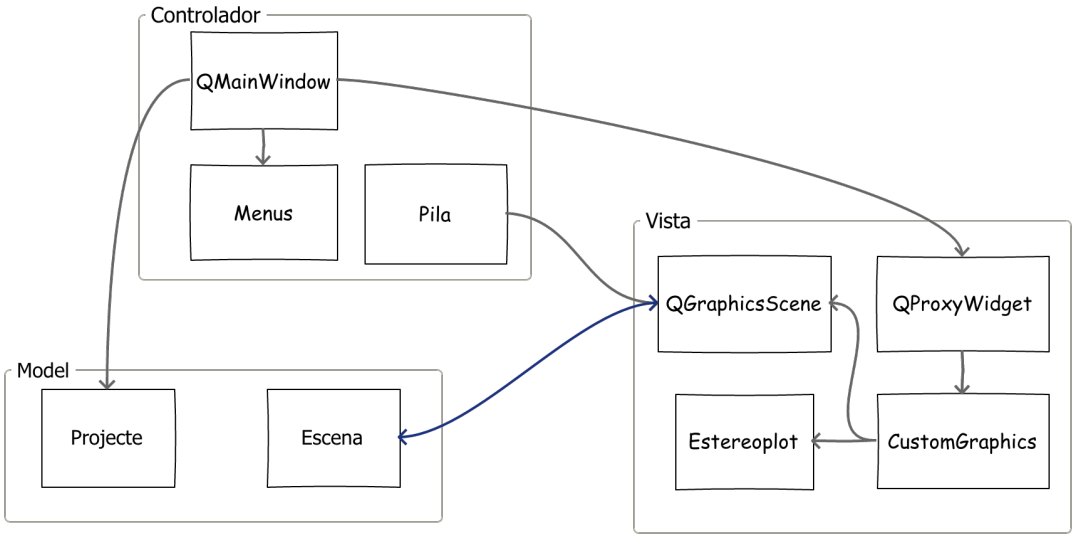


Figura 20: Diagrama MVC

3.2.1 Model

Aquest bloc és on es gestionen i es guarden les dades. El Model està format per dos models de dades, que són la base de l'aplicació.

El primer model de dades és el *Projecte* (figura 21). Aquest està connectat amb el controlador i és un model de dades auto-contingut. Es pot definir el model del projecte com un objecte contenidor en el que s'efectuen una quantitat mínima de accions de escriptura i serveix com a objecte de referència de l'aplicació. Un projecte conté informació per a poder llegir o carregar el subprojecte.

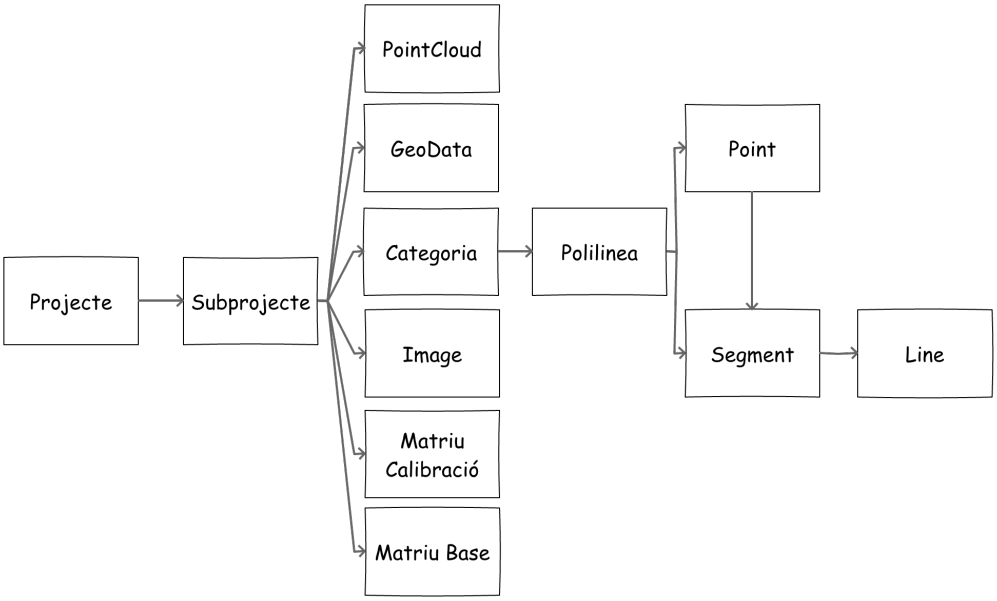


Figura 21: Projecte

El subprojecte és un objecte auto-contingut, que conté la informació mínima i rellevant que necessita el projecte. Conté una imatge, les matrius de cal·libració i de canvi de base, el Point Cloud, el fitxer de relació dels models i les polylines que s'han generat en el subprojecte.

El segon model de dades és l'*escena*. Aquest model actua d'objecte abstracte que conté tots els objectes gràfics generats per l'aplicació. Aquest model està lligat directament amb el objecte `QGraphicsScene` que conté la vista, ja que les modificacions efectuades tant al model com en aquest objecte es veuen reflectides en ambdós objectes.

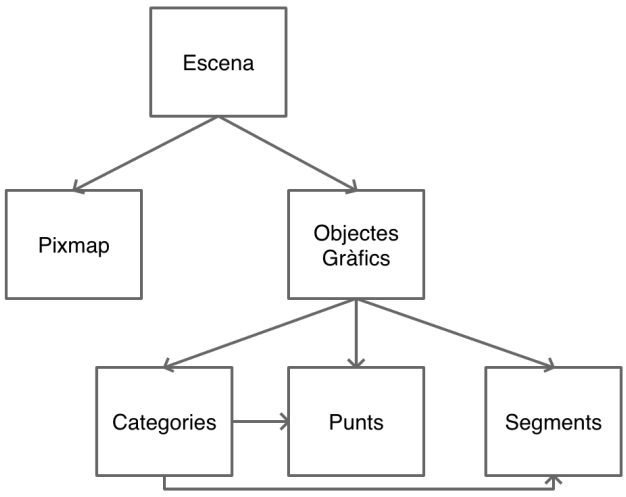


Figura 22: Diagrama de classes de l'Escena

3.2.2 Controlador

El controlador està format per la classe contenidora de l'aplicació, *MainWindow*, els *menús* que ens permeten tenir certa interacció amb la vista i la *pila*.

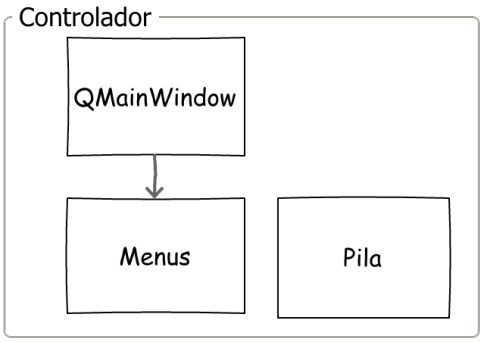


Figura 23: Controlador

MainWindow és la classe principal del programa, a més de ser la classe més gran del controlador, ja que gestiona la comunicació entre events. Aquesta classe, a més a més de connectar el model i la vista, gestiona tots els menús i objectes que formen part de l'aplicació. En ella es poden trobar referències i crides a objectes que formen part de la vista, que permeten certa interactivitat. En aquesta classe també es fa la gestió del projecte, fent servir una classe auxiliar i es carreguen els punts guardats en el projecte a l'escena.

El següent bloc, molt relacionat amb la classe *MainWindow*, són els *menús*. Aquests són objectes de QT, els events dels quals, es gestionen mitjançant *MainWindow*. Els *menús* són una de les parts més importants en la usabilitat de la interfície, ja que ens permeten fer una interfície útil i una interconnexió entre les accions que es volen fer i la vista.

El bloc final i un dels més difícils de crear ha sigut la *pila*. Aquest objecte és un dels més importants per a garantir flexibilitat a l'usuari, doncs dona la robustesa que es necessita alhora de gestionar la digitalització de punts sobre la imatge (veure taula 3).

Per a assegurar la robustesa de la edició, s'ha dissenyat, agafant la *pila* com a base, un sistema per a desfer i refer accions possibles sobre objectes. Aquestes accions són:

- ADD: Afegir un Punt a l'escena (també afegeix una línia si és possible)
- DELETE: Eliminar un Punt a l'escena (també elimina una línia si és possible)
- JOIN: Eliminar un punt intermig i unificar els extrems.
  - Exemple: Punt0-Línia0-Punt1-Línia1-Punt2 → Punt0-Linea0-Punt2
- SPLIT: Afegir un punt en una línia.
  - Exemple: Punt0-Línia0-Punt1 → Punt0-Línea0-Punt2-Línia1-Punt1

En la Figura 3 es resumeixen aquestes operacions amb els paràmetres necessaris per a garantir recuperar l'estat en una operació de 'Undo' o de 'Redo'.

Acció	Punt	Categoria
ADD	P1	G0
DELETE	P0	G7
JOIN	P2	G1
SPLIT	P3	G2
...	...	...

Taula 3: Estructura Stack Undo/Redo

Es tenen dues piles lligades entre si, la pila del UNDO i la pila del REDO. Per al fer un UNDO, s'extreu el primer element de la pila del UNDO, i es coloca a la pila del REDO. Un cop fet això, es realitza l'acció contrària a la que s'havia fet per a deixar consistent l'escena. En les taules 4 i 5 es mostra un exemple de funcionament de les Piles. En aquest exemple, es veu que s'havia fet un DELETE del punt 1 corresponent a la categoria 0 (veure primera fila de la taula 4). Per tant, en l'acció REDO apliquem un ADD de l'element punt 1 de la categoria 0 a l'escena, retornant-ho a la seva posició inicial i fent el link corresponent del punt.

Acció	Punt	Categoria
DELETE	P1	G0
DELETE	P0	G0
JOIN	P2	G0
SPLIT	P3	G1

Taula 4: Pas 1: Pila UNDO d'exemple

Acció	Punt	Categoria
DELETE	P1	G0

Taula 5: Pas 2: Pila REDO d'exemple

Acció	Punt	Categoria
DELETE	P0	G0
JOIN	P2	G0
SPLIT	P3	G1

Taula 6: Pas 2: Pila REDO d'exemple després d'extreure el primer element

Després de realitzar l'acció, la taula del UNDO (taula 6), perd el que era el primer element, deixant l'acció anterior com a opció a refer i es posa aquesta operació a la pila del REDO (taula 5). En el cas de efectuar una altre acció diferent a la desfeta sobre l'escena, es buida la taula del REDO. En cas contrari, en fer REDO, es torna a posar l'acció desfeta a la taula del UNDO, aplicant l'acció a l'escena i buidant la taula (5), per a assenyalar que ja s'ha refet l'acció.

3.2.3 Vista

La vista està formada per *QProxyWidget*, *CustomGraphicsView*, *QGraphicsScene* i l'*estereoplot*. Aquests objectes permeten la visualització gràfica on es pot interactuar amb els elements.

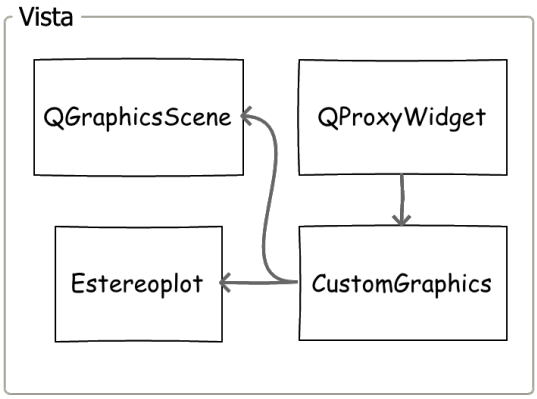


Figura 24: Vista

El primer objecte a tractar, *QProxyWidget*, fa de pont entre el *Controlador* i *CustomGraphicsView*, fent que aquest últim es quedi fixat sobre la imatge carregada i gestioni tots els seus events d'entrada. A més, dóna una capa de seguretat que fa més fàcil la gestió de la digitalització i permet que els punts siguin independents de la imatge.

El següent element és *CustomGraphicsView*, una classe *custom* que permet la creació i gestió dels objectes de l'escena, a més de gestionar el pintat del *estereoplot*. Aquest element ha sorgit per la necessitat de gestionar de forma independent els punts de la imatge, donant una capa extra d'extracció per motius de seguretat, de cara a la gestió de l'*escena*. Aquest element gestiona la *pila*, actuant sobre les dades de *QGraphicsScene*.

QGraphicsScene és un element que conté els punts digitalitzats i enllaça el model i la vista. Està gestionat i contingut per *CustomGraphicsView*. Aquest element conté una referència als objectes continguts a l'escena, això fa que en actualitzar les dades de l'element, també canviï l'*escena*.

Finalment, l'*estereoplot* és un dels elements creats per a l'aplicació per a representar de forma gràfica la relació de les dades digitalitzades 2D amb el model 3D. És creat per *QGraphicsView* i gestionat per *CustomGraphicsView*. Per a mostrar les dades 2D relacionades amb el model 3D s'han de seguir els següents passos:

1. Cal trobar el punt 3D del Point Cloud associat al punt digitalitzat.
2. Seguidament, amb el conjunt de punts de la polilínia digitalitzada ja trobats amb els seus associats al Point Cloud, s'ha d'ajustar un pla aproximat que contingui aquests punts.
3. Es calcula el vector normal associat al pla i s'aplica la convenció *Dip Direction* [12], que permet calcular la projecció del vector normal a l'esfera que representa l'esteroplot.

,

Cadascun d'aquests passos dependrà de la forma d'emmagatzemament del Point Cloud, quin mètode geomètric es faci servir per ajustar plans a un conjunt de punts. En el capítol d'implementació i resultats es detallen els mètodes i les representacions utilitzades.

La convenció de *Dip Direction* s'utilitza per aquests tipus de dades i és un llenguatge comú en l'anàlisi de les dades geològiques.

$$Dip_{Direction} = \left\{ \begin{array}{cc} \text{x} = 0 \ \& \ \text{y} = 0 & 0 \\ \text{x} < 0 \ \& \ \text{y} = 0 & \text{-90} \\ \text{x} \ \geq 0 \ \& \ \text{y} = 0 & 90 \\ \text{y} < 0 & 180 + \arctan x/y * 180/\pi \\ \text{y} > 0 \ \& \ \text{x} < 0 & 360 + \arctan x/y * 180/\pi \\ \text{y} > 0 \ \& \ \text{x} > 0 & \arctan x/y * 180/\pi \end{array} \right\} \tag{3.1}$$

$$Dip = \left\{ \begin{array}{cc} \text{x} = 0 \ \& \ \text{y} = 0 & 0 \\ \text{else} & \frac{x^2+y^2}{z} * \frac{180}{\pi} \end{array} \right\} \tag{3.2}$$

3.3 Scene

Aquest objecte, emmarcat dins del model, conté la informació més rellevant i necessària per a contenir les dades de l'aplicació: les Categories, les Polylinies, els Punts i els Segments.

3.3.1 Categoria

Es tracta d'un objecte que conté tant els segments com els punts generats a l'escena. Les categories permeten diferenciar fàcilment els objectes que pertanyen al mateix tipus i a més permeten ocultar-los, per a que l'usuari pugui centrar-se en treballar, sense trobar un conjunt enorme de línies que pertanyin a altres grups.

3.3.2 Punt

Aquest objecte és l'element bàsic de l'escena, sobre el que es construeixen la resta d'elements. Es pot generar un nou punt en fer clic sobre la imatge, estant en mode edició. Com a objecte bàsic, la seva finalitat és, a més de marcar gràficament la localització del clic generat, ser l'element atòmic en el qual es pot realitzar accions i defineix de forma unívoca un punt de la imatge amb un punt d'una fractura en el *point cloud*.

3.3.3 Segment

Aquest objecte es genera a partir de dos punts. Els segments contenen una llista doblement enllaçada el punt inicial i el punt final. La finalitat dels segments és fer de ajuda gràfica, per tal de facilitar a l'usuari la identificació i seguiment de les polylines.

3.3.4 Polylinies

Una polilínia és un conjunt tancat d'elements, començat i acabat per un punt, formats per les següents seqüencies:

- Punt: Quan la polilínia és un cas degenerat i es composa d'un sol punt.
- Punt-Línia-Punt: Objecte més natural i utilitzat. Aquest objecte permet descriure les direccions de les fractures i resseguir-les.

Com s'ha remarcat anteriorment, tant el Punt (CustomElipse) com la Línia (CustomLine) tenen una propietat que permet regenerar la polilínia a partir d'un objecte . Això és degut a que aquests objectes han estat dissenyats per formar una llista doblement enllaçada. Aquesta propietat s'ha afegit des del començament com un mètode per a optimitzar la memòria i permetre una forma més flexible, mitjançant l'ús de punters, d'accedir als objectes.

Per a fer això, els punts contenen dues línies, un mateix punt es inici una línia i final d'una altra. Els punts, a més, actualitzen les línies (si les tenen) quan es mouen. Les polilínies es guarden en una `llista` per categoria , on per les propietats definides als objectes que les formen, es guarda només l'últim punt que forma aquesta polilínia.

3.4 Point Cloud

El Point Cloud és un objecte que conté un conjunt finit, però molt gran, de punts, els quals formen part d'un model 3D. En el cas de l'aplicació dissenyada, està definit com un fitxer de text pla, que conté els punts separats amb espais. Per a poder fer la seva connexió amb les imatges, es necessiten 3 objectes més, la matriu de calibració, la matriu de canvi de base del Point Cloud i el fitxer de dades geològiques.

En les dades de tipus ortofoto no es fan servir les matrius. En el projecte actual només s'ha tractat aquest tipus d'imatges, però s'han deixat les funcions implementades amb les matrius per a facilitar la seva ampliació a altres tipus de dades.

La matriu de calibració és aquella que es genera a partir de les dades de calibració (figura 26) obtingudes de la càmera amb la que s'han obtingut les dades fotomètriques.

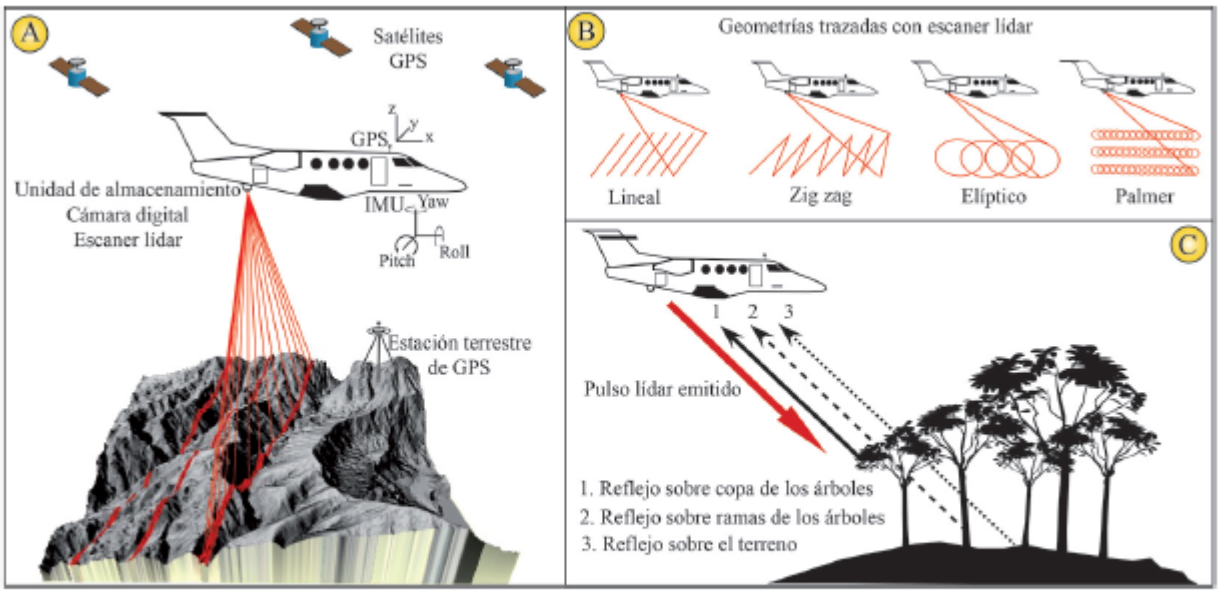


Fig. 1: A.) Esquema explicativo de funcionamiento de un equipo básico de lídar aéreo transportado generado con imágenes lídar del sector este del cráter del volcán Poás. B.) Geometrías que se pueden trazar con el escáner lídar durante un barrido. C) Esquema sobre el orden de retorno de un pulso lídar emitido desde un avión.

Figura 25: Exemple funcionament recollida de dades LiDAR

Després de generar-la, aquesta s'aplica al model 2D digitalitzat, per a corregir l'aberració i la distorsió generada per la lent de la càmera. En la matriu de calibració es defineixen els següents atributs:

- Gradient (Pitch)
- Inclinació (Yaw)
- Rotació (Roll)
- Z de la posició dins del sistema de coordenades del LiDAR
- Y de la posició dins del sistema de coordenades del LiDAR
- X de la posició dins del sistema de coordenades del LiDAR
- Distancia Focal
- x origen de píxel
- y origen de píxel
- 1er coeficient distorsió radial
- 3er coeficient distorsió radial
- Mida píxel x
- Mida píxel y
- Coeficient de transformació obliqua
- 5é coeficient distorsió radial
- coeficient P1 distorsió tangencial
- coeficient P2 distorsió tangencial

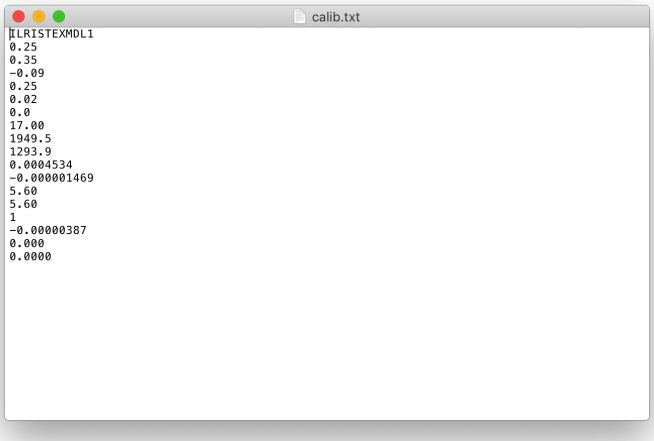


Figura 26: Estructura fitxer dades geològiques

La matriu de canvi de base serveix per a canviar de base els punts del Point Cloud, en cas de fer servir un projecte no ortonormal. Aquesta matriu permet corregir les dades per a trobar una relació correcta entre el model 2D i el 3D.

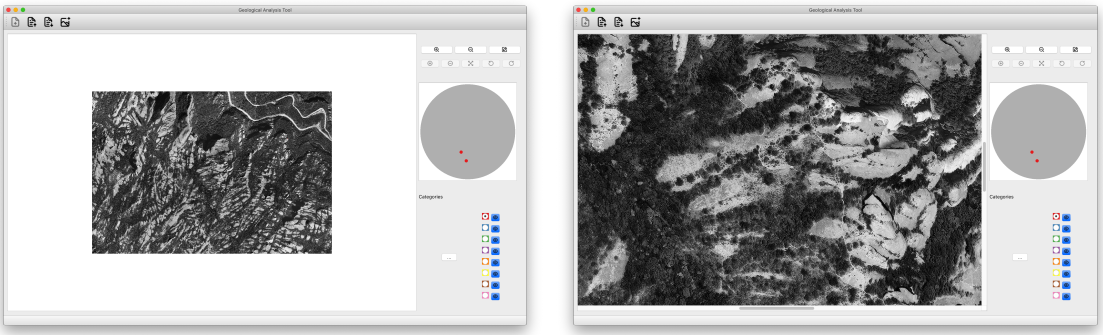
El fitxer de dades geològiques serveix per a poder saber on és l'origen de coordenades i en quin sentit creixen els eixos. Està codificat de la manera mostrada en la figura 27 i té la següent informació: Mida de píxel, 3 valors arbitraris, origen coordenades x i origen de coordenades y.

Figura 27: Estructura fitxer dades geològiques

4 Resultats i Simulacions

4.1 Editor 2D

Per a implementar l'editor, s'ha desenvolupat una aplicació gràfica bàsica basada en QT, en la que s'ha afegit una instància d'una classe nativa de Qt, QGraphicsView, per a poder mostrar una imatge. Posteriorment a la mostra de la imatge, s'han implementat les dues primeres interaccions, apropar i allunyar la imatge.



(a) Acció d'allunyar realitzada a l'aplicació (b) Acció d'apropar realitzada a l'aplicació

Figura 28: Accions de Zoom

Seguidament, ja tenint una base que mostrava una imatge, s'ha implementat el sistema de pintat de l'escena. Per a poder fer un element independent de la imatge, s'ha optat per un sistema de capes simple. És a dir, s'ha encastat un objecte interactiu a sobre de la imatge, de manera que les coordenades de l'escena d'aquest objecte corresponguessin a la imatge.

Durant aquest procés, per a gestionar els events d'aquest nou objecte interactiu, s'ha optat per a crear una nova classe anomenada CustomGraphicsView. Aquesta gestiona els events que succeeixen dins d'ella i els events que modifiquen els elements de l'escena. Per tal de gestionar els events de CustomGraphicsView s'utilitza un sistema de flags, que s'actualitzen al realitzar una acció a QMainWindow. Aquestes accions són l'activació de l'edició, l'adició de elements, l'esborrat i el moviment d'un element.

La implementació de l'adició de punts (figura 29) s'ha fet de la següent manera. Primer es comprova si hi ha un punt de partida en la categoria que s'està editant, anomenat **LastPoint**. Cada categoria pot tenir el seu propi LastPoint. Seguidament es genera el nou punt i en cas existeixi el LastPoint, es genera una línia de connexió entre el Last Point i el punt nou. Aquesta línia s'afegeix a l'escena i a la categoria adient, afegint-ne també el punt, tant a l'escena com a la categoria escaient. Ara el nou punt passarà a ser el nou LastPoint. En cas de no haver-hi aquest punt de partida, només s'afegeix el nou punt a l'escena i a la seva categoria.

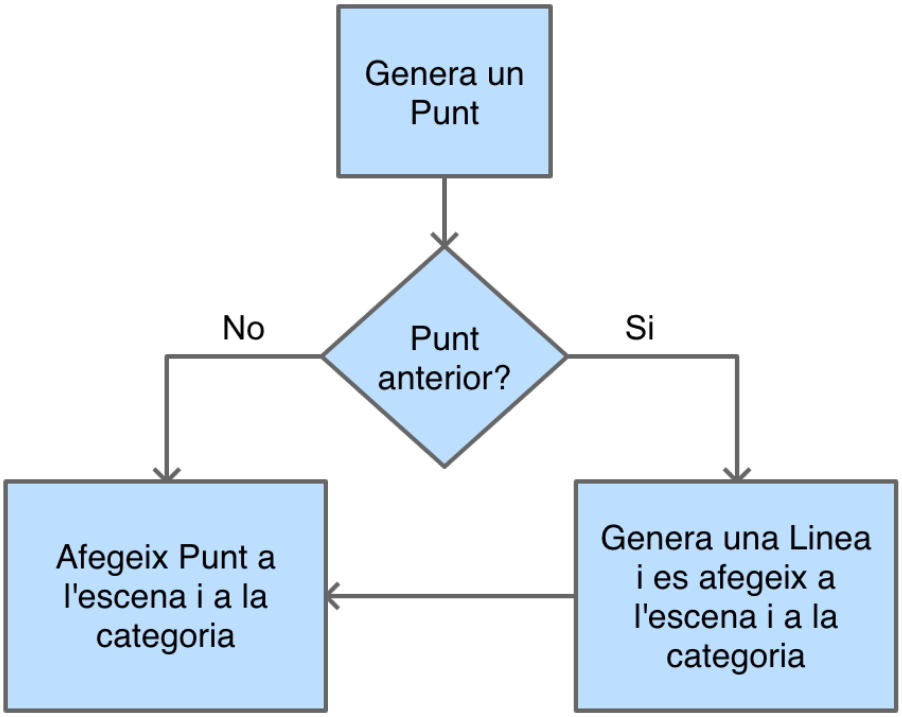


Figura 29: Diagrama de decisió Acció Adició

La implementació de l'esborrat d'elements (figura 30) s'ha fet de la següent manera. En clicar amb el botó dret sobre un punt, amb la acció d'esborrat seleccionada al menú d'edició, aquest menú respon delegant l'event, fet que fa que sigui capturat pel CustomGraphicsView seleccionant l'element que ha respòs. Fent això, es comprova que l'element a eliminar sigui el correcte i llavors es comprova la seva situació. En cas de tenir una **CustomLine** abans i un altra després, s'efectuarà l'acció de unió, eliminant el punt seleccionat de la categoria i de l'escena, conjuntament amb la línia posterior a ella i unint el punt anterior i posterior entre sí. En cas de tenir només una CustomLine, s'elimina el punt i aquesta línia de forma normal, canviant la línia final del punt anterior a null.

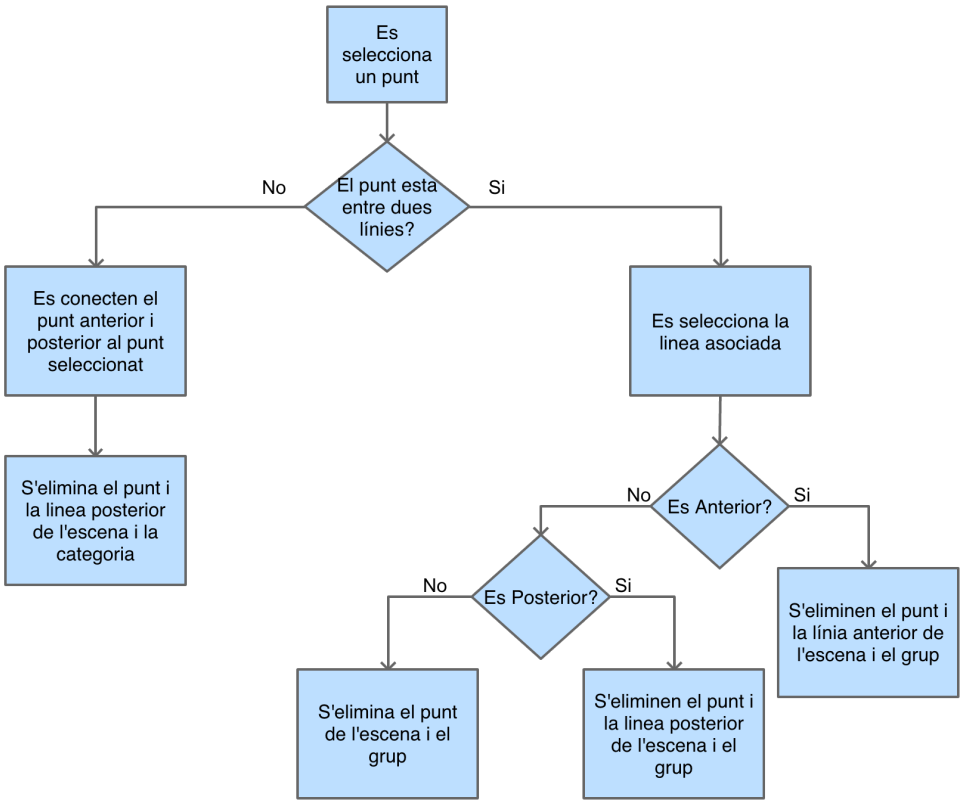


Figura 30: Diagrama de decisió Acció Borrat

La implementació del moviment d'elements(figura 31), s'ha fet servir en conjunt events predefinits a Qt i un event modificat, de manera que en el CustomGraphics-View es gestioni l'actualització de les línies associades als punts. A més, en cas de prémer sobre una línia, es fa la acció de intersecció, generant un punt i una línia nova, que s'afegeixen entre els punts anteriors. Totes les accions d'edició, excepte la del moviment s'afegeixen a la pila de UNDO.

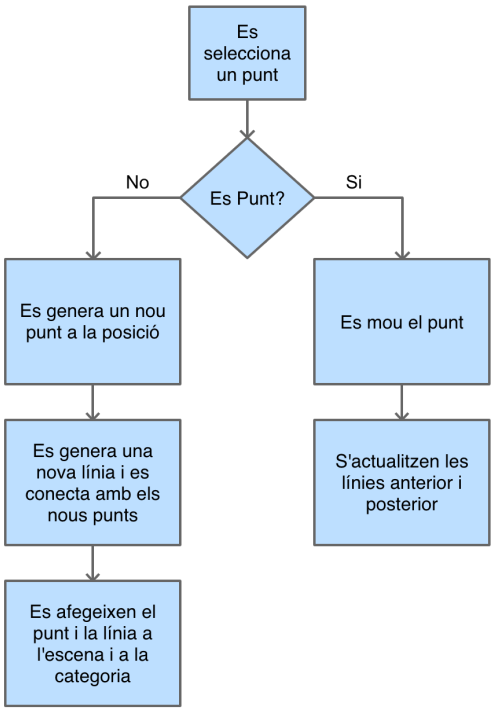


Figura 31: Diagrama de decisió Acció Borrat

Una vegada aconseguit poder pintar sobre aquest objecte, s'han implementat els objectes que es fan servir per a la digitalització. Aquests són **CustomEllipse** i **CustomLine**. Per a fer la gestió s'ha implementat de forma que CustomEllipse conté la CustomLine anterior i posterior i, alhora, CustomLine conté el CustomEllipse anterior i posterior (figura 32). És a dir, són estructures doblement encadenades.

Figura 32: Enllaçat de CustomEllipse i CustomLine

D'aquesta manera, s'obté de forma natural una llista doblement enllaçada que permet generar a posteriori les polylines.

Després d'aconseguir aquesta fita, s'han generat les interaccions de UNDO i REDO. Per a fer-ho, s'ha dissenyar una pila (taula 3) per a la interacció de UNDO i un altre pila per a la interacció de REDO. Per a implementar-ho, es comprova l'acció que s'ha fet anteriorment i s'efectua l'acció contraria, de forma que es fa efectiva l'acció de desfer o refer, tal i com s'ha mostrat a l'exemple del capítol de disseny (veure secció 3.2.2).

4.2 Projecte

El projecte és l'objecte on es guarden els subprojectes. Aquests últims, guarden una estructura amb una imatge, les matrius relacionades, el Point Cloud i les polylines generades. Inicialment, en el programa Geo150, un projecte es definia com un document de text pla, en format txt, relacionat amb un altre document en format txt, on es guardaven les dades generades formant logs de les sessions i estudis que s'havien fet. En el nou projecte, s'ha considerat dissenyar una estructura auto-continguda que permeti guardar tota aquesta informació en una única estructura. A més, per a la seva implementació es proposa la utilització de la llibreria JSON (nlohmann json[17]). l'estructura del projecte es mostra a la figura 33. Les estructures internes del projecte i del subprojecte es detallen a les figures 34 i 35.

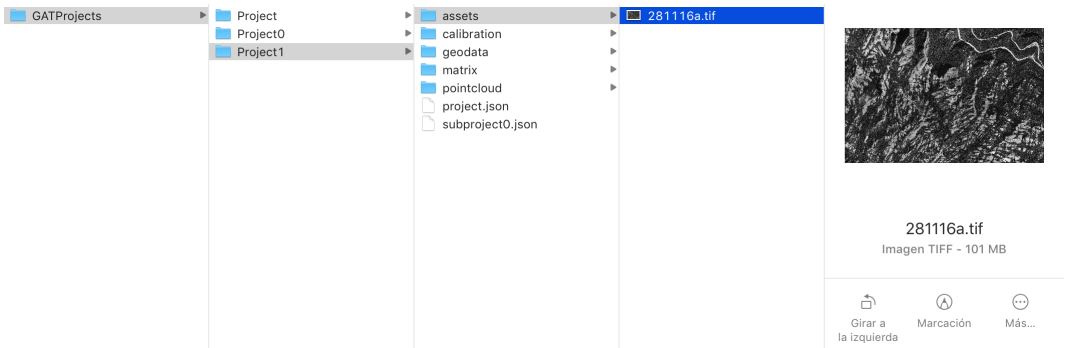


Figura 33: Estructura de carpetes del projecte

```
{
    "0": "../../../../GATProjects/Project1/subproject0.json"
}
```

Figura 34: Estructura interna del projecte

En implementar aquesta estructura, en fer un nou projecte, es genera la ruta relativa a les dades, depenent del sistema operatiu en qüestió. Per fer-ho efectiu, es comprova el sistema operatiu que s'esta utilitzant, fent servir comandes internes de c++11. Tot seguit es genera de la mateixa manera la carpeta de projecte, comprovant que la carpeta no existeixi.

Les carpetes es generen de manera iterativa, assegurant el control de el projecte en qüestió. Com que el més natural és carregar directament una imatge, es genera un nou projecte que contindrà inicialment la imatge carregada i, quan l'usuari editi sobre ella, es guardaran els punts i les polilínies en aquest nou projecte.

```
{
    "calibration": "",
    "geodata": "../../../../GATProjects/Project1/geodata/geodata0.tfw",
    "image": "../../../../GATProjects/Project1/assets/281116a.tif",
    "matrix": "",
    "pointcloud": "../../../../GATProjects/Project1/pointcloud/PointCloud0.txt",
    "polylines": {
        "group0": null,
        "group1": {
            "line0": [
                [3197.0, 1683.0],
                [3185.0, 1699.0],
                [3166.0, 1731.0]
            ]
        },
        "group2": {
            "line0": [
                [3231.0, -1534.0],
                [3132.0, -1475.0],
                [3027.0, -1414.0],
                [2925.0, -1397.0],
                [2859.0, -1448.0]
            ],
            "line1": [
                [3279.0, 1403.0],
                [3249.0, 1387.0],
                [3178.0, 1367.0],
                [3115.0, 1341.0],
                [3058.0, 1318.0],
                [2990.0, 1305.0],
                [2921.0, 1296.0],
                [2851.0, 1274.0]
            ]
        },
        "group3": null,
        "group4": null,
        "group5": null,
        "group6": null,
        "group7": null
    }
}
```


Figura 35: Estructura interna del subprojecte

En el procés de salvar la informació d'un projecte, el procediment és el següent. Primer es guarda la informació general del projecte i seguidament es guarda la informació corresponent al subprojecte que s'esta fent servir. Per fer-ho, es recorren tots els punts corresponents a cadascuna de les polylines i es guarda la seva localització, identificant la seva categoria i el número de línia dins de la categoria.

Per a carregar les dades es fa el procés invers, llegint les dades generals del projecte, carregant el fitxer `projecte.json` corresponent i carregant les dades del primer subprojecte. Per a llegir les dades del subprojecte es carrega la imatge, les matrius, les dades geològiques i el Point Cloud, aquest últim fent servir un objecte de la llibreria *pcl*[13] anomenat PointCloud, que permet gestionar i guardar de manera eficient un núvol de punts.

A més el programa permet canviar entre els subprojectes, fent servir el menú `View/Next Image` o `View/Previous Image`. Això canvia un identificador que determina el subprojecte, descarrega i guarda el subprojecte actual i en carrega el següent, depenent de l'acció de l'usuari En cas de no haver-hi posterior o anterior, no fa res.

4.3 Connexió model 2D-3D

Per a efectuar aquesta connexió s'ha implementat la càrrega de les dades geogràfiques en un projecte. Aquestes determinen l'origen de coordenades de les dades i la relació entre la distància del Point Cloud i la mida del píxel.

Seguint els passos citats en la secció 3.2.3, s'han fet les següents accions:

1. Per a trobar el punt 3D corresponent a un punt 2D de la imatge, s'aplica la matriu de transformació de la resolució de la imatge a la seva posició en el point cloud. Per a fer aquest pas, es parteix del punt 2D digitalitzat en coordenades de la escena i es converteixen, orientant l'origen a dalt a la esquerra de la imatge i multiplicant les noves coordenades per la mida del píxel. Seguidament, es suma l'origen de coordenades a les coordenades x i y calculades.

2. Un cop es té un punt 2D en coordenades de point cloud, es busca el punt del point cloud més proper. Donat que el point cloud està optimitzat a la llibreria pcl, es fa una cerca directa del punt més proper utilitzant la distància euclidiana amb tots els punts del Point Cloud.

3. S'afegeix el punt trobat a l'objecte CustomElipse corresponent. En cas de haver-hi CustomLine, es troba el punt mig de la línia i es cerca en el Point Cloud i s'afegeix a l'objecte en qüestió.

4. Amb tots els punts trobats en el Point Cloud, es troba el pla que els aproxima. Per això es poden fer servir molts mètodes. En aquest projecte s'ha utilitzat el mètode de Planar-Ransac [14]. *Canviaries aixo del model?*

4.3.1 Estereoplot

Per a mostrar la connexió entre ambdós models, la imatge i el point cloud, a partir dels punts trobats anteriorment, s'ha implementat l'objecte estereoplot.

Aquest és un objecte gràfic on es representa la projecció del vector normal del pla aproximat que conté els punts associats de la Polylinia digitalitzada al Point Cloud.

En una primera versió només es calculava el pla utilitzant el mètode Ransac, Primerament, només es calculava el pla, fent servir un model planar utilitzant Ransac i amb els coeficients retornats, es generava un punt de l'estereoplot.

En una segona iteració, tal i com ho interpreten els geòlegs, es calcula la projecció del vector fent servir la convenció Dip Direction [12]. Aquesta convenció, basant-se en el vector normal al pla, calcula l'angle i la inclinació del pla.

Amb aquestes dades, fent servir les fórmules detallades a la secció 3.2.3 es passa de l'angle i la inclinació a la projecció de les dades a un estereoplot.

L'objecte estereoplot és un QGraphicsView que té com a base un cercle centrat al mig del QGraphicsView.

Per a visualitzar la projecció d'un pla a l'estereoplot, una vegada obtingudes les coordenades a l'estereoplot, es passen a coordenades de l'escena. Aquest pas implica normalitzar les coordenades (restant el centre de l'estereoplot més la meitat del radi del punt per a cada coordenada) i multiplicant per un factor màxim (ja que les coordenades oscil·len entre -1 i 1). Aquesta transformació esta continguda a la linia 767 del fitxer 'CustomGraphicsView.cpp'.

4.4 Test d'usabilitat i validació

Durant la realització d'aquest projecte, els usuaris han fet proves amb les dues de les tres versions funcionals de l'aplicació.

En provar la primera versió, on els usuaris van provar l'editor 2D, on van fer moltes edicions 2D, amb nombroses operacions de UNDO i REDO, els usuaris van ser molt receptius amb tots els canvis de la interfície, trobant que l'aplicació era ràpida i amena, validant el disseny de la interfície. Tot i així, van trobar més requeriments que en un principi no s'havien tingut en compte com l'enumeració de les polilínies i una millor gestió d'events de l'editor (a vegades es seleccionava un punt amb el ratolí, però l'editor no capturava be l'event).

També van demanar una millor identificació de les categories mitjançant el color i l'ampliació del gruix de les línies en el moment de fer zoom de la imatge.

En l'ultima versió, amb l'editor, la gestió de projectes i l'estereoplot afegit, els usuaris han demanat el darrer últim retoc, corresponent a canviar la manera de calcular la reperesentació dels plans en l'estereoplot. Es en aquest punt on s'ha canviat el càlcul de la normal al pla pel calcul del Dip Direction[12] de la normal al pla, tal i com s'ha comentat abans.

Els usuaris han validat la funcionalitat de l'aplicació i el disseny gràfic. Tot i així, s'han trobat en petits defectes d'interacció amb el ratolí acoblat al teclat, errors que són inherents a l'ús d'alguns objectes de la llibreria Qt.

Després d'aquesta prova, es recomana la utilització de l'aplicació fent servir un ratolí extern i fent les accions de manera pautada, de manera que no es mantingui polsat massa temps el botó esquerra.

Finalment, els usuaris han donat el vist-i-plau del disseny, afegint com a possible millora en una propera versió valors de les dades que s'estant mostrant en l'estereoplot, com pot ser la inclinació del pla de la polylínea seleccionada.

5 Conclusions i feina futura

En aquest projecte s'ha desenvolupat una eina inicial per a agilitzar l'anàlisi de dades geològiques i fer un lligam entre les dades fotomètriques i els models 3D captats pels geòlegs, tal i com es va marcar a l'inici del projecte.

En primer lloc, s'ha dissenyat una interfície gràfica, fàcil d'usar, i amb informació rellevant pels geòlegs.

Cal remarcar també que s'ha millorat la part de digitalització de punts 2D, on s'han inclòs operacions de UNDO i REDO, per a facilitar la feina d'edició. En la part de digitalització s'han tingut en compte l'optimització del recurs de memòria per a poder carregar projectes de grans volums de dades. L'edició es pot gestionar a nivell de punt, polilínia i categoria.

S'ha definit una estructura de projectes en JSON per poder gestionar les dades associades a un projecte de forma fàcil, robusta i versàtil. En aquesta gestió de projecte s'inclou tant la informació inicial com les dades digitalitzades.

Finalment, s'ha dissenyat i implementat la visualització del lligam de les dades 2D al point cloud 3D en un estereoplot, eina utilitzada pels geòlegs per a validar les digitalitzacions.

Tot aquesta aplicació s'ha validat amb els geòlegs durant tot el projecte, incloent les millores proposades i consultant tots els temes durant tot el desenvolupament del projecte.

Tot i això, encara queda feina per fer i per poder donar als usuaris una eina millor i més refinada.

5.1 Feina Futura

Per a continuar aquest projecte, s'haurien de fer les següents millores:

- Millorar el feedback del usuari: mitjançant algun element no invasiu que permeti a l'usuari saber en tot moment que esta fent o mostrant de dades com la inclinació del pla de la polylinea o dades necessàries pel client.
- Afegir visualització 3D: part que no es va poder incloure al projecte per falta de temps, però possible gracies a les llibreries de gestió de Point Clouds utilitzades en el projecte.

A Manual tècnic

A.1 Instal·lació: Requeriments mínims i passos a seguir

- Passos a seguir en Apple
  Per a executar el codi en iOs, només s'ha d'executar el application-bundle generat per al deployment.

- Passos a seguir en Windows
  Per a executar el codi en windows, només s'ha d'executar el executable generat per al deployment.

En cas de no poder executar-se, s'haurà de fer fent servir les opcions de desenvolupador descrites en el manual del desenvolupador.

A.2 Manual del desenvolupador

Els requeriments de hardware utilitzats per a desenvolupar l'aplicació han estat:

- Macbook Pro mid 2012
  CPU: INTEL i7-3615QM CPU @ 2.30GHz
  GPU: NVIDIA GeForce GT 650M
  RAM: 8GB

- Torre windows
  CPU: AMD 6350-FX
  GPU: ASUS GeForce GTX 750-OC
  RAM: 16GB

Per a poder desenvolupar l'aplicació, és necessari instal·lar les següents llibreries:

- Eigen 3
- PCL
- boost
- flann
- VTK

Per a instal·lar-les en iOs, és aconsellable fer servir Homebrew [**?**], fent servir les següents comandes:

```
brew cask install pcl
```

Amb aquesta comanda s'assegura la instal·lació de les llibreries auxiliars associades a PCL, en aquest cas s'instal·len Eigen, boost, flann i VTK.

En cas de no tenir Homebrew, s'ha d'accedir a `http://pointclouds.org` i seguir els passos de instal·lació descrits a la pàgina.

El kit a utilitzar en iOs, és el següent:

- C Compiler: Clang x86 64 bits
- C++ Compiler: Clang++ x86 64 bits
- debuger: System LLDB at Xcode.app
- cMake tool: cmake GNU

A la configuració de projecte, s'ha d'afegir a la variable path, els següents directoris:

- /usr/local
- /usr/local/bin
- /usr/local/sbin
- /usr/local/Cellar

Per a instal·lar-les en Linux, es pot fer utilitzant binaris precompilats o compilar el codi de la llibreria directament. La llibreria conté explicacions per a instal·lar-la als següents sistemes:

- Ubuntu
```
sudo add-apt-repository ppa:v-launchpad-jochen-sprickerhof-de/pcl
sudo apt-get update
sudo apt-get install libpcl-all
```
- Debian
```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-key 19274DEF
sudo echo 'deb http://ppa.launchpad.net/v-launchpad-jochen-sprickerhof-de/pcl/
maverick main' >> /etc/apt/sources.list
sudo apt-get update
sudo apt-get install libpcl-all
```

A més de instal·lar les llibreries anteriors amb `sudo apt-get install &Llibreria&`

Per a desenvolupar el codi a Windows, és necessari instal·lar la llibreria PCL, fent servir el instal·lador creat per els desenvolupadors de la llibreria o compilant la llibreria al sistema. En cas de fer servir Windows, per simplesa i facilitat, es té un document anomenat `CMakelists.txt`, el qual es pot utilitzar com a document d'inici a QT.

Per a obtenir el codi font d'aquest projecte s'ha de descarregar del repositori següent: `https://github.com/IvanSevilla/GAT/releases/tag/v1.0`. Un cop descarregat el codi font, es descomprimeix en una carpeta, en la qual es troba un conjunt de projectes de prova, una carpeta amb el codi compilat per a mac i la carpeta amb el codi font, anomenada "GAT". Per a poder carregar el projecte a Qt, només cal obrir projecte i seleccionar el fitxer .pro o el fitxer CMakeLists.txt per a que carregui el projecte.

Referències

[1] García-Sellés, D.: *PROJECTE 150, Reference Guide Version Beta for Windows Vista, 7 and 8.* January, 2014

[2] NOAA.: *What is LIDAR?* National Ocean Service website, `https://oceanservice.noaa.gov/facts/lidar.html` accés el 14/06/2019.

[3] Bertotti, G.; Hardebol, N.; Taal-van Koppen, J.K.; Luthi, S.M.: *Toward a quantitative definition of mechanical units*: new techniques and results froman outcropping deep-water turbidite succession (Tanqua-Karoo Basin, SouthAfrica). AAPG Bulletin 91, 1085–1098, `http://dx.doi.org/10.1306/03060706074` Agost 2017

[4] Gigli, G. ; Casagli, N.:Semi-automatic extraction of rock mass structural data from *high resolution LIDAR point clouds.* International Journal of Rock Mechanics and Mining Sciences - INT J ROCK MECH MINING SCI. 48. 187-198. `https://doi.org/10.1016/j.ijrmms.2010.11.009` Gener 2011

[5] Hardebol, N.J.; Berttoti,G.: *Digifract*:A software and data model implementation for flexible acquisition and processing of fracture data from outcrops `https://www.sciencedirect.com/science/article/pii/S0098300412003706` Abril 2013

[6] Tavani, S.; Granado, P.; Corradetti, A.; Girundo, M.; Iannace, A.; Arbués, P.; Muñoz, J.; Mazzoli, S.: Building a virtual outcrop, extracting geological information from it, and sharing the results in Google Earth via *OpenPlot* and Photoscan: An example from the Khaviz Anticline (Iran). Computers & Geosciences. `https://www.sciencedirect.com/science/article/pii/S009830041300277X` Febrer 2014.

[7] The Qt Company:*Qt*: Cross-platform software development for embedded & desktop `https://www.qt.io` Juny 2019

[8] Django Software Foundation: *Meet Django* `https://www.djangoproject.com` Juny 2019

[9] Ronacher, A.: *Flask is Fun* `flask.pocoo.org` Juny 2019

[10] Pivotal Software: *Spring*: the source for modern java `https://spring.io` Juny 2019

[11] Guillem, W: Introducción a la proyección *estereográfica*. `https://www.geovirtual2.cl/Geoestructural/prak02.htm` Juny 2019

[12] Johnson, S: Understanding *strike* and *dip* on geologic maps. `https://fractalplanet.wordpress.com/2014/03/15/understanding-strike-and-dip-on-geologic-maps/` Juny 2019

[13] Bogdan Rusu, R; Cousins, S. : *3D is here*: Point Cloud Library (PCL), IEEE International Conference on Robotics and Automation (ICRA), 9-13 Maig de 2011 http://www.pointclouds.org/ accés el 15/05/2019

[14] PointClouds.org: Tutorial: Plane model segmentation. http://pointclouds.org/documentation/tutorials/planar_segmentation.php Juny 2019

[15] StackOverFlow question: Make a *VB-dll* and load it in C++ application https://stackoverflow.com/questions/10949455/make-a-vb-dll-and-load-it-in-c-application accés el 11/02/2019

[16] Som, G.: Crear una *DLL normal de Windows con Visual Basic 6.0* explicado paso a paso http://www.elguille.info/vb/avanzado/crear_dll_windows_con_vb6_explicado.htm accés el 11/02/2019

[17] Lohmann, N.: C++ *Modern JSON Library* https://github.com/nlohmann/json/releases accés el 14/02/2019

[18] Lohmann, N.: Serialization / Deserialization https://github.com/nlohmann/json#serialization--deserialization accés el 24/04/2019

[19] StackOverFlow question: How can I return a list of matrices from Rcpp to R? https://stackoverflow.com/questions/44795190/how-can-i-return-a-list-of-matrices-from-rcpp-to-r accés el 30/01/2019

[20] The Qt Company: Chapter 1: *Writing a Unit Test* http://doc.qt.io/qt-5/qttestlib-tutorial1-example.html accés el dia 30/01/2019

[21] Coppola, D.: C++ unit testing with Qt Test – part 1 – introduction http://blog.davidecoppola.com/2017/11/cpp-unit-testing-with-qt-test-introduction/ accés el 30/01/2019

[22] Qt Forum: How to set Auto resize GUI in Qt depending on the screen size https://forum.qt.io/topic/94227/how-to-set-auto-resize-gui-in-qt-depending-on-the-screen-size accés el 20/02/2019

[23] StackOverFlow question:How to get rid of tiffwarning message https://stackoverflow.com/questions/27608124/imagemagick-how-to-get-rid-of-tiffwarnings-768-message-about-unknown-field-wh accés el 21/2/2019

[24] @mitchcurtis : Slate - Pixel Art Editor https://github.com/mitchcurtis/slate/ accés el 24/02/2019

[25] StackOverFlow question: How to layer independent widgets in Qt https://stackoverflow.com/questions/9176473/how-to-layer-independent-widgets-in-qt accés el 24/2/2019

[26] StackOverFlow question: How to add an image on the top of another image https://stackoverflow.com/questions/4375433/how-to-add-an-image-on-the-top-of-another-image accés el 24/02/2019

[27] StackOverFlow question: Click event for QGraphicsView Qt https://stackoverflow.com/questions/16039020/click-event-for-qgraphicsview-qt accés el 24/02/2019

[28] Qt Forums : QGraphicsView & mouse events https://forum.qt.io/topic/37334/solved-qgraphicsview-mouse-events/3 accés el 25/02/2019

[29] StackOverFlow question: Storing a List of Objects https://stackoverflow.com/questions/1309921/storing-a-list-of-objects accés el 10/03/2019

[30] Qt Bugreports : No touch events handling (QGraphicsProxyWidget / QWidget) https://bugreports.qt.io/browse/QTBUG-45737 accés el 21/03/2019

[31] Legotskaya, E. : Qt/C++ - Lesson 023. Moving QGraphicsItem on QGraphicsScene with mouse help https://evileg.com/en/post/86/ accés el 25/03/2019

[32] Baron, B.: Qt QGraphicsScene click, select, move, resize, delete QGraphicsItems https://gist.github.com/benjbaron/532a476560f3d7085bda accés el 25/3/2019

[33] Thread on Qt Centre :Position of QGraphicsItem on Scene. https://www.qtcentre.org/threads/65173-Position-of-QGraphicsItem-on-Scene accés el 27/03/2019

[34] Iconscout: Unicons https://iconscout.com/unicons?#start accés el 14/04/2019

[35] Question at CodeProject: How to write log file in C++ https://www.codeproject.com/questions/97485/how-to-write-log-file-in-c accés el 04/05/2019

[36] StackOverFlow question: Fastest way to check if a file exist using standard C++/C++11/C? https://stackoverflow.com/questions/12774207/fastest-way-to-check-if-a-file-exist-using-standard-c-c11-c accés el 04/05/2019

[37] StackOverFlow question: Fastest way to check if a file exist using standard C++/C++11/C? https://stackoverflow.com/questions/34963738/c11-get-current-date-and-time-as-string accés el 04/05/2019

[38] StackOverFlow question: `https://stackoverflow.com/questions/6252060/check-folder-path` accés el 14/05/2019

[39] Howell, M: *Homebrew*: The missing package manager for macOS (or Linux) `https://brew.sh` accés el 25/06/2019