Universitat de Girona
Escola Politècnica Superior

Projecte fi de grau

Estudi: Grau en Enginyeria Informàtica

Títol: Implementació de tècniques de percepció NDT i compàrativa amb tècniques ICP

Document: Memòria

Alumne: Ricard Segura Duran

Tutor: Pau Vial Serrat
Departament: Arquitectura i Tecnologia de Computadors
Àrea: Arquitectura i Tecnologia de Computadors

Convocatòria (mes/any): Juny 2022

Universitat de Girona
Escola Politècnica Superior

Projecte Fi de Grau

Implementació de tècniques de percepció NDT i comparativa amb tècniques ICP

Autor:
Ricard Segura Duran

Juny 2022

Grau en Enginyeria Informàtica

Tutors:
Pau Vial Serrat
Narcís Palomeras Rovira

Resum

En aquest projecte es presenta el disseny i la implementació d'una llibreria en llenguatge C++ que permet resoldre el problema de registre de núvols de punts acústics procedents d'entorns submarins. Es tracta d'una llibreria que es pot executar en temps real, per tant, un dels objectiu de disseny ha estat l'eficiència computacional del codi. Alhora, la llibreria s'ha especialitzat per sensors acústics, que generen núvols de punts menys densos i molt més sorollosos que els generats per lidars o càmeres de profunditat. L'aplicació de la llibreria és processar les dades generades per sonars muntats en Vehicles Submarins Autònoms que realitzen tasques de mapeig, avançant així amb l'exploració científica del fons marí i el manteniment de nombroses infraestructures submarines.

Agraïments

Per començar vull agrair molt especialment a l'equip de VICORB i al professorat del Grau en Enginyeria Informàtica de la UdG. Gràcies a tothom he pogut participar en el desenvolupament d'un projecte que no hagués pogut dur a terme pel meu compte. He aconseguit molts coneixements al desenvolupar el projecte i solucionar els errors i he pogut conèixer a gent molt bona que m'ha ajudat a entendre millor les coses i aprendre'n de noves.

Índex

Capítol 1

Introducció

En aquest projecte es descriu el desenvolupament i implementació d'una llibreria de tècniques de percepció acústica per a la robòtica submarina.

1.1 Motivacions

El projecte s'ha dut a terme per a generar una ajuda als investigadors que treballin en un entorn submarí. Aquest entorn és tan complex com desfavorable per a les aplicacions robòtiques, pel fet que la majoria de sistemes de percepció i comunicació que s'utilitzen en l'àmbit terrestre no funcionen en aquest entorn per culpa de l'aigua. L'aigua actua com un gran atenuador de les ones electromagnètiques, anul·lant els sistemes de comunicació, també dificulta el camp de visió de les càmeres òptiques, escurçant el camp de visió a uns pocs metres. Altres sensors que es poden utilitzar són els làsers subaquàtics però poden ser perillosos per a qualsevol humà que hi estigui a prop i encara es troben en desenvolupament. Un altra alternativa són els sensors acústics que fan servir ones mecàniques, que són més lentes i més sorolloses, per a construir un sistema de comunicació i percepció. Entre aquestes opcions s'ha fet servir els sensors acústics, encara que siguin més lents i més sorollosos, perquè són els que ens permeten arribar a veure més lluny. Per la qual cosa els algoritmes que processen les dades han de ser més avançats i amb una robustesa i complexitat majors que els altres.

Una tecnologia emergent en el camp de l'exploració submarina són els Vehicles Autònoms Submarins. Els AUV són plataformes robòtiques dotades de sistemes de percepció i control que operen sota l'aigua i que permeten efectuar diferents tasques com localització, mapatge i planificació de trajectòria, d'una manera autònoma. Un cop s'estableix la missió a realitzar, el robot comença a executar la tasca i la intervenció humana és limitada a la supervisió del robot, mirant que es mantingui en funcionament i recollint-lo un cop ha acabat amb la seva missió.

Fent ús d'aquestes plataformes robòtiques, es pot avançar amb l'exploració científica de fons marí i el manteniment autònom de nombroses infraestructures submarines.

Quan es volen generar mapes a partir de l'associació de vistes adquirides amb un sensor de rang com pot ser un sonar, si només confiem en el sistema de navegació a la deriva del robot, és a dir, la integració dels sensors inercials del vehicle, s'obtenen mapes incoherents. Això vol dir, que quan es revisita una zona ja visitada les vistes no encaixen. Per això ens cal un sistema que processi en línia les dades que es van rebent del sensor per registrar les vistes i evitar incoherències en els mapes.

1.2 Objectius

L'objectiu principal d'aquest projecte és el desenvolupament de forma conjunta amb l'estudiant Miguel Malagón d'una llibreria implementada en llenguatge C++, on s'implementin diferents variants de la tècnica de registre de núvols de punts coneguda com a Normal Distributions Transform per adaptar-la al món acústic. Els objectius desglossats d'aquest projecte són:

- Estudiar i entendre un article científic.
- Aprendre com utilitzar el middleware ROS en una aplicació robòtica.
- Fer la definició de l'estructura de classes per a una llibreria.
- Generació d'un codi eficient computacionalment per ser executat en temps real en una aplicació submarina.
- Aprendre com realitzar la correcta documentació d'un codi amb el programa Doxygen sobre un repositori que es farà públic un cop acabada la llibreria.
- Estudi i aprenentatge de tècniques equivalents de l'estat de l'art.
- Avaluació del codi amb dades reals, obtingudes amb robots submarins al mar.
- Comparativa de l'algorisme implementat amb altres tècniques de referència de l'estat de l'art, per comprovar la implementació.
- Introducció de nous algorismes per la millora del Front-End de la tècnica de registre, és a dir, proposar nous mètodes per aprendre un Gaussian Mixture Model a partir d'un núvol de punts.

1.3 Entorn de desenvolupament

El projecte es realitza en l'entorn de l'Institut VICORB, concretament en el Centre d'Investigació en Robòtica Submarina ubicat al Parc Tecnològic i Científic de la UdG. Aquest complex, un dels més avançats en la robòtica submarina a escala europea, està compost per dos edificis. Per una banda, el primer edifici allotja oficines i taller. Per altra banda, el segon edifici disposa d'una sèrie d'instal·lacions ideals per al testatge dels robots: un tanc de proves de 6x12 metres i 5 metres de profunditat, un pont grua per a la immersió i recuperació i una sala de control situada per sota del nivell de l'aigua que permet una visió directa de l'interior del tanc a través d'una gran finestra subaquàtica.

El laboratori compta amb tres AUVs operatius desenvolupats íntegrament pel grup de recerca. Aquests robots disposen d'una àmplia gamma de sensors i equipaments d'última generació, tals com unitats per a la navegació (DVL, sensor de pressió, INS, GPS), càmeres d'alta definició, mòdems acústics i també sonars d'imatges, d'escombrats laterals o de rang.

Per a la realització d'experiments al mar, el grup disposa d'una embarcació de 7 metres d'eslora equipada amb un receptor GPS-RTK, AHRS i USBL. A més de tot això, el laboratori compta amb tècnics altament capacitats per donar suport a doctorats, investigadors i estudiants per realitzar la preparació dels robots i ajudar en la realització d'experiments al mar.

Capítol 2

Viabilitat

2.1 Paràmetres necessaris per al desenvolupament del projecte

Per al desenvolupament d'aquest projecte és necessari disposar, per una banda, d'un ordinador amb sistema operatiu Ubuntu que sigui força potent per als diversos càlculs que es realitzaran. Per altra banda, cal tenir tot el sistema referent al submarí i enregistrament de les dades.

A part d'això, cal un sistema de connexió remot entre ordinador i submarí. Per això, és necessari disposar d'un encaminador per a donar cobertura en una àrea força amplia, com més amplia millor.

2.2 Costos dels paràmetres necessaris

Segons els paràmetres explicats anteriorment el cost serà més o menys elevats segons les especificacions i les diferents opcions per a cada paràmetre. Per a l'encaminador cal que tingui un bon senyal i capacitat de transmetre i rebre com més dades millor, que tingui banda 5 GHz és opcional, però molt recomanable, la banda necessària per a l'encaminador és la de 2,4 GHz. En el mercat actual un bon router que compleixi amb les nostres necessitats i que sigui ben valorat, pot valdre entre 30 a 150 euros.

L'ordinador haurà de complir diverses especificacions. Per a la memòria RAM es necessita que tingui entre 8 i 16 GB. En el nostre cas, com es farà en el sistema operatiu Ubuntu i les dades a mostrar no són molt pesades, no cal que incorporem cap targeta gràfica. Com a processador, un Intel i7 o un AMD Ryzen 7 serien el millor. En cas que per disponibilitat no fos possible trobar cap dels dos, un Intel i7 o AMD Ryzen 5 serien els substituts. Processadors que tinguin menor capacitat potser serien justos en algun moment de l'execució. Si s'opta per comprar un ordinador que ja vingui muntat amb aquests processadors, la memòria RAM i tots els altres components necessaris per a l'ordinador, el preu podria variar entre uns 600-700€ fins a uns 1400-1500€.

Malauradament, no es disposa d'un preu o un rang final per al conjunt de components que formen el submarí que es fa servir al CIRS (figura 2.1). No

obstant, si s'hagués de pagar el seu valor sense cap ajut, seria impossible de dur a terme aquest projecte per a un estudiant de grau normal.

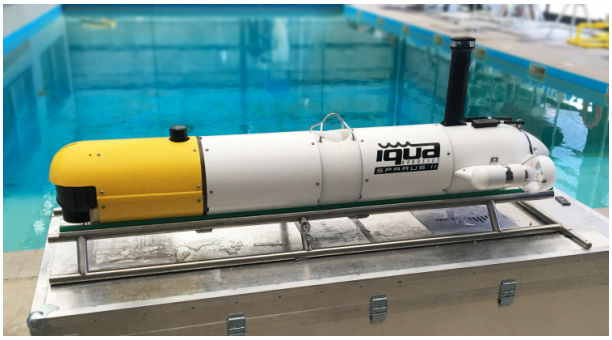

Figura 2.1: Exemple d'un AUV SPARUS II.

2.3 Costos de desenvolupament

Un cop definits els paràmetres necessaris per a poder iniciar el projecte, es pot tenir en compte el cost que suposaria per a una empresa, el fet de contractar un treballador per a fer aquesta feina.

Serà necessari un enginyer informàtic que porti a terme totes les tasques que s'han fet en aquest projecte. Aquesta persona ha d'estar cinc mesos treballant dedicant com a mínim quatre hores cada dia, excepte dels dies festius que hi hagi. Si el treballador cobra uns 12 € l'hora, el salari brut total per a aquesta persona seria d'uns 5000 €, tenint en compte que l'empresa hauria de pagar una mica més per contractar a aquesta persona a l'empresa.

Capítol 3

Metodologia

Per a aquest projecte s'ha seguit una metodologia de desenvolupament de tipus àgil, on per cada una o dues setmanes es defineix com un sprint per a cada funcionalitat. S'ha realitzat d'una forma iterativa i incremental, on cada versió nova que es genera millora la versió anterior del projecte, refinant i millorant les diferents parts i on alhora es van generant noves característiques.

Com que el projecte es va començar al gener, només es disposaven de quatre mesos per a la realització del projecte, i hi havia moltes tasques a fer: estudi previ de la tècnica, desenvolupament de les interfícies, documentació, implementació i ús d'altres tècniques en l'estat de l'art, etc., es va fer un desglossament en etapes i una posterior separació de la feina de cada etapa a fer fent servir aquest tipus de metodologia.

El desenvolupament d'aquest projecte és particular: hi ha una part de desenvolupament conjunt amb un altre estudiant i una part de desenvolupament individual. La part comuna s'ha desenvolupat amb l'estudiant Miguel Malagón, que realitza un projecte amb la mateixa particularitat. Ambdós projectes es basen en la implementació de la mateixa llibreria per després fer les evolucions pertinents per a la realització de cada projecte. S'ha decidit de fer la part base de la llibreria conjuntament per trencar el gel en la temàtica amb companyia i alleugerir la càrrega de feina evitant haver de fer dues vegades la mateixa implementació. Alhora, fer el disseny de la llibreria conjuntament, ha permès acordar-lo i desenvolupar els dos projectes sobre la mateixa idea. Per aquest motiu els projectes s'han desglossat en dues parts, una part és la part comuna de familiarització, disseny i implementació de la part base de la llibreria i, una segona part, per al treball individual d'experimentació i ampliació corresponents.

Per a la part comuna s'ha fet una separació en diferents etapes. De cada etapa es mira al principi que és el que hauria d'haver-hi i que caldria fer. A partir d'aquí es comenta amb el professorat per veure si és correcte el plantejament i, posteriorment, es reparteix la feina a fer. En la part individual també s'ha realitzat una divisió de la feina en etapes. S'ha definit en etapes incrementals, començant per l'aprenentatge d'altres tècniques de l'estat de l'art, passant a una primera implementació amb aquestes tècniques fins a arribar a fer experiments, proves i comparacions amb la tècnica implementada

en aquest projecte. Per últim, el redactat dels resultats obtinguts en fer la comparativa de les tècniques.

Capítol 4

Planificació

4.1 Estratègia per al desenvolupament

Tenint en compte la metodologia explicada al capítol 3, es va fer una primera reunió d'inici del projecte. En aquesta reunió es va fer la planificació de la feina per al projecte. Es van desglossar les tasques de desenvolupament principals per a la part conjunta per realitzar els tres tipus d'objectes en diferents terminis. Posteriorment, es duria a terme una documentació parcial de la part conjunta i es començaria a fer la part única de cada estudiant (figura 4.1 sobre la planificació).

4.2 Planificació tasques

A la figura 4.1 es mostra el diagrama de Gantt de la planificació del projecte on s'hi mostra la planificació organitzada en setmanes i tasques. De les tasques que es veuen les que componen la part comuna del projecte són les etapes de la 1 fins a la 7, començant per a l'aprenentatge de la tècnica i una primera implementació, passant per al desenvolupament de les diferents components de la llibreria, fins a arribar a la documentació de tot el que es va fer en la part conjunta del projecte.

Les tasques que van a continuació d'aquestes, són les etapes en què s'ha dividit la part individual. En la part individual es realitza l'aprenentatge i ús d'altres tècniques en l'estat de l'art (ICP i GICP). Duent a terme al final una comparativa i una obtenció de resultats d'aquestes tècniques amb la que s'ha implementat en la llibreria. Després de fer la comparativa, s'han afegit les etapes que tracten dels nous constructors amb K-means i Expectation Maximization.

4.3 Comentaris respecte a la planificació

Ja des d'un principi, va haver-hi canvis respecte a la planificació original, l'etapa inicial d'aprenentatge amb una primera implementació es va allargar una setmana més. Com que era una tècnica nova amb la qual no s'havia

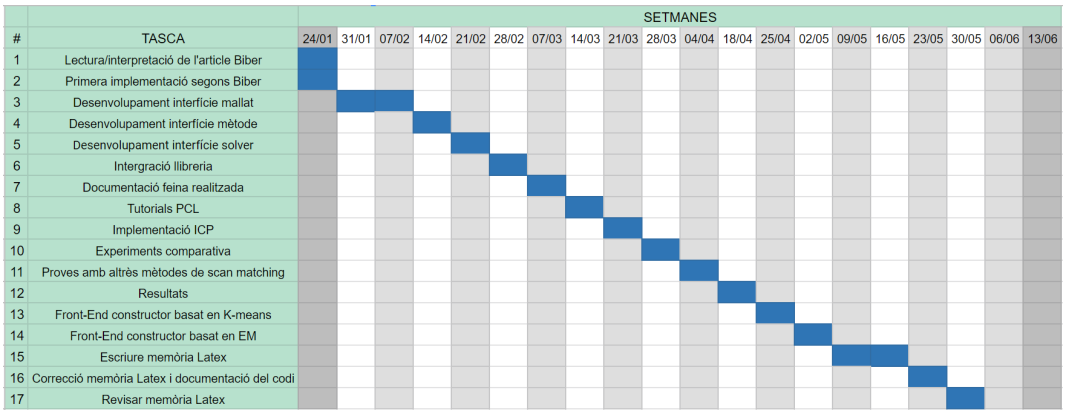

Figura 4.1: Planificació inicial per al projecte.

treballat anteriorment, va ser més costos a l'hora d'entendre com funcionava i es va decidir allargar aquesta etapa.

El desenvolupament d'aquest projecte no ha estat diferent que d'altres projectes, hi ha hagut diversos problemes al llarg de tot el projecte que han afectat la planificació. Per dificultats amb diverses d'aquestes tasques, diferents etapes van portar més temps de l'esperat, per la qual cosa, totes les etapes següents es van veure afectades, arribant així a què hi hagués múltiples etapes en una mateixa setmana.

Capítol 5

Marc de treball i conceptes previ

L'entorn submarí és un domini complex i desfavorable per a les aplicacions robòtiques, ja que la majoria de sistemes de percepció i comunicació utilitzats en l'àmbit terrestre no hi funcionen. L'aigua actua com un gran atenuador de les ones electromagnètiques i, en conseqüència, els sistemes de comunicació Wi-Fi o de localització GPS no hi funcionen. Alhora, el camp de visió de les càmeres òptiques no arriba més enllà dels pocs metres si no hi ha partícules en suspensió a l'aigua, cosa també difícil de garantir. A més, si es volen fer servir sensors làsers subaquàtics - que encara es troben en fase de desenvolupament - s'han d'emprar feixos molt potents que són molt perillosos per a qualsevol humà que hi estigui a prop. Per tant, sota l'aigua cal recórrer a fenòmens acústics per construir sistemes de comunicació i percepció, ja que les ones mecàniques sí que s'hi propaguen bé. No obstant això, les ones mecàniques són ordres de magnitud més lentes que les ones electromagnètiques i, en conseqüència, proporcionen menys dades i molt més sorolloses. En conseqüència, els algoritmes que les processen han de ser més avançats, amb una robustesa i complexitat superiors.

Els Vehicles Autònoms Submarins (AUV) són plataformes robòtiques dotades de sistemes de percepció i control que operen sota l'aigua i que permeten efectuar tasques de localització, mapatge o planificació de trajectòria de manera totalment autònoma, és a dir, sense interacció amb cap altre sistema. Per tant, un cop definida la missió a realitzar, el robot se submergeix i la intervenció humana es limita només a la supervisió de què el sistema del robot es mantingui en funcionament i, si no, anar-lo a recollir quan per flotabilitat arribi a la superfície. Actualment, els AUVs són una tecnologia a l'alça que ha de permetre avançar en l'exploració del fons marí i en el manteniment autònom de nombroses infraestructures submarines. Tots els estudis mostren un interès creixent per aquest tipus de tecnologia i la necessitat d'aquests robots per moltes aplicacions marines: des de l'exploració científica del fons marí per interessos biològics, geològics o arqueològics; fins a la inspecció de plataformes petrolieres, parcs eòlics, gasoductes o línies elèctriques.

En l'actualitat els AUVs comercials són àmpliament utilitzats per a l'exploració a mar obert. En aquest mode de funcionament, els robots segueixen trajectòries predefinides adaptant només la profunditat per mantenir una alçada constant respecte el fons marí, la qual cosa es detecta amb sensors

de rang acústic. Tanmateix, és necessari dotar aquests vehicles de capacitats avançades que els permetin desenvolupar tasques més intel·ligents, com l'exploració o la inspecció de determinades zones que tenen una estructura tridimensional complexa i que no puguin ser resoltes amb trajectòries predefinides. Aquest tipus d'entorns tenen un gran interès científic i industrial, encara que actualment només són accessibles per submarinistes professionals o per Vehicles Operats Remotament (ROV) que fan servir un cable umbilical.

El mapatge de zones submarines emprant sensors de rang acústic muntats en AUVs actualment està limitat per l'acumulació de deriva durant la navegació del robot amb sensors inercials. La deriva provoca que quan es revisita un espai ja explorat les vistes no concordin i el mapa resultant sigui incoherent. Per evitar-ho, cal dotar el robot d'un sistema d'associació de vistes que permeti determinar el desplaçament del robot que registra les dues vistes. El resultat del registre es pot utilitzar en una aplicació de Mapeig i Localització Simultanis (SLAM) ([Durrant-Whyte 2006] i [Bailey 2006]) que, combinat amb la navegació a la deriva, permeti determinar la localització del robot en un mapa coherent del seu entorn construït a partir de les dades del sensor de rang. El problema de SLAM és un problema fonamental dins la comunitat de la robòtica mòbil. En aquest treball es vol avançar en aquest problema implementant una eina de registre de núvols de punts especialitzada per núvols de punts adquirits amb sensors de rang acústics. Aquests núvols de punts tenen la propietat de ser poc densos - és a dir, d'estar formats per un nombre de punts reduït - i d'estar contaminats per molt de soroll procedent del sensor.

5.1 El problema de registre

Donat un robot en moviment equipat amb un sensor de rang que a l'instant t_0 observa un obstacle i a l'instant t_1 observa el mateix obstacle des d'una altra posició a l'espai; el problema de registre vol determinar quin és el desplaçament del robot entre t_0 i t_1 a partir de les observacions de l'entorn. A la figura 5.1 esquerra es pot veure que els núvols de punts captats per un sensor de rang no quadren si s'associen segons el resultat de la navegació a la deriva del robot, és a dir, a partir de la integració de les mesures dels sensors inercials del robot. En canvi, a la figura 5.1 dreta es pot veure com aplicant un algorisme de registre les vistes quadren i es pot generar un mapa coherent de l'entorn del robot. Per tant, el problema de registre determina la transformada ${}^{k}T_{k+1}$ entre les posicions consecutives k i $k+1$ del robot.

Els algorismes de registre es poden classificar en dues categories. Per una banda, la categoria dels mètodes punt a punt inclou tots aquells mètodes basats en l'algorisme del Punt més Proper Iteratiu [ICP 022]. Es tracta de

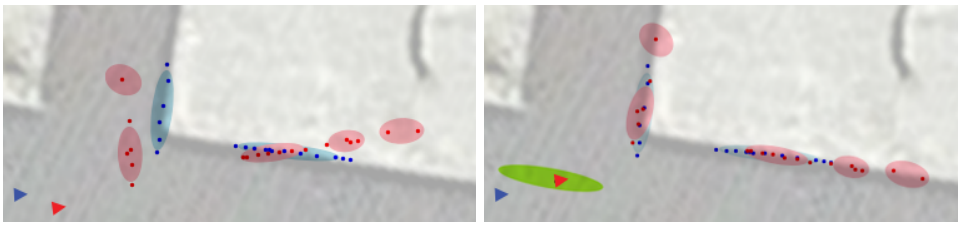

Figura 5.1: Exemple del problema de registre amb dades d'un Mechanical Scanning Profiling Sonar muntat en un AUV captades a l'escullera del Port de Sant Feliu de Guíxols. Esquerra: Associació de vistes a partir del sistema de navegació a la deriva. En blau el núvol de punts de referència i en vermell el núvol de punts mòbil. Dreta: Vistes registrades.

mètodes iteratius on en cada iteració s'associa cada punt de l'escaneig mòbil al punt més proper de l'escaneig de referència i es busca la transformació entre els escanejos que minimitza la distància euclidiana. Quan les associacions no canvien o la distància no millora, l'algorisme ha convergit.

Per altra banda, la categoria dels mètodes basats en camps inclou tots aquells mètodes on almenys algun dels dos núvols de punts s'expressa mitjançant un Model de Mescles de Gaussianes (GMM). Un GMM és un model molt emprat per la comunitat de l'aprenentatge automàtic per classificar dades de manera no supervisada que té la següent forma:

$$p(x|\theta) = \sum^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \qquad \text{amb} \qquad \sum^{K} \pi_k = 1,$$

on $\mathcal{N}(x|\mu_k, \Sigma_k)$ és un model Gaussià amb mitjana μ_k i covariància Σ_k i π_k és el pes de la component gaussiana k per assegurar que la suma de totes les distribucions integra a u. Existeixen diverses tècniques per fitar un GMM a un núvol de punts, a partir d'ara anomenades Front-End. Un dels Front-Ends més bàsics consisteix a generar una component gaussiana per cada punt de l'escaneig i donar a totes les components la mateixa covariància. Aquesta tècnica l'utilitza l'algorisme ICP Generalitzat [Generalized-ICP 022].

Dins de la categoria de mètodes basats en camps podem distingir entre els mètodes Punt a Distribució (P2D) i Distribució a Distribució (D2D). Els mètodes P2D busquen la solució de Maximum Likelihood de l'escaneig mòbil $\mathcal{M}$ expressat en forma de núvol de punts i l'escaneig fixe $\mathcal{F}$ modelat per un GMM:

$$p(\mathcal{D}|\theta) = \prod^{\mathcal{D}} p_{\mathcal{F}}(x = R(\phi)q_d + t|\pi, \mu, \Sigma) = \prod^{\mathcal{D}} \sum^{K} \pi_k \mathcal{N}(x = R(\phi)q_d + t|\mu_k, \Sigma_k),$$

on $p_{\mathcal{F}}(x|\pi,\mu,\Sigma)$ modela l'escaneig fixe, $\mathcal{D} = \{q_1,\ldots,q_n\}$ és el núvol de punts mòbil i $\theta = (t,\phi)$ és la rotació i la translació que es volen determinar. Es tracta d'un mètode costós computacionalment, ja que cal avaluar el model de $\mathcal{M}$ per cada punt de $\mathcal{F}$.

Els mètodes D2D minimitzen la distància $\mathcal{L}^2$ entre dos núvols de punts modelats amb un GMM. La distància $\mathcal{L}^2$ entre dos GMM té la forma analítica derivada a [Jiang 2011] que permet expressar el problema com:

$$\theta^* = \arg\max_{\theta} \mathcal{L}^2(\theta) = \arg\max_{\theta} \sum_{i=1}^{n_{\mathcal{F}}} \sum_{j=1}^{n_{\mathcal{F}}} \pi_{\mathcal{F}i}\pi_{\mathcal{M}j}\mathcal{N}(0|\mu_{\mathcal{F}i} - R(\phi)\mu_{\mathcal{M}i} - t, \Sigma_{\mathcal{F}i} + R(\phi)\Sigma_{\mathcal{M}i}R(\phi)^T),$$

on $p_{\mathcal{F}}(x|\pi,\mu,\Sigma)$ modela l'escaneig fixe, $p_{\mathcal{M}}(x|\pi,\mu,\Sigma)$ modela l'escaneig mòbil i $\theta = (t,\phi)$ és la rotació i la translació que es volen determinar. Es tracta d'un mètode menys costós computacionalement, ja que modelar els dos escanejos per mitjà de GMMs permet comprimir les dades.

5.2 La Transformada de Distribucions Normals (NDT)

La tècnica de registre coneguda com a Transformada de Distribucions Normals (NDT) va ser proposada per primer cap a [Biber 2003]. Es tracta d'una tècnica de registre de núvols de punts basada en camps que primer es va plantejar com a P2D, donant lloc a la metodologia NDT-P2D. Més endavant, a [Stoyanov 2012] es va formular el problema D2D, donant lloc a la metodologia D2D.

El tret distintiu i característic de totes les metodologies NDT és el seu Front-End. A diferència de la tècnica GICP anteriorment presentada, la tècnica NDT té per objectiu comprimir el núvol de punts complint $N << K$, on N és la dimensió del núvol de punts i K és el nombre de components del GMM. Per fer-ho, es genera una graella cartesiana uniforme sobre el núvol de punts i s'assigna cada punt a una casella. Per cada casella amb un nombre suficient de punts p_{min} es genera una component k:

$$\begin{aligned}
\mu_k &= \frac{1}{N_k}\sum^{N} x_n, \\
\Sigma_k &= \frac{1}{N_k}\sum^{N} (x_n - \mu_k)(x_n - \mu_k)^T, \\
\pi_k &= \frac{N_k}{N},
\end{aligned}$$

on $X = (x_0, \ldots, x_{N_k})$ són els punts de l'escaneig associats a la cel·la k.

En aquest treball es volen proposar altres metodologies per al Front-End provinents de la comunitat de l'aprenentatge automàtic que permetin generar les components del GMM tenint en compte l'estructura implícita a l'escaneig, en lloc d'imposar una graella cartesiana.

Com que les funcions objectives dels problemes d'optimització que defineixen les metodologies P2D i D2D són contínues i diferenciables, la tècnica NDT suggereix resoldre'ls aplicant el mètode de Newton (algorisme 10.1 de [Bierlaire 2015]). Es tracta d'un mètode iteratiu definit a partir de l'aproximació d'ordre 2 de la Sèrie de Taylor que requereix el càlcul de la primera i la segona derivada de la funció objectiu. Encara que l'avaluació de la segona derivada de la funció incrementi la complexitat del càlcul, la seva convergència és major que pels algorismes d'una sola derivada.

5.3 Algorisme K-means

L'algorisme de classificació anomenat K-means (cap 9.1 [Bishop 2006]) té com a objectiu la classificació de conjunts de N punts en K grups, on cada punt s'assigna només al grup més proper segons la distància euclídia. Aquesta classificació de les dades pot servir per fitar-hi un GMM, establint una component gaussiana per cada classe.

L'algorisme de K-means comença assignant de manera aleatòria cada punt del núvol de punts $\{x_1, ..., x_N\}$ a una de les K components. Feta l'assignació comença un procés iteratiu d'expectació i maximització. Donades les assignacions de punts a cada component, s'estableix com a centroïde de la component k la mitjana dels punts assignats. Donats nous centroïdes, es generen noves assignacions assignant cada punt a la component amb el centroïde més proper segons la distància euclídia. Aquest procediment es va repetint fins que les assignacions s'estabilitzen.

Les assignacions punt a component es descriuen amb una variable binària, anomenada variable latent, $r_{nk} \in \{0, 1\}$, on $k = 1, ..., K$, que descriu en quina de les k components el punt x_n s'assigna.

Es defineix una mesura J, anomenada inèrcia,

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2 \tag{5.1}$$

que representa la suma de les distàncies quadrades de cada punt x_n amb el seu centroïde μ_k. L'algorisme K-means minimitza la inèrcia del data set determinant r_{nk} i μ_k.

Per generar assignacions r_{nk} donats uns centroïdes μ_k s'aplica:

$$r_{nk} = \begin{cases} 1, & \text{if } d_E = argmin_k \, ||x_n - \mu_k|| \\ 0, & \text{otherwise.} \end{cases} \tag{5.2}$$

Per calcular els centroïdes μ_k donades unes assignacions r_{nk} s'aplica l'estimador de "minimum likelihood"per una distribució normal:

$$\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}. \tag{5.3}$$

El denominador d'aquesta expressió és igual al nombre d'observacions assignades a la component k. Per tant, el centroïde μ_k és igual a la mitjana de cada punt x_n que han estat assignats a la componet k. Per aquest motiu, aquesta tècnica és coneix com l'algoritme de K-means (K-mitjanes).

5.4 Algorisme d'Expectació-Maximització

L'algorisme d'Expectació-Maximització (cap 9.2.2 [Bishop 2006]) és una tècnica iterativa que s'utilitza per optimitzar els paràmetres de models probabilístics que depenen de variables que no es poden observar, com seria el cas d'un GMM.

Aquest algoritme alterna iterativament una etapa d'expectació (E), on es determina l'esperança del likelihood en base als valors actuals dels paràmetres, i una etapa de maximització (M), on es determinen els paràmetres del model maximitzant el likelihood assignat en l'etapa d'expectació (E). Determinats els paràmetres del model comença una nova etapa d'expectació.

Per fitar un GMM, a l'etapa E es determinen les variables latents del model $z_{nk} \in (0, 1)$, on $k = 1, ..., K$ i $n = 1, ..., N$, anomenades responsabilitats:

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_j \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)} \tag{5.4}$$

Aquesta equació se separa en les dues fórmules següents, una fa el càlcul de punt amb component i l'altre engloba la suma d'aquesta primera operació aplicada a cada component.

$$\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k) \tag{5.5}$$

$$\sum_j \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j) \tag{5.6}$$

A diferència de l'algorisme de k-means, en lloc d'assignar cada punt a una sola component, l'algorisme EM determina quina responsabilitat té cada punt n en cada component k.

A l'etapa M es determinen els paràmetres del GMM donades les responsabilitats. Per fer-ho s'aplica l'estimador de 'maximum likelihood' per a un GMM, que té la següent forma

$$\begin{aligned}
\pi_k &= \frac{N_k}{N}, \\
\mu_k &= \frac{1}{N_k}\sum_{n=1}^{N}\gamma(r_{nk})x_n \\
\Sigma_k &= \frac{1}{N_k}\sum_{n=1}^{N}\gamma(r_{nk})(x_n-\mu_k)(x_n-\mu_k)^T, \\
N_k &= \sum_{n=1}^{N}\gamma(r_{nk}).
\end{aligned}$$

on π_k és el pes de la component k, μ_k el seu centroide, Σ_k la seva covariança i n el nombre de punts assignats.

Per mirar el log likelihood de les assignacions que s'han trobat en una iteració, s'ha de realitzar l'equació següent:

$$lnp(X|\pi,\mu,\Sigma) = \sum_{n=1}^{N}\ln\{\sum_{k=1}^{K}\pi_k\mathcal{N}(x_n|\mu_k,\Sigma_k)\} \tag{5.7}$$

Capítol 6

Requisits del sistema

6.1 Requisits funcionals

Aquesta secció conté tots els requisits funcionals que ha de tenir la nostra llibreria de cares a fer-la publica per a altres investigadors:

- Programació en C++
- Recepció de dades
- Extracció dels núvols de punts (2D o 3D)
- Extracció de l'odometria
- Creació d'objectes de Gaussian Mixtures Model
- Creació de Scan Matching Methods amb distribució P2D o D2D
- Calcul score, primera i segona derivades amb Scan Matchin Method
- Creació de Solvers amb Newton Method o LineaSearch Newton Method
- Calcul de l'optim amb el Solver

6.2 Requisits no funcionals

Aquesta secció conté tots els requisits no funcionals de la nostra llibreria:

- Llibreria del Front-End
- Plots de Gaussian Mixtures Models
- Plots d'objectes Scan Matching Mehods
- Plots d'objectes Solver
- Paràmetres com a shared pointers o constants

Capítol 7

Estudi i decisions

Al llarg del desenvolupament del projecte s'han fet servir diversos softwares i programes. En alguns casos s'han utilitzat de principi a fi, mentre que en altres, han estat descartades posteriorment per complexitat o per incompatibilitat amb altres eines o amb el mateix sistema.

7.1 Descripció del maquinari

Com s'ha especificat anteriorment en el capítol 2, la màquina en la qual s'han de fer les operacions cal que sigui força potent per a la part de processadors i memòria. Per a la part de la memòria RAM s'ha optat per un ordinador amb una capacitat de 16 GB, de processador s'ha fet servir un Intel-i7, tot i que es podria haver fet servir un Intel-i5 o el corresponent processador d'AMD. Com que en el nostre cas s'ha utilitzat el sistema operatiu Ubuntu i les dades a mostrar no han estat molt pesades, no s'ha incorporat cap targeta gràfica.

7.2 Programari per al desenvolupament

Per escriure codi i gestionar les versions amb més comoditat han calgut dos softwares bàsics. Un és el Visual Studio Code, com a editor de codi, i l'altre és la plataforma Bitbucket, com a gestió de versions Git.

El Visual Studio Code [VisualSC 022] s'ha fet servir tant per l'edició de codi com per pujar al repositori les noves modificacions. Es tracta d'un editor de codi font que ha estat construit sobre el framework anomenat Electron [Electron 022]. Aquest editor disposa de diferents opcions i terminals per a la realització de comandes Git del projecte.

Bitbucket [Bitbucket] és una eina d'allotjament de codi i col·laboració per a equips basada en Git [Git 022]. Amb aquesta eina s'ha generat un repositori on s'ha guardat tant el codi de la branca principal com totes les altres branques que s'han generat per a la realització de les tasques.

7.3 Llibreries i Frameworks emprats

7.3.1 Llibreries

Pel desenvolupament d'aquest projecte s'han emprat tres llibreries diferents: la llibreria Eigen [Eigen], la lliberia Matplotlib per a C++ [Matplotlib 022] i la Point Cloud Library [PCL 022]. La llibreria Eigen [Eigen] s'ha fet servir des de el principi del projecte, ja que permet definir objectes matricials i realitzar operacions matricials en C++. És l'equivalent en C++ de la llibreria NumPy de Python [Python 022]. Eigen és una llibreria en C++ que es troba templaritzada per a l'àlgebra lineal. Permet definir matrius i vectors i disposa de solvers numèrics i molts algoritmes de l'àlgebra lineal.

La llibreria de Matplotlib [Matplotlib 022] per a ús en C++ és una llibreria creada per poder fer servir la llibreria Matplotlib nativa del llenguatge Python en C++, amb l'objectiu de poder visualitzar dades. El problema d'aquesta llibreria és que els objectes escrits en C++ són limitats i per moltes tasques encara cal recòrrer a la versió de Python.

La Point Cloud Library [PCL 022] és una llibreria pública amb diferents algoritmes utilitzats en el processament de núvols de punts i processament de geometria 3D. És una llibreria de referència per les comunitats de percepció aplicada a la robòtica mòbil.

7.3.2 Frameworks

Pel desenvolupament d'aquest projecte s'han fet servir dos frameworks: un és ROS i l'altre és Boost. El Robotic Operation System (ROS) [ROS] és un framework, on s'agrupen un conjunt d'eines, llibreries i convencions informàtiques que tenen com a objectiu fer que sigui més simple la creació i programació amb robots.

El Boost [Boost 022] és un framework creat per a estendre les capacitats del llenguatge de programació C++. Aquest conjunt de llibreries aporta ajuda a l'hora de dur a terme tasques i estructures tals com l'àlgebra lineal, generació de nombres aleatoris, multiprocessament, etc.

7.4 Programari descartat

En aquesta secció es nombren els diferents softwares que han estat descartats durant el desenvolupament del projecte. Com a editor de codi també hi havien les opcions de Gedit [Gedit 022], Atom [Atom 022] i Sublime Text [SublimeText 022]. No obstant, com que el Visual Studio Code [VisualSC 022]

també permet l'ús del terminal i la realització de comandes Git, les altres van ser descartades.

Per a mostrar les dades en plots es varen provar varis softwares, entre ells openGL [OpenGL 022], Gnuplot [Gnuplot 022] i Matlab [Matlab 022]. No obstant, per problemes de compatibilitat, ja sigui amb el mateix repositori amb softwares que ja es feien servir, van ser rebutjats.

Capítol 8

Anàlisi i disseny del sistema

8.1 Anàlisi del problema

Després de fer una primera implementació senzilla de la tècnica sense la utilització de classes, es va observar que la tècnica es pot dividir en tres parts relacionades entre elles. Per una banda, s'han de fitar un Gaussian Mixtures Models (GMM) a un núvol de punts. Per una altra, s'han d'establir les funcions que defineixen el problema de registre com un problema d'optimització i, finalment, solucionar el problema d'optimització.

La primera part és el que anomenem Front-End, el qual donat un núvol de punts fita un GMM. Un GMM és un model probabilístic format per un sumatori de distribucions gaussianes tal com s'observa a l'equació 5.1. Aquestes distribucions, anomenades components k, estan definides per una mitjana μ_k, una matriu de covariància Σ_k i un pes π_k; on la suma dels pesos de totes les components ha de ser 1. Per tant, el Front-End permet passar d'una representació discreta de la superfície detectada a un model continu, a més de fer una compressió de les dades.

La segona part és el mètode de registre. Aquest defineix el problema de registre com un problema d'optimització, definint variables de decisió - increment de pose entre els dos scans a registrar - i funció de cost - que poden definir els dos problemes plantejats a la secció 5.1: Punt a Distribució (P2D) i Distribució a Distribució (D2D) -. Per una banda, el mètode P2D registra un scan parametritzat com a núvol de punts contra un scan parametritzat amb un GMM. Per tant, la funció de cost maximitza el likelihood del núvol de punts sobre el model avaluant tots els punts del núvol a totes les components del GMM. Per altra banda, el mètode D2D registra dos scans parametritzats amb GMM. Per tant, la funció de cost defineix la divergència entre les dues distribucions per minimitzar-la.

Per últim, el solucionador utilitza mètodes numèrics per resoldre el problema d'optimització definit pel mètode de registre. En aquest treball ens centrem en mètodes del gradient, és a dir, mètodes basats en les derivades de la funció de cost.

8.2 Disseny del sistema

L'element base de la llibreria és la classe GaussianMixtureModel que permet emmagatzemar un GMM i inclou mètodes de visualització. El Front-End s'ha implementat com una llibreria de funcions, les quals actuen com constructors de la classe GaussianMixturesModel, donant un núvol i certs paràmetres d'entrada per obtenir un GMM de sortida. Al no tenir una única tècnica per fitar un GMM a un núvol de punts, s'ha decantat per crear aquesta llibreria. Altrament, els constructors que tinguin els mateixos tipus d'argument, s'haurien de diferenciar amb un paràmetre extra que especifiqui quina tècnica es vol utilitzar. Generant un constructor que crida altres constructors segons el valor d'aquest paràmetre, complicaria innecessàriament la creació d'un GMM. En canvi, utilitzant la llibreria podem donar diferents noms als constructors sense complicar-los.

Els mètodes de registre s'han implementat amb una interfície i un seguit de subclasses. Això ens permet utilitzar les funcions definides per la interfície que estan implementades en cada subclasse sense haver de saber quina és la subclasse, el que ens permet donar al solucionador un punter a un objecte del tipus de la interfície i que aquest utilitzi les funcions implementades a les subclasses. Les funcions que utilitza el solucionador són compute_score(), compute_score_and_gradient() i compute_score_gradient_and_hessian(). Gràcies a la interfície, el solucionador funciona amb qualsevol de les subclasses sense la necessitat d'implementar un solucionador per cada ScanMatchingMethod. La interfície està templaritzada per poder definir subclasses tant en casos bidimensionals com tridimensionals, ja que les diferents funcions depenent fortament de les dimensions de les variables de decisió. Alhora, s'ha decidit que els scans a registrar siguin atributs de la interfície definits al construir la classe. Per tant, per cada registre particular cal construir un mètode particular. D'aquesta manera s'evita haver de passar els scans complerts cada vegada que es vol avaluar el cost o les seves derivades, fent transparent el solucionador dels scans a registrar.

El solucionador, igual que els mètodes de registre, s'ha implementat amb una interfície, ja que es volen fer diferents variacions de solucionadors que es puguin utilitzar de la mateixa manera. Es defineixen unes funcions a implementar en totes les subclasses, com la funció compute_optimum(), que és l'encarregada de fer l'optimització. Aquesta interfície també es troba templaritzada segons la dimensió, però a diferència de l'anterior, les subclasses també ho estan, ja que el mètode de Newton i les modificacions implementades segueixen la mateixa estructura independentment de la dimensió.

L'estructura bàsica del codi per utilitzar la llibreria és la següent. En primer lloc, cal fitar un GMM als scans pertinents que es volen registrar:

```
shared_ptr<GaussianMixturesModel<2>> gmm = ndt_constructor(scan1, 3,
    5);
```

A continuació, es defineix el mètode de registre a utilitzar passant els scans a registrar en la seva forma correcta segons el mètode triat:

```
PointsToDistribution2D p2d = PointsToDistribution2D(gmm,scan2);
shared_ptr<ScanMatchingMethods<3>> method =
    make_shared<PointsToDistribution2D>(p2d);
```

Tot seguit, es defineix el solucionador que es vol aplicar passant el mètode ja construït:

```
LineSearchNewtonMethod<3> lsnm (meth, 0.0001, 0.0001, 100,
    pow(10,-4), 0.99, 2.0, 100);
shared_ptr<Solver<3>> solv =
    make_shared<LineSearchNewtonMethod<3>>(lsnm);
```

Finalment, es resol el problema d'optimització cridant el mètode compute_optimum() del solucionador, passant-li una inicialització per la solució que normalment prové del sistema de navegació a la deriva del robot:

```
solution = solv->compute_optimum(make_tuple(t,h))
```

El solucionador retorna el valor de la transformada que registra els dos scans i, alhora, una matriu de covariança que mesura la incertesa de la solució donada.

El diagrama amb l'estructura final de les classes de la llibreria es mostra a la figura 8.1. S'hi mostren les interfícies i les herències implementades. Pel que fa a la implementació de la llibreria de Front-Ends, en aquest projecte s'han implementat les funcions ndt_constructor(), kmeans_constructor() i em_constructor(). La funció ndt_constructor() està basada en la tècnica NDT, presentada a la secció 5.2. La tècnica NDT clusteritza un núvol de punts projectant una graella sobre el núvol. Per cada cel·la amb un mínim nombre de punts, genera una component al GMM fitant una distribució gaussiana. La funció kmeans_constructor() està basada en l'algorisme de K-means, presentat a la secció 5.3 . Finalment, la funció em_constructor() està basada en l'algorisme d'Expectació-Maximització aplicat per fitar un GMM, descrit a la secció 5.4.

De la interfície ScanMatchingMethods s'han implementat 2 subclasses. La classe Points2Distrtibution2D defineix un registre P2D en dues dimensions, mentre que la classe Distrtibution2Distrtibution2D defineix un regis-

tre D2D bidimensional. Les implementacions s'han separat completament ja que, tot i que la funció de cost només varia entre P2D i D2D, els càlculs de les derivades són completament diferents en el cas bidimensional i en el cas tridimensional. El codi és impossible de templaritzar. Destacar que la classe Distribution2Distribution3D encara no està implementada i forma part del treball futur del projecte.

Finalment, de la interfície Solver s'han implementat 2 subclasses. Totes tenen de base la implementació de la classe NewtonMethod, la qual aplica el mètode de Newton per resoldre un problema de minimització sobre la funció de cost del registre i obtenir la transformació òptima entre els scans. En aquest algorisme, segons les derivades primera i segona de la funció objectius definides a les implementacions de ScanMatchingMethod, a cada iteració es decideix una direcció i un pas per a l'optimització. La classe LineSearchNewtonMethod afegeix al mètode de Newton un algorisme basat en les condicions de Wolfe que determina la mida òptima del pas que defineix el mètode de Newton a cada iteració.

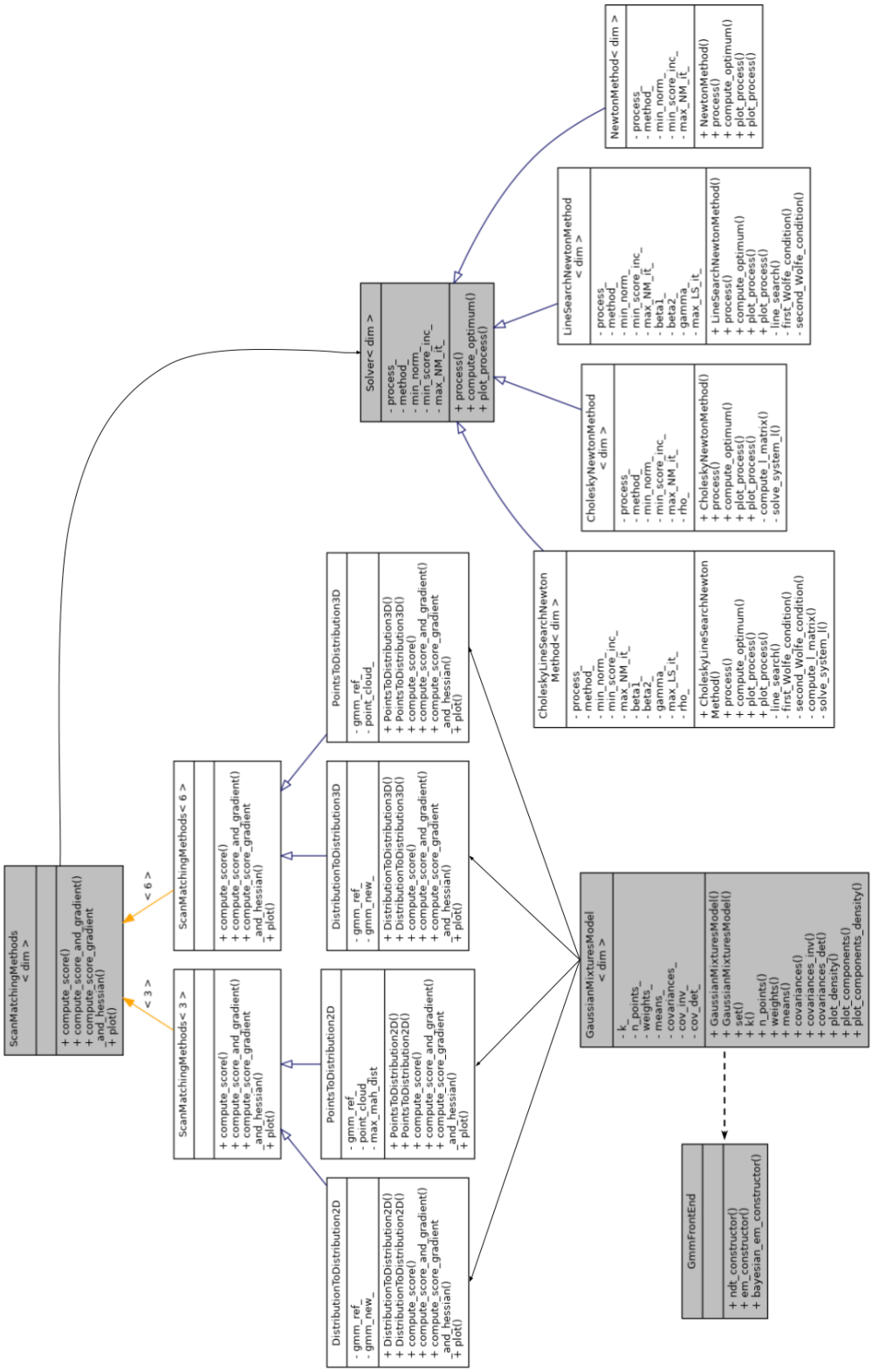

Figura 8.1: Diagrama de classes

Capítol 9

Implementació i proves

Es pot accedir al codi desenvolupat en aquesta proposta i a la seva documentació amb els enllaços que es troben a l'annex 14.

9.1 Aprenentatge inicial

Com s'ha explicat en els capítols anteriors, durant les dues setmanes inicials, es va realitzar una primera versió per a l'aprenentatge de la tècnica de registre. Es va llegir, estudiar i seguir un article científic per a aprendre sobre aquesta tècnica i quina es la seva finalitzat. Els fitxers creats on es realitzaven proves, es van ubicar localment dins d'un ordinador i no es van fer servir o afegir posteriorment a la llibreria.

9.2 Implementació de la llibreria

9.2.1 Objecte Gaussian Mixtures Model

Tota la llibreria es basa en la utilització de Gaussian Mixtures Models, els quals són models probabilístics definits per múltiples distribucions gaussianes. Presentats a la secció 5.1.

Per tant, es va decidir implementar la classe GaussianMixturesModel per emmagatzemar les dades necessaries que conformen un GMM. Un GMM es pot definir amb els paràmetres següents: K número de components del model, $\phi_{i=1...K}$ pes de cada component on la suma de totes ha de ser 1, $\mu_{i=1...K}$ la mitjana de la component i i $\sigma_{i=1...K}$ covariància de la component i. Aquests paràmetres es guarden com atributs de l'objecte de la següent manera:

```
class GaussianMixturesModel{
    private:
        int k_;

        shared_ptr<vector<int>> n_points_;
        shared_ptr<vector<double>> weights_;
        shared_ptr<vector<Eigen::Matrix<double,2,1>>> means_;
```

```
        shared_ptr<vector<Eigen::Matrix<double,2,2>>> covariances_;
```

Els mètodes que utilitzen l'objecte GaussianMixturesModel necessiten de manera recurrent el càlcul del determinant i de la inversa de la matriu de covariància. Per estalviar temps de càlcul, es va decidir precalcular-los en el constructor de la classe i emmagatzemar-los en llistes com la resta d'atributs.

```
        shared_ptr<vector<Eigen::Matrix<double,2,2>>> cov_inv_;
        shared_ptr<vector<double>> cov_det_;
            ...
}
```

Aquests mètodes estan tots implementats per dues dimensions.

Per poder visualitzar els GMM que es generen, s'implementen tres mètodes de plot per representar-los en gràfics. Aquest mètodes són plot_density(), plot_components() i plot_components_density().

```
void plot_density(int index, const int& resolution = 200);
void plot_components(int index, const double& scalar = 4.0);
void plot_components_density(const std::vector<Eigen::Vector2d>&
    scan,int index, const int& resolution = 200, const double& scalar
    = 4.0);
```

El primer, plot_density(), genera una imatge de mida resolution x resolution, on es discretitza per cada pixel el valor de la funció de cost del GMM avaluat en aquell punt seguint l'equació 5.1 , tal i com es veu en el següent tall de codi.

```
for(int i=0; i<resolution; i++){
    for(int j=0; j<resolution; j++){
            ...
        for(int k=0;k<k_;k++){
            img[(resolution-1-j)*resolution + i] += weights_->at(k) *
                ( 1 / ( 2 * M_PI * sqrt(cov_det_->at(k)) ) ) * ( exp(
                -(0.5) * ((punt_actual-means_->at(k)).transpose() *
                cov_inv_->at(k) * (punt_actual-means_->at(k)))(0)) );
        }
    }
}
```

El segon, plot_components(), mostra les mitjanes de cada component i dibuixa una el·lipse al voltant d'aquestes per representar la covariància. Aquesta el·lipse es calcula amb els vectors i valors propis de la covariància i es discretitzen 10 punts que pertanyen a aquesta per fer una línia que els recorri

tots formant així una forma amb el perfil de l'el·lipse.

El tercer, plot_components_density(), genera els gràfics dels dos mètodes anteriors junts en una mateixa figura. D'aquesta manera, es poden comparar més fàcilment les dues imatges.

El la figura 9.1, es veu un exemple dels gràfics generats per les funcions, a l'esquerra es veu la densitat del GMM i a la dreta les components tal com s'ha explicat anteriorment, també s'ha afegit una guia visual a la dreta per veure la graella utilitzada per generar el GMM.

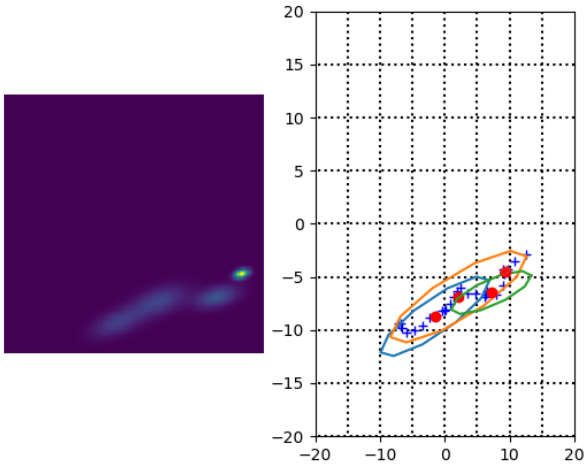

Figura 9.1: Diagrama de classes

9.2.2 Frot-End NDT

A la llibreria de Front-Ends hem definit la funció ndt_constructor() que es basa en la tècnica de les Normal Distribution Transform presentada a la secció 5.2 . És un mètode que agrupa els punts d'un núvol de punts en components projectant una graella cartesiana sobre el núvol. A cada casella amb un mínim nombre de punts li ajusta una distribució gaussiana, generant així múltiples distribuccions que defineixen un GMM.

```
shared_ptr<GaussianMixturesModel<2>> ndt_constructor(const
    std::vector<Eigen::Matrix<double,2,1>>& scan, const int&
    points_threshold, const double& cell_size, const double&
    vaps_min_ratio=-1){
```

La funció ndt_constructor() necessita com a paràmetres: scan que és un núvol de punts que el donem en forma de vector de matrius de 2x1 d'Eigen,

points_threshold és el mínim de punts que ha d'haver en cada component, cell_size el tamany de les caselles quadrades en les que es dividirà la graella en les mateixes unitats que scan, i per últim vaps_min_ratio indica en cas de ser > 0, si s'ha de corregir la covariància en cas de que sigui massa deformada, fent que els valors propis segueixin el ratio donat.

Per fer el control d'aquesta graella s'ha fet ús d'una matriu d'enters que guarda l'index dels vectors on s'emmagatzema la informació de cada component del GMM. D'aquesta manera s'aconsegueix una lookup table que ens indica quina posició dels vectors s'ha d'actualitzar a partir de les coordenades del punt.

Per calcular la covariància amb la fòrmula convencional de l'equació es necessita tenir el valor de la mitjana ja calculada. Per evitar fer una estructura de dades molt gran en memòria, no es guarda cadascun dels punts que corresponen a cada casella de la graella. Aquest fet ens obliga a calcular tant la mitjana com la covariància de manera incremental, les equacions utilitzades són les explicades a la secció 5.2.

Aquest càlcul es fa al codi de la següent manera:

```
for(int i=0;i<scan.size();i++){
        ...
    // actual is the index of the component the point is in
    covariances[actual] += scan[i] * scan[i].transpose();
}

for(int i=0; i<covariances.size();i++){
    covariances[i] = (1.0 / (n_points[i]-1)) * (covariances[i] -
       n_points[i] * ( means[i] * means[i].transpose() ));
           ...
}
```

Un cop obtingudes totes les components del GMM es comprova que totes tinguin el mínim de punts establert i s'eliminen les que no. Al haver eliminat les sobrants es fan la resta de càlculs necessaris, com ara donar el pes de cada component, el qual és el número de punts que hi ha a cada component dividit per el total per així tenir els pesos normalitzats. En aquest punt és on es fa la segona part del càlcul de la covariància tal com s'ha explicat.

A la figura 9.2 veiem la relació entre entre número de punts del núvol original, el numero de components generades i el temps d'execució. Com era d'esperar tots aquests són proporcionals entre si.

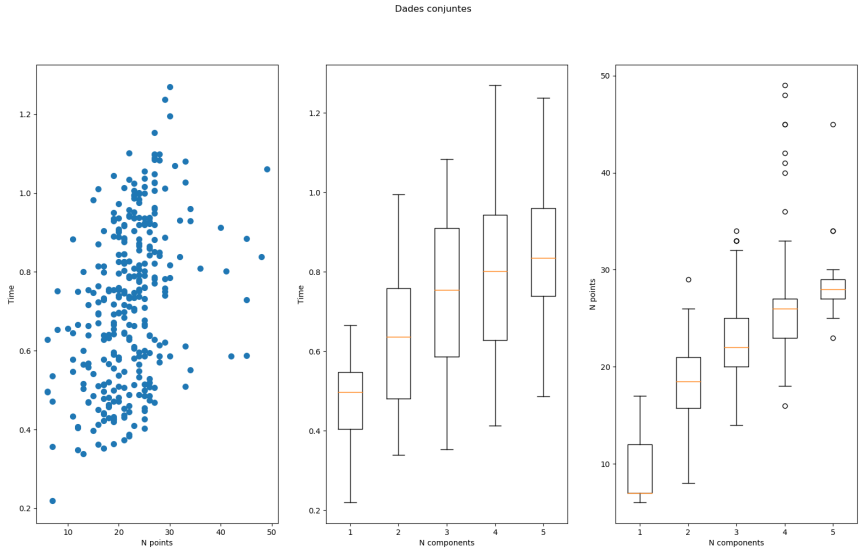

Figura 9.2: Comparació entre número de punts del núvol original, el numero de components generades i el temps d'execució al generar un GMM amb la tècnica NDT

9.2.3 Mètodes de registre

Els mètodes de registre defineixen la funció de cost d'un registre donada una transformació entre els dos núvols de punts. Per solucionar el problema de registre, el mètode també ha de definir la primera derivada per obtenir el gradient que s'utilitzara posteriorment al solucionador i la segona derivada que dona la matriu hessiana utilitzada també en el soluccionador.

A la interfície ScanMatchingMethods es defineixen aquestes funcions que han d'implementar tots els mètodes de registre, compute_score(), compute_score_and_gradient(), compute_score_gradient_and_hessian() i plot().

```
template<int dim>
class ScanMatchingMethods{
    public:
        virtual double compute_score(Eigen::Vector<double,dim> t) = 0;

        virtual tuple<double,Eigen::Vector<double,dim>>
            compute_score_and_gradient (Eigen::Vector<double,dim> t) =
            0;

        virtual tuple<double,Eigen::Vector<double,dim>,
            Eigen::Matrix<double,dim,dim>>
            compute_score_gradient_and_hessian(Eigen::Vector<double,dim>
```

```
            t) = 0;

        virtual void plot(int index, vector<float> process,
            Eigen::Matrix<double,dim,1> t, Eigen::Matrix<double,dim,1>
            optimal)=0;
};
```

Els mètodes per calcular el cost, gradient i hessiana només necessiten la transformació que es vol aplicar al segon registre, ja sigui 2D amb forma $[x, y, rz]$, com 3D amb forma $[x, y, z, rx, ry, rz]$.

9.2.3.1 P2D

Degut a les diferències entre els casos 2D i 3D s'ha decidit no templaritzar la classe i fer objectes concrets pels casos bidimensional i tridimensional simplificant així el codi d'aquests, dels quals només s'ha implementat el 2D de moment.

La primera subclasse d'aquesta interfície és PointsToDistribution2D. En aquest objecte els dos escanejos a registrar són un GMM i un núvol de punts, passats al constructor i que es guarden com atributs de l'objecte.

```
class PointsToDistribution2D : public ScanMatchingMethods<3>{
    private:
        shared_ptr<GaussianMixturesModel<2>> gmm_ref_;
        shared_ptr<vector<Eigen::Vector<double,2>>> point_cloud_;
    public:
        PointsToDistribution2D(const shared_ptr
            <GaussianMixturesModel<2>>& gmm_ref, const shared_ptr
            <vector <Eigen::Vector<double,2>>>& point_cloud);
            ...
}
```

Com ja s'ha explicat, les subclasses de ScanMatchingMethod han d'implementar la funció de cost i les seves derivades.

```
double compute_score(Eigen::Vector<double,3> t);
tuple<double,Eigen::Vector<double,3>>
    compute_score_and_gradient(Eigen::Vector<double,3> t);
tuple<double,Eigen::Vector<double,3>,Eigen::Matrix<double,3,3>>
    compute_score_gradient_and_hessian(Eigen::Vector<double,3> t);
```

En la funció compute_score() es calcula el cost després d'aplicar la transformació t al núvol de punts, tal com es veu en l'equació 9.1, on N es el

numero de punts, K el numero de components,w_j el pes de la component, Σ_j la covariància d'una component, μ_j la mitjana d'una component i p_i un punt x_i del núvol després d'aplicar la transformació t.

$$Score = -\sum_{i=0}^{N}\sum_{j=0}^{K} w_j \frac{1}{2\pi\sqrt{det(\Sigma_j)}} exp(-\frac{1}{2}((p_i-\mu_j)^T\Sigma_j^{-1}(p_i-\mu_j))) \quad (9.1)$$

La funció compute_score_and_gradient() a més de fer el mateix càlcul del cost que l'anterior també calcula la primera derivada de la funció de cost al aplicar la transformació t seguint l'equació:

$$\begin{aligned} Gradient &= (g_0, g_1, g_2)^T, \\ g_0 &= -\textstyle\sum_{i=0}^{N}\sum_{j=0}^{K} A(1,0)^T, \\ g_1 &= -\textstyle\sum_{i=0}^{N}\sum_{j=0}^{K} A(0,1)^T, \\ g_2 &= -\textstyle\sum_{i=0}^{N}\sum_{j=0}^{K} A(-sin(t_2), -cos(t_2); cos(t), -sin(t_2))p_i, \\ A &= Score_{ij}(\mu_j - p_i)^T\Sigma_j^{-1}. \end{aligned} \quad (9.2)$$

La funció compute_score_gradient_and_hessian() caclcula tot l'anterior més la matriu hessiana seguint l'equació: 9.3

$$\begin{aligned} Hessiana &= (h_{00}, h_{01}, h_{02}; h_{01}, h_{11}, h_{12}; h_{02}, h_{12}, h_{22}), \\ h_{00} &= -\textstyle\sum_{i=0}^{N}\sum_{j=0}^{K} Score_{ij}(((\lambda^T\Sigma_j^{-1}(1,0)^T)^2) - (A(1,0)^T)), \\ h_{01} &= -\textstyle\sum_{i=0}^{N}\sum_{j=0}^{K} Score_{ij}(\lambda^T\Sigma_j^{-1}(0,1)^T * \lambda^T\Sigma_j^{-1}(1,0)^T - A(0,1)^T), \\ h_{02} &= -\textstyle\sum_{i=0}^{N}\sum_{j=0}^{K} Score_{ij}(\lambda^T\Sigma_j^{-1}\delta_{rot}x_i\lambda^T\Sigma_j^{-1}(1,0)^T - A\delta_{rot}x_i), \\ h_{11} &= -\textstyle\sum_{i=0}^{N}\sum_{j=0}^{K} Score_{ij}((\lambda^T\Sigma_j^{-1}(0,1)^T)^2 - B(0,1)^T), \\ h_{12} &= -\textstyle\sum_{i=0}^{N}\sum_{j=0}^{K} Score_{ij}(\lambda^T\Sigma_j^{-1}\delta_{rot}x_i\lambda^T\Sigma_j^{-1}(0,1)^T - B\delta_{rot}x_i), \\ h_{22} &= -\textstyle\sum_{i=0}^{N}\sum_{j=0}^{K} Score_{ij}((\lambda^T\Sigma_j^{-1}\delta_{rot}x_i)^2 + (-(\delta_{rot}x_i)^T\Sigma_j^{-1}\delta_{rot}x_i + \lambda^T\Sigma_j^{-1}\delta\delta_{rot}x_i))), \\ \lambda &= \mu_j - p_i, \\ \delta_{rot} &= (-sin(t_2), -cos(t_2); cos(t_2), -sin(t_2)), \\ \delta\delta_{rot} &= (-cos(t_2), sin(t_2); -sin(t_2), -cos(t_2)), \\ A &= (\Sigma_j^{-1}(1,0)^T)^T, \\ B &= (\Sigma_j^{-1}(0,1)^T)^T. \end{aligned} \quad (9.3)$$

9.2.3.2 D2D

L'implementacío de la classe s'ha fet sense templaritzar, separant així els casos bidimensional i tridimensional en objectes diferents per simplicitat del codi. Dels quals, de moment només s'ha implementat el cas 2D.

La segona subclasse de la interfície ScanMatchingMethods és DistributionDistribution2D. En aquest objecte els dos escanejos a registrar són dos GMM, passats al constructor i que es guarden com atributs de l'objecte.

```
class DistributionToDistribution2D : public ScanMatchingMethods<3>{
    private:
        shared_ptr<GaussianMixturesModel<2>> gmm_ref_;
        shared_ptr<GaussianMixturesModel<2>> gmm_new_;

    public:
        DistributionToDistribution2D(const
            shared_ptr<GaussianMixturesModel<2>>& gmm_ref, const
            shared_ptr<GaussianMixturesModel<2>>& gmm_new);
            ...
};
```

Igual que amb l'anterior i a totes les subclasses de ScanMatchingMethod s'han d'implementar la funció de cost i les seves derivades.

```
double compute_score(Eigen::Vector<double,3> t);
tuple<double,Eigen::Vector<double,3>>
    compute_score_and_gradient(Eigen::Vector<double,3> t);
tuple<double,Eigen::Vector<double,3>,Eigen::Matrix<double,3,3>>
    compute_score_gradient_and_hessian(Eigen::Vector<double,3> t);
```

On la funció de cost està definida per l'equació 9.4, on K_1 és el número de components del primer GMM, K_2 el número de components del segon GMM, w_i el pes de la component, Σ_i la covariància d'una component, μ_i la mitjana d'una component i t el vector de 3 components que dona la transformació en $[x, y, rz]$.

$$\begin{aligned} Score &= -\sum_{i=0}^{K_1}\sum_{j=0}^{K_2} w_i w_j exp(-\tfrac{1}{2}(\lambda^T(\Sigma_i + rot(t_2)\Sigma_j rot(t_2)^T)^{-1}\lambda)), \\ \lambda &= \mu_i - rot(t_2)\mu_j - (t_0, t_1)^T, \\ rot(x) &= (cos(x), -sin(x); sin(x), cos(x)). \end{aligned} \tag{9.4}$$

La funció compute_score_and_gradient() calcula tant el cost, com la primera derivada que és el gradient, aquest càlcul es fa després d'aplicar la

transformació t, seguint l'equació següent:

$$\begin{aligned}
Gradient &= (g_0, g_1, g_2)^T, \\
g_0 &= -\sum_{i=0}^{K_1}\sum_{j=0}^{K_2} Score_{ij}\lambda^T(\Sigma_i + rot(t_2) * \Sigma_j * rot(t_2)^T)^{-1}(1,0)^T, \\
g_1 &= -\sum_{i=0}^{K_1}\sum_{j=0}^{K_2} Score_{ij}\lambda^T(\Sigma_i + rot(t_2) * \Sigma_j * rot(t_2)^T)^{-1}(0,1)^T, \\
g_2 &= -\sum_{i=0}^{K_1}\sum_{j=0}^{K_2} Score_{ij}B, \\
\lambda &= \mu_i - rot(t_2)\mu_j - (t_0, t_1)^T, \\
A &= \lambda^T(\Sigma_i + rot(t_2)\Sigma_j rot(t_2)^T)^{-1}C(\Sigma_i + rot(t_2)\Sigma_j rot(t_2)^T)^{-1}; \\
B &= (\lambda^T(\Sigma_i + rot(t_2)\Sigma_j rot(t_2)^T)^{-1}\delta_{rot}(t_2)\mu_j) + \tfrac{1}{2}(A\lambda) \\
C &= \delta_{rot}(t_2)\Sigma_j rot(t_2)^T + rot(t_2)\Sigma_j \delta rot(t_2)^T; \\
rot(x) &= (cos(x), -sin(x); sin(x), cos(x)),, \\
\delta_{rot}(x) &= (-sin(x), -cos(x); cos(x), -sin(x))
\end{aligned} \tag{9.5}$$

Per últim la compute_score_gradient_and_hessian() fa tots els càlculs anteriors i també obté la matriu hessiana, ja que l'equació per aquesta és molt més complexa s'ha deixat en forma de codi per que sigui més llegible.

```
for(int i=0; i<gmm_ref_->k(); i++){
    for(int j=0; j<gmm_new_->k(); j++){
        lamb = mu->at(i) - rotacio * mu_new->at(j) - translacio;
        eta = (sigma->at(i) + rotacio * sigma_new->at(j) *
            rotacio.transpose()).inverse();
        lamb_T_eta = lamb.transpose() * eta;
        hmk = w->at(i) * w_new->at(j) * 1.0 * exp( -0.5 * (lamb_T_eta
            * lamb));

        dR_mu_M = d_rot * mu_new->at(j);
        d_eta = d_rot * sigma_new->at(j) * rotacio.transpose() +
            rotacio * sigma_new->at(j) * d_rot.transpose();
        A = lamb_T_eta * d_eta * eta;
        B = (lamb_T_eta * dR_mu_M)(0) + 0.5 * (A * lamb);
            ...
        lamb_T_eta_d_tras0 = lamb_T_eta * d_tras.col(0);
        lamb_T_eta_d_tras1 = lamb_T_eta * d_tras.col(1);

        h_00 -= hmk * pow( lamb_T_eta_d_tras0,2) -
            float(d_tras.col(0).transpose() * eta * d_tras.col(0));
        h_01 -= hmk * (lamb_T_eta_d_tras1 * lamb_T_eta_d_tras0 -
            d_tras.col(0).transpose() * eta * d_tras.col(1));
        h_02 -= hmk * ( B * (lamb_T_eta_d_tras0) -
            (d_tras.col(0).transpose() * eta * dR_mu_M + A *
            d_tras.col(0)));
        h_11 -= hmk * ( pow(lamb_T_eta_d_tras1,2) -
```

```
            d_tras.col(1).transpose() * eta * d_tras.col(1) );
        h_12 -= hmk * ( B * (lamb_T_eta_d_tras1) -
            (d_tras.col(1).transpose() * eta * dR_mu_M + A *
            d_tras.col(1)));

        G2 = - dR_mu_M.transpose() * eta * dR_mu_M ;
        G3 = - 2.0 * (A * dR_mu_M)(0);
        G4 = float( lamb_T_eta * d_d_rot * mu_new->at(j) );
        G5_aux = d_d_rot * sigma_new->at(j) * rotacio + 2.0 * ( d_rot
            * sigma_new->at(j) * d_rot ) + rotacio * sigma_new->at(j)
            * d_d_rot - 2.0 *( d_eta * eta * d_eta );
        G5 = 0.5 * float( lamb_T_eta * G5_aux *
            lamb_T_eta.transpose() );

        h_22 -= hmk * ( pow(B,2) + G2 + G3 + G4 + G5 );
    }
}

hessian = Eigen::Matrix<double,3, 3>{{h_00,h_01,h_02},
                                     {h_01,h_11,h_12},
                                     {h_02,h_12,h_22}};
```

9.2.4 Solver

El solucionador es l'encarregat de resoldre el problema de registre expressat com a problema d'optimització. La variable objectiu és la transformació entre els dos escanejos. Els solucionadors que s'implementen estan basats en mètodes del gradient, concretament el mètode de Newton i les seves variacions. Per això, a la definició dels mètodes ha calgut definir també les derivades primera i segona de la funció objectiu.

```
template <int dim>
class Solver{
    private:
        vector<double> process_;
        shared_ptr<ScanMatchingMethods<dim>> method_;
        double min_norm_;
        double min_score_inc_;
        int max_NM_it_;

    public:
        virtual vector<float> process() = 0;
```

```
        virtual tuple<int,Eigen::Matrix<double,dim,1>,
            Eigen::Matrix<double,dim,dim>> compute_optimum(const
            tuple<Eigen::Vector<double, dim>,
            Eigen::Matrix<double,dim,dim>>& t_init) = 0;
        virtual void plot_process(int index,
            Eigen::Matrix<double,dim,1> t, Eigen::Matrix<double,dim,1>
            optimal) = 0;
};
```

Per a la implementació del solucionador, s'ha definit una interfície anomenada Solver, amb els mètodes compute_optimum(), process() i plot_process(). També els atributs min_norm_, min_score_inc_ i max_NM_it_ que defineixen els criteris de parada. L'atribut method_ de tipus ScanMatchingMethods permet definir les funcions de cost i les seves derivades per l'optimització. Finalment, a l'atribut de tipus vector process_ es guarda el valor del score de cada iteració del solucionador.

compute_optimum() és el mètode encarregat de resoldre l'optimització, retornant la transformació òptima juntament amb la inversa de la matriu hessiana, que proporciona una mesura de la incertesa del registre, i un enter, que permet saber quin dels criteris ha causat la parada de l'algoritme. El mètode process() és un getter de l'atribut i plot_process() crida a la funció de plot del method_ per fer els gràfics dels resultats de l'optimització i el seu procés.

9.2.4.1 Newton Method

La classe NewtonMethod implementa el mètode de newton per resoldre un problema de minimització sobre la funció de cost per obtenir la millor transformació per fer el scan matching. Com que aquest és el mètode bàsic no trobem cap diferència amb els atributs i mètodes definits per la interfície.

```
template <int dim>
class NewtonMethod: public Solver<dim>{
    private:
        vector<float> process_;
        shared_ptr<ScanMatchingMethods<dim>> method_;
        double min_norm_;
        double min_score_inc_;
        int max_NM_it_;

    public:
        NewtonMethod(const shared_ptr<ScanMatchingMethods<dim>>&
            method, const double& min_norm, const double&
```

```
            min_score_inc, const int& max_NM_it);

        vector<float> process();

        tuple<int,Eigen::Matrix<double,dim,1>,
            Eigen::Matrix<double,dim,dim>> compute_optimum(const
            tuple<Eigen::Vector<double, dim>,
            Eigen::Matrix<double,dim,dim>>& t_init);

        void plot_process(int index, Eigen::Matrix<double,dim,1> t,
            Eigen::Matrix<double,dim,1> optimal);
};
```

En el mètode compute_optimum() hi ha implementat l'algoritme del mètode de Newton. En aquest s'itera modificant el valor d'una transformació llavor a partir de les seves derivades aplicant

$$x_{n+1} = x_n - \alpha \frac{f'(x_n)}{f''(x_n)}, \tag{9.6}$$

obtenint cada cop un valor més proper al desitjat fins que algun dels criteris de parada pari l'execució del bucle. Aquests són els ja definits a l'interfície, que paren el mètode quan s'arriba a un numero d'iteracions, quan la millora del cost es molt petita o quan el pas es molt petit i per tant ja es molt proper a la solució real.

A continuació, es mostra un tros del codi del mètode compute_optimum(). Com es veu, el càlcul del nou òptim no es suma directament, ja que no es pot composar la part de la rotació amb la suma. Per tant, es crida una funció que multiplica les matrius de transformació generades per el vector t i les torna a deixar en forma de vector.

```
//initialize variables
while(!stop_criteria){
    d=-(hessian.inverse() * gradient);

    optimal = composite_o_plus(optimal, d*0.01);

    score_and_derivates =
        method_->compute_score_gradient_and_hessian(optimal);

    score = get<0>(score_and_derivates);
    gradient = get<1>(score_and_derivates);
    hessian = get<2>(score_and_derivates);
```

```
    process_.push_back(score);

        //update stop_criteria

}
// return the result
```

El resultat a retornar pot tenir diferents implicacions depenent del criteri que ha parat l'optimització. Si l'algorisme convergeix, es retorna l'òptim i un 0 que indica convergència. Si s'ha arribat al numero màxim d'iteracions establert, es compara el cost de l'estat inicial amb el final, i es retorna el millor juntament amb un 1 per representar que no s'ha convergit. Si durant el procés iteratiu el problema queda mal condicionat amb una matriu hessiana no invertible, es trenca l'optimització i es retorna la llavor amb un 2 que indica que l'optimització s'ha trencat.

```
tuple<int,Eigen::Matrix<double,dim,1>,Eigen::Matrix<double,dim,dim>>
    result;
if(bad_conditioning){
    result = make_tuple(2,initial, hess);
}
else if(i>=max_NM_it_){
    if(process_.front()<process_.back()){
        result = make_tuple(1,initial, hess);
    }
    else{
        result = make_tuple(1,optimal,hessian);
    }
}
else if(found){
    result = make_tuple(0,optimal,hessian);
}

return result;
```

9.2.4.2 Line search

La classe LineSearchNewtonMethod també implementa el mètode de Newton per resoldre un problema de minimització, però afegeix el mètode de line_search i el càlcul de les condicions de Wolfe en les funcions first_Wolfe_condition i second_Wolfe_condition per poder determinar la mida òptima del pas en la

direcció que defineix el mètode de Newton. Aquestes noves funcions també tenen una sèrie de constants que es defineixen al constructor i es guarden com atributs de la classe. Els nous atributs són: max_LS_it_ que posa un màxim d'iteracions al line_search, beta1_ i beta2_ són constants que s'utilitzen en l'avaluació de la primera i segona condició de Wolfe, i finalment gamma_ que s'utilitza pel càlcul del factor α pel pas del mètode de Newton.

```
template <int dim>
class LineSearchNewtonMethod: public Solver<dim>{
    private:
            ...
        double beta1_;
        double beta2_;
        double gamma_;
        int max_LS_it_;

        double line_search(const
            tuple<double,Eigen::Vector<double,dim>,
            Eigen::Matrix<double,dim,dim>>& score_and_derivates, const
            Eigen::Matrix<double,dim,1>& d, const
            Eigen::Matrix<double,dim,1>& xk, const int& iter);
        bool first_Wolfe_condition(const double& alpha, const
            Eigen::Matrix<double,dim,1>& d, const
            Eigen::Matrix<double,dim,1>& gradient, const double&
            score, const double& score_plus);
        bool second_Wolfe_condition(const
            Eigen::Matrix<double,dim,1>& d, const
            Eigen::Matrix<double,dim,1>& gradient, const
            Eigen::Matrix<double,dim,1>& gradient_plus);

    public:
        LineSearchNewtonMethod(const
            shared_ptr<ScanMatchingMethods<dim>>& method, const
            double& min_norm, const double& min_score_inc, const int&
            max_NM_it, const double& beta1, const double& beta2, const
            double& gamma, const int& max_LS_it);
                ...
};
```

En la mètode compute_optimum(), igual que a l'anterior, s'hi troba implementat l'algoritme del mètode de Newton. Aquest algorisme itera modificant el valor d'una transformació llavor a partir de les seves derivades, tal com es veu en l'equació 9.6, fins que es troba la solució òptima, quan els criteris de

parada aturen l'optimització. No obstant, en aquest cas α no és una constant fixada, sinó que a cada iteració se'n busca un valor òptim gràcies al mètode line_search. En conseqüència, ara el mètode compute_optimum() segueix la següent estructura:

```
    ...
while(!stop_criteria){
    d=-(hessian.inverse() * gradient);

    alpha = line_search(score_and_derivates, d, optimal);

    optimal = composite_o_plus(optimal, d*alpha);
        ...
}
```

Al mètode line_search se li ha de passar el mètode de registre, la direcció que es vol seguir i l'òptim actual.

```
double line_search(const tuple<double,Eigen::Vector<double,dim>,
    Eigen::Matrix<double,dim,dim>>& score_and_derivates, const
    Eigen::Matrix<double,dim,1>& d, const
    Eigen::Matrix<double,dim,1>& xk)
```

Amb aquesta informació es pot implementar l'algorisme explicat a la pàgina 274 del llibre [Bierlaire 2015]. Aquest consisteix en modificar α segons si es compleixen les les condicions de Wolfe. Per això es defineixen dues variables reals que estableixen el limit esquerra i dreta d'α. A cada iteració es comprova si es compleix la primera condició de Wolfe. En cas que no sigui així, s'actualitza el limit dret d'α al valor actual d'aquesta i dóna com a nou valor d'α el put mig entre els limits dret i esquerra. En cas de que es compleixi la primera condició però no la segona, s'actualitza el límit esquerra i es torna a donar el punt mig com a nova α. No obstant, si el límit dret encara val infinit per l'inicialització el nou valor d'α és l'anterior multiplicat per gamma_. Aquest algoritme s'executa fins que es compleixen les dues condicions i, en conseqüència, es troba la solució o s'arriba al màxim d'iteracions.

La funció first_Wolfe_condition avalua la primera condició de Wolfe

$$score\_plus \leq score + \alpha\beta_1 gradient^T d$$

que, quan és falsa, indica que el pas és massa llarg. La funció demana els paràmetres: alpha, que és la α actual; d, que és la direcció que estableix el Mètode de Newton; gradient, que és el gradient actual; score, que és el cost actual; i score_plus, que és el cost fent el pas actual multiplicat per α. En

aquest mètode s'avalua l'igualtat

La funció second_Wolfe_condition avalua la segona condició de Wolfe

$$(gradient\_plus^T d) \geq (\beta_2 gradient^T d)$$

que, quan és falsa, indica que el pas és massa curt. La funció demana els paràmetres: d, que és la direcció que estableix el Mètode de Newton; gradient, que és el gradient actual; i gradient_plus, que és el gradient fent el pas actual multiplicat per α.

9.3 Comparativa amb ICP i GICP

Per validar la tècnica de registre implementada es planteja fer una comparativa amb altres tècniques de l'estat de l'art desenvolupades per i aplicades en sensors làser. Concretament, es vol comparar la tècnica implementada de GMM-Registration amb les tècniques ICP [ICP 022] i GICP [Generalized-ICP 022] aplicades a núvols de punts provinents de sonar adquirits sota l'aigua. A diferència dels núvols de punts adquirits amb làser, el sonar proporciona núvols de punts molt menys densos i amb molt més soroll. Per tant, es vol veure com reaccionen aquestes tècniques a núvols de punts amb aquestes característiques i si la tècnica de GMM registration presenta una millor adaptació.

Per al registre de núvols de punts aplicant la tècnica ICP [ICP 022], s'utilitza la implementació existent a la Point Cloud Library [PCL 022]. Es tracta d'una implementació molt utilitzada dins la comunitat de la robòtica mòbil. La metodologia ICP està basada en l'algorisme d'Expectació-Maximització on per cada iteració es minimitza la distància euclídia entre els aparellaments de punts fets. Es tracta d'un algorisme que no garanteix la convergència en un mínim global i, per tant, és important proporcionar a l'algorisme la millor inicialització possible. En el nostre cas, utilitzem l'increment de posició entre els dos escanejos mesurat amb el sistema de navegació a la deriva del robot.

Per al registre de núvols de punts aplicant la tècnica GICP [Generalized-ICP 022], s'utilitza la implementació existent a la Point Cloud Library [PCL 022]. Es tracta d'una implementació també molt utilitzada dins la comunitat de la robòtica mòbil. A diferència del ICP, un dels dos núvols de punts es modela a través d'un GMM - fitat situant una component gaussiana a cada punt de l'escaneig - i es resol amb un P2D on es minimitza la distància de Mahalanobis entre les assignacions. Igual que l'ICP, és un algorisme que no garanteix mínims globals i, per tant, també s'utilitza la mesura del sistema de navegació a la deriva del robot com a inicialització.

Per utilitzar les tècniques ICP i GICP, és necessari fer uns imports de la PCL per a obtenir les classes que implementen cada tècnica i els objectes necessaris per encapsular les dades:

```
#include <pcl/registration/icp.h>
#include <pcl/registration/gicp.h>
#include <pcl/io/pcd_io.h>
#include <pcl/point_types.h>
#include <pcl/search/impl/search.hpp>
#include <pcl/common/common_headers.h>
#include <pcl/registration/warp_point_rigid_3d.h>
#include <pcl/registration/transformation_estimation_2D.h>
```

A la Point Cloud Library hi ha implementats molts tipus. D'aquests objectes els que més s'han fet servir han estat els PointCloud, dissenyats per contenir núvols de punts. Els punts de dins són del tipus PointXYZ, que té la possibilitat d'incorporar dades en l'eix Z. No obstant, no ha estat necessari guardar cap valor en aquest atribut perquè només s'han tractat les dades en dues dimensions. No s'ha utilitzat el tipus bidimensional, PointXY, perquè no deixa realitzar posteriorment les crides als mètodes ICP i GICP.

Tot seguit, es declaren els objectes que permeten fer el registre, ja que cada tècnica està implementada en una classe diferent:

```
pcl::IterativeClosestPoint<pcl::PointXYZ, pcl::PointXYZ> icp;
pcl::GeneralizedIterativeClosestPoint<pcl::PointXYZ, pcl::PointXYZ>
    gicp;
```

Per inicialitzar aquests objectes s'han realitzat les crides dels mètodes 'setInputSource()' i 'setInputTarget()'. La funció d'aquests mètodes és passar un punter cap a un núvol de punts, sigui el conjunt de referència o el móbil, als objectes de tipus ICP i GICP.

```
icp.setInputSource(current_pointCloud);
icp.setInputTarget(reference_pointCloud);
gicp.setInputSource(current_pointCloud);
gicp.setInputTarget(reference_pointCloud);
```

En el cas del GICP cal fixar el nombre de veïns que s'han de tenir en compte per als càlculs de les covariàncies. Si no es fixa, surten errors amb els casos que no tinguin veïns. Per això s'ha fixat en 1, perquè tingui com a mínim un únic veí en tots els casos.

```
gicp.setCorrespondenceRandomness(1);
```

Amb els objectes definits i inicialitzats es fa la crida al mètode 'align()' perquè s'executi el registre.

```
icp.align(Final, matrix_icp_align);
gicp.align(gFinal, matrix_icp_align);
```

El primer argument que s'ha de passar en aquest mètode és un objecte de tipus PointCloud de la PCL. Aquest objecte és necessari perquè emmagatzema els punts ja transformats segons la transformada trobada. El segon argument és una matriu de floats de quatre per quatre dimensions on s'escriu la transformada resultant del registre. Aquest segon argument és opcional però en el nostre cas el el necessitem ja que és el resultat que volem determinar.

Per aplicar la tècnica GMM-Registration, s'ha creat una funció templaritzada segons la dimensió del problema anomenada 'gmm_register()'. L'objectiu d'aquesta funció és poder fer dos registres consecutius sobre els mateixos núvols de punts i gestionar el resultat que es retorna en funció de si el solver convergeix o no. Concretament, primer, es vol fer un registre D2D per acostar-nos ràpidament a la solució i, a continuació, un registre P2D per refinar el resultat. Com és sabut, el mètode D2D representa els dos escanejos amb GMM fent el registre més àgil. En canvi, el mètode P2D només comprimeix un núvol de punts amb un GMM i, al disposar de més dades, es pot assolir un resultat més precís.

La funció 'gmm_register()' rep tres objectes com a arguments. Dos dels objectes són solvers, on el primer ha d'estar construït amb un mètode D2D i el segon amb un mètode P2D. No importa el tipus de solver utilitzat, pot ser qualsevol herència de la interfície solver. L'últim argument és una tupla formada per un vector i una matriu de doubles templaritzades per dimensió, les quals proporcionen la llavor per a la optimització. El resultat del registre es retorna amb una tupla del mateix tipus, on el vector conté la transformada del registre i la matriu la seva incertesa.

```
template <int dim>
inline tuple<Eigen::Vector<double, dim>, Eigen::Matrix<double, dim,
    dim>> GMM_register(const shared_ptr<Solver<dim>>& solver_d2d,
    const shared_ptr<Solver<dim>>& solver_p2d, const
    tuple<Eigen::Vector<double, dim>, Eigen::Matrix<double, dim,
    dim>>& t_init);
```

Dins el mètode 'gmm_register()', primer, es fa la crida al mètode 'compute_optimum()'

del primer solver amb la inicialització rebuda per paràmetre. Si el registre convergeix, es fa la crida de l'altre solver utilitzant com a llavor el resultat del primer. Si el primer registre no convergeix, es fa la crida al segon solver utilitzant com a llavor la inicialització rebuda per paràmetre. Si el segon registre convergeix, la funció retorna com a resultat aquest segon registre. Si el segon registre no convergeix i el primer registre sí que ho ha fet, la funció retorna el primer registre. Si cap dels dos solvers convergeix, la funció retorna la inicialització rebuda per paràmetre ja que és el millor resultat que es disposa.

El registre amb la tècnica GMM-Registration s'inicialitza amb el resultat del sistema de navegació a la deriva, ja que també és un mètode que no garanteix mínims globals. Es fa un registre doble, enllaçant un D2D amb un P2D, ambdós registres solucionats amb el solver 'LineSearchNewtonMethod()':

```
tuple<Eigen::VectorXd, Eigen::MatrixXd> gmm_register_return =
    GMM_register(solver_p2d, solver_d2d, make_tuple(delta_odometry_x,
    mat) );
```

Finalment, per poder efectuar la comparativa entre les tres tècniques de registre proposades, es genera una figura on es mostra la inicialització donada pel sistema de navegació a la deriva i els diferents registres obtinguts. Per fer-ho s'utilitza la llibreria Matplotlib [Matplotlib 022]. Exemple de figura obtinguda d'una de les comparatives realitzades al data set de Sant Feliu (figura 9.3).

9.4 Ampliació de la llibreria Front-Ends

El Front-End implementat de base, la funció 'ndt_constructor()', genera els GMMs projectant una graella sobre un núvol de punts. En aquesta secció es mostra l'ampliació que s'ha fet de la llibreria Front-End implementant dues noves funcions per fitar GMMs a núvols de punts. Per fer-ho, s'implementen els algorismes presentats a les seccions (5.3) i (5.4).

9.4.1 Front-End basat en l'algorisme K-means

En aquesta secció es detalla la implementació de la funció de la llibreria de Front-Ends basada en l'algorisme de K-means presentat a la secció (5.3).

```
template<int dim>
shared_ptr<GaussianMixturesModel<dim>> k_means_constructor(const
    std::vector<Eigen::Matrix<double,dim,1>>& scan, const int&
```

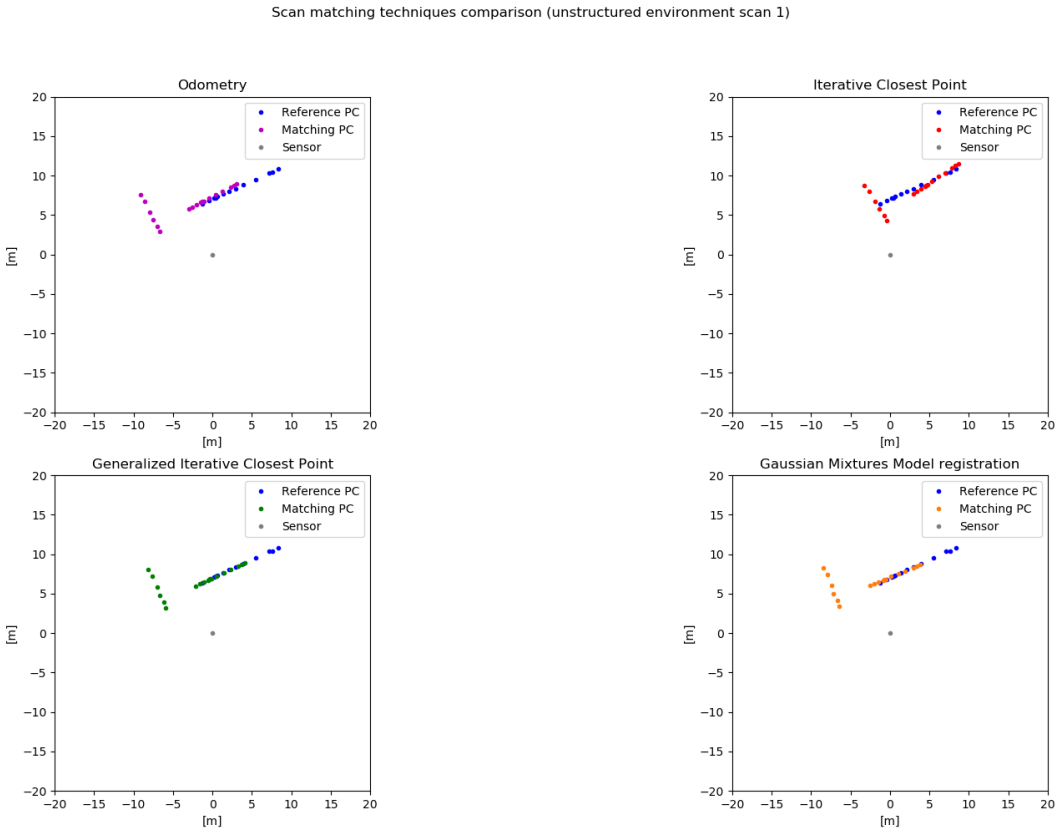

Figura 9.3: Exemple de la figura de la comparativa.

```
points_threshold, const int& k, const int& n_init, const int&
max_iter_lloyd, const bool& calc_cov = false)
```

Aquesta funció rep per paràmetre les dades del núvol de punts, el nombre mínim de punts que han de tenir assignats tots els grups al final del bucle, el nombre de components a crear, el màxim nombre de vegades que es realitzarà el mètode K-means, el nombre màxim d'iteracions que tindrà el bucle intern del K-means i la dada per indicar si es vol o no realitzar el càlcul de les covariàncies (sent aquesta última opcional). Si el booleà és fals, es genera un GMM amb covariàncies isotròpiques unitàries.

El mètode s'ha realitzat fent una esquematització començant amb els paràmetres i clàusules fins a arribar a les equacions del mateix algoritme, aquest esquema s'introduirà a continuació.

Abans de començar els càlculs es comprova que $K > 0$, és a dir, que com a mínim es busqui un model d'una component i que $K < N$, és a dir, que hi hagi menys components que dades. Si aquestes condicions no es compleixen

es retorna un GMM buit.

$$K == 0 \text{ OR } K > N \tag{9.7}$$

Estructura de la implementació que s'ha seguit en la funció 'k_means_constructor()':

```
Parametritzacio
Bucle de solucions K-means
    Inicialitzacio dels components
    Bucle algorisme
        Bucle del nuvol de punts
            E step
            M step
        Avaluacio criteri de parada
    Comprovar si hi ha una millor solucio per retornar
```

Com que aquesta tècnica pot ser que no convergeixi, s'ha ficat un bucle general que s'executarà 'n_init' vegades i es retornarà la millor solució trobada. Al principi de cada volta sempre es realitzen els mateixos procediments.

Creació dels objectes temporals. Es creen els objectes necessaris per a la tècnica. Els objectes més rellevants són la matriu amb els indicadors binaris que relacionen punts amb grups i els vectors per a les dades μ_k i Σ_k on es guardaran les respectives dades de les components.

```
Eigen::MatrixXi old_labels = Eigen::MatrixXi::Zero(n_samples,
    n_clusters);
vector<Eigen::Matrix<double,dim,1>>
    old_means(n_clusters,Eigen::Matrix<double,dim,1>::Zero());
vector<Eigen::Matrix<double,dim,dim>>
    old_covariances(n_clusters,Eigen::Matrix<double,dim,dim>::Zero());
```

Randomització dels grups. A cada volta el mètode fica en un vector les posicions de les observacions, passades per paràmetre, en un ordre aleatori i se n'extreuen K observacions que s'utilitzaran com a llavor de cada centroïde.

```
for(int i=0; i < n_samples; i++) seeds[i] = i;
    std::random_shuffle ((seeds), (seeds+n_samples));
for(int i=0; i < n_clusters; i++) old_means[i] = scan.at(seeds[i]);
```

Un cop definides les llavors dels centroïdes, es comença el bucle que aplica l'algorisme de k-means. Com a màxim se'n realitzen 'max_iter_lloyd' iteracions.

Dins d'aquest bucle es creen nous objectes, iguals que els anteriors, però amb la diferència que aquests són temporals.

```
Eigen::MatrixXi new_labels = Eigen::MatrixXi::Zero(n_samples,
    n_clusters);
double new_inertia = 0;
vector<int> new_points(n_clusters,0);
vector<Eigen::Matrix<double,dim,1>>
    new_means(n_clusters,Eigen::Matrix<double,dim,1>::Zero());
vector<Eigen::Matrix<double,dim,dim>>
    new_covariances(n_clusters,Eigen::Matrix<double,dim,dim>::Zero());
```

A continuació es crea un bucle que recorre tot el data set per efectuar l'etapa E i l'etapa M. En l'etapa (E) es duen a terme els càlculs per mirar quin centroïde és el que es troba més a prop per a cada punt. En fer aquesta comprovació per cada punt es va emplenant la matriu amb els indicadors binaris, deixant a 0 totes les posicions [punt, grup] a on no s'assignarà el punt, i a 1 on s'assigna el punt:

```
new_labels(iter_punt, nearest_cluster) = 1;
```

A l'etapa (M) només es fan dues operacions. S'incrementa el nombre de punts que es troben assignats a la component que ha sortit com a més propera i s'actualitza la nova mitjana μ_k de la component k:

```
new_points[nearest_cluster]++;
new_means[nearest_cluster] += actual_point;
```

Un cop acabat el bucle del núvol de punts es fa una comprovació que tots els grups tinguin un mínim nombre de punts assignats, per evitar matrius de covariança singulars. En cas que això passi es diu que s'ha trobat un solució degenerada i es trenca el bucle de l'algorisme de k-means i es torna al bucle exterior. El valor del llindar el defineix el paràmetre del constructor 'points_threshold'.

```
for(int i = 0; i < n_clusters; i++) {
    if(new_points[i] < points_threshold) {
        inferior = true;
        break;
    }
}
```

Alhora, si la inèrcia de la solució no millora es dona per finalitzat l'algorisme i es dóna al bucle exterior la solució trobada.

```
if( inferior || new_inertia == old_inertia)
```

Altrament, es guarden les dades dins de les variables inicials perquè es puguin fer servir per l'etapa E de la nova iteració.

```
old_inertia = new_inertia;
old_means = new_means;
old_covariances = new_covariances;
```

Al final de cada volta del bucle general on es realitza 'n_init' vegades l'algorisme k-means, hi ha un condicional que comprova cada cop si la solució temporal que s'ha trobat és diferent de la que ja hi ha guardada i de si té una millor inèrcia. Si aquest és el cas, es sobreescriuen les dades a establir dins del GMM i, en cas contrari, no es fa cap modificació d'aquests valors i continuarà fent les iteracions que li quedin al bucle.

En finalitzar les iteracions del bucle general, es fa una comprovació sobre el resultat. Com que no és assegurat que es trobi una solució, es mira si s'ha trobat una solució en alguna de les iteracions realitzades de la que es puguin extreure les dades per al GMM, si és així, fa els passos necessaris per ficar les dades correctament. I en el cas de no haver aconseguit cap solució, es fa una única passada a l'algorisme de k-means amb la que s'obtindran possibles valors a establir al GMM.

En el cas de voler realitzar el càlcul de les covariàncies, passant com a cert el paràmetre 'calc_cov' de la funció 'k_means_constructor()' i si durant l'últim pas de comprovació s'ha trobat una solució per a introduir al GMM, es farà un càlcul per a cada component on s'agafarà el seu recompte de punts assignats i la mitjana d'aquella component per a obtenir així la covariància.

```
covariance_current = (1.0 / (count_current - 1)) *
    (covariance_current - count_current * (means_current *
    means_current.transpose()));
```

Aquesta operació es duu a terme al final perquè si no el cost seria major, ja que en si, el k-means no realitza el càlcul per la covariància dels components i si l'ha de realitzar mentre fa les altres operacions a cada iteració, el cost pujaria considerablement.

9.4.2 Front-End basat en l'algorisme d'Expectació-Maximització

En aquesta secció es detalla la implementació de la funció de la llibreria de Front-Ends basada en l'algorisme d'Expectació-Maximització presentat a la secció (5.4).

```
template<int dim>
```

```
shared_ptr<GaussianMixturesModel<dim>> em_constructor(const
    std::vector<Eigen::Matrix<double,dim,1>>& scan, const int&
    points_threshold, const int& k, const int& n_init, const int&
    max_iter_lloyd, const bool& init_kmeans = false)
```

Aquesta funció rep per paràmetre les dades del núvol de punts, el nombre mínim de punts que han de tenir assignats tots els grups al final del bucle, el nombre de components a crear, el màxim nombre de vegades que es realitzarà el mètode EM, el nombre màxim d'iteracions que tindrà el bucle intern de l'algorisme EM i la dada per indicar si es vol o no inicialitzar les dades a l'inici de cada volta del bucle general amb els resultats de fer una crida al constructor de K-means (sent aquesta última opcional) .

El mètode s'ha realitzat fent una esquematització començant amb els paràmetres i clàusules fins a arribar a les equacions del mateix algoritme, aquest esquema s'introduirà a continuació.

Abans de començar els càlculs es comprova que $K > 0$, és a dir, que com a mínim es busqui un model d'una component i que $K < N$, és a dir, que hi hagi menys components que dades. Si aquestes condicions no es compleixen es retorna un GMM buit (equació 9.7).

Estructura de la implementació que s'ha seguit en la funció 'em_constructor()':

```
Parametritzacio
Bucle de solucions K-means
    Inicialitzacio dels components
    Bucle algorisme
        Bucle del nuvol de punts
            E step
            M step
        Avaluacio criteri de parada
    Comprovar si hi ha una millor solucio per retornar
```

Com que aquesta tècnica pot ser que no convergeixi, s'ha ficat un bucle general que s'executarà 'n_init' vegades i es retornarà la millor solució trobada. Al principi de cada volta sempre es realitzen els mateixos procediments.

Creació dels objectes temporals. Es creen els objectes necessaris per a la tècnica. Els objectes més rellevants són la matriu de responsabilitats entre dades i components i els vectors per a les dades π_k, μ_k i Σ_k on es guardarden els paràmetres de cada component.

```
vector<double> old_weights;
vector<Eigen::Matrix<double,dim,1>> old_means;
vector<Eigen::Matrix<double,dim,dim>> old_covariances;
```

A continuació, es generen les llavors de cada component. El paràmetre 'init_kmeans' indica si es vol inicialitzar l'algorisme d'EM a partir d'una solució obtinguda per l'algorisme K-means o amb una inicialització aleatòria. Per generar una llavor de K-means es crida la funció "k_means_constructor()".

```
shared_ptr<GaussianMixturesModel<dim>> a_init;
a_init = k_means_constructor(scan, points_threshold, k, n_init,
    max_iter_lloyd, index, true);
old_weights = *a_init->weights();
old_means = *a_init->means();
old_covariances = *a_init->covariances();
```

Per generar una llavor aleatòria, el mètode fica en un vector les posicions de les observacions, passades per paràmetre, en un ordre aleatori i se n'extreuen K observacions que s'utilitzaran com a llavor de cada centroïde.

```
for(int i=0; i < n_clusters; i++) {
    old_weights.push_back( (1.0/n_clusters) );
    old_means.push_back(scan.at(seeds[i]));
    old_covariances.push_back(Eigen::Matrix<double,dim,dim>::Identity());
}
```

Un cop definides les llavors dels centroïdes, es comença el bucle que aplica l'algorisme d'EM amb un màxim de 'max_iter_lloyd' iteracions.

Dins d'aquest bucle es creen els nous objectes, iguals que els anteriors, però amb la diferència que aquest són temporals.

```
vector<double> new_weights(n_clusters, 0);
Eigen::MatrixXd new_labels = Eigen::MatrixXd::Zero(n_samples,
    n_clusters);
vector<Eigen::Matrix<double,dim,1>>
    new_means(n_clusters,Eigen::Matrix<double,dim,1>::Zero());
vector<Eigen::Matrix<double,dim,dim>>
    new_covariances(n_clusters,Eigen::Matrix<double,dim,dim>::Zero());
```

A continuació es crea un bucle que recorre tot el data set per efectuar l'etapa E i l'etapa M.

En l'etapa (E) s'avaluen les responsabilitats del punt x_i amb cada component, utilitzant els valors dels paràmetres actuals. Per a obtenir les responsabilitats utilitzant els valors dels paràmetres actuals es fa servir l'equació 5.4 amb:

```
for(int i=0; i<n_clusters; i++){
    sumatori_cluster[i] = old_weights[i] * ( 1 / ( pow( 2 * M_PI,
```

```
        (dim/2) ) ) ) * ( 1 / ( pow(
        old_covariances[i].determinant(), (0.5) ) ) ) * exp( -0.5 *
        (actual_point-old_means[i]).transpose() *
        old_covariances[i].inverse() * (actual_point-old_means[i]) );
    sumatori += sumatori_cluster[i];
}
```

A l'etapa (M) s'actualitzen els paràmetres del GMM seguint les formules (equació 5.4.1), (equació 5.4.2) i (equació 5.4.3).

El problema amb aquesta etapa recau en l'equació de la covariància (equació 5.4.3) que depèn de les dades de la nova mitjana (μ_k). Això fa que sigui necessari recórrer com a mínim dues vegades tot el conjunt de núvol de punts. Una volta on es calcularien les mitjanes dels components (μ_k) i una altra volta per a calcular les covariàncies.

Per solucionar aquest problema, s'ha modificat l'equació (5.4.3) per obtenir els mateixos resultats, amb la diferencia en rendiment. Aquesta modificació el que fa es, treure la dependència de les noves mitjanes (μ_k) i reduir el nombre de bucles que s'han de realitzar, resultant així en l'equació (9.8).

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(r_{nk})(x_n * x_n^T) - (\mu_k * \mu_k^T) \tag{9.8}$$

En realitzar aquestes modificacions en la formula (equació 5.4.3) per reduir el nombre de bucles a fer, ha calgut fer les modificacions adients en el codi per a l'etapa M.

Per a portar a cap totes les operacions i fer servir la nova equació de la covariància (equació 9.8), s'han hagut de fer algunes modificacions en les operacions que es feien amb els objectes. Durant el primer bucle, s'ha ficat que s'acumulin les dades de les observacions, d'aquesta forma un cop que s'obtinguin les dades de les noves mitjanes (μ_k), es pot realitzar la nova equació i així obtenir les noves covariàncies sense necessitat de realitzar cap altre bucle.

Un cop acabat el bucle del núvol de punts on es realitza l'algorisme d'EM, s'ha avaluat el likelihood de la solució trobada aplicant l'equació 5.7.

En el cas que el likelihood no millorés respecte el valor optingut a l'iteració anterior, es considerara que l'algorisme ha convergit perque no a trobat una millor solucioi es trenca el bucle:

```
if(fabs(new_likelihood - old_likelihood) < 0.00001d)
```

S'ha fet que mentre es van realitzant les diverses operacions en l'etapa M es van fent comprovacions per detectar si una solució es degenerada, a on,

es trencaria el bucle de l'algorisme. Una solució degenerada s'identifica quan hi ha com a mínim una component a on no hi ha més de dues responsabilitats que siguin major a '1.0E-10'. En aquests casos la solució en degenera i apareixen covariàncies nul·les.

Altrament, es guarden les dades dins de les variables inicials perquè es puguin fer servir per l'etapa E de la nova iteració de l'algorisme EM.

```
old_weights = new_weights;
old_means = new_means;
old_covariances = new_covariances;
```

Al final de cada volta del bucle general on es realitza 'n_init' vegades l'algorisme EM, hi ha un condicional que comprova cada cop si la solució temporal que s'ha trobat és diferent de la que ja hi ha guardada i de si té una millor log likelihood, si és aquest el cas, se sobreescriuen les dades a establir dins del GMM, i en cas contrari, no es fa cap modificació amb aquests valors i continuarà fent les iteracions que li quedin al bucle.

En finalitzar les iteracions del bucle general, es fa una comprovació sobre el resultat. Com que no és assegurat que es trobi una solució, es mira si s'ha trobat una solució en alguna de les iteracions realitzades de la que es puguin extreure les dades per al GMM, si és així, fa els passos necessaris per ficar les dades correctament. En el cas de no haver aconseguit cap solució, es realitza una crida al constructor de K-means per retornar una solució encara que sigui aleatòria.

Capítol 10

Implantació i resultats

10.1 Data sets utilitzats

Per als diferents experiments de validació proposats en aquest projecte s'utilitzen dos data sets adquirits en un entorn submarí amb un Mechanical Scaning Profiling Sonar muntant en un AUV. Aquest tipus de sensor està format per un sonar perfilador - un únic feix acústic molt prim - que rota al pla accionat per un actuador mecànic. El sensor es munta paral·lel al pla de moviment de l'AUV i, per tant, s'obtenen núvols de punts 2D de l'entorn del robot. El sensor està configurat amb un camp de visió de 270º simètrics respecte de l'eix de moviment del robot, tal com es mostra a la figura (10.1).

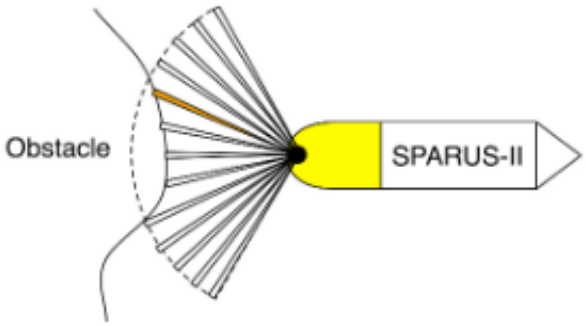

Figura 10.1: Exemple funcionament Sonar.

Com a AUV s'utilitza el vehicle desenvolupat pel Centre d'Investigació en Robòtica Submarina (CIRS) Sparus II. Es tracta d'un vehicle holonòmic en forma de torpede equipat amb sensors inercials (DVL i IMU). Disposa d'una paylod frontal on es munta el Mechanical Scaning Profiling Sonar. Per adquirir el dataset el robot s'opera des d'una barca a través d'una boia en mode debug.

Un dels datasets està adquirit en un entorn estructurat amb formes cartesianes i parets verticals, fent irrellevants els canvis de profunditat del robot. El dataset és correspon a una ruta en bucle entre uns blocs de formigó situats a l'espigó principal del port de Sant Feliu de Guíxols (Girona) mostrats a la figura 10.2 esquerra. L'altre dataset està adquirit en un entorn natural amb

formes molt irregulars. El dataset es correspon a una volta i mitja a l'illot anomenat la Galera situat al Golfet del Cap de Creus (Girona) mostrat a la figura 10.2 dreta.

Figura 10.2: Entorn on s'han adquirit els datasets. Esquerra: Espigó principal del port de Sant Feliu de Guíxols. Dreta: Illot la Galera situat al Golfet del Cap de Creus.

Com que els datasets estan adquirits amb l'AUV submergit, no es disposa de senyal de GPS - ja que les ones electromagnètiques només penetren pocs centímetres dins de l'aigua - i, per tant, no es disposa del ground truth de la trajectòria seguida pel robot. En conseqüència, no es disposa d'una mesura exacta de la trajectòria del robot i no es pot mesurar l'error de cada tècnica de registre. La comparativa només podrà ser qualitativa basant-se en l'observació visual dels registres assolits.

10.2 Adquisició i processat de les dades

L'AUV Sparus II funciona amb una arquitectura de control anomenada COLA2 implementada sobre el middleware ROS [ROS]. L'arquitectura ens permet subscriure'ns als topics on les dades es van publicant en temps real a mesura que són adquirides pels sensors del robot.

Per rebre els escanejos del sonar ens hem de subscriure al topic '$/sparus2/ndt\_navigator/scan\_builder/sonar\_scan\_and\_delta\_pose$', on es publica l'escaneig i l'increment de posició del robot durant la seva adquisició. Per extreure les dades d'aquest tòpic s'utilitza una funció anomenada 'get_scan()'. Aquesta funció es crida cada cop que es rep un missatge pel topic que és de tipus 'PointCloudAndPoseWithCovarianceStamped'. Per fer servir els punts l'escaneig cal fer una conversió del tipus de ROS al tipus Vector de la llibreria Eigen [Eigen]. Per fer-ho només cal un bucle on es van desant les dades en aquest vector. Per a obtenir les dades d'odometria es cri-

dar un mètode anomenat 'get_SO2_projection()', proporcionat pel COLA2, que projecta els increments de posició tridimensionals publicats al topic de ROS al pla d'escaneig del robot.

```
void get_scan( const
    ndt_scanmatching_cpp::PointCloudAndPoseWithCovarianceStamped&
    data );
std::tuple<Eigen::Vector3d, Eigen::Vector3d> get_SO2_projection(
    const geometry_msgs::PoseWithCovariance& odom );
```

Aquest es un exemple de com es veuria utilitzar el COLA2 i el RVIZ per veure les dades que es reben del AUV 10.3.

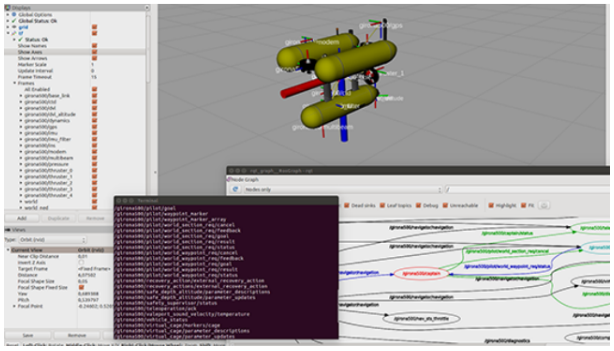

Figura 10.3: Exemple d'ús del COLA2.

10.3 Comparativa amb ICP i GICP

En aquesta secció es mostren els resultats de la comparativa proposada a la secció 9.3 utilitzant els dos data sets presentats a la secció 10.1. La comparativa només s'ha pogut realitzar de manera qualitativa, ja que al utilitzar dades reals subaquàtiques no es disposa de ground truth perquè sota l'aigua no hi funciona amb precisió cap sistema de posicionament absolut. Per tant, les comparatives només es poden fer per inspecció visual dels resultats.

Per poder fer la comparativa s'ha generat una figura per cada escaneig rebut, on es mostren els resultats obtinguts per cada tècnica i l'associació que ens dóna l'odometria, que serveix de llavor als tres mètodes. Un cop feta la figura per cada escaneig, s'ha generat un document Excel per gestionar els resultats de la comparativa. En aquest document, estructurat en tres pàgines, a la primera pàgina es fa el recompte de les diferents observacions que s'han fet dels escanejos pels dos data sets en diferents matrius. A la segona i a la tercera pàgina, es fan les comparacions dels diferents escanejos del data

set de Sant Feliu i del data set de la Galera respectivament. Aquesta comparació s'ha fet a partir de la mateixa graella de l'Excel, on per cada fila es realitza l'observació d'una única figura. Al mateix temps, s'ha fet una distribució per columnes per les tres tècniques utilitzades. Per cada tècnica s'han comptabilitzat:

- Encerts: comparativa per mirar quants dels resultats obtinguts per una tècnica són millors que els resultats que s'haguessin obtingut amb l'odometria.
- Erronis: comparativa per mirar quants dels resultats obtinguts per una tècnica són pitjors que els resultats que s'haguessin obtingut amb l'odometria.
- Retards: comparativa per mirar quants dels resultats obtinguts per una tècnica tenen l'efecte passadís. Aquest efecte succeeix en escaneigs formats per línies paral·leles. Com que el registre només es pot fer en la direcció normal a les línies, no tenim informació en l'altre direcció i el registre tendeix a para el robot, no deixant-lo avançar.
- Millors: comparativa que s'utilitza per tenir el recompte de quantes vegades una tècnica ha encertat i és millor comparant-la amb l'altra tècnica.
- Semblants: comparativa a on, independentment de si són encerts o erronis, les dues tècniques han assolit uns resultats molt similars.

10.3.0.1 Resultats de l'ICP

Si s'analitzen el resultats del registre ICP de manera individual, es veu que la majoria dels resultats són o bé incorrectes (no té sentit el resultat obtingut respecte a l'odometria) o pitjors que els resultats de l'associació utilitzant només l'odometria. En poques ocasions s'aconsegueix un bon registre.

Alguns exemples d'aquests casos, de les dades que s'han obtingut dels dos data sets (secció 10.1), són (figura 10.3.0.1) i (figura 10.3.0.1) on els resultats són pitjors que aplicant només l'odometria. (figura 10.3.0.1) i (figura 10.3.0.1) exemples a on els resultats han estat incorrectes, i com últims exemples, (figura 10.3.0.1) i (figura 10.3.0.1) on han donat resultats correctes.

10.3.0.2 Comparativa amb GICP

Un cop descartada la tècnica ICP [ICP 022], es fa la comparativa de la tècnica GICP [Generalized-ICP 022] amb la tècnica implementada de GMM Regis-

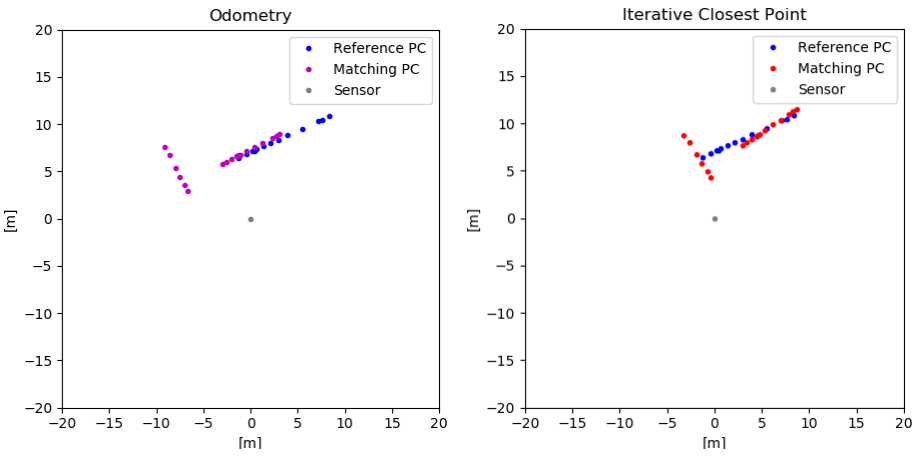

Figura 10.4: Dataset (Sant Feliu) exemple de resultat pitjor que odometry (ICP)

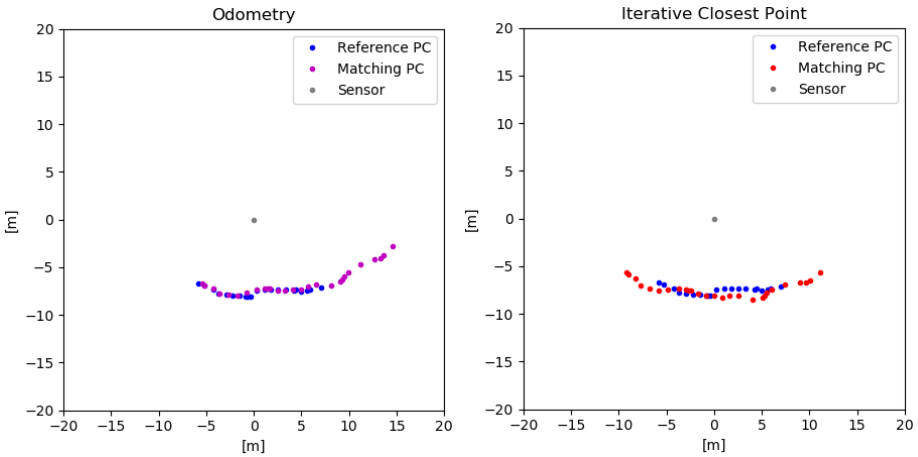

Figura 10.5: Dataset (Galera) exemple de resultat pitjor que odometry (ICP)

tration (secció 5.2). Un cop feta la comparativa segons els criteris definits anteriorment, a la taula 10.1 es mostren els resultats pel data set de Sant Feliu de Guíxols (secció 10.1) i a la taula 10.2 es mostren els resultats pel data set de la Galera (secció 10.1).

No obstant, es pot aclarir el resultat de la comparativa utilitzant les dades relatives que es mostren a la taula 10.3 pel data set de Sant Feliu de Guíxols (secció 10.1) i a la taula 10.4 pel data set de la Galera (secció 10.1).

D'aquestes taules es veu que les dues tècniques funcionen de manera similar amb poques diferències qualitatives entre elles. Pel que fa als encerts els resultats són molt propers al 50% per les dues tècniques en els dos data sets. Pel que fa al retard, es veu com la tècnica de GMM Resgistration en genera menys, amb una diferència del 10% pel data set la Galera (secció 10.1) i del

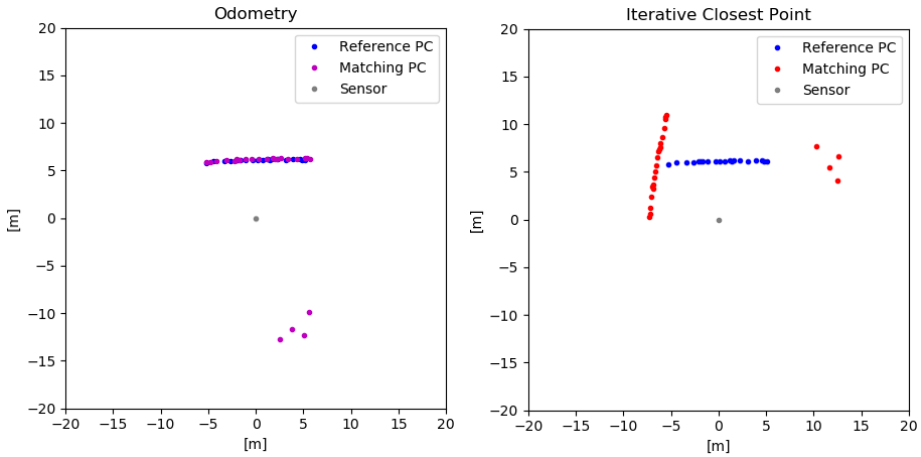

Figura 10.6: Dataset (Sant Feliu) exemple de resultat incorrecte (ICP)

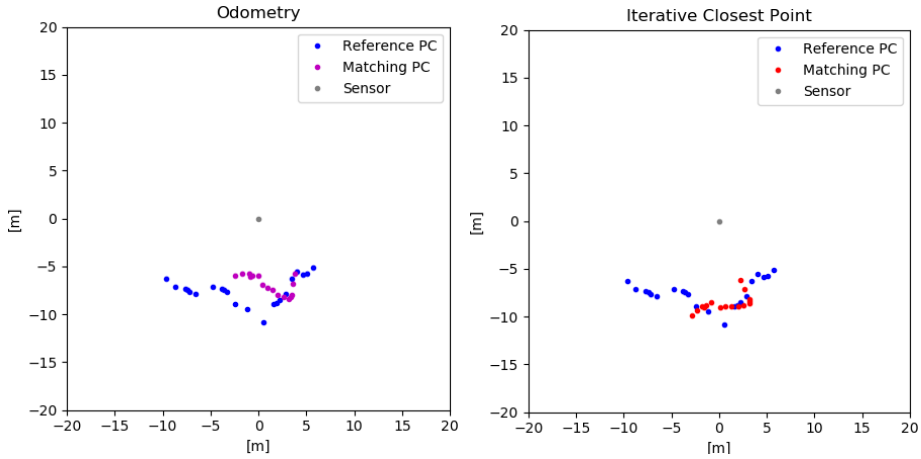

Figura 10.7: Dataset (Galera) exemple de resultat incorrecte (ICP)

15% pel data set de Sant Feliu de Guíxols (secció 10.1). Alhora, es veu que les diferències són mínimes pels dos data sets, fet que ens mostra que les dues tècniques s'adapten tant a entorns amb estructura (port) com en entorns sense estructura (pedres naturals). Aquesta adaptació és un punt a favor per les dues tècniques.

A les figures (10.3.0.2) i (10.3.0.2) es mostren casos de retards produïts en aplicar la tècnica GICP pels diferents data sets. Es pot veure com la tècnica mou lleugerament el núvol de punts mòbil per arribar a una solució. A la figura (10.3.0.2) es pot veure com aplicant la tècnica GICP a un dels scans del data set de Sant Feliu de Guixols, ha realitzat un petit desplaçament, en comparació a l'odometria, per intentar registrar millor els punts. Això ha fet que realitzés un desplaçament en el mateix sentit d'on venia l'AUV. A la figura (10.3.0.2) es pot veure com aplicant la tècnica GICP a un dels scans

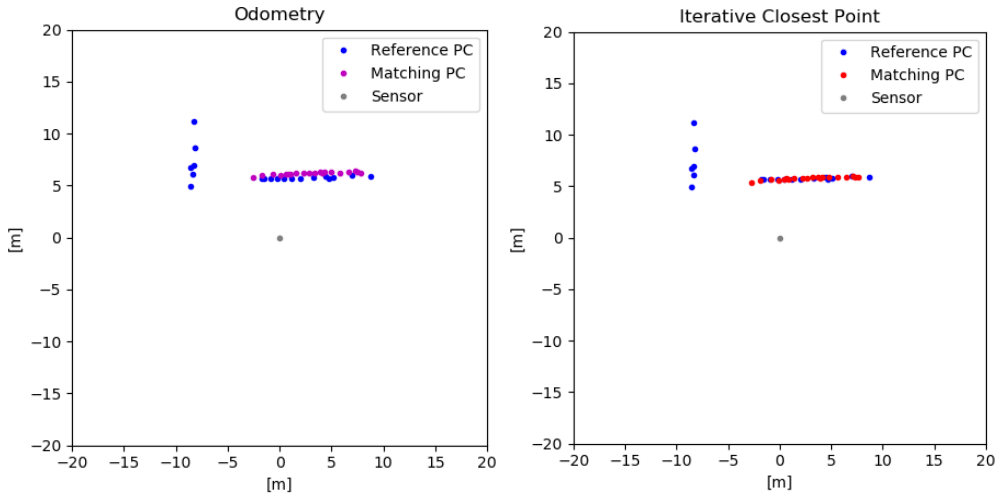

Figura 10.8: Dataset (Sant Feliu) exemple de resultat correcte (ICP)

Feliu	Encerts	Erronis	Retards	Millor	Semblants
GICP	41	21	28	21	32
GMM	43	19	21	19	

Taula 10.1: Comparacions amb el dataset de Sant Feliu.

del data set de la Galera, ha realitzat un petit desplaçament, en comparació a l'odometria, per intentar registrar millor els punts. Això ha fet que realitzés un desplaçament en el mateix sentit d'on venia l'AUV.

A les figures (10.3.0.2) i (10.3.0.2) es mostren casos amb retards produïts en aplicar la tècnica GMM pels diferents data sets. Es pot veure com la tècnica mou lleugerament el núvol de punts mòbil per arribar a una solució. A la figura (10.3.0.2) es pot veure bé el retard, ja que ha registrat desplaçant l'escaneig mòbil cap a l'esquerra per ficar-los just a sobre l'escaneig fix. A la figura (10.3.0.2) el registre ha fet el mateix però girant 180º l'escaneig mòbil.

A les figures (10.3.0.2) i (10.3.0.2) es mostren casos a on cap de les dues tècniques ha produït un resultat correcte pels diferents data sets. Com es veu a la figura (10.3.0.2) el GICP ha registrat l'escaneig mòbil més amunt d'on hauria d'estar, mentre que el GMM ha fet el contrari.

Galera	Encerts	Erronis	Retards	Millor	Semblants
GICP	190	68	84	101	101
GMM	176	82	68	93	

Taula 10.2: Comparacions amb el dataset de la Galera.

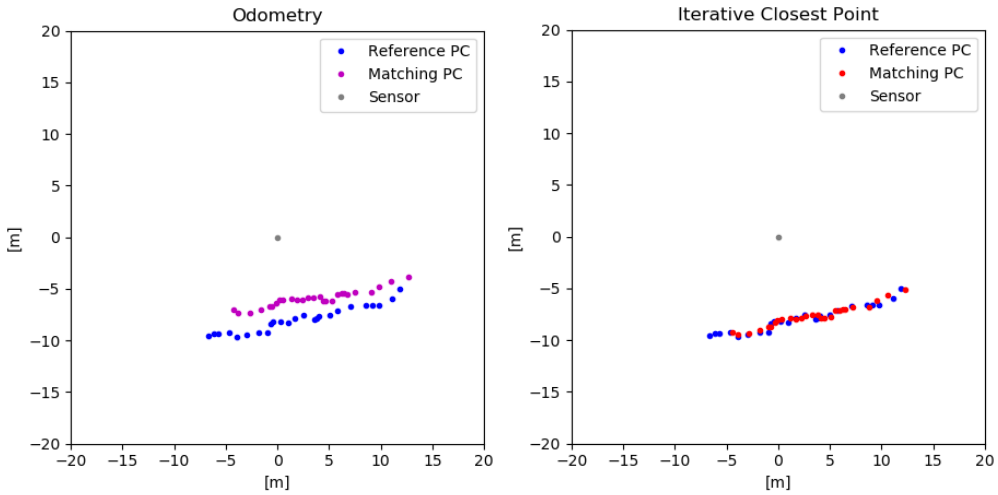

Figura 10.9: Dataset (Galera) exemple de resultat correcte (ICP)

Feliu	Encerts	Erronis	Retards	Millor	Semblants
GICP	48,81%	52,5%	57,14%	52,5%	51,61%
GMM	51,19%	47,5%	42,86%	47,5%	

Taula 10.3: Comparacions amb el dataset de Sant Feliu en percentatges.

A les figures (10.3.0.2) i (10.3.0.2) es mostren casos a on només la tècnica GICP ha trobat un resultat correcte pels diferents data sets. La tècnica GICP ha aconseguit registrar els escanejos, mentre que la tècnica GMM Registration dóna resultats incorrectes.

A les figures (10.3.0.2) i (10.3.0.2) es mostren casos a on només la tècnica GMM Registration ha trobat un resultat correcte pels diferents data sets. La tècnica GMM Registration ha aconseguit registrar els escanejos, mentre que la tècnica GICP dóna resultats incorrectes.

A les figures (10.3.0.2) i (10.3.0.2) es mostren casos a on les dues tècniques han produït resultats correctes pels diferents data sets.

Els resultats pels data sets complerts es poden consultar l'annex (15). En aquesta memòria només s'han seleccionat els resultats més rellevants.

En conclusió, observem que la tècnica implementada té un comportament molt semblant a la tècnica GICP. No obstant, la nostra tècnica és capaç d'es-

Galera	Encerts	Erronis	Retards	Millor	Semblants
GICP	51,91%	45,33%	55,26%	52,06%	39,15%
GMM	48,09%	54,67%	44,74%	47,94%	

Taula 10.4: Comparacions amb el dataset de la Galera en percentatges.

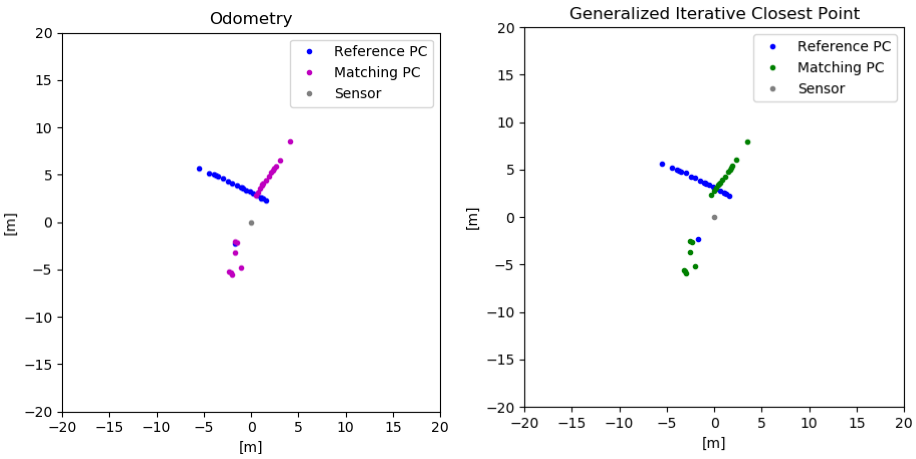

Figura 10.10: Dataset (Sant Feliu) exemple de retard amb GICP

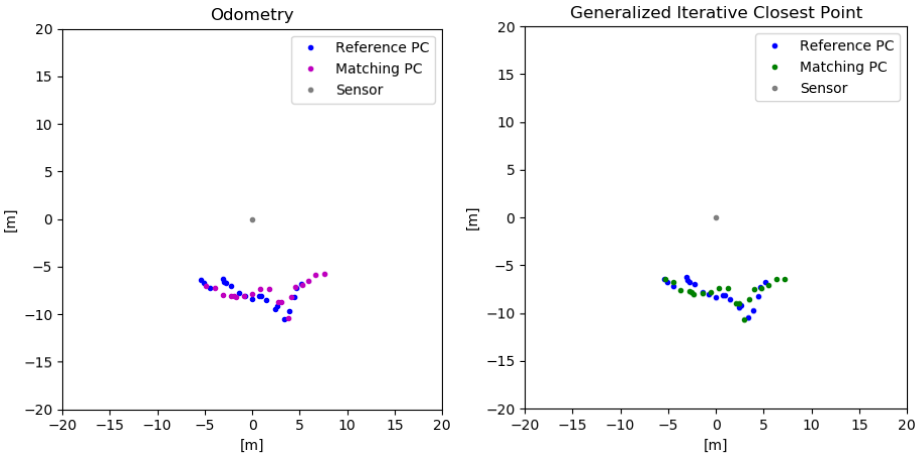

Figura 10.11: Dataset (Galera) exemple de retard amb GICP

timar de manera automàtica la incertesa del registre. Aquesta estimació de la incertesa no es coneix per cap implementació de la tècnica GICP. Per tant, la nostra implementació manté els resultats del GICP i aporta un mecanisme d'estimació de la incertesa.

10.4 Ampliació de la llibreria Front-Ends

10.4.1 Front-End basat en l'algorisme K-means

Per posar en marxa la funció 'k_means_constructor()' implementada segons es mostra a la secció 9.4.1, primer, s'han realitzat uns experiments per determinar la convergència de l'algorisme. L'algorisme pot convergir perquè la

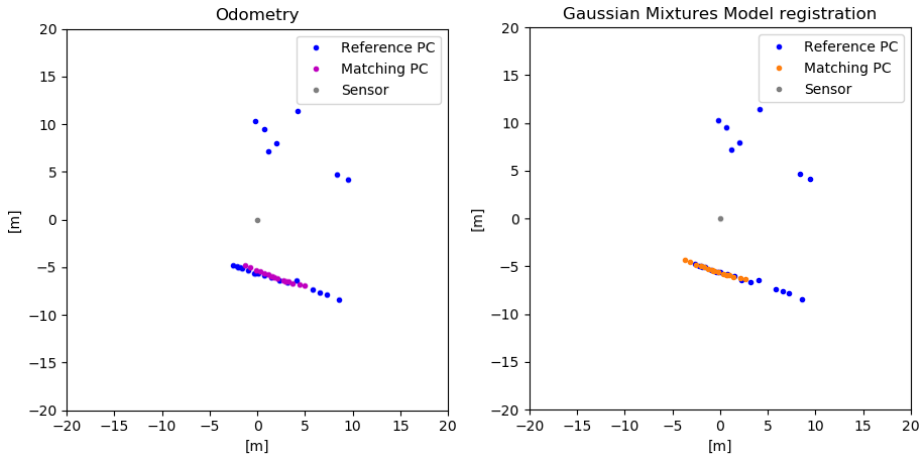

Figura 10.12: Dataset (Sant Feliu) exemple de retard amb GMM

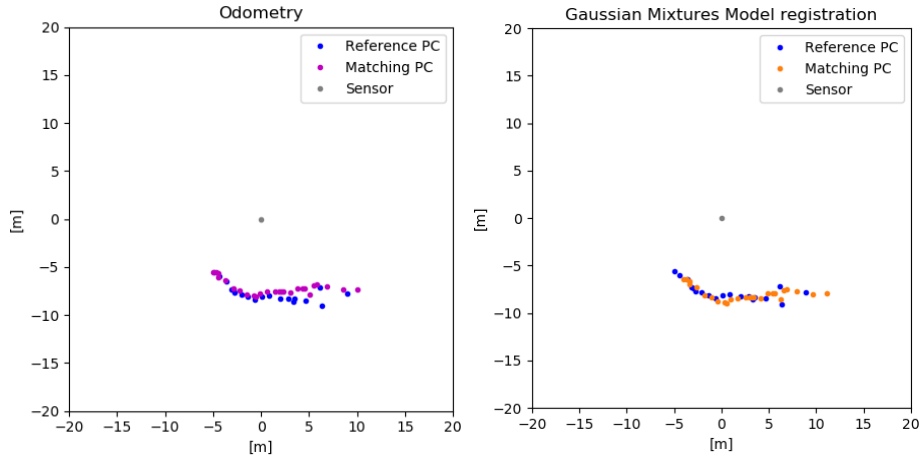

Figura 10.13: Dataset (Galera) exemple de retard amb GMM

inèrcia de la solució no millora o perquè les assignacions de la matriu de responsabilitats no canvien. La inèrcia de la solució es calcula segons l'equació 5.1. Per quantitzar els canvis en les assignacions calculem la norma de Forbenius de la matriu resultat de restar la matriu d'assignacions de la iteració anterior amb la matriu d'assignacions actual. A les figures (10.22) es pot veure l'evolució d'aquests indicadors per a 300 iteracions de l'algorisme de k-means. Com es veu, l'evolució entre els dos indicadors és semblant i per això a la implementació final només utilitzem la inèrcia com a criteri de parada. A partir de l'anàlisi dels resultats fixem un llindar de 50. Tal com es veu a les figures (10.23) aquest llindar és suficient per assegurar una bona eficiència computacional.

Un cop establert el criteri de parada provem el Front-End amb les dades

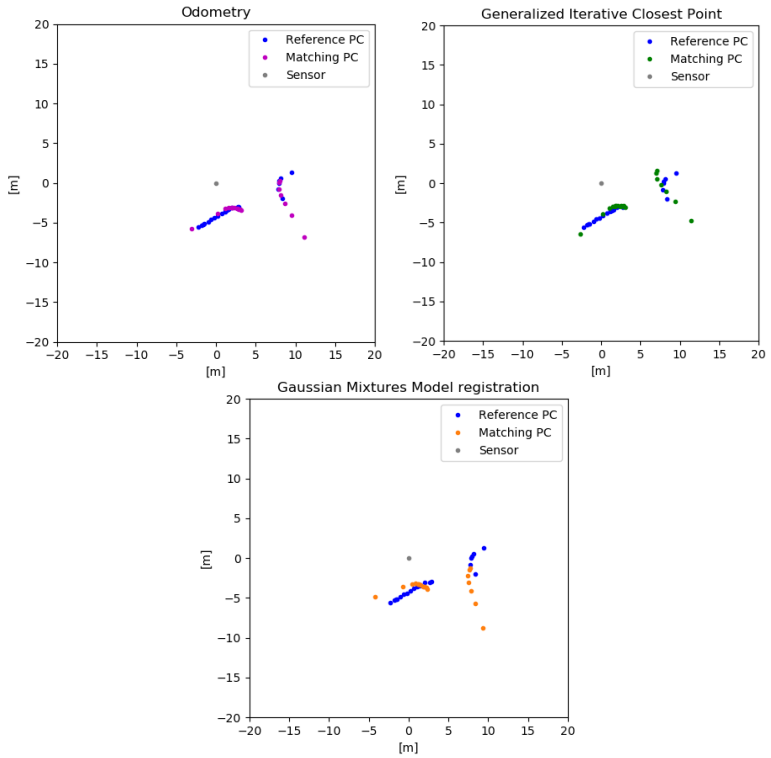

Figura 10.14: Dataset (Sant Feliu) exemple de resultats incorrectes GICP i GMM

dels data sets presentats a la secció (10.1).

Els paràmetres introduïts per a la realització de les proves han estat els següents:

- Scan: núvol de punts en dues dimensions.
- Points_threshold: el nombre mínim s'ha fixat a 2.
- k: el nombre de components s'ha fixat a 5 components per solució.
- n_init: El nombre de vegades que es realitzara l'algorisme de K-means s'ha fixat a 10 iteracions.
- max_iter_lloyd: El nombre de voltes s'ha fixat com a 50 iteracions.
- calc_cov: S'han realitzat diverses proves amb valor true o false.

Els resultats obtinguts en aplicar aquesta tècnica amb aquests paràmetres han estat satisfactoris, uns exemples dels resultats obtinguts son les figures (10.24) i (10.25) dels data sets de Sant Feliu de Guixos i de la Galera respectivament. Com es pot veure, aplicant aquesta tècnica amb un nombre de k =

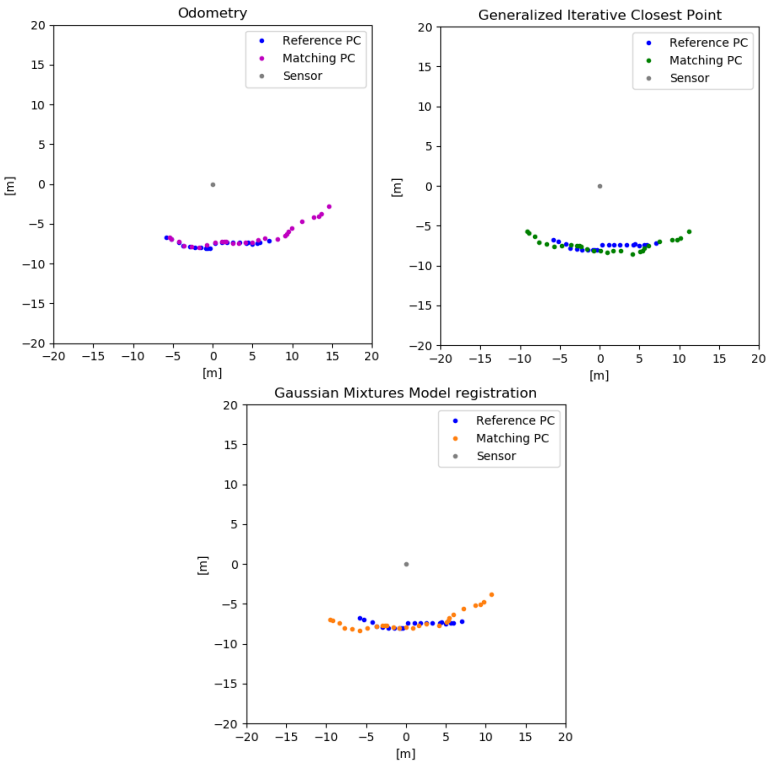

Figura 10.15: Dataset (Galera) exemple de resultats incorrectes GICP i GMM

5, fa que l'algorisme generi 5 components diferents que primer han obtingut uns valors aleatoris i posteriorment han assolit uns valors molt bons per a realitzar la representació de les dades. En les figures (10.26) i (10.27) es pot veure que només una part de les figures ha mostrat un resultat mentre que l'altre s'ha quedat en blanc. En aquests casos no s'ha generat un error. El que ha succeït és que el resultat té un valor tan petit que no pot ser representat i llavors el Matplotlib ([Matplotlib 022]) no el pot generar.

10.4.2 Front-End basat en l'algorisme d'Expectació-Maximització

Per posar en marxa la funció 'em_constructor()' implementada segons es mostra a la secció 9.4.2, primer, s'han realitzat uns experiments per determinar la convergència de l'algorisme. L'algorisme pot convergir perquè el log likelihood de la solució no millora o no convergir perquè la solució es degenera. El log likelihood de la solució es calcula segons l'equació 5.7.

A la figura (10.4.2) es pot veure l'evolució d'aquest indicador per a 50 iteracions de l'algorisme d'EM. Com es veu, l'evolució del log likelihood arriba a un punt en el que segueix amb el mateix valor fins al final, el que significa que no troba una millor solució. A partir de l'anàlisi dels resultats fixem un

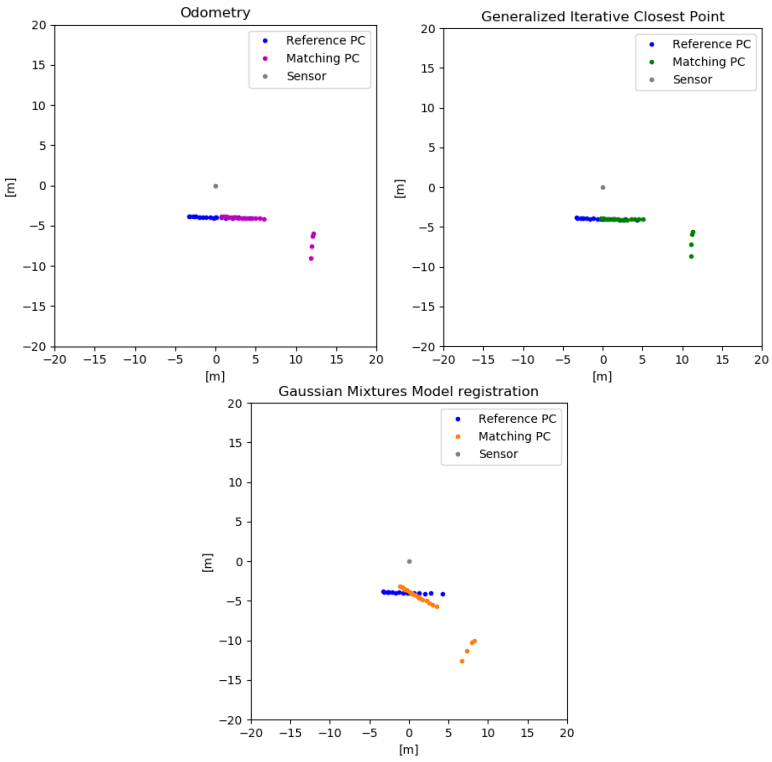

Figura 10.16: Dataset (Sant Feliu) exemple de resultat només correcte GICP

llindar de 50. Tal com es veu a les figures (10.4.2) i (10.4.2) aquest llindar és suficient per assegurar una bona eficiència computacional i aplicant el criteri de parada del log likelihood fa que convergi molt abans del llindar.

Un cop establert el criteri de parada provem el Front-End amb les dades dels data sets presentats a la secció 10.1.

Els parametres introduits per a la realitzacio de les proves han estat els següents:

- Scan: núvol de punts en dues dimensions.
- Points_threshold: el nombre mínim s'ha fixat a 2.
- k: el nombre de components s'ha fixat a 5 components per solució.
- n_init: El nombre de vegades que es realitzara l'algorisme de K-means s'ha fixat a 10 iteracions.
- max_iter_lloyd: El nombre de voltes s'ha fixat com a 50 iteracions.
- init_kmeans: S'han realitzat diverses proves amb valor true o false.

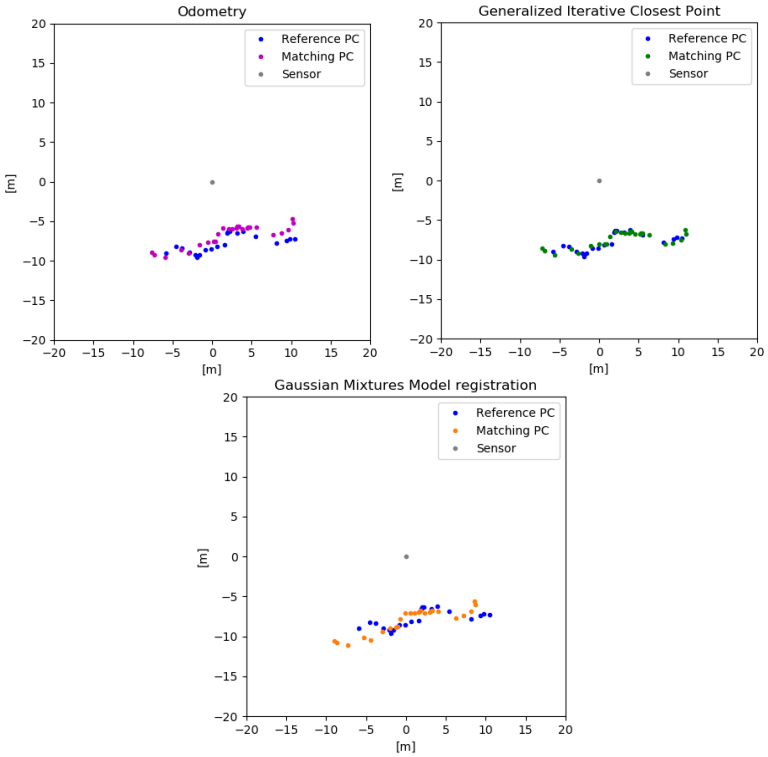

Figura 10.17: Dataset (Galera) exemple de resultat només correcte GICP

Els resultats obtinguts en aplicar aquesta tècnica amb aquests paràmetres han estat satisfactoris, uns exemples dels resultats obtinguts són les figures (10.4.2) i (10.4.2) dels data sets de Sant Feliu de Guixos i de la Galera respectivament. Com es pot veure, aplicant aquesta tècnica amb un nombre de $k = 5$, fa que l'algorisme generi 5 components diferents que primer han obtingut uns valors aleatoris i posteriorment han assolit uns valors molt bons per a realitzar la representació de les dades.

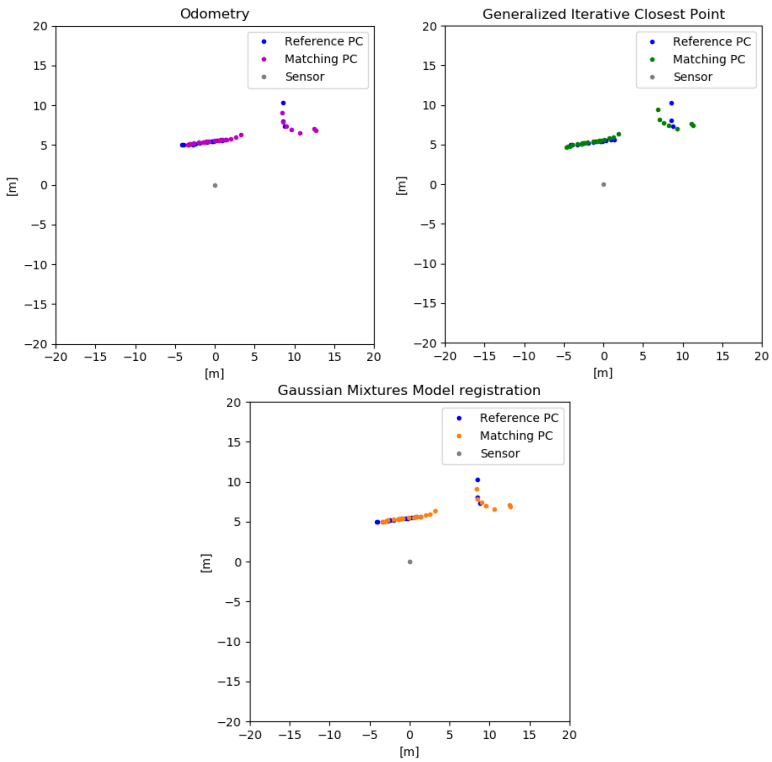

Figura 10.18: Dataset (Sant Feliu) exemple de resultat només correcte GMM

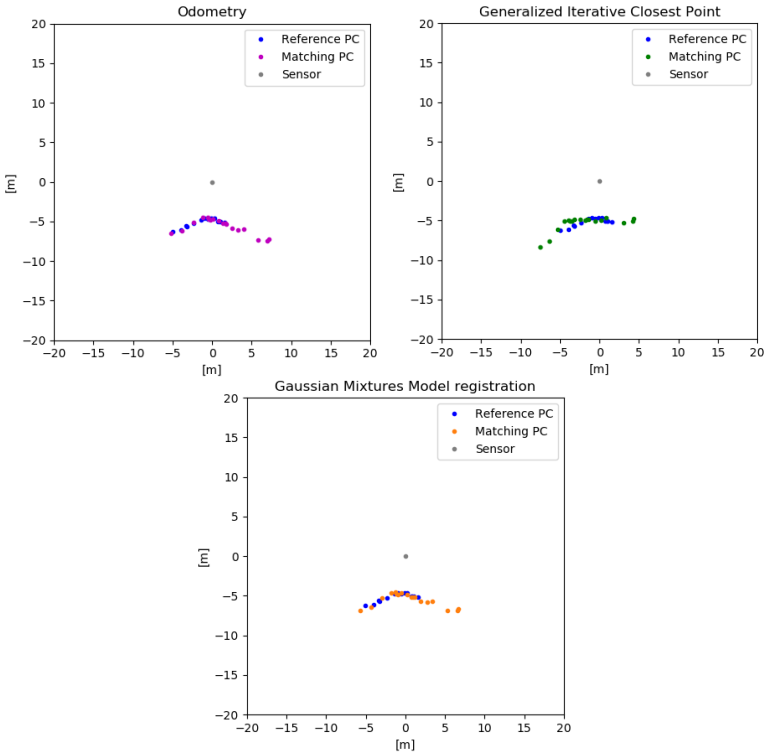

Figura 10.19: Dataset (Galera) exemple de resultat només correcte GMM

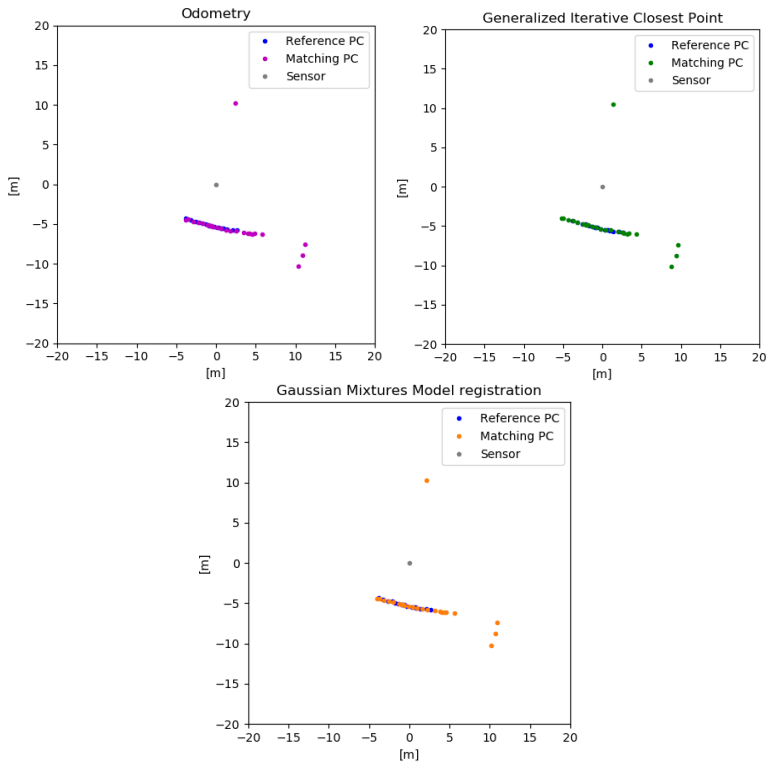

Figura 10.20: Dataset (Sant Feliu) exemple de resultats tots dos correctes

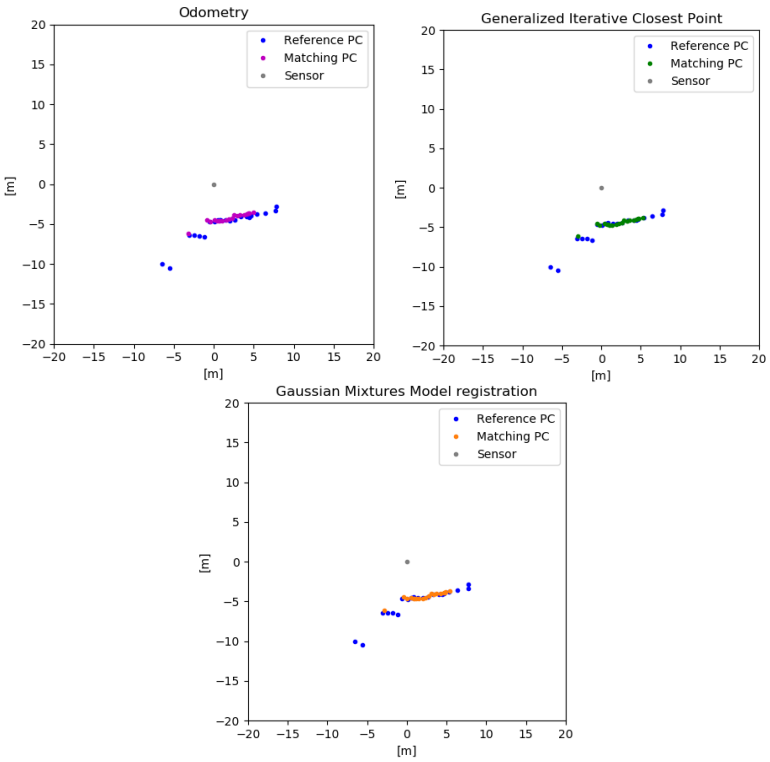

Figura 10.21: Dataset (Galera) exemple de resultats tots dos correctes

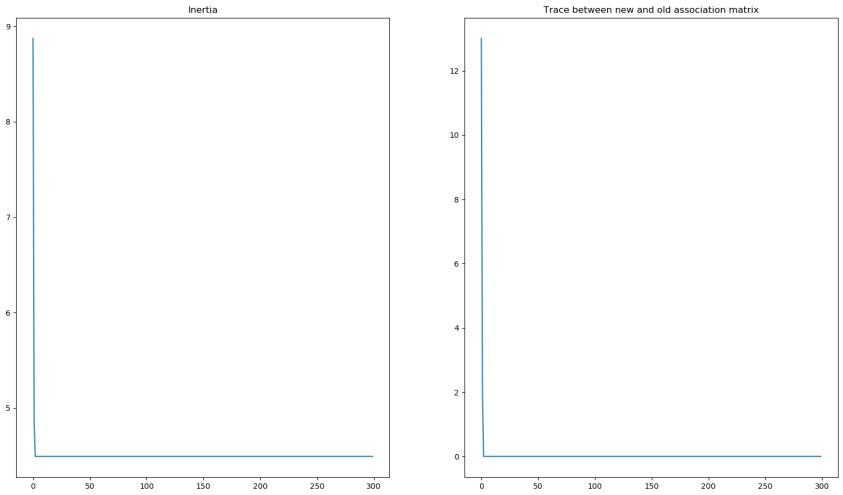

Figura 10.22: Seguiment de l'inertia i traça entre matrius nova i antiga amb màxim de 300 iteracions.

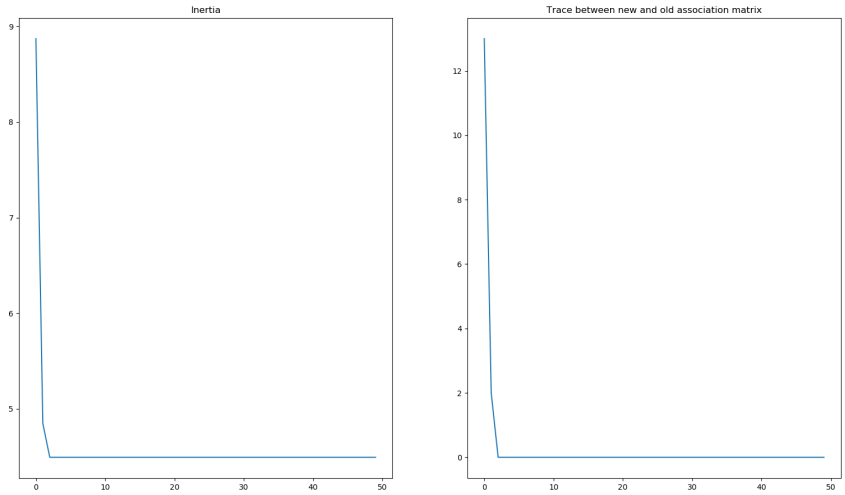

Figura 10.23: Seguiment de l'inertia i traça entre matrius nova i antiga amb màxim de 50 iteracions.

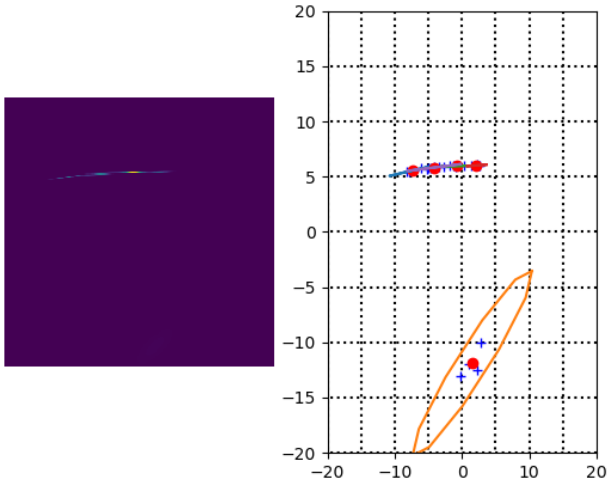

Figura 10.24: Dataset (Sant feliu) Plot de GMM fet amb constructor K-means.

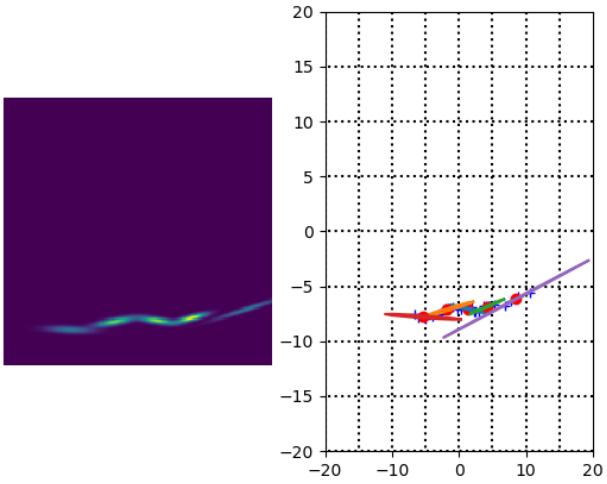

Figura 10.25: Dataset (Galera) Plot de GMM fet amb constructor K-means.

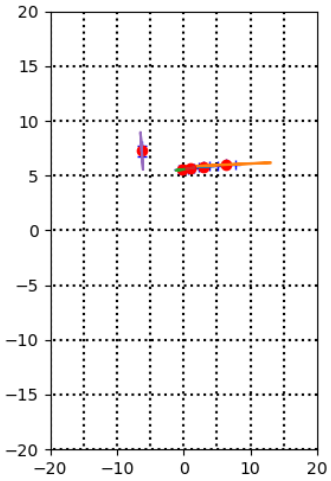

Figura 10.26: Dataset (Sant feliu) Plot de GMM fet amb constructor K-means resultat petit.

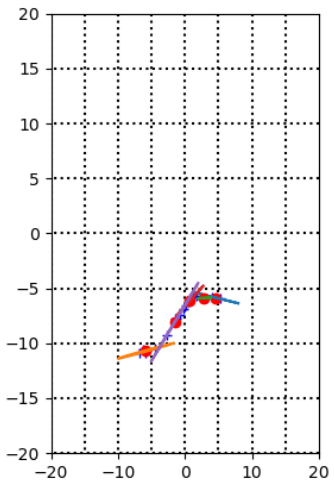

Figura 10.27: Dataset (Galera) Plot de GMM fet amb constructor K-means resultat petit.

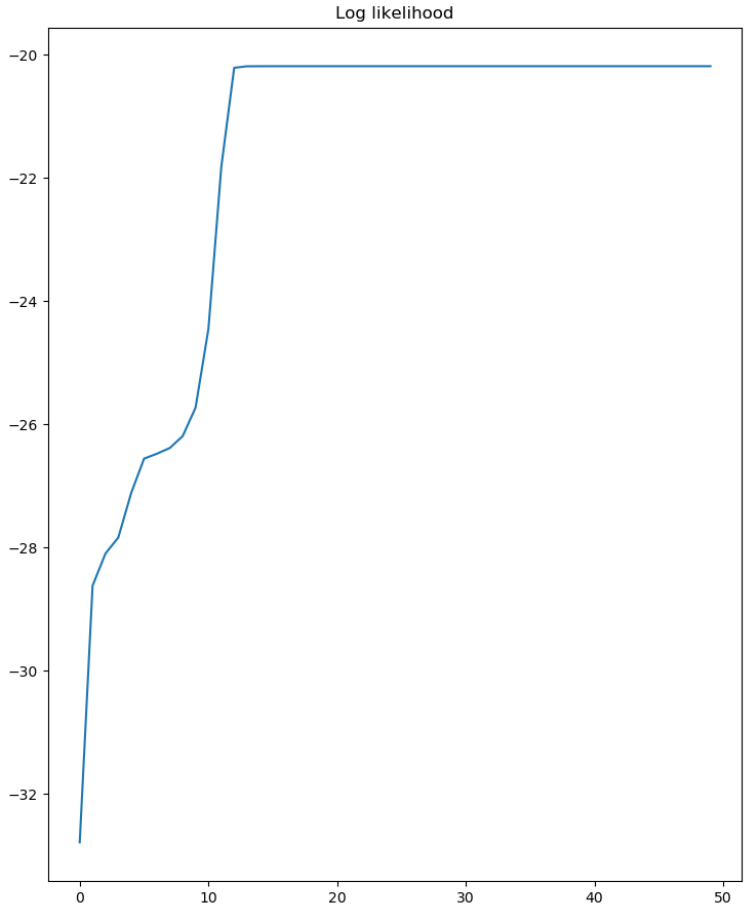

Figura 10.28: Plot d'un seguiment de log likelihood del constructor EM.

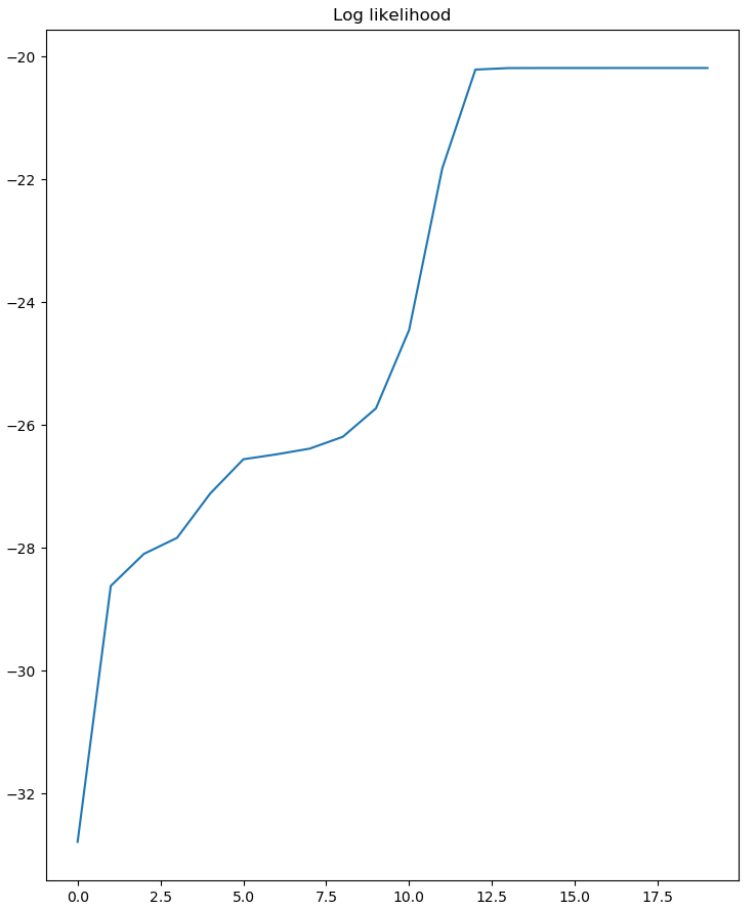

Figura 10.29: Dataset (Galera) Plot d'un seguiment de log likelihood del constructor EM on s'atura per possible cas de degeneració o perquè els valors són quasi iguals.

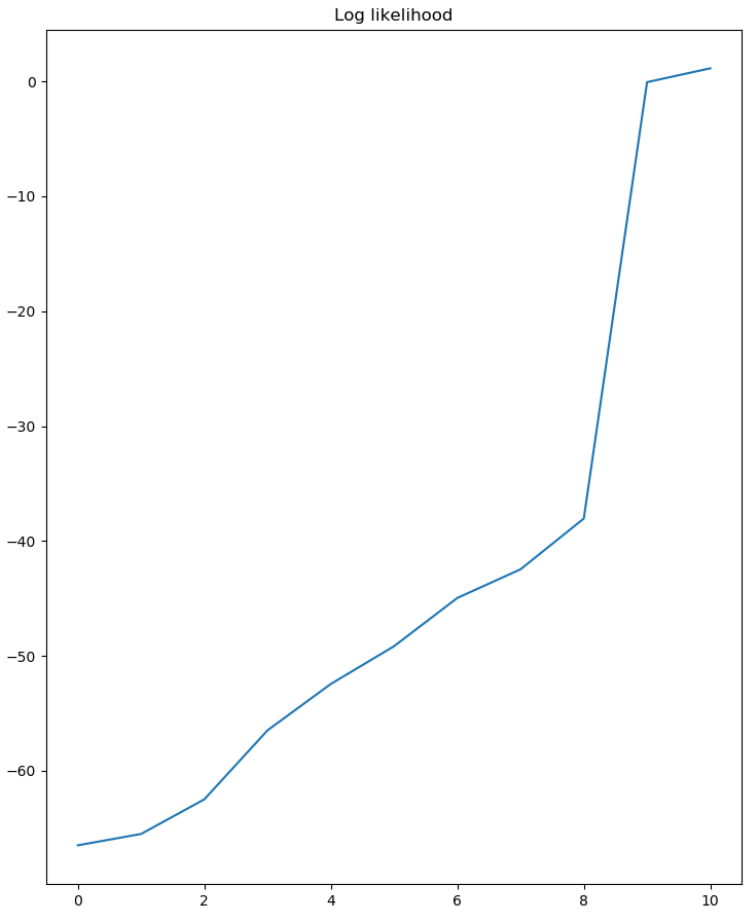

Figura 10.30: Dataset (Galera) Plot d'un seguiment de log likelihood del constructor EM on s'atura per possible cas de degeneració o perquè els valors són quasi iguals.

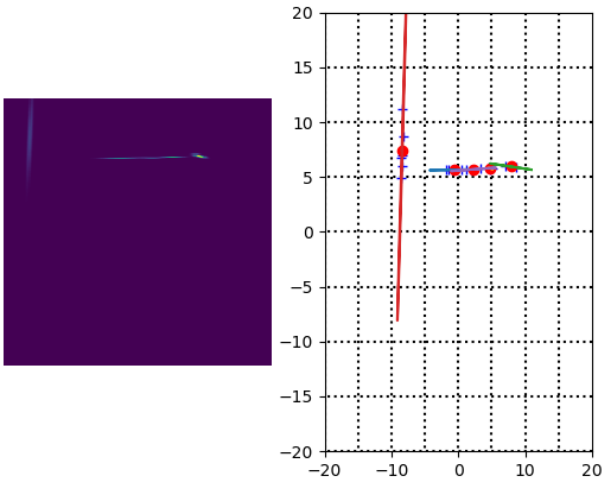

Figura 10.31: Dataset (Sant feliu) Plot de GMM fet amb constructor EM.

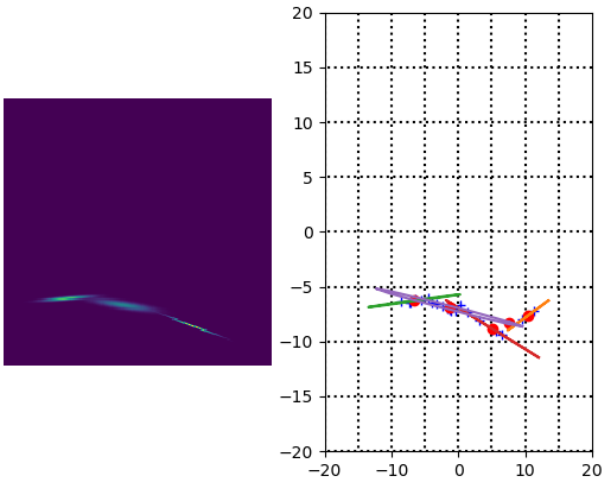

Figura 10.32: Dataset (Galera) Plot de GMM fet amb constructor EM.

Capítol 11
Conclusions

En aquest treball s'ha implementat una llibreria per al registre de núvols de punts adquirits amb sensors de rang acústic muntats en AUVs. Aquesta llibreria extrapola al món submarí i acústic la tècnica de registre de l'estat de l'art per a sensors làsers coneguda com a NDT. La llibreria s'ha implementat en llenguatge C++ per tenir una millor execució en temps real de les dades adquirides. S'està treballant a crear un repositori públic perquè el codi generat estigui disponible per altres investigadors o professionals del món submarí.

El projecte va començar fent lectura de l'article científic on es publica la primera versió de la tècnica NDT. Per facilitar la comprensió de la metodologia se'n va fer una primera implementació senzilla. A partir d'aquesta primera versió, es va definir l'estructura de classes de la llibreria basant-se en les diferents parts de codi generades. Aquesta estructura està formada per tres components conceptuals: el Front-end, el Mètode i el Solver.

Per a manejar els GMM, models probabilístics per a distribucions normals de dades, que es creïn i fer les crides pertinents, s'ha generat l'objecte GaussianMixturesModel. Aquest objecte enregistra les dades del GMM a través del mètode "set()", el mètode rep els diferents vectors necessaris per al GMM (vector amb les dades de π_k, vector amb les dades de μ_k, vector amb les dades de Σ_k).

El primer component anomenat front-end ha estat pensat com una llibreria, conté els constructors de les tècniques que crearan els GMM a retornar. Aquesta llibreria el que porta a cap es, fent una crida a un dels seus mètodes, farà la conversió d'un núvol de punts passat per paràmetres, i les dades que calguin, depenen del mètode cridat, a un GMM on hi hauran les dades ja enregistrades a dins.

Dins d'aquesta llibreria, s'ha definit la funció 'ndt_constructor()', aquest mètode rep un núvol de punts i aplicant la tècnica anomenada NDT genera un GMM. Per a generar un GMM la tècnica emprada en aquest constructor es basa en generar una graella per sobre els punts rebuts, aquests punts seran separats llavors en cel·les de la graella i a partir d'aquesta separació de dades, es comença a aplicar la tècnica per calcular la informació a enregistrar dins del GMM.

El component anomenat ScanMatchingMethods ha estat pensat per a ser un objecte pare, la seva funció és definir la funció de cost del problema de

registre, escrit com a problema d'optimització, i les seves derivades primera i segona (Jacobiana i Hessiana).

Tè diversos mètodes, 'compute_score()' per retornar el cost del problema, 'compute_score_and_gradient()' per retornar el cost i la primera derivada i el mètode 'compute_score_gradient_and_hessian()' que retorna tot.

De la interfície ScanMatchingMethods s'han implementat dues herències: PointsToDistribution2D i DistributionToDistribution2D. Aquests objectes duen a terme la mateixa tasca que el pare, es a dir, retornar el valor del likelihood i les seves derivades donada una transformada entre els dos núvols de punts. L'objecte PointsToDistribution2D implementa la tècnica P2D per escanejos bidimensionals, on es registra un escaneig representat en forma de núvol de punts contra un altre escaneig modelat per un GMM. En canvi, l'objecte DistributionToDistribution2D implementa la tècnica D2D per escanejos bidimensionals, on es registra un escaneig representat en forma de GMM contra un altre escaneig modelat per un GMM.

El component anomenat Solver ha estat pensat per a ser un objecte pare, la seva funció és la de resoldre un problema de computació on es calcula la transformació òptima a realitzar. Aquest objecte té un mètode principal, la funció 'compute_optimum()' que realitza la resolució del problema.

De la interfície Solver s'han implementat dues herències: NewtonMethod i el LineSearchNewtonMethod. L'objecte NewtonMethod implementa un solucionador basat en el mètode Newton pur. Donades la primera i la segona derivada de la funció de cost del problema, el mètode retorna directament el següent pas d'optimització. En canvi, l'objecte LineSearchNewtonMethod implementa un solucionador basat en el mètode de Newton amb un algorisme de Line Search per determinar una longitud òptima pel pas de minimització. Per fer-ho s'aplica una tècnica basada en comprovar les condicions de Wolfe.

Quan es completi la implementació de la llibreria, el repositori es vol fer públic perquè altres investigadors la puguin utilitzar. Per aquest motiu, des de l'inici s'ha treballat amb la documentació del codi que s'anava generant. Per fer-ho, s'ha fet servir el programa Doxygen que permet crear una bona documentació del codi ficant comentaris especials directament en el codi. D'aquesta manera es pot generar una documentació del codi concisa on mirant les capçaleres de les definicions es pot entendre que realitza cada apartat.

Fetes totes les implementacions descrites, es va fer una comparativa de la nostra llibreria amb altres llibreries públiques que implementen altres tècniques de l'estat de l'art, concretament, les tècniques ICP i GICP implementades a la Point Cloud Library. D'aquesta manera, es pot avaluar com es comporta la implementació que proposem. Per resoldre el problema de registre amb la

GMM Registration Library s'utilitza la funció "ndt_constructor()çom a Front-End. Pel que fa al mètode es vol fer un registre doble on primer s'executa un registre D2D per acostar-nos ràpidament a la solució i, a continuació, un registre P2D per refinar la solució. La tècnica P2D utilitza totes les dades de l'escaneig, sent més lenta, però assolint més precisió. Per fer el doble registre s'ha implementat la funció "GMM_register()", aquest mètode el que fa és agafar el millor resultat aplicant Solvers P2D i D2D. Per portar a cap aixó primer realitza amb el solver D2D una aproximacio més rapida, en cas de que trovi una solució realitza un altre aproximació amb els resultats obtinguts del D2D pero amb l'altre Solver P2D. En el cas que el D2D no arribi a una solucio es fa el P2D i en tots dos casos, sempre s'agafa el millor valor, sent el valor base el millor en cas de que cap trovi una solucio. Finalment, com a solucionador, s'utilitza l'objecte NewtonMethodLineSearch per aprofitar l'ajuda a la convergència que aporta l'algorisme de Line Search que inclou.

La comparativa es fa utilitzant dades reals obtingudes mitjançant sensors acustics que han estat muntats en un AUV, Vehicles Autònoms Submarins, aquestes plataformes robòtiques dotades de sistemes de percepció i control operen sota l'aigua i han efectuat unes tasques de localització, mapatge o planificació de trajectòria de manera totalment autònoma, és a dir, sense interacció amb cap altre sistema. S'ha ha muntat un sensor especial anomenat Mechanical Scaning Profiling Sonar, aquest sonar realitza un escobreig d'una zona amb un unic feix que va modificant l'angle constantment. Les dades han estat obtingudes de dos datasets, un dels datasets està adquirit en un entorn estructurat amb formes cartesianes i parets verticals, fent irrellevants els canvis de profunditat del robot. El dataset és correspon a una ruta en bucle entre uns blocs de formigó situats a l'espigó principal del port de Sant Feliu de Guíxols (Girona). L'altre dataset està adquirit en un entorn natural amb formes molt irregulars. El dataset es correspon a una volta i mitja a l'illot anomenat la Galera situat al Golfet del Cap de Creus (Girona).

Com que els datasets s'han adquirit amb l'AUV submergit, no s'ha disposat de senyal de GPS, ja que les ones electromagnètiques només penetren pocs centímetres dins de l'aigua i, per tant, no es disposa del ground truth de la trajectòria seguida pel robot.

D'aquests resultats no es poden extreure uns valors quantitatius per fer una comparació quantitativa amb altres tècniques en l'estat de l'art, però si es poden fer comparacions qualitatives. Si els resultats obtinguts en aplicar la tècnica es mostren d'una forma visual com figures, i posteriorment s'afegeixen els resultats en aplicar les tècniques en l'estat de l'art (dins de la mateixa figura per fer-ho més senzill), es possible realitzar en aquest cas una comparació que fos qualitativa entre la tècnica que s'ha implementat i les altres

tècniques a comprovar de l'estat de l'art.

Aquest codi es pot executar tant de forma directa amb el robot en funcionament com amb dades que han estat previament guardades.

La comparativa s'ha executat utilitzant el middleware ROS. Amb aquest programari s'ha utilitzat per realitzar una subscripcio a un node amb el que es pot rebre les dades que s'han guardat previament dels datasets de Sant Feliu de Guixols i la Galera.

Dels resultats de la comparativa es conclou que la tècnica ICP és la més simple en termes de equacions i que per a aquest motiu els resultats que s'obtenen no són tan bons com els que s'obtenen de la tècnica GICP, que aplica la teoria de la probabilitat i té un funcionament semblant al de la tècnica desenvolupada en aquest projecte. Per aquest motiu i perque també donava pitjors resultats (falla més), la tècnica ICP ha estat rebutjada. De les comparatives realitzades entre la tècnica GICP i la implementada en aquest projecte, s'ha vist que tant per un entorn estructurat (formes artificials) o un entorn desestructurat (formes naturals, pedres, etc.) les dues tècniques obtenen un resultat molt semblant. La tècnica GICP obté uns casos més que la de GMM on el resultat es millor, però no hi ha gaire diferencia si es mira en global tota la comparativa feta tant en el data set de Sant Feliu de Guixols com en el de la Galera. En conclusió, s'ha determinat a partir de les comparatives fetes que la tècnica GICP i la tècnica implementada en la llibreria són molt semblants, però amb la diferencia que la tècnica NDT retorna les dades de la incertesa, per lo que es podria debatre acabant amb que seria millor fer servir la que s'ha implementat per aquesta rao.

Feta la comparativa es proposa ampliar la llibreria de Front-Ends implementant altres algorismes que permeten fitar un GMM a un núvol de punts, diferents al Front-End proposat per la tècnica NDT. Concretament es proposen dues noves funcions que implementen algorismes procedents de la comunitat de l'aprenentatge automàtic: 'k_means_constructor()' i 'em_constructor()'. La primer funció aplica l'algorisme K-means i la segona aplica l'algorisme d'Expectació-Maximització.

La tècnica de classificació anomenada K-means té com a objectiu la classificació de núvols de N punts en K grups, on cada punt s'assigna només al grup més proper segons la distància euclídia. En canvi, la tècnica d'Expectació-Maximització és una tècnica iterativa que s'utilitza per optimitzar els paràmetres de models probabilístics que depenen de variables que no es poden observar, com seria el cas de les variables latents d'un GMM. És a dir, les variables que indiquen a quines components s'associa cada punt del núvol de punts. Per tant, la diferencia principal entre aquestes tècniques és que l'algorisme K-means assigna cada punt a només una component, mentre que

l'algorisme d'Exepctació-Maximització fa assignacions probabilístiques d'un punt a totes les components del model.

Per implementar aquests nous mètodes s'ha seguit un procediment incremental. Primer, es defineixen els objectes i les iteracions que ha de realitzar cada algorisme. Posteriorment, s'introdueixen les primeres equacions i es mostren els resultats d'haver aplicat aquestes operacions per així comprovar que els resultats siguin correctes. Es fa primer una operació i si el resultat és correcte es defineix la següent operació. Aquest procediment es segueix fins a arribar a definir totes les operacions de cada algorisme, tenint aixi la implementació base i funcionant de cada tècnica.

Un cop feta la primera implementació s'ha buscat per cadascuna de les funcions, quin es el seu criteri de parada. Per al K-means s'ha fixat com a criteri de parada la inèrcia i per a l'algorisme d'Expectació-Maximització el likelihood de la solució. Si en cap dels dos casos aquesta mesura no millora respecte la iteració anterior, significa que l'algorisme ha convergit i es pot trencar el bucle. Aplicant aquest criteri, els algorismes no necessiten fer el màxim nombre de voltes que s'ha definit per paràmetre, sinó que, pararan en convergir. D'aquesta manera, s'estalvia tot el temps que passaria l'algorisme fent càlculs i obtenint els mateixos resultats. Experimentalment veiem que per núvols de punts bidimensionals aquestes funcions triguin de l'ordre dels [100,400] ms. Sense criteri de parada trigaven de l'ordre dels segons.

Amb les noves funcions del Front-Ends s'ha aconseguit fitar GMMs més genèrics que els fitats amb el constructor NDT, ja que no s'imposa l'estructura cartesiana que utilitza la tècnica NDT. És a dir, en comptes de seguir una graella, s'utilitza un nombre fix de components sense una estructura prèvia. Una millora de les noves funcions respecte la tècnica NDT és que es pot deixar fixat un nombre de components a crear i així tenir un nombre estable de clústers en cada solució trobada. En el cas de la tècnica NDT aquesta opció no existeix perquè, en realitzar la comprovació de nombre mínim de punts, eliminarà les components que no compleixin aquesta regla. Això implica que el model tingui un nombre irregular de components entre totes les solucions que trobi aquesta tècnica.

En conclusió, en aquest projecte s'ha realitzat la implementació d'una llibreria en C++ basada en la tècnica NDT. Alhora, s'aporten dades de la seva comparativa amb altres tècniques en l'estat de l'art. Finalment, s'ha ampliat la llibreria de funcions Front-End implementant els algorismes de K-means i d'Expectació-Maximització.

Capítol 12

Treball futur

La llibreria s'ha deixat en un estat de funcionament semi-eficient, es realitzen les operacions correctament i s'obtenen resultats. No obstant, encara queda feina per tenir-la finalitzada de cares a fer-la pública. Per aquest motiu hi ha certs punts que es poden realitzar com a treball futur per a la millora o ampliació de la llibreria:

- Fer un estudi de temps amb els nous mètodes que s'han afegit en el Front-End i fer les millores necessàries en quant eficiència de memòria i temps d'execució.
- Implementar un Front-End basat en GMM bayesians, que permeti determinar automàticament el nombre de components necessàries pel GMM.
- Els Front-Ends implementats s'han templaritzat segons la dimensió dels punts però només s'han testejat amb núvols de punts bidimensionals. Caldria fer-ne l'experimentació tridimensional.

Aquestes són unes idees per a realitzar com a treball futur, ja fos jo mateix continuant amb el desenvolupament i l'expansió de la llibreria, com si fos un altre estudiant que volgués realitzar el seu TFG amb VICORB. El codi ha estat documentat perquè el pugui continuar una persona que no l'ha tocat anteriorment.

Bibliografia

[Atom 022] Atom. *atom*, (Accedit: Abril 2022). Disponible a https://atom.io/. (Citat a la plana 22.)

[Bailey 2006] T. Bailey and H. Durrant-Whyte. *Simultaneous localization and mapping (SLAM): part II*. IEEE Robotics & Automation Magazine, vol. 13, pages 108–117, 2006. (Citat a la plana 12.)

[Biber 2003] P. Biber and W. Strasser. *The normal distributions transform: a new approach to laser scan matching*. Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003), vol. 3, pages 2743–2748, 2003. (Citat a la plana 14.)

[Bierlaire 2015] M. Bierlaire. Optimization: Principles and algorithms. EPFL Press, 2015. (Citat a les planes 15 and 45.)

[Bishop 2006] Christopher M. Bishop. Pattern recognition and machine learning. Springer, 2006. (Citat a les planes 15 and 16.)

[Bitbucket] Bitbucket. *Built for professional teams*. Disponible a https://bitbucket.org/. (Citat a la plana 21.)

[Boost 022] Boost. *boost*, (Accedit: Maig 2022). Disponible a https://www.boost.org/. (Citat a la plana 22.)

[Durrant-Whyte 2006] H. Durrant-Whyte and T. Bailey. *Simultaneous localization and mapping: part I*. IEEE Robotics & Automation Magazine, vol. 13, pages 99–110, 2006. (Citat a la plana 12.)

[Eigen] Eigen. *C++ template library for linear algebra*. Disponible a https://eigen.tuxfamily.org/index.php?title=Main_Page. (Citat a les planes 22 and 60.)

[Electron 022] Electron. *electron framework*, (Accedit: Abril 2022). Disponible a https://www.electronjs.org/. (Citat a la plana 21.)

[Gedit 022] Gedit. *gedit*, (Accedit: Abril 2022). Disponible a https://wiki.gnome.org/Apps/Gedit. (Citat a la plana 22.)

[Generalized-ICP 022] Generalized-ICP. *Generalized-ICP*, (Accedit: Abril 2022). Disponible a https://www.robots.ox.ac.uk/~avsegal/resources/papers/Generalized_ICP.pdf. (Citat a les planes 13, 46 and 62.)

[Git 022] Git. *git*, (Accedit: Abril 2022). Disponible a https://git-scm.com/. (Citat a la plana 21.)

[Gnuplot 022] Gnuplot. *gnuplot*, (Accedit: Abril 2022). Disponible a http://www.gnuplot.info/. (Citat a la plana 23.)

[ICP 022] ICP. *A method for registration 3-D shapes*, (Accedit: Abril 2022). Disponible a https://www.robots.ox.ac.uk/∼avsegal/resources/papers/Generalized_ICP.pdf. (Citat a les planes 12, 46 and 62.)

[Jiang 2011] B. Jiang and B. C. Vemuri. *Robust point set registration using Gaussian Mixture Models*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 33, pages 1633–1645, 2011. (Citat a la plana 14.)

[Matlab 022] Matlab. *matlab*, (Accedit: Maig 2022). Disponible a https://www.mathworks.com/products/matlab.html. (Citat a la plana 23.)

[Matplotlib 022] Matplotlib. *matplotlib-cpp*, (Accedit: Abril 2022). Disponible a https://github.com/lava/matplotlib-cpp. (Citat a les planes 22, 49 and 70.)

[OpenGL 022] OpenGL. *opengl*, (Accedit: Maig 2022). Disponible a https://www.opengl.org/. (Citat a la plana 23.)

[PCL 022] PCL. *The Point Cloud Library (PCL) is a standalone, large scale, open project for 2D/3D image and point cloud processing.*, (Accedit: Maig 2022). Disponible a https://pointclouds.org/. (Citat a les planes 22 and 46.)

[Python 022] Python. *python*, (Accedit: Abril 2022). Disponible a https://www.python.org/. (Citat a la plana 22.)

[ROS] Open Robotics ROS. *ROS - Robot Operating System (Instal·lació en Ubuntu)*. Disponible a http://wiki.ros.org/noetic/Installation/Ubuntu/. (Citat a les planes 22 and 60.)

[Stoyanov 2012] T. Stoyanov, M. Magnusson, H Andreasson and A. J. Lilienthal. *Fast and accurate scan registration through minimization of the distance between compact 3D NDT representations*. The International Journal of Robotics Research, vol. 31, pages 1377–1393, 2012. (Citat a la plana 14.)

[SublimeText 022] SublimeText. *sublime text*, (Accedit: Abril 2022). Disponible a https://www.sublimetext.com/. (Citat a la plana 22.)

[VisualSC 022] VisualSC. *visual studio code*, (Accedit: Abril 2022). Disponible a https://code.visualstudio.com/. (Citat a les planes 21 and 22.)

Capítol 14

Annex A: Repositori

En aquest annex, s'afegeix tan l'enllaç que porta cap al repositori que conté tot el codi del projecte com l'enllaç de la documentació que s'ha generat amb Doxygen.

Un cop dins, es podrà veure tota la informació referent a aquest repositori, les diferents branques, còmits realitzats, modificacions fetes i el mateix codi.

Pàgina de Bitbucket on es troba guardat el repositori: NDT_ScanMatching_CPP

Dins del Bitbucket es pot escollir quina branca es vol mirar, d'aquestes branques hi ha tres que són a on s'han guardat els diferents codis que s'han desenvolupat.

- Branca anomenada 'part_conjunta', en aquesta branca s'ha desat tot el codi base per al funcionament de la llibreria.
- Branca anomenada 'iterative_closest_point', en aquesta branca s'ha desat totes les modificacions i proves, respecte a la llibreria base, que s'han fet per a dur a terme la comparativa entre tècniques.
- Branca anomenada 'front_end', en aquesta branca s'ha desat les noves definicions dels constructors generats en l'ampliació del projecte.

Enllaç web per a accedir a la documentació generada del projecte amb Doxygen: DOCUMENTACIÓ_NDT_ScanMatching_CPP

Per a poder veure la documentació, cal obrir el fitxer anomenat 'index.html' que es troba dins de l'enllaç.

Capítol 15

Annex B: Resultats estesos

En aquest annex, s'introdueix un enllaç que porta cap a una carpeta en el Google Drive, on s'han guardat en subcarpetes tots els resultats obtinguts en realitzar diferents proves.

Pàgina de Google Drive on s'han guardat els resultats: RESULTATS TFG

Capítol 16

Manual d'usuari i/o instal·lació

16.1 Instal·lació de les dependències

16.1.1 ROS

16.1.1.1 Instal·lació del software

```
1) sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu
   $(lsb_release -sc) main" >
   /etc/apt/sources.list.d/ros-latest.list'
2) sudo apt install curl
3) curl -s
   https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc
   | sudo apt-key add -
4) sudo apt install ros-noetic-desktop-full
5) source /opt/ros/noetic/setup.bash
6) echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
7) source ~/.bashrc
8) sudo apt install python3-rosdep python3-rosinstall
   python3-rosinstall-generator python3-wstool build-essential
9) sudo apt install python3-rosdep
10) sudo rosdep init
11) rosdep update
```

16.1.1.2 Creació del workspace

```
1) mkdir -p ~/catkin_ws/src
2) cd ~/catkin_ws/
3) catkin_make
4) source devel/setup.bash
```

16.1.2 ROSMON

```
1) sudo apt install ros-${ROS_DISTRO}-rosmon
2) source /opt/ros/${ROS_DISTRO}/setup.bash
```

16.1.3 CATKIN TOOLS

```
1) sudo apt-get install pip
2) sudo pip3 install -U catkin_tools
```

16.1.4 Visual Studio Code

```
1) Obrir un programa en Ubuntu anomenat Ubuntu Software
2) Buscar el programa Visual Sudio
3) Descarregar la versio anomenada code
```

16.1.5 Eigen

```
1) Primer de tot cal descarregar i descomprimir la versio 3.4.0
    1.1) Aquesta versio es pot baixar amb aquest link
        "https://gitlab.com/libeigen/eigen/-/archive/3.4.0/eigen-3.4.0.tar.gz"
2) Obrir terminal dintre de la carpeta i executar aquestes
    comandes:
    2.1) mkdir build_dir
    2.2) cd build_dir
    2.3) cmake ../../eigen-3.4.0
    2.4) sudo make install
```

16.1.6 Boost

```
1) Anar a "https://www.boost.org/users/download/", descarregar el
    programa i descomprimir
```

```
2) Afegir en el includePath del fitxer json al vscode el nou path
   cap a la carpeta a on ha estat descomprimit
```

16.1.7 Point Cloud Library

```
1) Per a descarregar aquesta llibreria cal executar aquesta
   comanda per terminal:
   1.1) sudo apt install libpcl-dev
```

16.1.8 Posada en marxa

```
1) Crear les carpetes "icp_plots" i "ndt_plots" dins de la
   carpeta "catkin_ws/logs"
2) Descarregar dins de la carpeta "/catkin_ws/src" el repositori
   "ndt_scanmatching_cpp" amb el projecte
3) Si han estat generades les carpetes "build" i "devel" dintre
   de catkin_ws han de ser esborrades
4) Executar la comanda "catkin build" en un terminal amb path cap
   a la carpeta "catkin_ws"
5) Un cop realitzat aquests passos, obrint el repositori del
   projecte amb el programa Visual Studio, obrir el fitxer JSON
   (els fitxers JSON es troben en el directori vscode) amb les
   propietats i afegir en la propietat de "includePath" les
   linies:
   5.1) "/usr/local/include/eigen3",
   5.2) "/opt/ros/noetic/include",
   5.3) "~/catkin_ws/devel/include",
   5.4) "/usr/include/pcl-1.10",
   5.5) "/usr/include/python3.8"
```

16.2 Manual d'usuari

Un cop realitzada la instal·lació de totes les dependències de la llibreria es procedirà a explicar com s'utilitza i quines dades es fan servir o es recomanen.

Aquesta llibreria s'executarà a partir de les definicions que es facin en el fitxer "scan_matcher.cpp", aquest fitxer es troba dins de la carpeta "/src"del repositori.

En aquest fitxer s'han definit tant els includes com els diferents namespaces que es fan servir amb l'Eigen i la classe principal ScanMatcher. Dins d'aquesta classe hi ha creat el mètode "get_scan", a on s'afegiran les definicions i execucions dels diferents objectes i mètodes que vulguem.

Abans de crear cap objecte de la llibreria és necessari realitzar uns procediments per a l'obtenció de les dades rebudes pel tòpic, les dades que s'han d'aconseguir són el núvol de punts i les dades de la posició de l'AUV.

16.2.1 Extracció de les dades rebudes pel tòpic

Per extreure el núvol de punts de les dades, primer es crea un vector de VectorDd d'Eigen, on la D indica la dimensió en què volem que es tractin les dades (2D o 3D).

```
std::vector<Eigen::Vector3d> scan3d (data.points.size());
std::vector<Eigen::Vector2d> scan2d (data.points.size());
```

Un cop definit el vector desitjat es defineix un bucle amb el qual es desaran les dades del dataset, per agafar un dels valors de les coordenades (XYZ) d'un punt qualsevol s'ha de seguir la sintaxi "data.points[num_punt].x", on la 'x' serà el valor que volem agafar i 'data' el nom de la variable rebuda per paràmetre al mètode "get_scan".

En el cas de les dades d'odometria de l'AUV són diferents, en aquest cas s'han de crear dos vectors tridimensionals d'Eigen, on es guardaran les dades de l'odometria en realitzar la crida al mètode de projection. Aquest mètode cal passar les dades de 'data.pose' perquè porti a cap els càlculs adients i retorni una tupla amb les dades ja extretes.

```
Eigen::Vector3d delta_odometry_x;
Eigen::Vector3d delta_odometry_P;
std::tie( delta_odometry_x, delta_odometry_P ) = get_SO2_projection(
    data.pose );
```

16.2.2 Utilització de la llibreria

En fer l'extracció de les dades, es pot començar a fer els tractaments que es vulguin realitzar.

L'objecte base per a començar és el GMM, per fer-ho hi ha diverses opcions. Si es vol fer per a dues dimensions, es deixarà fixat amb un dos en la declaració, i en cas de voler que sigui tridimensional s'establirà com a tres.

```
shared_ptr<GaussianMixturesModel<2>> gmm_2d;
```

```
shared_ptr<GaussianMixturesModel<3>> gmm_3d;
```

Un cop creat l'objecte, cal cridar un dels constructors del Front-end que s'encarregarà de crear d'una forma o l'altre segons el constructor que cridem, i les dades que se li proporcionin per paràmetre.

Hi ha tres constructors, per al moment, on tots creen un GMM aplicant una tècnica o un altre.

El constructor anomenat "ndt_constructor", crea el GMM fent servir una graella per a realitzar les agrupacions dels punts. Aquest mètode rep per paràmetre el següent conjunt de dades:

- La dada que conté el núvol de punts, es tracta d'un vector de dues o tres dimensions segons el que s'ha definit prèviament, que contindrà tots els punts d'un únic scan.
- El 'points_threshold' defineix el nombre mínim de punts que cada grup ha d'assolir, en cas que un grup no compleixi amb aquesta regla serà exclòs del GMM, per al desenvolupament del projecte aquest valor s'ha fixat com 3, després de realitzar proves es va veure que 2 era molt petit i >4 retornava poca informació.
- La dada amb la mida que es volen definir les cel·les de la graella, depèn de la mida que es vulguin fer les cel·les pot sortir uns resultats o uns altres, en el cas d'aquest projecte la mida que s'ha fet servir és de 5.
- La ràtio que es vol mantenir entre els valors d'Eigen de les covariàncies, la que s'ha utilitzat es de 0.2 per aquest projecte.

Un altre constructor, anomenat "k_means_constructor", s'ha basat en la tècnica de K-means Clustering (5.3) per a generar el GMM en comptes d'utilitzar una graella com a base.

Aquest mètode té els mateixos paràmetres inicials que el "ndt_constructor", el scan i el 'points_threshold', però amb la diferència que s'han definit unes altres dades a continuació per a introduir:

- La dada del nombre de grups que es vol generar, és el nombre que es farà servir per a generar K grups, cal que sigui positiu i compleixi la norma (9.7).
- El nombre d'iteracions generals a realitzar, com que pot ser que no trobi un resultat algunes vegades per la randomització es defineix aquest valor perquè realitzi diversos cops l'algoritme i així intentar obtenir-ne algun.

- El nombre d'iteracions internes màximes que es realitzen mentre fa les operacions per trobar millors resultats.
- L'últim valor és de tipus booleà, s'utilitza per indicar si es vol o no realitzar el càlcul de les covariàncies, en aquest no és necessari ficar res, ja que per defecte està negat, però si es vol fer es pot ficar true.

L'últim constructor que s'ha definit és el de "em_constructor", aquest constructor com el de "k_means_constructor"aplica una altra tècnica diferent, però amb la diferència que aquest aplica la d'Expectation Maximization (5.4).

Té els mateixos paràmetres que el constructor anterior amb la diferència que en compte de tenir l'últim valor per si es vol calcular o no la covariància, es fa servir per dir si es vol agafar unes dades inicials fent servir el del K-means al principi de cada iteració global.

Exemples de les possibles crides:

```
gmm_2d = ndt_constructor(scan2d, 3, 5, 0.2);
gmm_2d = k_means_constructor(scan2d, 2, 5, 7, 50, false);
gmm_2d = em_constructor(scan2d, 2, 5, 10, 30, true);
```

Un cop creat el GMM, si es defineix una variable temporalment en la classe ScanMatcher per guardar els nous GMM que es creïn, es podrà executar els solvers a continuació, ficant un condicional per mirar si hi ha com a mínim un guardat.

Els solvers calen que es creïn previament els objectes de tipus P2D o D2D, i posteriorment definir el tipus de solver que es vulgui fer servir.

Per crear els objectes de Point to Distribution (P2D) o Distribution to Distribution (D2D) es poden fer de les següent forma.

Aquest és un exemple de com definir la distribució, en aquest cas de tipus P2D que després s'ha establert com un tipus ScanMatchingMethods amb el make_shared. El gmm_reference és la variable utilitzada per guardar l'anterior GMM creat i el vector fa referència al nou núvol de punts rebut.

```
PointsToDistribution2D exemple_p2d =
    PointsToDistribution2D(gmm_reference,
    make_shared<std::vector<Eigen::Vector2d>>(scan2d));
shared_ptr<ScanMatchingMethods<3>> smm_p2d =
    make_shared<PointsToDistribution2D>(exemple_p2d);
```

Aquest és un altre exemple però fet amb D2D:

```
DistributionToDistribution2D exemple_d2d =
    DistributionToDistribution2D(gmm_reference, gmm_current);
```

```
shared_ptr<ScanMatchingMethods<3>> smm_d2d =
    make_shared<DistributionToDistribution2D>(exemple_d2d);
```

Com a últim pas, queda crear el solver del tipus desitjat, sigui NewtonMethod, LineSearchNewtonMethod o Cholesky i després fer la transició a la classe solver pare.

Depenent del tipus de NewtonMethod que es vulgui crear, s'hauran de passar unes dades en concret per paràmetre.

Si es fa servir el NewtonMethod, els paràmetres que s'han de passar són els següents:

- El ScanMatchingMethod que es vol fer servir, creat prèviament.
- El nombre que indiqui la norma mínima del gradient de transformació per ser considerat òptim.
- El nombre per indicar el valor mínim de diferència entre el score actual i l'anterior per ser considerat òptim.
- El nombre màxim d'iteracions que realitzarà el NewtonMethod.

En els casos del LineSearch i Cholesky fan servir els mateixos que el NewtonMethod, però afegint uns paràmetres més al final:

- Els valors que es faran servir per a la Beta1 i la Beta2, van per separat.
- El valor per a la Gamma que es farà servir en el mètode de LineSearch.
- El valor per indicar el nombre màxim d'iteracions que farà el mètode LineSearch.

Exemple de creació de solver:

```
LineSearchNewtonMethod<3> lsnm (smm_d2d, 0.0001, 0.0001, 100,
    pow(10,-4), 0.99, 2.0, 100);
shared_ptr<Solver<3>> solv =
    make_shared<LineSearchNewtonMethod<3>>(lsnm);
```