SLAM amb imatges de Side Scan Sonar

Alumne: DANIEL MORENO LINARES
Directors: ANTONI BURGUERA, GABRIEL OLIVER
Universitat de les Illes Balears

L'objectiu central d'aquest Treball Final de Màster és determinar la trajectòria seguida per un Vehicle Autònom Submarí i construir un mosaic del fons marí a partir de dades proporcionades per sensors acústics (Side Scan Sonar i Doppler Velocity Log). Les tècniques de *Simultaneous Localization And Mapping* i *Extended Kalman Filter*-SLAM han demostrat ser clau en aquest context. Per tal d'assolir l'objectiu esmentat s'ha creat una llibreria que consisteix en un conjunt de classes en Matlab[1] per al processament d'aquestes dades i la implementació de l'algorisme EKF-SLAM amb *landmarks* extrets manualment. El càlcul de la trajectòria està ideat per fer-se en temps real mentre es mesura amb el DVL i el SSS. Durant la realització del treball s'ha experimentat amb les dades obtingudes de l'AUV EcoMapper durant el 2on Workshop del projecte TRIDENT al Port de Sóller.

1 INTRODUCCIÓ

Els AUV (*Autonomous Underwater Vehicles*) són d'utilitat per explorar i mapejar el fons marí, recollir dades de paràmetres ambientals o participar en tasques de reconeixement. Un aspecte necessari per dur a terme aquestes tasques és la localització que consisteix en determinar la posició de l'AUV, és a dir, tenir un model de l'entorn i la posició del robot dins del mateix.

Per poder dur a terme la localització els AUV van equipats amb sensors que permeten tenir informació de l'orientació, la profunditat, la velocitat a la que es mouen dins l'aigua i el temps transcorregut. Amb aquestes dades es pot procedir a aplicar un algorisme de *dead reckoning* (DR) per estimar la posició. Precisament el DR té origen en la navegació i consistia en anar fent triangulacions segons la brúixola, les velocitats i el temps transcorregut per predir on era el vaixell.

En l'actualitat els AUV disposen de sensors DVL (*Doppler Velocity Log*) per mesurar la velocitat dins l'aigua i brúixoles més precises. El fabricant proporciona un rang d'error pels instruments, de manera que es pot estimar l'error en el càlcul de la posició amb una certa probabilitat gràcies a algorismes com l'EKF (*Extended Kalman Filter*) [1]. L'EKF és un filtre gaussià adaptatiu, això vol dir que treballa amb distribucions normals per a les probabilitats de les mesures, estimacions i actualitzacions i és adaptatiu perquè durant les iteracions pondera dinàmicament les mesures i estimacions segons les probabilitats.

Degut a l'elevada autonomia dels AUV l'error acumulat en l'estimació de la posició va creixent de manera que el problema de l'SLAM (*Simultaneous Localization And Mapping*) es fa especialment important. L'SLAM consisteix en anar formant el mapa de l'entorn a la vegada que es calcula la localització de l'AUV amb la informació dels sensors i la reobservació del propi mapa. Una forma de definir el mapa és construir una llista de *landmarks*. Els *landmarks* són punts fàcilment identificables de l'entorn com formacions naturals ben definides o marques introduïdes per l'home. Quan s'observa un *landmark* se li assigna la posició i quan es torni a observar es pot corregir la posició en base a la observació anterior. En un primer plantejament aquests *landmarks* s'haurien de poder extreure de forma automatitzada a partir de les imatges submarines proporcionades pel sensor SSS (*Side Scan Sonar*).

L'SLAM basat en EKF (EKF-SLAM) és un dels algorismes més utilitzats en l'actualitat per la facilitat d'implementació i a l'hora d'entendre'l i s'utilitza per a estimar l'estat del robot amb les següents accions:

1. Estimant i actualitzant l'estat amb la informació dels sensors.
2. Actualitzant l'estat amb la reobservació de *landmarks*.
3. Afegint els nous *landmarks* a l'estat.

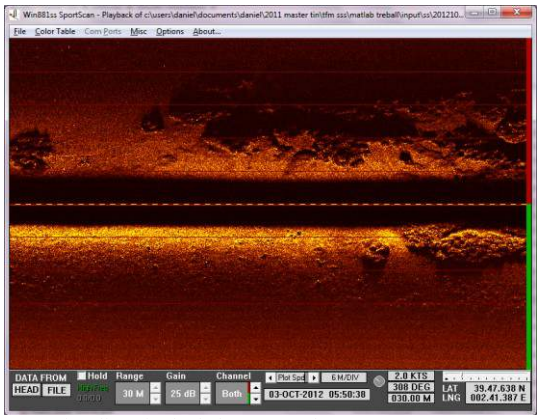

Figura 1 Visualització de dades SSS amb WIN881SS.

En aquest treball el Grup de Sistemes, Robòtica i Visió (SRV) es va proposar crear unes llibreries reaprofitables en Matlab per a aplicar l' EKF-SLAM en temps real sobre DVL i SSS aprofitant les dades recollides per Pablo Rodríguez del CSIC amb l'AUV EcoMapper durant el 2on workshop del projecte TRIDENT al Port Sóller a l'octubre del 2012. A la Figura 1 es pot observar un exemple de la representació gràfica de les dades del SSS amb el visor WIN881SS de Imagenex. Les dades del SSS estan pensades per ser visualitzades per un operari que pugui interpretar-les. Un dels nostres objectius serà poder arribar a formar mosaics submarins col·locant aquestes imatges sobre la trajectòria de manera autònoma i extreure'n els *landmarks* per formar el mapa.

Així doncs aquest treball parteix del conjunt de dades reals generades per l'EcoMapper com són els logs DVL amb les que reconstruir el recorregut i les dades del SSS que formen imatges acústiques del fons. A partir d'aquí els passos marcats per arribar a la solució proposada han estat els de la seqüència de la Figura 2:

1. Interpretar les dades DVL i reconstruir la trajectòria mitjançant un filtre EKF.
2. Reconstruir les imatges submarines a partir de les dades del SSS sobre les trajectòries per crear un mosaic del fons marí.
3. Extreure característiques dels mosaics submarins per obtenir un conjunt de *landmarks*. L'extracció es farà manualment.
4. Aplicar l'algorisme d'EKF-SLAM i comprovar que millora la localització del robot submarí.

Figura 2 Divisió de les tasques de recerca principals del TFM.

La detecció automàtica de punts característics a la imatge mosaic no ha estat possible per a un fons rocós d'aquestes característiques i s'ha optat per definir la posició dels *landmarks* manualment tal i com s'explica a la Secció 4.5.

[1] Es pot trobar el directori amb el codi font del TFM per fer proves a: http://dmi2.uib.es/burguera/TFMSSS/

La resta del treball s'estructura de la següent manera. A la Secció 2 es fa un repàs als treballs previs de robòtica sobre localització i mapeig simultani, l'algorisme EKF-SLAM i la detecció de característiques en imatges submarines. A la Secció 3 es presenta l'estudi i recerca inicial de la informació referent a l'EcoMapper, el format de les dades que s'hauran de tractar, l'algorisme EKF, les possibilitats de l'orientació a objectes amb Matlab i el disseny de la solució. A la Secció 4 s'explica el desenvolupament de la reconstrucció de les trajectòries amb l'EKF, els mosaics, la generació de *landmarks* i la implementació final de l'EKF-SLAM amb exemples i disseny tècnic. A la Secció 5 es presenten els resultats de la comparació de trajectòries i mosaics. Finalment a la Secció 6 s'exposen el resum amb les conclusions i possibles treballs futurs.

2 TREBALLS PREVIS

Els treballs previs inclouen la localització i l'SLAM, la generació de mosaics i imatges del fons marí i la detecció i correspondència dels *landmarks* en les imatges. Tal com es llegeix en els següents sub apartats els algorismes d'SLAM estan força madurs però la generació de mosaics i l'extracció i identificació de *landmarks* no està resolt.

2.1 Localització i SLAM

Tal com hem introduït l'SLAM tracta de resoldre el fet de situar un robot mòbil en un entorn a priori desconegut i que utilitzant els sensors i actuadors pugui anar localitzant-se i construint un mapa del que l'envolta. Aquest problema ha estat tema d'estudi de la comunitat científica durant els darrers 20 anys.

A l'hora de predir o calcular la posició del robot és necessari conèixer la limitació dels sensors i actuadors. Aquests poden ser imprecisos i introdueixen errors i soroll a les mesures, de manera que seran necessaris mètodes probabilístics com els filtres de Bayes. A més el model de l'entorn ha de ser computacionalment tractable, la qual cosa s'aconsegueix, entre d'altres, amb l'EKF presentat per Rudolf E. Kalman al 1960 [1].

L'EKF esdevé una de les solucions més esteses al problema de localització i modelat simultani per la facilitat d'implementació, per ser eficient computacionalment i pels bons resultats. Als treballs realitzats per Randall Smith, Mattew Self i Peter Cheeseman a finals de 1980 se'ls coneix com a EKF-SLAM.

A aquesta sèrie de treballs se suma el de Krakiwsky [2] on combina el *dead reckoning*, el GPS i el *map matching* amb un filtre de Kalman. Si bé sota l'aigua no hi ha estructures clares per construir un mapa sí és interessant llegir com combina el GPS i el *dead reckoning*.

De forma molt semblant al que es planteja en aquest TFM I. Tena presenta una solució d'SLAM amb mapes estocàstics per millorar la navegació a partir de mosaic i extracció manual de landmarks [3].

Per plantejar la solució s'han utilitzat els treballs de Antoni Burguera [4] i de David Ribas [5] es planteja l'SLAM sobre imatges de sonar amb una formulació coneguda en l'assignatura de Mètodes Numèrics del Màster TIN de la Universitat de les Illes Balears. Aquest treballs, però, es basen en imatges de sonar d'escombrat mecànic en comptes del SSS.

2.2 Mosaics submarins

Per generar imatges del fons marí els sonar són els millors dispositius. Altres mitjans més directes com les càmeres no són tan efectives perquè la llum s'atenua massa recorreguts als pocs metres. Així doncs la incidència dels polsos sonors permeten mesurar llargues distàncies i durant els darrers 20 anys s'han fet econòmicament més assequibles.

En la generació de mosaics la percepció de realisme augmenta quan s'hi afegeix la informació de batimetria 3D. Aquesta informació de la profunditat del fons no sempre està disponible com és en el cas de les dades d'aquest TFM. En el cas que es disposés d'informació batimétrica llavors es podria associar més informació als possibles *landmarks*.

L'objectiu inicial era poder extreure *landmarks* a partir dels mosaics generats durant la navegació enlloc de les imatges obtingudes directament del SSS. Tal com es podrà llegir a la Secció 2.3 els treballs que extreuen *landmarks* i busquen correspondències es basen en les imatges obtingudes del SSS i no de cap mosaic generat.

2.3 *Landmarks*

Els *landmarks* són els punts característics de la imatge que permeten anar construint el mapa de l'entorn i permeten corregir la posició del robot un cop es reobserven. Així doncs són els *landmarks* els que permetran fer SLAM sempre i quan es puguin detectar i establir la correspondència amb les observacions anteriors.

Les imatges sonar típicament s'interpreten per operaris però aquest procediment és subjectiu, de forma que s'estudia el processament automàtic d'imatges sonar amb l'objectiu d'identificar objectes, formacions o classificar textures i tipus de fons marins.

La dificultat d'identificar objectes radica en la natura de les imatges sonar. L'aparença d'un objecte dins la imatge pot canviar segons s'enfoqui d'un cantó o de l'altre degut a la seva forma, per aquest motiu els primers treballs se centren en objectes simètrics. Y. Petillot descriu com detectar mines amb *Markof Random Fields* (MRF) directament amb imatges sonar sense mosaic [6]. Funciona per a fons plans sense complicacions de relleu ni inclinació i es basa en la forma de l'objecte i la llargada de la seva ombra. A la Figura 3 es pot veure una representació.

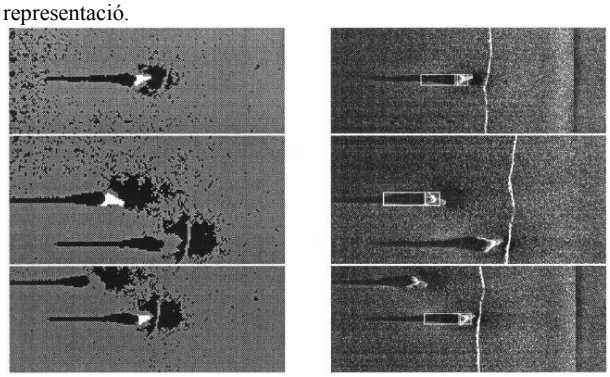

Figura 3 Detecció de mines en imatges SSS amb *Markof Random Fields* (MRF) basant-se en la forma i la longitud de l'ombra.

J. Aulines juntament amb Y. Petillot [7] treballa amb SLAM i proposa els descriptors de Haar per a trobar objectes sobre el fons marí. Aquest tipus de descriptor es basa en fitxers amb la informació com la de la Figura 4 en cascada segons com estigui format l'objecte a detectar. En el cas del seu treball es defineixen les formes per a trobar pedres segons la llargada de les seves ombres.

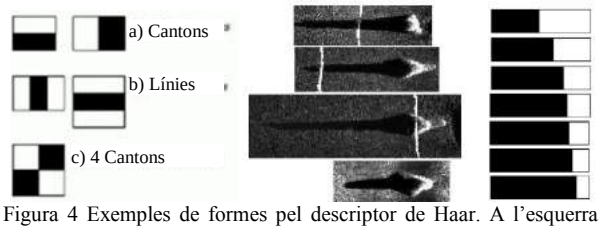

Figura 4 Exemples de formes pel descriptor de Haar. A l'esquerra plantilles i a la dreta exemples de *landmarks*.

La limitació és que aquest descriptor serveix per a pedres aïllades sobre fons plans. La detecció és del 99% i la correspondència entre *landmarks* és alta però amb errors de falsos positius.

Un altre treball és el de Stalder, Bleuler i Tamaki [8] on treballen per reposicionar el robot amb la correspondències amb els *landmarks* segons els mapes de característiques *Haralick*. La seva idea es basa en la segmentació de la imatge i l'energia dels segments per a identificar i catacteritzar els *landmarks*. A la Figura 5 es pot observar una representació gràfica de la correspondència i l'energia associada a un objecte *landmark*.

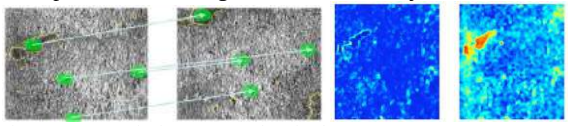

Figura 5 Detecció i correspondència de *landmarks* per segmentació i descripció estadística de la energia dels segments.

Com a refinament es podria pensar en detectar i identificar els *landmarks* segons formacions i relacions espacials amb la resta de *landmarks*. Aquesta idea apareix a Daniel S. [9] al 1998 on es treballa amb arbres de decisió a l'hora de trobar correspondències entre *landmarks*. A la Figura 6 es representen 2 imatges amb objectes reconeguts, la jerarquia dels objectes identificats i les correspondències establertes.

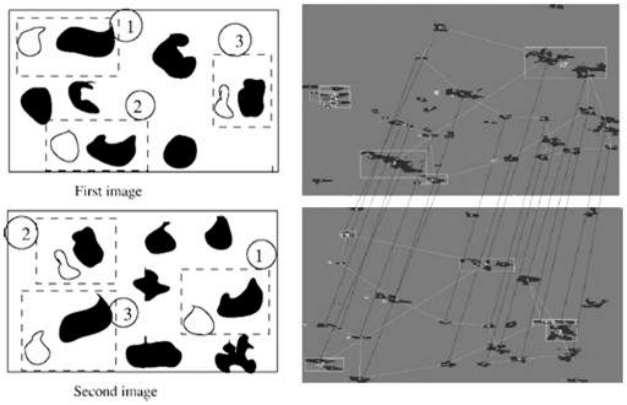

Figura 6 Representació del treball de Daniel S. on s'estableixen jerarquies entre objectes identificats per millorar la correspondència entre *landmarks* en les reobservacions.

En aquest apartat s'han repassat alguns dels treballs relacionats amb l'extracció i identificació de correspondència dels *landmarks* en imatges sonar, les dues tasques més importants per a poder realitzar l'SLAM de forma efectiva i que encara no està resolt pel cas de les imatges del SSS.

3 ESTUDI PREVI I DISSENY

Com a punt de partida del treball ha estat necessari l'estudi de les característiques del vehicle EcoMapper i dels seus sensors DVL i *Side Scan Sonar* (SSS). També ha estat necessari entendre les especificacions per als formats d'arxius d'entrada, aprendre l'algorisme d'EKF-SLAM, validar la viabilitat de Matlab amb Orientació a Objectes i fer un disseny de la solució global. En els següents punts es presenta aquesta informació.

3.1 L' EcoMapper

L' EcoMapper de YSI és l'AUV amb el que es va capturar el conjunt de dades. És un vehicle submarí en forma de torpede fàcilment manejable per una persona. Pesa 35kg, té una autonomia de 8h i pot arribar a 200m de profunditat. A la Taula 1 es mostra un resum de les seves característiques principals.

Aquest model de robot va ser adquirit per l' Institut de Tecnologia Marina del CSIC al 2010 [10] i des de llavors està al servei dels grups investigadors. Durant el 2on Workshop del projecte TRIDENT a l'octubre del 2012 aquest vehicle va participar de la mà de Pablo Rodríguez per a realitzar una demostració del seu funcionament. Durant 3 dies es van capturar dades en diverses sessions de prova i les utilitzades en aquest treball són en les dades de la missió de demostració a l'entrada del Port de Sòller amb 7 trams de dades SSS, alguns d'ells superposats, d'on s'identificaran manualment *landmarks* coincidents.

Tamany	140 cm L x15 cm Ø (forma torpede)
Pes	35 kg
Autonomia	8 h a 2 nusos
Capacitat de les bateries	600 W/h
Comunicació	Wifi
Navegació	GPS, DVL, brúixola i seguiment del fons i de la superfície

Taula 1. Característiques principals de l'EcoMapper.

El funcionament autònom s'aconsegueix realitzant la programació d'una missió per part de l'operari a través d'una aplicació proporcionada pel fabricant i en la que es poden indicar els punt del camí (*way points*), la profunditat i les accions a realitzar com fer fotos amb la càmera incorporada, desplaçar-se a una alçada constant de 10 metres per sobre del fons marí capturant imatges amb el *SSS* o finalitzar en un punt determinat de la superfície. A través de Wifi es pot comunicar amb l'estació base per comandar-lo manualment o obtenir-ne les dades recollides.

A la Figura 7 es pot veure una imatge i la situació dels diferents sensors. Per a la navegació l'EcoMapper disposa de la brúixola, el DVL i el GPS explicats a la Secció 3.2 i per a la captura de dades del fons el SSS detallat a la Secció 3.3 i la càmera fotogràfica.

3.2 DVL i fitxers de *log*

Per a calcular la navegació l'EcoMapper combina el sensor GPS quan està a la superfície i el DVL, la brúixola i l'altímetre quan està sota l'aigua. En aquest sub apartat es descriu breument el fonament físic del GPS i del DVL i el fitxer de log generat durant la navegació.

El GPS (*Global Positioning System*) és un sistema que permet calcular la posició d'un element receptor situat a la superfície de la terra a partir de la senyal transmesa per un conjunt de satèl·lits. Els satèl·lits tenen el rellotge sincronitzat i transmeten periòdicament, de manera que el receptor pot triangular la localització pel retard del senyal i de la posició dels satèl·lits de qui rep senyal. L'error pot ser de 15m amb 4 satèl·lits o menor de 2.5m si es triangula amb 9 satèl·lits.

El sensor DVL es basa en l'efecte *Doppler* per a determinar les

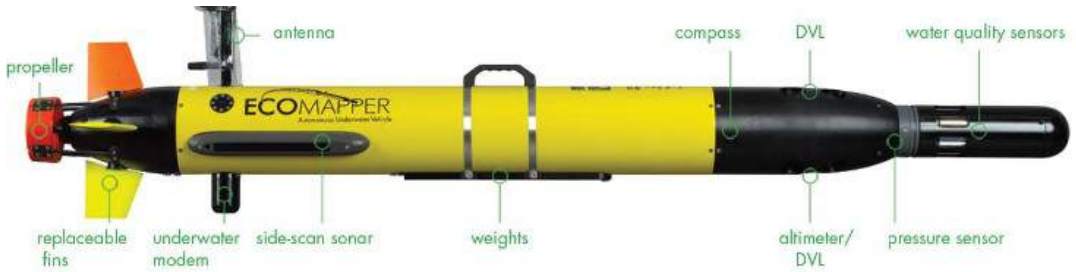

Figura 7 Imatge de l'AUV EcoMapper de YSI. Es poden veure els diferents sensors GPS, DVL, SSS o sensors de la qualitat de l'aigua [11].

velocitats de l'AUV. A través de transductors acústics emet ones amb una certa inclinació cap al terra o la superfície de l'aigua, de manera que si es mou pot calcular les velocitats per les variacions de freqüència a l'eco rebut degudes a l'efecte *Doppler*. Quan té una columna d'aigua de més de 7m pot fer *water tracking* però normalment fa el seguiment del terra *bottom tracking* de 1 a 40m de distància respecte el fons. És el sistema principal per a la navegació sota l'aigua.

Durant la missió l'EcoMapper guarda les dades dels sensors en diferents fitxers de logs. La freqüència a la que s'enregistren les dades és de 1 Hz perquè és el valor per defecte i per assegurar la robustesa de les dades. Els logs generats són els següents 3 arxius de text amb columnes de valors separats per punt i coma [12]:

1. DVL log file .**dvl**: conté les dades de localització, velocitats i altres paràmetres de l'aigua calculats per l'AUV. Aquest és l'arxiu utilitzat a l'hora de treure les dades de sensors.
2. Water profile up file .**pfu**: conté les dades referents a la columna d'aigua que queda per sobre del robot.
3. Water profile down file .**pfd**: en aquest cas és la columna d'aigua que queda per sota del robot.

L'arxiu principal és el .dvl on l'EcoMapper enregistra les dades de sensors utilitzades en aquest treball per a la localització. Alguns dels camps que conté són: latitud i longitud, data i hora, profunditat i alçada sobre el fons marí, velocitats en els eixos x, y i z o paràmetres de l'aigua com temperatura o salinitat. De tots els valors els interessants a l'hora de programar el *dead reckoning* en Matlab són l'orientació, el temps, les velocitats i l'alçada i profunditat.

Columna	Descripció	Exemple
Latitude	Latitud en coordenades geodèsiques.	39.79414
Longitude	Longitud en coordenades geodèsiques.	2.691231
Time	Temps en format 24h.	05:22:52.73
Date	Data en format *M/D/Y*	10/3/2012
C True Heading	Compàs amb graus a partir del nord en sentit horari.	289.755727
Sample number	Número de mostra en el log	310
Fix Type	tipus de tracking 0 cap, 1 water i 2 bottom. Són 0 o 2 la majoria.	2
Fix Quality		9
X Speed (m/s)	velocitat cap endavant	0.027
Y speed (m/s)	velocitat lateral (positiu cap a babord)	0.159
Z speed (m/s)	velocitat vertical (positiu cap al fons)	-0.016
Track time	temps desde l'inici de les mesures en segons	309
Altitude (m)	altitud respecte el fons marí, no corregit internament	11.35
Depth (m)	la profunditat sota la superfície de l'aigua.	9.68
Altres ...	Existeixen moltes altres columnes referents a paràmetres de temperatura, velocitat de l'aigua o altres que no s'han utilitzat.	

Taula 2 Algunes de les columnes de l'arxiu DVL.

A les especificacions del fabricant no es detalla exactament com l'AUV calcula la navegació. De forma molt general descriu que a la superfície s'empra el GPS i sota la superfície realitza a) un *bottom tracking* si el fons no està a més de 40m o b) un *dead reckoning* amb l'orientació, velocitats i profunditat quan està a més de 40m de distància del fons [11]. El resultat és que en el fitxer *.dvl* hi ha els camps calculats descrits a la Taula 2 i les columnes de latitud i longitud es tindran de referència com la trajectòria DVL calculada pel propi vehicle.

En aquest treball es calcularà la navegació del vehicle amb un *dead reckoning* propi que utilitzarà els camps de velocitats, orientació i profunditat. També es prendran els camps de longitud i latitud com a mesures de correcció GPS quan el vehicle estigui a la superfície. D'aquesta forma, calculant la posició amb les dades dels sensors, es podrà aplicar l'EKF i no es dependrà de processos de localització externs no documentats.

3.3 Side Scan Sonar i el format XTF

Per a poder reconstruir les imatges del fons marí ha estat necessari conèixer el funcionament del SSS i el format XTF (*eXtended Triton Format*) amb que es guarden les dades del sonar.

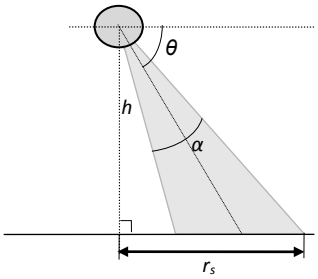

Figura 8 Paràmetres que caracteritzen el sensor SSS com l'angle d'obertura α, l'amplada mesurada r_s (*slant range*) i l'angle θ a que estan els transductors dels sensors respecte l'horitzontal.

El SSS és un sensor per ecolocalització. Consisteix en un conjunt de transductors ceràmics capaços de transformar un impuls elèctric en una pressió i a l'inversa. El procediment és generar un pols sonor anomenat *ping* i mesurar durant un temps *t* la intensitat del so que va retornant després de rebotar contra el fons marí. Aquest temps *t* ha de ser suficient com per mesurar el so emès en l'angle d'obertura α que es vol mesurar tenint en compte que la velocitat mitja del so a l'aigua marina és de 1500m/s. L'amplada de fons marí que es mesura es coneix com a *slant range* i amb una freqüència d'ona baixa es poden mesurar fins a 60km però amb poca resolució metre que amb una freqüència alta la resolució pot arribar a 1.2cm. A la Figura 8 es presenten els paràmetres generals que caracteritzen les mesures del SSS com l'angle d'obertura α, l'alçada h respecte el fons, la amplada de fons marí r_s (*sland range*) mesurada i l'angle θ a la que està col·locat el sensor.

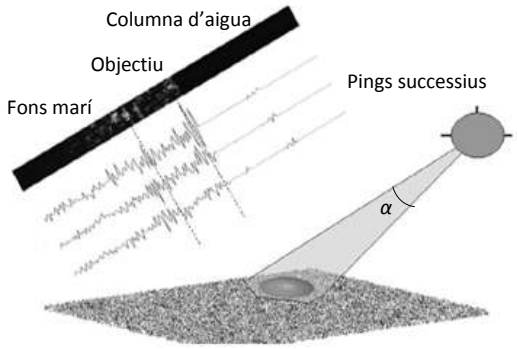

Figura 9 Exemple de com es mostra la imatge submarina a partir de *pings* successius com a intensitat de l'eco rebotat.

Idealment bastaria saber l'alçada *h*, l'angle d'obertura *α*, la velocitat del so i el temps *t* per projectar la intensitat mesurada sobre el fons marí però en la realitat hi ha moltes variables que afecten a la intensitat mesurada i al temps de resposta, com per exemple [13] la inclinació del robot, les característiques i inclinació del fons marí o la composició de l'aigua.

Com que és el propi vehicle el que avança no precisa de parts mecàniques per dirigir els sensors [14]. En la Figura 9 es representa com es va generant la imatge mentre el vehicle avança mesurant els diferents pings.

En el cas de l'EcoMapper aquest està equipat amb dos transductors SSS model YelowFin d'IMAGENEX com els de la Figura 10 situats a babord i estribord amb un angle θ de 20º

respecte la horitzontal del vehicle. A la Taula 3 es presenten les diferents possibilitats de configuració que ofereix.

Figura 10 IMAGENEX side scan sonar kit.

Especificacions hardware	
Interfície	RS-232 a 115.2 Kbps
Ample del transductor	330 kHz: 18º H x 60º V
	800 kHz: 0.7º H x 30º V
Resolució	2 costats: rang d'escala / 250
	1 costat: rang d'escala / 500

Taula 3 Característiques principals de l'IMAGENEX side scan sonar kit.

Per a les dades tractades en aquest treball l'EcoMapper es va configurar per a capturar els costats de babord i estribord amb una obertura α de 30º. La resolució és de 250 bytes per costat.

Per simplificar s'assumeix que la alçada respecte el fons durant la captura de dades SSS és constant a 10m, de manera que l'*slant range* r_s és de 30m per cada costat. La correspondència dels 60m total de *slant range* mesurats amb els 500 *bytes* de intensitat que es mesuren en cada ping amb una resolució de 12cm/píxel. En aquest punt ja es pot passar a descriure el format del fitxer que conté les dades SSS.

En aquest treball s'ha utilitzat el format obert XTF que tot i ser propietari de *Triton Imaging,® Inc* és un format obert, molt acceptat i àmpliament documentat.

El format XTF està pensat per poder introduir en un sol fitxer de forma flexible dades des de diferents canals o fonts de dades com dades sonar, altitud o batimetria. El funcionament és enregistrar pings de dades a mesura que arriben. A continuació es presenta un esquema de l'estructura del fitxer que s'enregistra en el cas del sonar.

XTF File

PingHeader	ChainHeader	Data	ChainHeader	Data
Ping 1	Chain 1		Chain 2	
Ping 2	Chain 1		Chain 2	
...				
Ping i	Chain 1		Chain 2	

Figura 11 Reperesentació del fitxer XTF pel cas de les dades SSS amb 2 canals de dades corresponents a babord i estribord.

En la Figura 11 es pot observar com s'estructura el fitxer XTF pel cas del sonar. Cada segon hi poden haver de 7 a 14 pings SSS depenent del temps que trigui el so en retornar als sensors. Cada ping de dades conté una capçalera amb les dades generals i el nombre de canals, que en aquest cas són 2: babord i estribord. Cada canal del ping té una capçalera descriptiva i li segueixen les dades que en aquest cas són els 250 bytes que representen la intensitat de l'eco mesurat. Aquestes dades són les que s'utilitzaran per formar els mosaics submarins i que s'explica a la Secció 4.3.

3.4 EKF-SLAM

Tal i com s'ha explicat a la introducció en aquest treball es planteja l'SLAM (*Simultaneous Localization And Mapping*) basat en l'algorisme EKF (*Extended Filter Kalman*). L'algorisme d'SLAM consisteix en diferents parts:

- Extracció dels *landmarks*,
- Associació de *landmarks*,
- Estimació de l'estat,
- Actualització de l'estat i
- Actualització dels *landmarks*.

L'extracció dels *landmarks* serà manual, de manera que en les observacions s'establirà un identificador per a poder associar sense error el *landmark* en les reobservacions. Per a l'estimació de l'estat s'utilitzarà un model dinàmic basat en les velocitats lineals. Per a l'estimació i actualització de l'estat i dels *landmarks* s'utilitzarà l'EKF.

L'algorisme EKF es presenta en la seva formulació general a la Taula 4. En les línees 2 i 3 es realitza la predicció de l'estat $\bar{x}_t$ amb un model dinàmic $g(u_t, x_{t-1})$ en funció de l'estat anterior x_{t-1} i de la informació del *dead reckoning* u_t. En les línees 4, 5 i 6 es realitza l'actualització a l'estat x_t segons les mesures z_t i la predicció anterior. La part corresponent a la construcció del mapa i la correcció per *landmarks* s'explica a la Secció 3.4.4.

1: **Algorithm Extended_Kalman_filter(x_{t-1}, $\sum_{t-1}$, u_t, z_t):**		
2: $\bar{x}_t = g(u_t, x_{t-1})$	}	Predicció
3: $\bar{\sum}_t = G_t \sum_{t-1} G_t^T + R_t$	}	
4: $K_t = \bar{\sum}_t H_t^T (H_t \bar{\sum}_t H_t^T + Q_t)^{-1}$	}	Actualització
5: $x_t = \bar{x}_t + K_t(z_t - h(\bar{x}_t))$	}	
6: $\sum_t = (I - K_t H_t) \bar{\sum}_t$	}	
7: **return** $x_t, \sum_t$		

Taula 4 Algorisme del filtre de Kalman estès [14].

En els següents subapartats es presenta l'algorisme desenvolupat pel cas concret de l'AUV i la formulació començant pel vector d'estats inicial i seguint per la predicció, les actualitzacions de l'estat i la observació i actualització per *landmarks*.

3.4.1 Vector d'estat inicial X_0

El primer pas és definir el vector d'estats amb la informació per predir la posició. L'EKF representa l'estat i totes les variables amb un vector de mitjanes i una matriu de covariàncies. El vector contindrà les coordenades de posició relativa a l'origen de la missió, l'angle θ respecte x i les velocitats dels 3 eixos i angular. Implícitament s'està assumint que el sistema és estable en els angles de *roll* i *pitch*. A continuació es representa breument l'estat columna amb els 8 valors:

X = [**x** : component x en metres respecte el punt inicial
y : component y en metres respecte el punt inicial
z : alçada z respecte el nivell del mar.
θ : angle *yaw* de rotació sobre l'eix z i respecte l'eix x
$\mathbf{v_x}$: component de la velocitat x
$\mathbf{v_y}$: component de la velocitat y
$\mathbf{v_z}$: component de la velocitat de l'eix z
ω : velocitat angular del θ]

En la inicialització de l'estat i la matriu de covariàncies els únics valors diferents a 0 són l'angle d'orientació θ i la seva variància a la matriu de covariàncies.

$$x_o = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \theta_0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}_0 , \sum_o \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_\theta^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (1)$$

L'estat X creixerà a mesura que es vagin observant *landmarks* dins del mosaic.

3.4.2 Estimació de l'estat $\bar{x}_t, \bar{\sum}_t$

Amb aquest vector d'estats ja és possible definir el model dinàmic del sistema per poder predir l'estat $\bar{X}$ que conté la posició i les velocitats passat un temps t a partir de l'estat anterior X_{t-1}. Aquest model dinàmic es planteja a l'equació 2 on es pot observar com les velocitats es mantenen constants i l'únic que es preveu són les coordenades lineals. Per evitar mesures d'alçada errònies z_t i v_z es multipliquen per la condició lògica $(z_{x_{t-1}} \leq 0)$ que fa que la profunditat z i la velocitat en v_z siguin 0 si el submarí es troba per sobre de la superfície.

$$\bar{x} = \begin{bmatrix} x \\ y \\ z \\ \theta \\ v_x \\ v_y \\ v_z \\ \omega \end{bmatrix}_t = \begin{bmatrix} x + v_x \cdot \mathrm{c}\,\theta \cdot t + v_y \cdot \mathrm{s}\,\theta \cdot t \\ y + v_x \cdot \mathrm{s}\,\theta \cdot t + v_y \cdot \mathrm{c}\,\theta \cdot t \\ (z + v_z \cdot t) \cdot (z_{x_{t-1}} \leq 0) \\ \theta \\ v_x \\ v_y \\ v_z \cdot (z_{x_{t-1}} \leq 0) \\ \omega \end{bmatrix} \quad (2)$$

Per completar la predicció és necessari el càlcul de matriu de covariàncies $\bar{\sum}_t$ segons la línia 3 de l'EKF on $R_t = W \cdot Q \cdot W^T$.

Com que la funció de predicció $g(u_t, x_{t-1})$ no és lineal primer s'ha de calcular la matriu jacobiana G_t amb les derivades parcials respecte l'estat x_{t-1} a l'equació 3 on T és el temps entre l'estat actual x_{t-1} i l'estat predit $\bar{x}_t$ i c i s són les funcions cosinus i sinus.

$$G_t = \frac{\partial g}{\partial X_T} =$$
$$\begin{bmatrix} 1 & 0 & 0 & -\mathrm{v_x T\,s}\theta - \mathrm{v_y T\,c}\theta & \mathrm{T\,c}\ \theta & -\mathrm{T\,s}\theta & 0 & 0 \\ 0 & 1 & 0 & \mathrm{v_x T\,c}\theta - \mathrm{v_y T\,s}\theta & \mathrm{T\,s}\theta & \mathrm{T\,c}\theta & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \mathrm{T} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \mathrm{T} \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

Posteriorment el renou de les mesures R_t. Q és el soroll gaussià estimat de les mesures de les velocitats i rotació de l'angle d'orientació i amb la matriu jacobiana W de les derivades parcials de $g(u_t, x_{t-1})$ respecte el soroll mesurat.

$$Q = \begin{bmatrix} {\sigma_{v_x}}^2 & 0 & 0 & 0 \\ 0 & {\sigma_{v_y}}^2 & 0 & 0 \\ 0 & 0 & {\sigma_{v_z}}^2 & 0 \\ 0 & 0 & 0 & (5\pi/180)^2 \end{bmatrix} \quad (4)$$

$$W = \begin{bmatrix} {T^2\,\mathrm{c}\,\theta}/{2} & -{T^2\,\mathrm{s}\,\theta}/{2} & 0 & 0 \\ {T^2\,\mathrm{s}\,\theta}/{2} & {T^2\,\mathrm{c}\,\theta}/{2} & 0 & 0 \\ 0 & 0 & {T^2}/{2} & 0 \\ 0 & 0 & 0 & {T^2}/{2} \\ T & 0 & 0 & 0 \\ 0 & T & 0 & 0 \\ 0 & 0 & T & 0 \\ 0 & 0 & 0 & T \end{bmatrix} \quad (5)$$

Un cop definides les fórmules de la previsió podem continuar amb l' actualització de l'estat.

3.4.3 Actualització de l'estat $x_t, \sum_t$

L'actualització es correspon amb les línies 4-6 de l'EKF de la Taula 4. La idea és calcular el guany de Kalman K_t que pondera la diferència entre l'estat previst i les mesures segons les covariàncies.

En aquest treball es realitza una actualització amb les coordenades del GPS quan l'AUV està a la superfície i tot seguit s'actualitza amb les dades de velocitats, fondària i orientació del DVL:

- Primera actualització GPS si està a la superfície:

$$h(x_t) = [x\ y\ z\ v_z]^T$$
$$z_{GPS} = [x\ y\ 0\ 0]^T$$
$$H_{GPS}(x) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}^T$$
$$Q_t = {\sigma_{GPS}}^2 \cdot I_{4x4}$$

- Segona actualització amb mesures DVL:

$$h(x_t) = [v_x\ v_y\ v_z\ \theta\ z]^T$$
$$z_{DVL} = [v_x\ v_y\ v_z\ \theta\ z]^T$$
$$H_{DVL}(x) = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$
$$Q_t = [{\sigma_{v_x}}^2 \quad {\sigma_{v_y}}^2 \quad {\sigma_z}^2 \quad {\sigma_\theta}^2 \quad {\sigma_z}^2] \times I_{4\times4}$$

3.4.4 Actualització per *landmarks*

Aquest apartat correspon a les tasques d'SLAM d'ampliació del mapa d'estat amb els *landmarks* i de l'actualització de l'estat segons les reobservacions de *landmarks*.

1: **function Update_with_landmaks(x_t, $\sum_t$, z_t, $\sum_z$):**
2: ids = []; Z_t = []; Q_t = [] ; h_t = []; H_t = []
3: **per totes les observacions** $z_t^i = (x, y, id)$ **dins** z_t
4: $[z_{id}, id]$ = funcio_similitud(z_t^i)
5: **si** $id < 0$ **llavors** augmenta l'estat
6: $x_t = \begin{pmatrix} x_t \\ \begin{bmatrix} x_{z_t^i} \\ y_{z_t^i} \end{bmatrix} \end{pmatrix}$; $\sum_t = \begin{pmatrix} \sum_t & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \sum_z \end{pmatrix}$
7: **altrament**
8: ids = [ids id]
9: $Z_t = \begin{bmatrix} Z_t \\ h_i(x_t, z_t^i) \end{bmatrix}$
10: $Q_t = \begin{bmatrix} Q_t & \cdots \\ \vdots & \sum_z \end{bmatrix}$
11: **fi si**
12: **fi per**
13: **si** ids > [] **llavors**
14: **per tot** *id* **dins** *ids* **fer**
15: $[h_i, H_i] = h\_per\_id_i(x_t, id)$
16: $h_t = \begin{bmatrix} h_t \\ h_i \end{bmatrix}$; $H_t = \begin{bmatrix} H_t \\ H_i \end{bmatrix}$
17: **fi per**
18: $K_t = \sum_t H_t^T (H_t \sum_t H_t^T + Q_t)^{-1}$
19: $x_t = x_t + K_t(Z_t - h_t)$
20: $\sum_t = (I - K_t H_t) \sum_t$
21: **fi si**
22: **return** x_t, $\sum_t$

Taula 5 Algorisme corresponent a la part de l'SLAM definit com a una funció.

L'estat x s'amplia quan els *landmarks* s'observen per primera vegada i s'actualitza la posició quan es re-observen. La actualització per *landmarks* és molt semblant a l'actualització de l'estat per la mesura dels sensors tret de que a priori no se sap quants *landmarks* es reobservaran. Aquesta funció es presenta a la Taula 5.

A l'igual que amb les observacions h del GPS o del DVL les observacions dels *landmarks* es comparen amb les observacions prèvies contingudes en el vector d'estat x i ponderant el guany de Kalman K_t es fa l'actualització. En aquest cas la formulació és diferent perquè les matrius de mesures Z_t, el renou de les mesures Q_t, les observacions h_t i la jacobiana de les observacions H_t es van construïnt a mesura que avança les iteracions sobre les observacions z_t. Tot seguit es comenta la formulació.

Tot i que els *landmarks* s'han definit manualment aquests se seguiran obtenint de la imatge mosaic que es va formant amb coordenades globals respecte el punt d'origen. Així doncs la matriu de mesures Z_t es construeix a partir de totes les observacions posant les coordenades absolutes a relatives respecte la posició x_t del robot amb la funció d'observació $h_i(x_t, z_i)$ de la Taula 6.

1: **function** $\boldsymbol{h_i(x_t,\ z_i)}$
2: $c = \cos\theta_{x_t}$; $s = \sin\theta_{x_t}$
3: $h = \begin{bmatrix} -x_{x_t}c - y_{x_t}s + x_{z_i}c + y_{z_i}s \\ x_{x_t}s - y_{x_t}c - x_{z_i}s + y_{z_i}c \end{bmatrix}$
4: **return** h

Taula 6 Funció d'observació per transformar les coordenades de la observació z_i d'absolutes a relatives respecte el robot segons x_t.

Les observacions h_t i la matriu de jacobianes de les observacions H_t es construeixen utilitzant la funció $h\_per\_id_i(x_t,id)$ de la Taula 7 que a la vegada utilitza la funció anterior. Els *landmarks* es guarden amb la posició global a l'estat x_t i es recuperen segons l'*id* a partir de la posició 9 del vector d'estats.

1: **function** $\boldsymbol{h\_per\_id_i(x_t,\ z_{id}, id)}$
2: $x_{z_{id}} = x_t[9+2\cdot(id-1)]$; $y_{z_{id}} = x_t[9+2\cdot(id-1)+1]$
3: $z_{id} = [x_{z_{id}}, y_{z_{id}}]^T$
4: $h_i = h_i(x_t,\ z_{id})$
5: $H_i =$
$$\begin{bmatrix} -c & -s & x_{x_t}s - y_{x_t}c - x_{z_{id}}c + y_{z_{id}}s & \underbrace{\ldots 0 \ldots}_{8+2i-2} & c & s & \underbrace{\ldots 0}_{2N-2i} \\ s & -c & x_{x_t}c + y_{x_t}s - x_{z_{id}}c - y_{z_{id}}s & & -s & c & \end{bmatrix}$$
4: **return** h_i, H_i

Taula 7 Funció d'observació per les observacions conegudes amb *id* que també retorna la jacobiana H_i.

Pel que fa al renou Q_t aquest també es construeix com a una matriu on hi ha tantes $\sum_z$ a la diagonal com observacions identificades. $\sum_z$ és la matriu de covariància de mesures de 2x2 amb el valor de $\sigma=0.05$ a la diagonal. Aquesta construcció es troba a la línia 10 de la Taula 5.

3.5 Matlab orientat a objectes

Uns dels requeriments inicials del treball era la utilització de Matlab con a entorn de programació. En aquest cas la versió utilitzada ha estat la R2011a.

Típicament el paradigma de programació amb Matlab és l'estructurat a base d'invocar funcions però en aquest cas s'ha plantejat la possibilitat que ofereix d'orientació a objectes per tal de simplificar el pas de paràmetres entre funcions, el nombre d'arxius i també agrupar el codi de la forma més lògica en classes.

A la Taula 8 es pot veure un exemple real de classe en Matlab i la instanciació d'un objecte des de la línia de comandes. La classe *SampleClass* estén la classe *handle* per poder passar l'objecte per referència en la crida a funcions. D'aquesta forma queda validada la possibilitat de l'ús de Matlab amb orientació a objectes a l'hora de dissenyar la solució.

SampleClass.m

```
classdef SampleClass < handle
    %SAMPLECLASS
    properties (Access = protected)
        value;
    end
    methods
        function obj = SampleClass()
            % Constructor
            obj.value = 0;
        end
        function v = getValue(obj)
            v = obj.value;
        end
        function setValue(obj, v)
            obj.value = v;
        end
    end
end
```

Exemple d'ús

```
>> sampleObj = SampleClass
sampleObj =
  SampleClass handle with no properties.
  Methods, Events, Superclasses
>> sampleObj.getValue
ans =
     0
>> sampleObj.setValue(10)
>> sampleObj.getValue()
ans =
    10
```

Taula 8 Exemple de classe amb Matlab i ús de la mateixa.

3.6 Disseny de la solució

Abans de passar a la implementació en aquest punt es planteja l'esquema de la solució. En la Figura 12 es poden observar els diferents blocs lògics: 1) dades d'entrada sonar, DVL i landmarks manuals, 2) sistema TFMSSS amb l'algorisme de localització, la generació de mosaics i la detecció de landmarks i 3) les dades de sortida com a imatges mosaic, trajectòries i observacions en arxius Matlab o capes d'informació geogràfica.

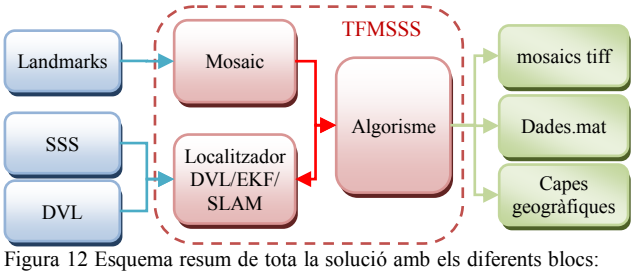

Figura 12 Esquema resum de tota la solució amb els diferents blocs: dades d'entrada, processament i les dades de sortida.

En un primer disseny es processaven les dades d'entrada en un sol bloc de localització però s'ha separat en 2 blocs per a poder comparar les diferents trajectòries amb un mateix algorisme. D'aquesta manera, amb una mínima configuració de l'execució es pot escollir processar les dades amb la trajectòria amb SLAM i les correccions per *landmarks*, només amb el filtre EKF o directament mostrar les dades de latitud o longitud enregistrades en el fitxer DVL.

Per descriure una mica l'algorisme ideat es presenta el pseudocodi de la Taula 9. Aquest algorisme treballa genèricament amb el localitzador DVL, EKF o amb SLAM. En la línia 2 fa una inicialització. Mentre hi hagi dades (línia 3) les processarà. Si les dades actuals són del DVL (4) realitzarà una predicció (5) i una actualització (6). En el cas que el localitzador sigui del tipus DVL la predicció no té efecte perquè només actualitza en funció de les dades DVL i no com l'EKF i l'SLAM que fan una predicció segons el model dinàmic. Si el localitzador

és de tipus SLAM tractarà de trobar *landmarks* en el mosaic (7) de manera que els guardarà en el vector d'estat i farà les correccions en el cas de les reobservacions.

```
1: algorisme TFM SSS(localitzador, dades):
2:    localitzador.inicialitzar()
3:    mentre no dades.fi
4:       si dades.DVL llavors
5:          [X,P] = localitzador.prediction(T)
6:          [X,P] = localitzador.update(dades.DVL)
7           si localitzador = 'SLAM' llavors
8:             [X,P] = localitzador.updateWithLandmarks(mosaic)
9:          fi si
10:      altrament si dades.SSS llavors
11:         mosaic.dibuixa(dadesSSS)
12:      fi si
7:    retorna trajectòria, mosaic
```

Taula 9 Algorisme TFM SSS per a un localitzador genèric i el dibuix del mosaic.

4 DESENVOLUPAMENT DEL TREBALL

Els següents punts es corresponen amb la seqüència descrita a la introducció: reconstruir la trajectòria amb *dead reckoning* i el filtre de Kalman, formar els mosaics submarins, extreure els *landmarks* i aplicar l'algorisme d'SLAM per refinar el recorregut.

4.1 Recorregut DVL

Per tal de poder situar les dades gràfiques del sonar el primer que es necessita és reconstruir el recorregut. L'EcoMapper enregistra una trajectòria amb un *dead reckoning* propi que combina dades GPS, de brúixola i velocitats del qual desconeixem el detall però que ens serveix per mostrar un primer recorregut de referència que es presenta en aquest punt.

El recorregut que enregistra l'EcoMapper es correspon amb les columnes de latitud i longitud del fitxer de dades *.dvl*. En una primera tasca s'ha programat la classe *DVLReader* per llegir i interpretar els logs del DVL, per cada fila que llegeix retorna una dada dins un objecte *DVLData* amb la informació dels sensors que ens interessa com la latitud i longitud, les velocitats, alçades o el *timestamp* de la dada. En els mètodes de consulta hi ha un booleà *ok* de sortida que indica la validesa de la dada consultada, per exemple, si la profunditat és major que 0 les dades GPS no seran vàlides.

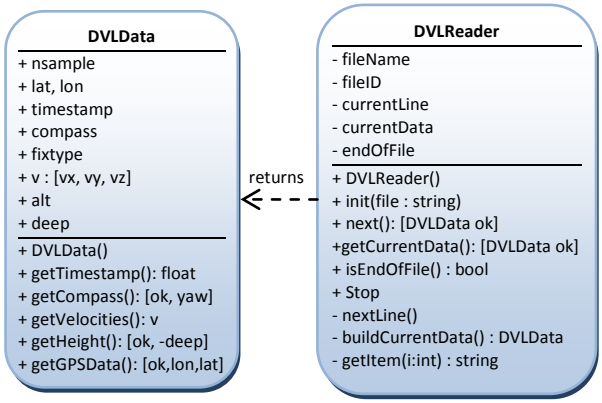

Figura 13 Classes per a la lectura del log DVL.

Per tal de treballar amb coordenades relatives a un punt inicial s'ha creat la classe *Utils* amb mètodes estàtics que permet establir una coordenada geodèsica com a origen del sistema de coordenades local i calcular les matrius de transformació per transformar les coordenades geodèsiques i cartesianes. Aquesta classe també conté altres mètodes que es comentaran més endavant.

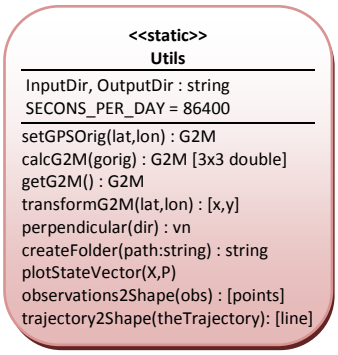

Figura 14 Classe estàtica *Utils* per compartir la informació sobre el punt origen i la matriu de transformació de coordenades geodèsiques a coordenades locals.

Amb l'ús d'aquestes tres classes *DVL* i *Utils* ja es pot realitzar una representació gràfica de la lectura del log DVL on es va pintant la trajectòria llegida transformada a metres relatius al punt d'origen, l'orientació del robot, el vector velocitat i la gràfica de la profunditat i l'alçada respecte el fons marí. Durant l'execució es pot observar com l'orientació i el vector velocitat pateixen canvis bruscos, la qual cosa indica que el DVL enregistra les mesures dels sensors sense aplicar cap filtre per suavitzar. A la Figura 15 es presenta una captura de l'execució d'aquesta representació gràfica.

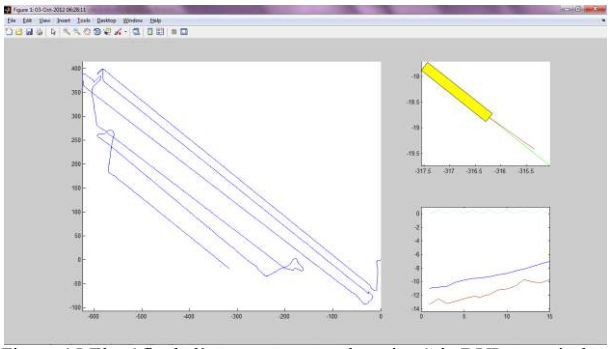

Figura 15 El gràfic de l'esquerra mostra la trajectòria DVL a partir de les dades enregistrades corresponents a les columnes de latitud, longitud. El gràfic de la dreta superior mostra les velocitats i orientació. El gràfic inferior dret mostra la profunditat i l'alçada respecte el fons marí.

El següent punt serà construir la trajectòria a partir de les dades dels sensors mitjançant el filtre de Kalman extés.

4.2 Trajectòria amb EKF

En aquest punt s'explica la implementació de l'EKF però abans es presenta la classe abstracta *LocBase* de la Figura 16 que permetrà comparar les trajectòries DVL, EKF i SLAM.

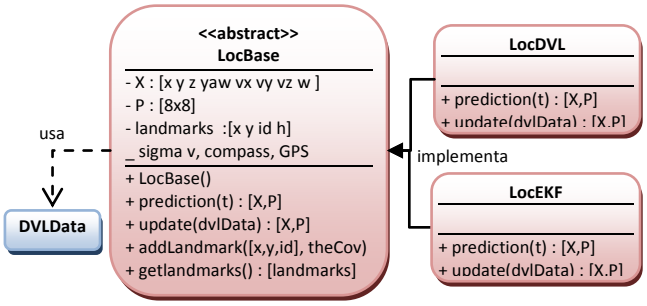

Figura 16 Classe abstracta *LobBase* per a la localització genèrica.

Aquesta classe conté el vector d'estats *x* i la matriu de covariàncies ∑ a més de la llista de *landmarks* identificats i les constants σ per les covariàncies de les mesures. Com es pot veure té els mètodes abstractes *prediction* i *update*. La raó d'aquesta classe és poder comparar més endavant els trajectòries canviant només el localitzador però utilitzant el mateix algorisme.

Havent vist la classe *LocBase* és fàcil identificar que la classe *EkfLoc* implementarà el mètode abstracte *prediction(t)* segons la formulació ideada a la Secció 3.4.2 i el mètode abstracte *update(DvlData)* segons el punt 3.4.3. En la Figura 17 es pot veure un primer exemple gràfic de la reconstrucció amb l'*EkfLoc* amb les dades del treball. Està representat en 3D per a poder mostrar la trajectòria a la superfície, segons la profunditat i l'alçada respecte al fons marí interpolat per formar una superfície.

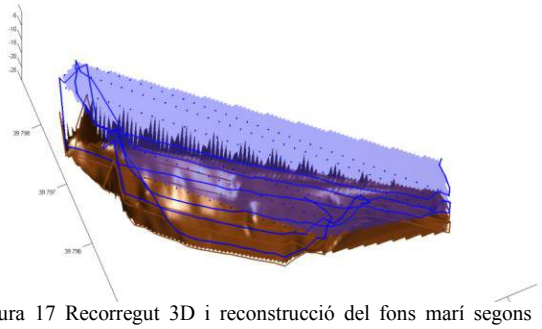

Figura 17 Recorregut 3D i reconstrucció del fons marí segons les dades processades amb el localitzador EKF.

El següent pas lògic abans de la trajectòria amb SLAM és extreure els punts característics *landmarks* a partir de les imatges sonar.

4.3 Lectura SSS

Amb l'objectiu de poder reconstruir el mosaic submarí ha estat necessari poder llegir i interpretar el contingut d'aquests fitxers, així que seguint les especificacions del format *XTF* s'ha programat la classe en Matlab que obre cada fitxer i va llegint les dades tal i com estan estructurades en pings.

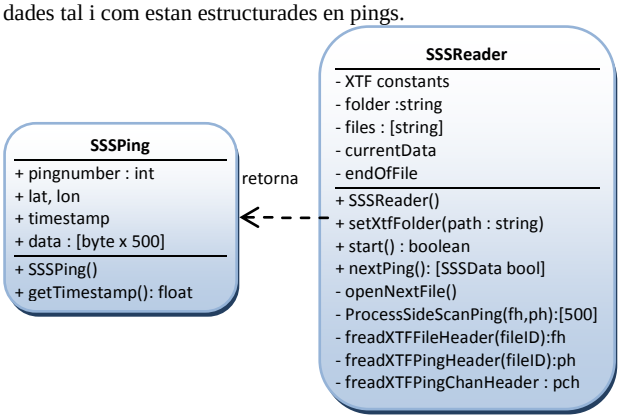

Figura 18 Classes per a la lectura de les dades del SSS.

A la Figura 18 es presenten les dues classes encarregades per a la lectura i obtenció de la informació del SSS. La classe *SSSReader* és semblant a la *DVLReader* amb la diferència que ara no llegeix un fitxer de log sinó els diferents fitxers corresponents als trams on l'*EcoMapper* tenia l'ordre de capturar dades. Així doncs amb el mètode *setXtfFolder(string)* és possible especificar el conjunt de dades que tractarà. Cada cop que s'invoqui a *nextPing* es retornarà un objecte *SSSPing* amb la informació del número de ping, el *timestamp* de l'ora d'enregistrament i els 500 bytes de dades corresponents a la mesura del sonar. El píxel 1 es correspon al punt més allunyat de babord i el 500 al punt més allunyat d'estribord.

En la Figura 19 es presenta una primera prova resultat d'anar posant les dades llegides en una imatge a mesura que es van llegint. Sense cap tractament d'imatge es pot observar la imatge del SSS en una escala de grisos de 0 a 255.

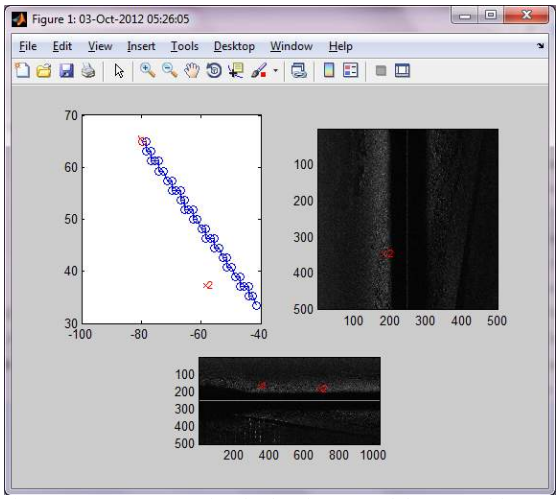

Figura 19 Reconstrucció de les imatges del SSS mitjançant les classes *SSSReader* i *SSSPing*.

Per tenir una idea del volum de dades es pot resumir que la missió d'exemple en el treball té 7 trams de dades, la majoria rectes, de fins a 400m, de 6 a 14 pings per segon de 500 bytes per tram amb un total de 2700 pings en el tram més llarg. El següent pas és projectar aquestes dades gràfiques en el fons marí.

4.4 Mosaics submarins

Un cop ja es té la posició del robot en cada moment i la informació del SSS ja es poden posicionar les dades llegides en forma de mosaic.

La solució adoptada és agrupar pings de les dades SSS en un *beam* amb la classe *Beam* i passar-la a l'objecte *Mosaic* per a que ho dibuixi dins la imatge. Un *beam* conté el conjunt de pings enregistrats pel SSS en un interval de temps *t* entre dos punts x_{ini} i x_{fi}. La raó és que el *SSSReader* pot retornar de 6 a 14 pings de dades *SSSPing* per segon sense informació exacte del *timestamp* dins del segon metre que les dades de posició es determinen amb el *DVLReader* que proporciona una dada nova cada segon. Així doncs cada segon es completarà un *beam* de *pings* que es dibuixarà en el mosaic. El *beam* contindrà la informació dels punts d'inici i final del tram i la informació gràfica dels pings. Si els pings arribessin amb informació del *timestamp* fins al milisegon llavors es podria predir la posició i dibuixar-lo on toca amb una amplada fixe.

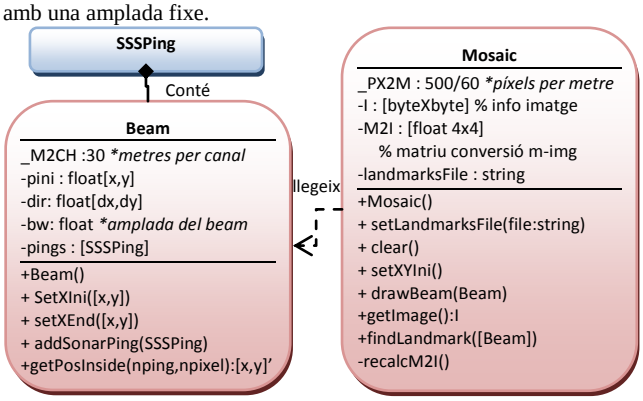

Figura 20 Classe *Mosaic* per formar la imatge a partir dels *pings* agrupats en *beams*.

En la Figura 20 es representen les classes necessàries per a la formació de mosaics. Els objectes *Beam* contenen la informació necessària per a projectar els *pings* de dades com són el punt inicial i final mentre que la classe *Mosaic* té el mètode *drawBeam(Beam)* que s'encarregarà d'incorporar i projectar la informació del *beam* i els *pings* en la matriu *bytes* que corresponen a la imatge. Internament la classe *Mosaic* manté una matriu *I* de bytes que es correspon amb el sistema de coordenades lineal a través de la matriu *M2I* que transforma de metres a píxels i que es recalcula cada cop que algun punt excedeix els límits de la imatge. La Figura 21 és l'exemple del mosaic que es forma amb el tram més llarg de dades SSS i la Figura 22 és un exemple d'un tram de dades SSS amb corba.

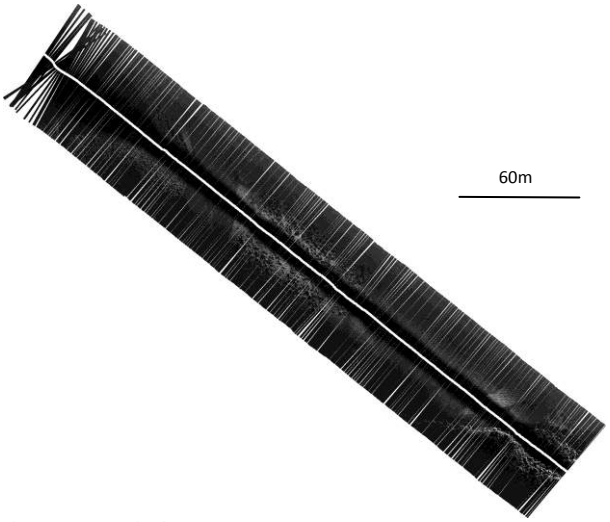

Figura 21 Mosaic d'un tram o transecte.

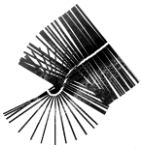

Figura 22 Tram de mosaic on es fa girar el vehicle i es veu clarament la falta d'informació entre els beams.

En la Figura 23 es mostra la formació del mosaic corresponent a tots els trams de dades SSS dibuixats amb el mateix objecte *Mosaic*. Aquesta imatge es pot obtenir amb el mètode *getImage():[n byte X m byte]*.

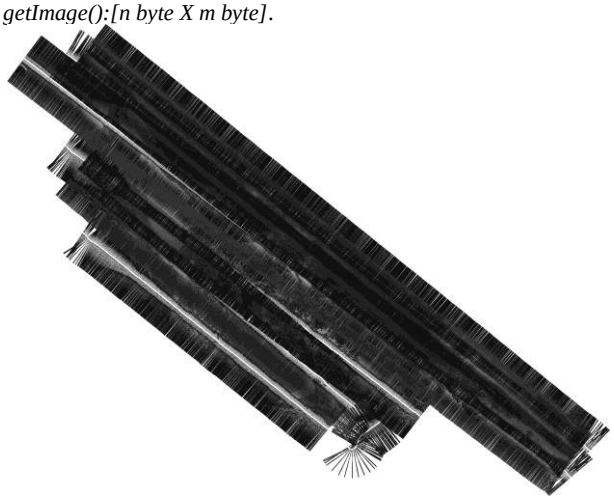

Figura 23 Exemple de formació de mosaic. Aquest és el resultat de dibuixar tots els beams dels diferents trams en un sol mosaic. S'ha tractat la imatge per substituir on no hi havia informació per blanc i per normalitzar els blancs per millorar la imatge.

4.5 *Landmarks*

La qüestió dels punts característics a les imatges ha estat la més difícil del treball. A partir de la formació de la imatge SSS a la Secció 4.3 es poden reconèixer algunes formacions rocoses que coincideixen o canvis de tipus de fons o pedres aïllades. La realitat és, però, que falten treballs previs sobre el reconeixement de punts característics en imatges sonar i la raó és la dificultat que això representa. Les ombres i la forma dels punts canvia si el sonar les enfoca des d'una banda o des de l'altre. El cervell humà és capaç de reconèixer, imaginar i reconstruir a partir d'una ombra i en un altre imatge establir la correspondència, però la complexitat d'aquesta tasca escapa a l'abast d'aquest TFM. És més, el fet de no poder identificar punts característics en les imatges lineals ha relaxat la qualitat dels mosaics generats, ja que per molt ben formats que estiguessin no hauria estat possible detectar els punts en el mosaic. Així doncs els *landmarks* s'han establert finalment manualment. Les figures Figura 24 i Figura 25 mostren la complexitat del problema. Corresponen a una regió on se superposen una part de 3 trams diferents de dades SSS, s'hi poden arribar a trobar fins a 9 punts coincidents 2 a 2 assenyalats amb el seu número identificador en vermell.

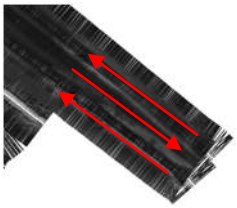

Figura 24 Trams amb solapament escollits per a l'exemple.

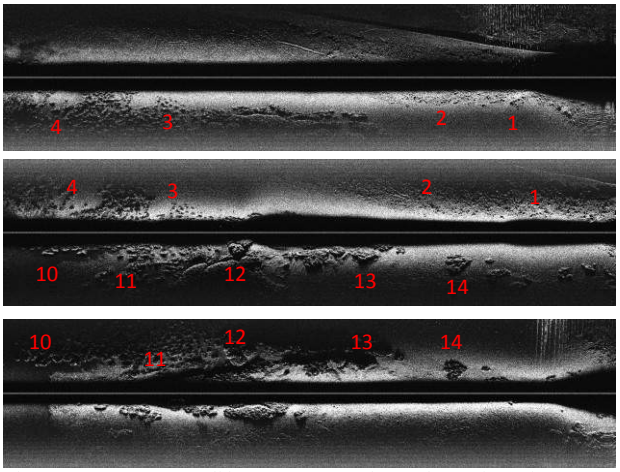

Figura 25 Tres trams diferents on se solapen les trajectòries. Les observacions tenen un identificador per relacionar els *landmarks*.

Per simular l'observació dels *landmarks* es defineix la matriu *landmarks* dins del conjunt de dades d'entrada amb les següents dades per cada columna. Aquesta matriu es guarda amb el nom de *landmarks.mat*.

$$landmark = [n^o\ ping,\ n^o\ píxel,\ id]^T$$

Cada observació de *landmark* conté el número de ping on apareix, el número de píxel dins del ping des de babord 0 fins al final d'estribord 255 i l'identificador del *landmark* per facilitar la funció d'observació. Programàticament es pot crear un event durant la simulació per capturar un clic del ratolí i guardar un nou landmark. Les observacions es llegeixen a través de la classe *Mosaic* que a partir d'un *beam* observa el número de cada ping i

retorna les observacions de *landmarks* ja amb les coordenades en metres que pertocarien dins del mosaic.

$$observation = [x, y, id]^T$$

En la Figura 26 es mostren les classes involucrades. Els *landmarks* es carreguen d'un fitxer *.mat* de Matlab des del *mosaic* i el *mosaic* retorna les observacions trobades segons els pings del *beam* que estigui observant. Per exemple, si s'invoca el mètode *findLandmarks(Beam)* passant com a paràmetre un objecte *Beam* que contingui *pings* amb nº de ping del 200 al 215, llavors el *mosaic* retornarà com a observacions tots els *landmarks* que es trobin en aquest rang de pings transformats de [nº ping, nº píxel, id]^T a [x, y, id]^T.

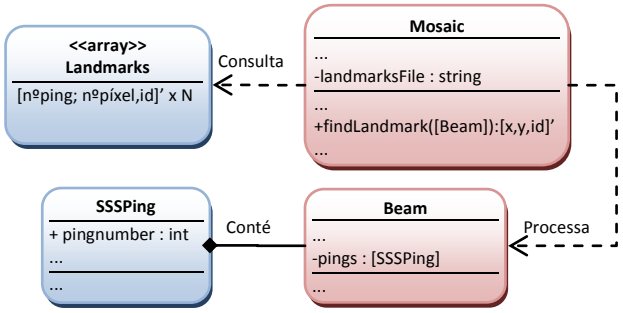

Figura 26 Classes per manejar els *landmarks*. El *mosaic* retorna les observacions segons el *beam* de *pings* que estigui processant.

Amb la simulació de l'observació de *landmarks* ja es pot plantejar la solució amb SLAM a la Secció 4.6.

4.6 SLAM

Tal com s'ha descrit a l'estudi previ l'algorisme d'SLAM es formula a partir de l'algorisme EKF. De la mateixa manera, per poder implementar l'SLAM es crea la classe *LocSlam* que és una extensió de la classe *LocEkf*. La classe *LocSlam* afegeix el mètode *updateWithLandmars(obs,Pz)* que conté el codi corresponent al punt 3.4.4 de la formulació ideada. A la Figura 27 es mostra el diagrama amb les classes necessàries per a l'execució de l'algorisme d'EKF-SLAM.

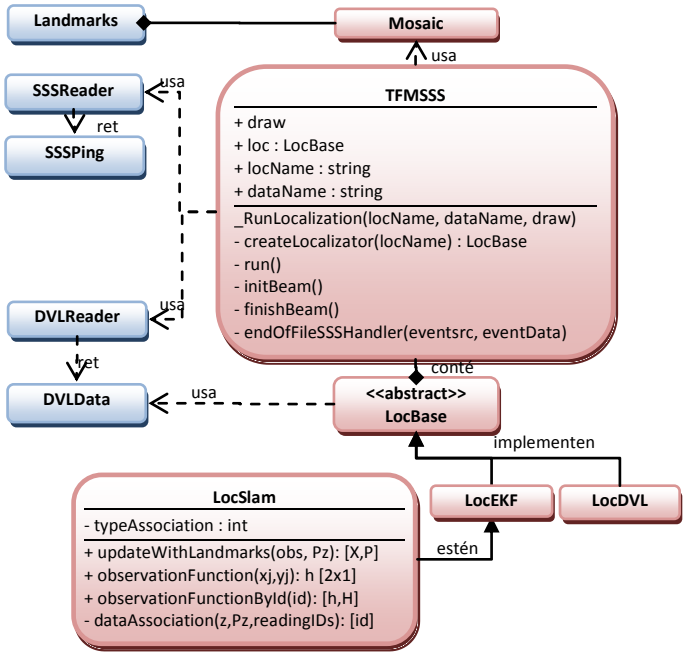

Figura 27 Classes necessàries per a la trajectòria EKF-SLAM.

Finalment s'introdueix la classe *TFMSSS* amb el mètode estàtic *RunLocalization(locName, dataName, draw)* per invocar l'execució de la localització especificant el localitzador DVL, EKF o SLAM, el conjunt de les dades d'entrada i mostrar o no l'execució per pantalla. A la Taula 10 es mostra el pseudocodi del mètode.

```
function RunLocalization(locName, dataName, draw)
  loc := createLocalizator(locName)
  dvlReader.ini(locName); sssReader.ini(locName)
  mentre no dvlReader.fi i sssReader.fi fer
    si dadesDVL.timestam < dadesSSS.timestamp llavors
      [x,P] = loc.prediction(timeStep)
      [x,P] = loc.update(dvlDades)
      mosaic.drawBeam(beam)
       obs = mosaic.findLandmarks(beam)
       si class(loc) == LocSlam llavors
         [x,P] = loc.updateWithLandmarks(obs)
      altrament
        loc.addLandmarks(obs)
      fi si
      dadesDVL := dvlReader.next()
    altrament
      beam.addSonarPing(dadesSSS)
      dadesSSS := sssReader.next()
    fi si
  fi mentre
return
```

Taula 10 Funció d'observació per transformar les coordenades de la observació

A continuació en la Figura 28 es pot veure un exemple de l'execució del mètode *TFMSSS.RunLocalization*. El resultat és que es guarden la trajectòria i les observacions en dos arxius matlab .mat, els mosaics submarins en imatges .tiff i també capes d'informació geogràfica en un directori de sortida.

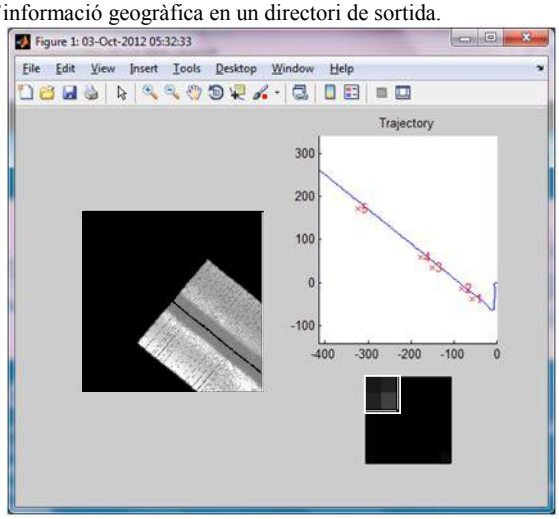

Figura 28 Execució del treball amb la localització per SLAM.

4.7 Geo localització de la informació

Un cop obtingudes les dades de sortida referents a observacions, trajectòries i imatges mosaics el següent pas lògic era voler veure-les en un visor d'informació geogràfica. Per a aquest fi s'ha utilitzat la Geographic Matlab Toolbox que permet afegir la informació que defineix cada capa geogràficament i guardar-les en format obert de la companyia Esri (*Environmental Systems Research Institute*).

Per transformar les observacions i les trajectòries a capes de punts i línies respectivament s'han implementat els mètodes *observations2Shape* i *trajectory2Shape* de la classe *Utils* i que transformen els punts de metres a estructures de dades amb

coordenades geodèsiques amb la funció *shapewrite* de Matlab. Bàsicament es afegir la informació de les coordenades que contenen el conjunt de dades.

Per guardar els mosaics com a imatges *.tif* geoposicionades la classe *Mosaic* retorna l'objecte amb les dades de referència de raster amb la funció *georasterref* dins del mètode *getRasterReference()*. Finalment es guarda la imatge amb la funció *geotiffwrite* de Matlab.

El visor GIS utilitzat per mostrar les dades ha estat el Quantum GIS [15] i s'han entrat les dades per a poder arribar a veure en un sol mapa el Port de Sóller i totes les capes de sortida generades.

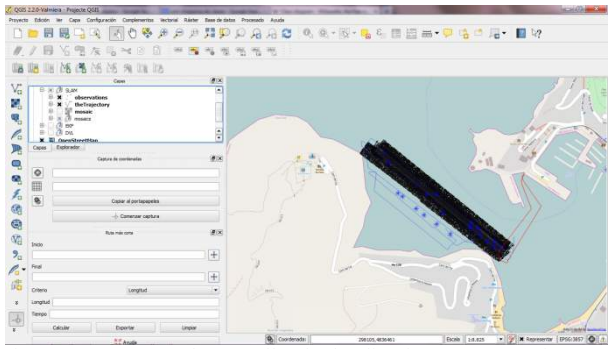

Figura 29 Captura de pantalla de l'aplicació Quantum Gis per visualitzar capes d'informació geogràfica.

Aquesta aplicació permet agrupar les dades de manera que es poden carregar i comparar trajectòries, observacions i mosaics generats amb les dades DVL, EKF o SLAM. També es poden sobreposar sobre un mapa descarregat del WMS (*Web Map Server*) de OpenStreetMap.

5 RESULTATS

Com a resultats es presenta el conjunt de classes implementades, les imatges de mosaics generades, els recorreguts calculats i el geo posicionament dels mosaics segons els recorreguts.

5.1 Conjunt de classes pel tractament de dades i proves

En la Figura 30 es presenten en forma d'esquema les classes i estructures que es corresponen amb el disseny de la solució proposat inicialment a la Secció 3.6. Hi ha 4 classes per les dades d'entrada, 2 pels mosaics i 6 pels algorismes de localització a més d'1 arxiu matlab amb els landmarks d'entrada i 2 per a les trajectòries i observacions.

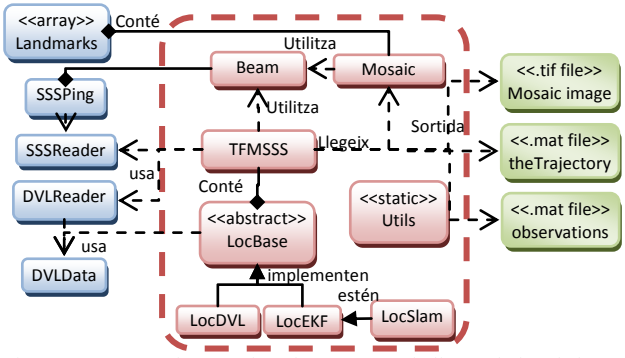

Figura 30 Resum de totes les classes per treballar amb les dades l'AUV, implementació de l'EKF-SLAM i la sortida generada.

Les classes d'entrada tenen una cohesió alta i un baix acoblament, la qual cosa ha permès provar per separat la lectura de dades SSS i DVL. La classe més dependent és la TFMSSS que coordina dins l'execució de l'algorisme la lectura de dades amb la creació de dades de sortida.

5.2 Imatges submarines i mosaics

Especificant el nom de la missió l'execució de l'algorisme processa les dades DVL, SSS i *landmarks* d'entrada per generar les imatges submarines. Per cada tram de captures de dades SSS es genera la imatge lineal i el mosaic individual. Al final de l'execució es guarda també el mosaic general. En la Figura 31 es mostren dos mosaics totals per a les missions de l'entrada del port i la bocana.

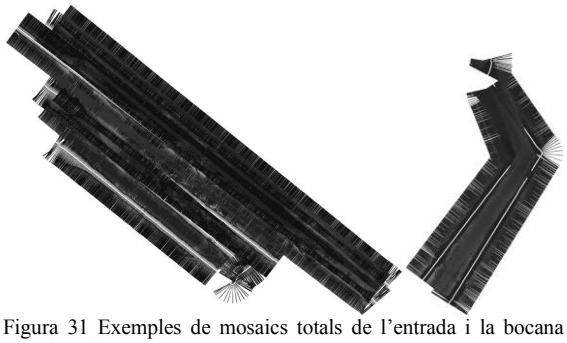

Figura 31 Exemples de mosaics totals de l'entrada i la bocana del Port de Sóller.

Per a mostrar els mosaics en aquest document s'ha augmentat la brillantor i el contrast original ja que els nivells de grisos són força baixos. També es poden observar espais entre els beams dibuixats, sobretot en el moment de girar. S'ha provat de dibuixar ping a ping en comptes d'agrupar els pings per cada segon, però s'ha descartat per no afegir més complexitat a l'algorisme, ja que els pings duen la informació del *timestamp* a nivell de segon i no en el *milisegon* en que s'han capturat.

Una altre característica negativa a destacar és que quan les dades gràfiques se solapen es calcula la mitja de les intensitats i el resultat és que el solapament es torna difuminat i els objectes perden detall. De totes formes el refinament a les imatges no influeix en la detecció dels *landmarks* ja que estan definits manualment.

5.3 Comparació de recorreguts

El fet de no disposar de la trajectòria real (*ground truth*) fa que el criteri de quina és la millor trajectòria es compliqui ja que no n'hi ha cap que a priori sigui la correcta. Per a poder-les comparar s'ha hagut de definir la funció de l'error per buscar la millor trajectòria. Aquesta funció es basa en la distància entre les observacions per un mateix *landmark* i es defineix com la mitja de les distàncies entre observacions per cada *landmark* reobservat. A la Figura 32 es pot veure gràficament la superposició de les diferents trajectòries i les seves observacions.

Per les variàncies σ de les velocitats, orientació o profunditat s'han escollit valors raonables a partir de la bibliografia. En canvi, per a les lectures GPS el valor σ_{GPS} s'ha establert a 0.01 a partir dels resultats obtinguts a la Taula 11 per tal de minimitzar l'error. Aquesta forma de calcular covariàncies és habitual en l'àmbit de l'SLAM.

σ_{GPS}	Error (μ, σ)		
	DVL	EKF	SLAM
1	4.3728, 3.1604	7.9091, 2.9536	4.9565, 5.0988
0.1	"	6.4164, 2.5219	4.2608, 2.9679
0.01	**"**	**3.4511, 2.2678**	**3.2705, 2.2496**
0.001	"	3.4228, 2.8796	3.3824, 2.3448
0.0001	"	4.9449, 3.2216	4.0541, 2.5885
0.000001	"	4.8427, 3.1383	3.8412, 2.5495

Taula 11. Resultat segons la variància de la mesura GPS σ_{GPS}.

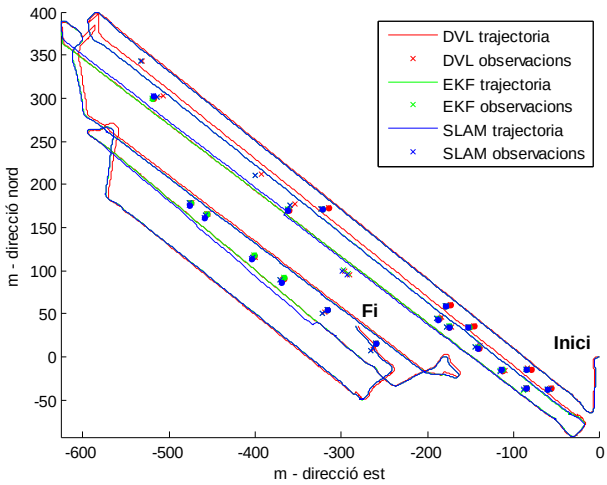

Figura 32 Superposició dels recorreguts DVL, EKF i SLAM i les diferents observacions. Amb una creu "x" es marca la primera observació del *landmark* i amb un punt les següents reobservacions.

El valor de σ_{GPS} afecta a l'actualització de la posició per a les trajectòries EKF i SLAM, com més alt és σ_{GPS} la trajectòria és més semblant a la que es calcularia sense correccions GPS i per tant presenta deriva. Com més baix és σ_{GPS} la trajectòria s'ajusta a les correccions i més gran és l'error entre les observacions. A la Figura 33 es presenta el detall de les trajectòries quan l'AUV surt a la superfície i realitza la correcció GPS per a diferents valors de σ_{GPS}. Es pot observar com la trajectòria EKF-SLAM en blau s'ajusta a la trajectòria DVL en vermell com més baix és σ_{GPS}.

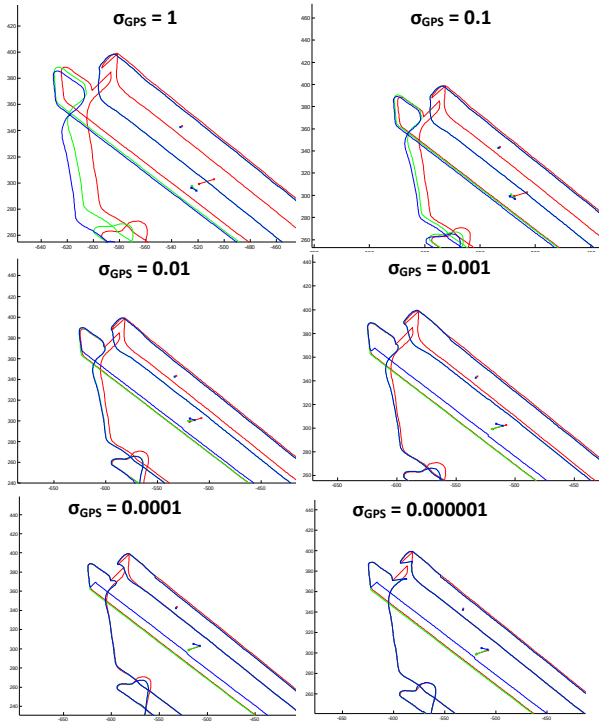

Figura 33 Detall de la superposició de les trajectòries segons la variància del GPS σ_{GPS}.

Un altre resultat a destacar és que l'aplicació de l'algorisme d'SLAM amb *landmarks* identificats millora la mesura d'error respecte la trajectòria DVL per tots els valors de σ_{GPS} excepte per σ_{GPS}=1. També es pot comprovar que l'aplicació d'SLAM millora en tots els casos respecte l'EKF gràcies a les correccions de les reobservacions. En la Figura 34 es pot veure en el detall d'un tram com la trajectòria SLAM realitza una correcció després de la reobservació d'un *landmark*.

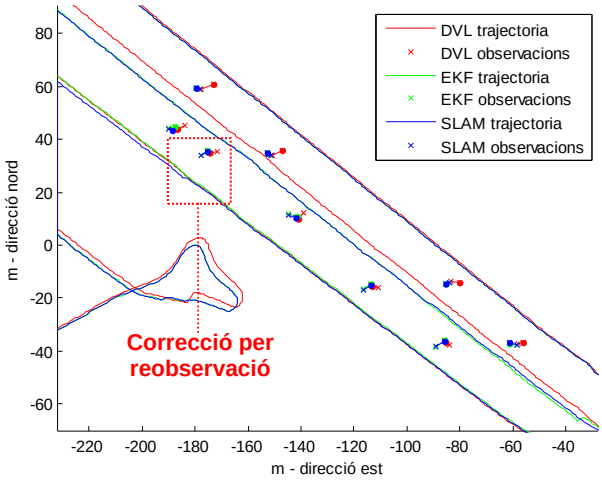

Figura 34 Detall de la superposició de les trajectòries en la que l'SLAM realitza una correcció quan reobserva *landmarks*.

El resultat de fer correccions per *landmarks* és que millora en tots els casos tal i com es pot comparar en el gràfic de la Figura 35 entre l'EKF o l'EKF-SLAM. En el gràfic es compara l'error per SLAM i EKF segons σ_{GPS}. Aplicant les correccions amb SLAM el resultat és estable mentre que l'EKF sí depèn del valor de la variància σ_{GPS} amb que s'apliquen les correccions per GPS.

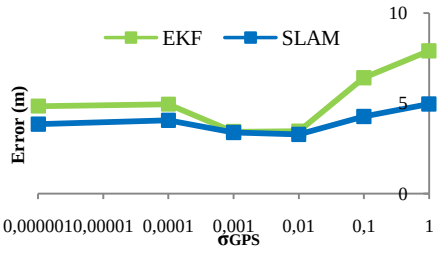

Figura 35 Comparació del valor de la funció error per a EKF i SLAM segons σ_{GPS}. L'error significa la mitja de separació en metres de les reobservacions.

5.4 Mosaics del fons marí geo-posicionades

A partir dels resultats de observacions, trajectòries i mosaics s'han creat les capes de punts, línies i *raster* per poder visualitzar sobre del mapa aquesta informació. En la Figura 36 es pot observar un mapa creat amb QGIS on se superposen dues trajectòries calculades amb l'algorisme de la solució.

Figura 36 En blau es mostra el recorregut de l'entrada del Port de Sóller en el que es basa aquest treball, en vermell un recorregut prova a la bocana.

El fet de poder superposar capes d'informació és útil per veure in situ per on es calcula que ha passat l'AUV o per dibuixar-hi els mosaics. A la Figura 37 es pot observar com es dibuixa un tram de dades SSS sobre del mapa podent identificar que la línia vertical que s'observa es correspon amb la paret de l'espigó de l'entrada.

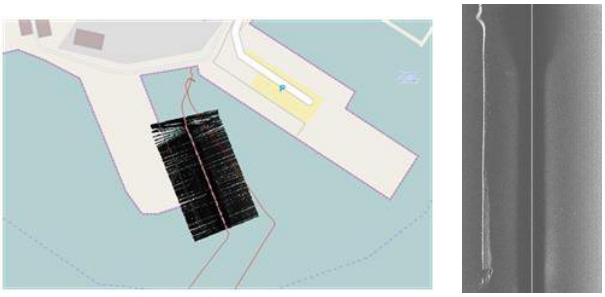

Figura 37 Detall amb el visor de l'espigó a l'esquerra i les imatges del tram lineal a la dreta.

6 CONCLUSIONS I TREBALLS FUTURS

S'ha presentat el problema de la localització i el mapeig simultani (SLAM) en la robòtica. A la Secció 2 s'han repassat els treballs previs sobre construcció de mosaics i extracció de característiques.

A la Secció 3 s'ha fet l'estudi i recerca de les característiques de l'AUV EcoMapper i de les dades DVL i SSS que proporciona. A partir del conjunt de dades s'ha ideat un model dinàmic per utilitzar-lo en la implementació de l'algorise *Extended Filter Kalman* amb SLAM i així dissenyar una solució per a la reconstrucció de trajectòries, mosaics submarins i extracció de *landmarks* amb diferents algorismes de localització.

A la Secció 4 s'ha presentat la implementació de les noves llibreries en Matlab orientat a objectes a través dels diagrames de classes i diversos exemples d'execució. També s'hi ha presentat la solució adoptada per obtenir els *landmarks* de forma manual a partir de les imatges sonar. També s'ha afegit la tasca de geo localitzar la informació a fi de tenir capes de punts, línies i *raster* d'observacions, trajectòries i mosaics submarins.

Finalment a la Secció 5 s'han presentat els resultats on es pot veure els mosaics generats amb la concatenació de *beams* de *pings* amb una alçada de 10m constant i una amplada de 60m. També es comparen els recorreguts segons les dades DVL, amb l'EKF i amb l'EKF-SLAM, amb una funció d'error basada en les reobservacions dels *landmarks*. Es comprova que el recorregut millora amb SLAM.

Per una altra banda resta la tasca pendent de poder extreure de forma automàtica els punts característics de les imatges. Per a poder arribar a identificar punts i formacions rocoses s'haurà d'afinar la creació d'imatges submarines tenint en compte més factors com l'alçada del robot, els angles, la superposició d'informació o fins i tot l'ús de batimetria o la segmentació del fons segons la textura. Tots aquests treballs futurs es podran continuar més fàcilment amb el conjunt de les classes i les llibreries creades en aquest TFM.

REFERÈNCIES

1. Kalman RE. A New Approach to Linear Filtering and Prediction Problems. Transactions of the ASME--Journal of Basic Engineering. 1960; 82(Series D): 35-45.
2. Krakiwsky EJ, Harris CB, Wong RV. A Kalman filter for integrating dead reckoning, map matching and GPS positioning. Position Location and Navigation Symposium, 1988 Record Navigation into the 21st Century IEEE PLANS'88, IEEE; 1988; 1988. p. 39-46.
3. Tena Ruiz I, Petillot Y, Lane DM. Improved AUV navigation using side-scan sonar. OCEANS 2003 Proceedings; 2003 Sept; 2003. p. 1261-8 Vol.3.
4. Burguera A, González Y, Oliver G. Underwater SLAM with robocentric trajectory using a mechanically scanned imaging sonar. Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on; 2011; 2011. p. 3577-82.
5. Ribas D, Ridao P, Neira Je. Underwater SLAM for structured environments using an imaging sonar: Springer, 2010.
6. Reed S, Petillot Y, Bell J. An automatic approach to the detection and extraction of mine features in sidescan sonar. Oceanic Engineering, IEEE Journal of. 2003; 28(1): 90-105.
7. Aulinas J, Fazlollahi A, Salvi J, LLadó X, Garcia R. Robust automatic landmark detection for underwater SLAM using side-scan sonar imaging. 11th International Conference on Mobile Robots and Competitions, Lisboa (Portugal); 2011; 2011. p. 70.
8. Stalder S, Bleuler H, Ura T. Terrain-based navigation for underwater vehicles using side scan sonar images. OCEANS 2008; 2008; 2008. p. 1-3.
9. Daniel S, Le Leannec F, Roux C, Soliman B, Maillard EPa, title=Concurrent mapping, localization using sidescan sonar, Tena Ruiz, I, et al. Side-scan sonar image matching. Oceanic Engineering, IEEE Journal of. 1998; 23(3): 245-59.
10. Rodríguez P, Piera J, Pujol N, others. New lightweight AUV at the Spanish Research Council. Instrumentation viewpoint. 2011; 11(13).
11. YSI. YSI EcoMapper AUV specifications. [Onlyne] 2014 [cited 2014 23 April]; Available from:
12. YSI. Eco Mapper Autonomous Underwater Vehicle YSI. 2010.
13. Blondel P, Murton BJ. Handbook of seafloor sonar imagery: Wiley Chichester,, UK, 1997.
14. Thrun S, Burgard W, Fox D. Probabilistic robotics: MIT press, 2005.
15. Team QD. QGIS Geographic Information System. Open Source Geospatial Foundation, 2009.

APÈNDIX

A. DIAGRAMA DE CLASSES COMPLET