Calcul ensembliste pour la robotique: développement de robots marins, sous marins, terrestres ou aériens pour l'exploration

Rapport d'avancement T0+12 mois (convention MRIS 2010-2011)

F. Le Bars, A. Bethencourt, J. Sliwka, L. Jaulin
ENSTA Bretagne
7 Novembre 2011

Descriptif de l'étude

Dans le cadre de cette étude, nous cherchons à réaliser des plateformes robotiques marines, sous-marines, terrestres et aériennes dans le but de valider des concepts et algorithmes divers, notamment en utilisant des méthodes ensemblistes telles que le calcul par intervalles. Les thématiques liées à ce projet sont variées : réalisation de robots simples et robustes, télé-opération, autonomie, localisation robuste, SLAM (Simultaneous Localization And Mapping), détection et reconnaissance d'objets, collaboration entre robots…

Cette année, nous avons travaillé sur différents types de robots et différents nouveaux concepts :

- Les robots sous-marins autonomes SAUC'ISSE et SARDINE de l'ENSTA Bretagne pour le concours SAUC-E et les données issues des robots sous-marins Redermor et Daurade du GESMA. Les problèmes principaux étaient les suivants : autonomie, contrôle, localisation robuste, détection et reconnaissance d'objets en milieu sous-marin, SLAM. Ces travaux étaient notamment liés aux thèses de Fabrice LE BARS et Jan SLIWKA.
- Meute de robots-buggys autonomes. La nouveauté était ici de réaliser un jeu de type « épervier » entre une meute de robots et un ou plusieurs humains, le but étant de montrer qu'une meute formée de robots simples (puissance de calcul limitée, vitesse, degrés de libertés, capteurs inférieurs par rapport à un être humain normal) peut être cependant capable de battre un être humain. Ces travaux ont été principalement réalisés par des étudiants de l'école : Benoît DELAUNAY et Olivier DEBANT.
- Développement d'un hovercraft autonome. Un tel robot peut être fabriqué rapidement et a un coût limité. Il peut être utilisé pour collaborer ou aider d'autres robots marins ou sous-marins, explorer rapidement une zone de manière répétitive et/ou régulière et ceci de manière discrète, participer à des missions de sauvetage (bouée de sauvetage intelligente)…
- Planeur autonome. Bien que les robots aériens motorisés autonomes soient maintenant de plus en plus courant, il semble que l'utilisation de planeurs autonomes ait été peu étudiée jusqu'à maintenant. De la même façon que des voiliers autonomes peuvent aller d'un point à un autre autant que des bateaux à moteurs mais en consommant moins d'énergie, un planeur pourrait aussi être utilisé pour aller d'un point à autre, rester dans une zone…
- Voilier autonome VAIMOS (Voilier Autonome Instrumenté pour Mesures Océanographiques de Surface) de l'IFREMER. Le but de ce robot est par exemple de remplacer les bouées munies de capteurs mesurant pendant plusieurs mois des

paramètres de la surface de l'eau. Contrairement à celles-ci, un voilier autonome peut quadriller une zone voulue sans dériver aléatoirement selon les courants et vents, et sans consommer beaucoup plus d'énergie. D'autres voiliers axés sur le bas coût, la simplicité et la robustesse pour des trajets de longue durée ont aussi été réalisés.

Déroulement de l'année

Comme tous les ans, plusieurs étudiants, stagiaires, doctorants et personnels ont été impliqués dans les activités de robotique à l'ENSTA Bretagne. L'année a comme d'habitude commencé par une présentation des robots existants aux étudiants et des initiations aux éléments de base de nos robots les lundis et mardis soirs : programmation C sous Windows et Linux, traitement d'images de webcams, découverte d'OpenCV (bibliothèque de traitement d'images), utilisation du boitier Labjack pour la commande de servomoteurs, CAO (Conception Assistée par Ordinateur)… Des sujets de projets et de stages ont ensuite été proposés aux étudiants : hovercraft, planeur, meute de buggys, voiliers. Divers cours dans le cursus des étudiants sont aussi en lien avec la robotique et le calcul ensembliste.

Meute de robots terrestres

L'idée de ce projet était de démontrer à travers une démonstration simple mais convaincante qu'une stratégie collective de robots peut s'avérer efficace face à des humains. Pour cela, un projet étudiant proposant la réalisation d'une équipe de robots mobiles devant participer à des jeux collectifs de plein air de type « épervier » contre des joueurs humains a été proposé. Les règles du jeu choisies étaient les suivantes :

1) Le jeu se déroule sur un terrain rectangulaire, plat et sans obstacle (de type terrain de football par exemple), avec plusieurs joueurs humains (habillés en rouge pour être repérés visuellement par les robots ou équipés d'un boitier de communication et localisation) face à plusieurs robots.
2) Ni les robots ni les joueurs ne peuvent sortir du terrain.
3) Pour les robots, l'objectif consiste à toucher des joueurs humains pour les éliminer. Dans la pratique, toucher signifie s'approcher à moins d'un mètre d'un joueur.
4) Les humains, quant à eux, doivent « survivre » aussi longtemps que possible en fuyant les robots. Il est interdit aux joueurs de sauter par dessus un robot. Dès qu'un joueur est touché, il est éliminé et sort du terrain.

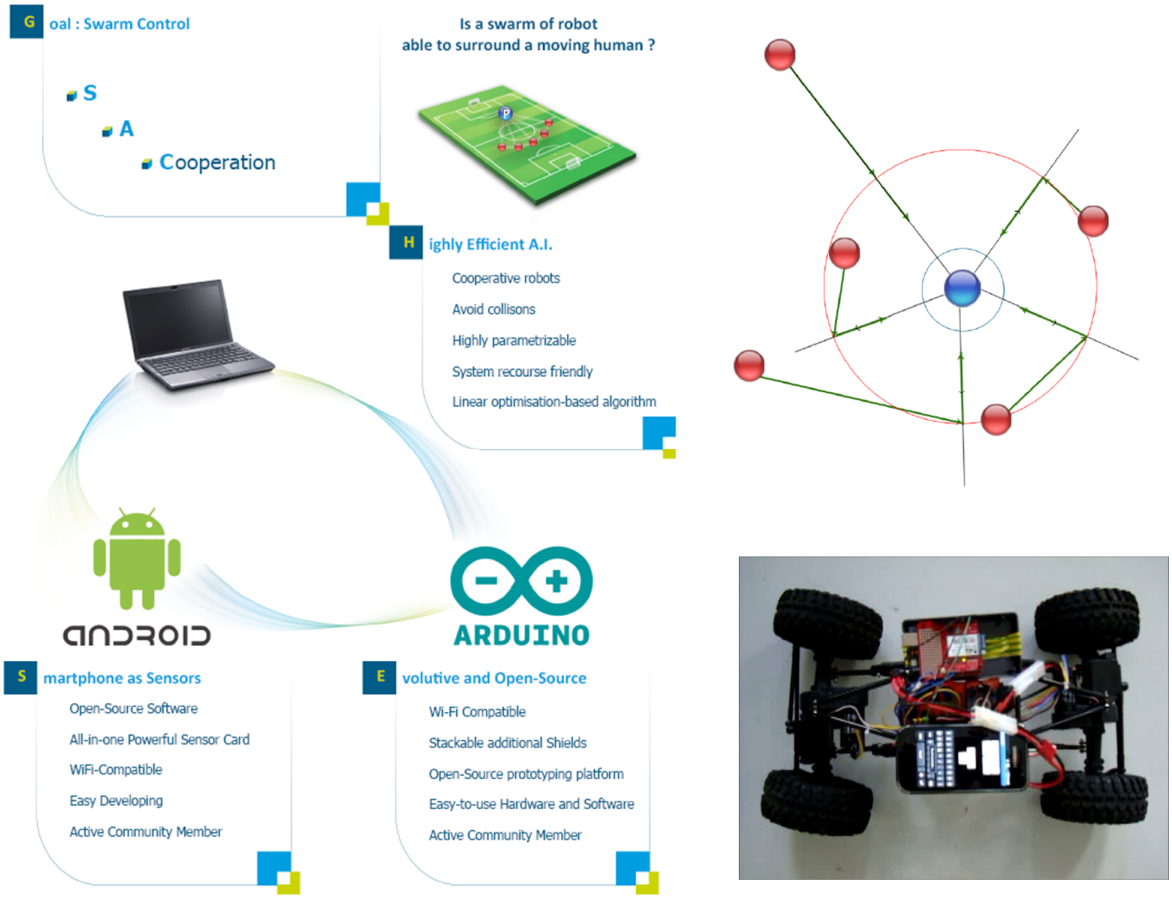


Figure 1 : Jeu des buggys

Les robots utilisés pour ce projet étaient des buggys miniatures de modélisme, munis d'un smartphone (avec GPS, boussole, accéléromètres, gyromètres comme capteurs intégrés permettant au buggy de se localiser précisément et Wifi et 3G pour communiquer avec les autres buggys de la meute) et d'une carte Arduino pour le contrôle des moteurs. Les joueurs étaient aussi munis d'un smartphone pour que les buggys connaissent aussi la position des joueurs précisément (la caméra des smartphones sur les buggys aurait aussi pu être utilisée à la place pour détecter les joueurs mais elle n'a pas été utilisée pour simplifier le problème de localisation des joueurs).

Figure 2 : Buggy et son électronique

La stratégie choisie était la suivante : tous les robots de la meute sont focalisés sur le même joueur à chaque instant et cherchent à l'encercler. Chaque robot va essayer de se positionner de manière à limiter les possibilités de mouvement du joueur et à entraver sa fuite, en s'appuyant éventuellement sur les bords du terrain de jeu. Des tests de l'approche ont été faits et validés en simulation pour un joueur contre 5 robots, et les robots réels ont été réalisés. Il reste à finaliser les tests avec les vrais robots

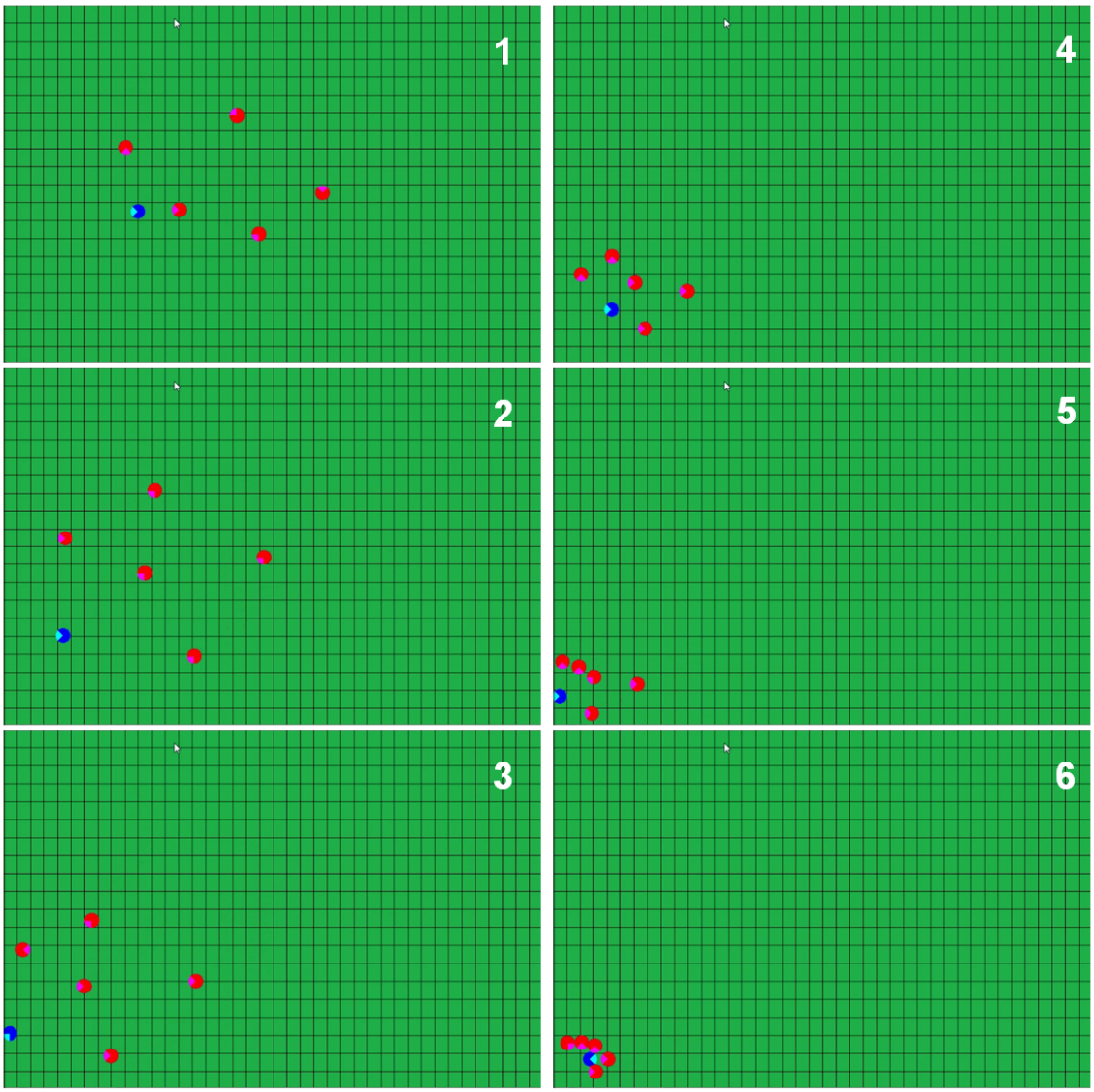


Figure 3 : Simulation de la stratégie d'encerclement d'un humain (en bleu) par 5 buggys (en rouge) collaborant en meute

Localisation robuste de sous-marins

Expérience de localisation

Dans le cadre du concours SAUC-E par exemple, les robots sous-marins doivent se localiser dans une piscine ou une marina. Avant d'implémenter l'algorithme de localisation sur notre sous-marin, nous avons validé l'approche en utilisant des données enregistrées lors d'une expérience effectuée par un robot sous-marin d'une autre université. L'expérience à considérer ici a été conçue à la base pour illustrer le fonctionnement d'un algorithme de

SLAM sous-marin [1]. Les données ont été recueillies au cours d'une étude approfondie d'un port de plaisance abandonné dans la Costa Brava (Espagne). L'AUV (sous-marin autonome) de L'Université de Girone a enregistré un jeu de données le long d'une trajectoire de 600 m qui comprenait une petite boucle autour du réservoir principal et une droite de 200 m à travers un canal étroit. L'ensemble de données comprenait des mesures d'un sonar sectoriel (Miniking Tritech), un DVL - Doppler Velocity Log - (Argonaut SonTek) et d'une centrale inertielle (Xsens MTI). Pour pouvoir valider le résultat du traitement de ces données, l'AUV a navigué près de la surface et a été muni d'un GPS qui a enregistré la trajectoire réelle du robot (vérité terrain). La figure suivante montre un échantillon de données sonar, les points verts indiquent que le faisceau sonar a trouvé un obstacle (qui est en fait un mur de la marina). Comme on peut le voir, les données sont bruitées et contiennent beaucoup de valeurs aberrantes. Les longs segments signifient que les obstacles sont hors de portée du sonar (c.à.d. au-delà de 50 m).

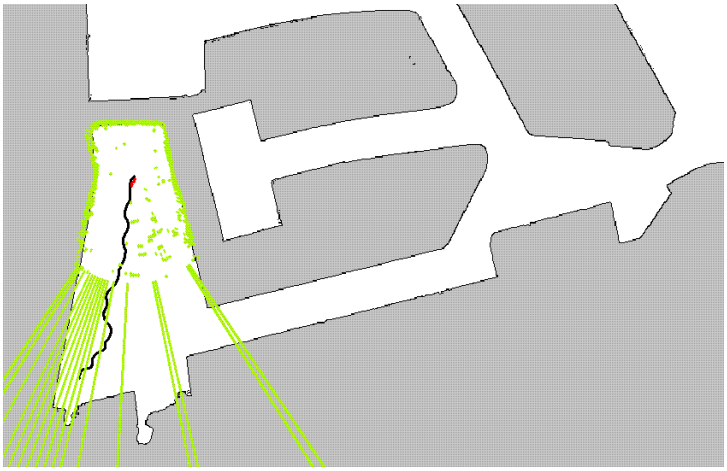


Figure 4 : Données du sonar prise dans une marina de la Costa Brava

Mise en équation du système

Considérons un système caractérisé par les équations d'état suivantes :

$$\mathbf{f_k} : \mathbb{R}^m \rightarrow \mathbb{R}^m, \mathbf{g_k} : \mathbb{R}^m \rightarrow \mathbb{R}^\ell$$
$$\mathbf{x_{k+1}} = \mathbf{f_k}(\mathbf{x_k})$$
$$\mathbf{y_k} = \mathbf{g_k}(\mathbf{x_k}).$$

Dans notre cas, $\mathbf{x}_k$ est la position du robot à l'instant k. $\mathbf{f_k}$ caractérise la dynamique du robot et $\mathbf{y}_k$ est le vecteur de sortie qui est ce qu'on mesure (ici les mesures sonar). $\mathbf{y}_k$ et $\mathbf{x}_k$ sont reliés par la fonction d'observation $\mathbf{g_k}$ qui exprime des relations géométriques entre la position, les mesures et la carte. On notera par $\mathbb{X}_i$ et $\mathbb{Y}_k$ les ensembles contenant $\mathbf{x}_k$ et $\mathbf{y}_k$ respectivement. En utilisant la formule récursive des équations d'états, nous obtenons le système d'équations suivant :

$$\begin{cases} \mathbf{x}_{k+1} = \mathbf{f}_k \circ \mathbf{g}_k^{-1}(\mathbf{y}_k) \\ \mathbf{x}_{k+1} = \mathbf{f}_k \circ \mathbf{f}_{k-1} \circ \mathbf{g}_{k-1}^{-1}(\mathbf{y}_{k-1}) \\ \dots \\ \mathbf{x}_{k+1} = \mathbf{f}_k \circ \mathbf{f}_{k-1} \circ \dots \mathbf{f}_{k-n} \circ \mathbf{g}_{k-n}^{-1}(\mathbf{y}_{k-n}) \\ \mathbf{x}_{k+1} = \mathbf{f}_k(\mathbf{x}_k) \\ \mathbf{x}_{k+1} \in \mathbb{R}^m, \mathbf{x}_k \in \mathbb{X}_k, \mathbf{y}_i \in \mathbb{Y}_i, i \in [k-n..k]. \end{cases}$$

Nous cherchons à connaitre $\mathbf{x}_{k+1}$ en fonction de l'état précédent $\mathbf{x}_k$ à partir de plusieurs mesures $\mathbf{y}_i, i \in [k-n..k]$. Comme nous allons le voir dans la partie suivante, un tel système d'équations se résout facilement avec les méthodes ensemblistes.

Méthodes ensemblistes et données aberrantes

Soit un système d'équations sous la forme :

$$\begin{cases} \mathbf{f}_0(\mathbf{x}) = \mathbf{0} & \\ \mathbf{f}_1(\mathbf{x}) = \mathbf{0} & \mathbf{f}_i : \mathbb{R}^m \to \mathbb{R}^{n_i}, \\ \dots & \mathbf{x} \in \mathbb{X}. \\ \mathbf{f}_n(\mathbf{x}) = \mathbf{0} & \end{cases}$$

Résoudre ce système d'équations revient à trouver l'ensemble des points qui satisfont toutes les équations i.e.

$$\mathbb{S} = \{\mathbf{x} \in \mathbb{R}^m, \forall i \in [0..n], \mathbf{f}_i(\mathbf{x}) = \mathbf{0}\}$$

Les méthodes ensemblistes permettent la manipulation d'ensembles (voir [2] et [3]). Par exemple, ces méthodes permettent de calculer une intersection d'ensembles $\mathbb{A} = \mathbb{B} \cap \mathbb{C}$, calculer une union $\mathbb{A} = \mathbb{B} \cup \mathbb{C}$, faire de l'inversion ensembliste $\mathbb{A} = f^{-1}(\mathbb{B})$, calculer l'image d'un ensemble par une fonction $\mathbb{A} = f(\mathbb{B})$… Pour résoudre notre système d'équations, nous allons donc calculer des sous ensembles solutions pour chaque équation. Nous obtenons donc pour chaque équation

$$\mathbb{X}_i = \mathbf{f}_i^{-1}(0).$$

Puisqu'on cherche les points qui appartiennent à tous les sous-ensembles solutions, la solution finale recherchée sera donc sous la forme suivante

$$\mathbb{S} = \bigcap_{i \in \{0..n\}} \mathbb{X}_i$$

Dans le cas où il y a des données aberrantes, certaines équations du système d'équation du robot qu'on souhaite résoudre ne seront pas satisfaites car elles se basent sur

des mauvaises données mesurées. Il faudra donc trouver un moyen de ne pas prendre en compte ces « fausses » équations. Le problème est qu'on ne sait pas quelles équations sont fausses. La résolution normale qui fait l'intersection de tous les sous ensembles solutions $\mathbb{X}_i$ nous donnera un l'ensemble vide comme solution. Ceci est normal car il n'y a aucun point qui satisfait toutes les équations puisque certaines d'entre elles sont fausses. Les méthodes ensemblistes nous apportent la solution avec un autre type d'intersection appelé l'intersection q-relaxée (voir [9] et [10]) exprimée par

$$\mathbb{S}_q = \bigcap_{i\in\{0..n\}}^{\{q\}} \mathbb{X}_i = \{\mathbf{x} \in \mathbb{R}^m, \exists \mathbb{I} \subset \{1,..,n\}, card(\mathbb{I}) = n-q, \forall i \in \mathbb{I}, \mathbf{x} \in \mathbb{X}_i\}$$

Un exemple avec 5 ensembles $\mathbb{X}_1, \mathbb{X}_2, \mathbb{X}_3, \mathbb{X}_4, \mathbb{X}_5$ est représenté ci-dessous.

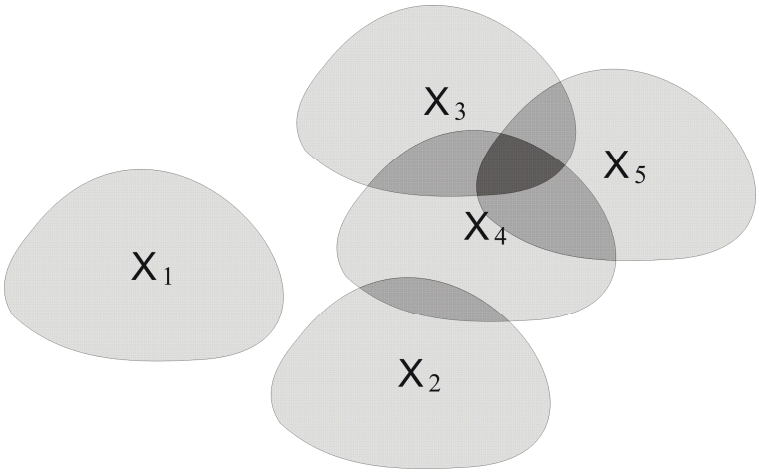


Figure 5 : Exemple de l'intersection relaxée de 5 ensembles

Dans l'exemple nous avons :

$$\bigcap_{i\in\{0..5\}}^{\{0\}} \mathbb{X}_i = \bigcap_{i\in\{1..5\}} \mathbb{X}_i = \emptyset$$

$$\bigcap_{i\in\{0..5\}}^{\{2\}} \mathbb{X}_i = \mathbb{X}_3 \cap \mathbb{X}_4 \cap \mathbb{X}_5$$ La partie en gris foncé

$$\bigcap_{i\in\{0..5\}}^{\{5\}} \mathbb{X}_i = \bigcup_{i\in\{1..5\}} \mathbb{X}_i$$

Résultats de la localisation dans la marina

La figure suivante montre une comparaison entre la trajectoire GPS de référence (en noir) et la trajectoire (en bleu) obtenue par la méthode Dead Reckoning - estimation aveugle -

qui est obtenue par fusion de données. Nous pouvons observer que la trajectoire obtenue par la méthode Dead Reckoning souffre d'une dérive non négligeable notamment en indiquant des positions à l'extérieur du canal.

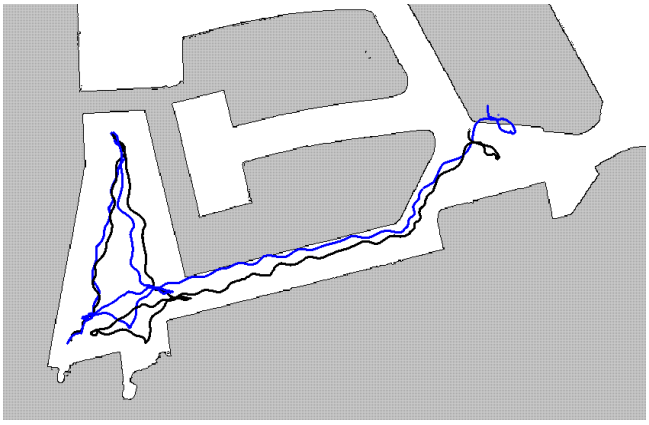


Figure 6 : Comparaison entre le Dead Reckoning et la trajectoire GPS de référence

La trajectoire calculée avec les méthodes ensemblistes est représentée sur la figure qui suit. Notre algorithme retourne la trajectoire sous forme de boites (en rose), mais nous prenons habituellement le centre de ces boites comme position réelle (en rouge). La trajectoire rouge suit la trajectoire GPS. L'exécution de l'algorithme est temps réel sur un ordinateur portable avec un processeur Intel Core 2 Duo.

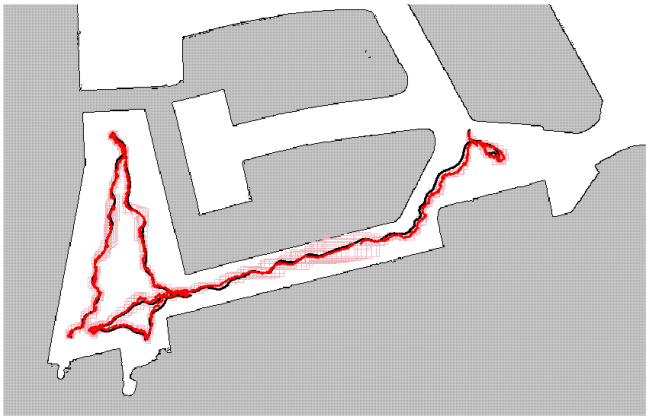


Figure 7 : Comparaison entre notre méthode et la trajectoire GPS de référence

Localisation de l'AUV SAUC'ISSE

Les équations d'évolution du sous-marin SAUC'ISSE dans le plan peuvent être écrites de cette manière:

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = u_2 - u_1 \\ \dot{v} = u_1 + u_2 - v \end{cases}$$

où x, y sont les coordonnées du robot, θ son orientation et v sa vitesse. Les entrées u_1 et u_2 sont les accélérations (ou forces) fournies par les propulseurs droit et gauche. Ce modèle normalisé (i.e. avec tous les coefficients mis à 1) correspond à un robot sous-marin à profondeur constante (la régulation en profondeur est considérée indépendante du mouvement dans le plan) sans roulis ni tangage (compensés mécaniquement).

Le système peut être discrétisé comme ceci:

$$\mathbf{x}(k+1) = \mathbf{f}_k(\mathbf{x}(k), \mathbf{u}(k))$$

où δ est le pas de temps, $\mathbf{x} = (x, y, \theta, v)$ est le vecteur d'état, $\mathbf{u} = (u_1, u_2)$ sont les entrées et

$$\mathbf{f}_k\left(\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}, \begin{matrix} u_1 \\ u_2 \end{matrix}\right) = \begin{pmatrix} x_1 + \delta . x_4 . \cos(x_3) \\ x_2 + \delta . x_4 . \sin(x_3) \\ x_3 + \delta . u_2(k) - \delta . u_1(k) \\ x_4 + \delta . u_1(k) + \delta . u_2(k) - \delta . x_4 \end{pmatrix}$$

On suppose que le robot se déplace dans un environnement de forme connue (localisation). Le sonar du robot permet de mesurer la distance entre les bords du bassin et le robot dans la direction pointée par l'angle courant du sonar (cette direction change en fonction du temps car c'est un sonar rotatif).

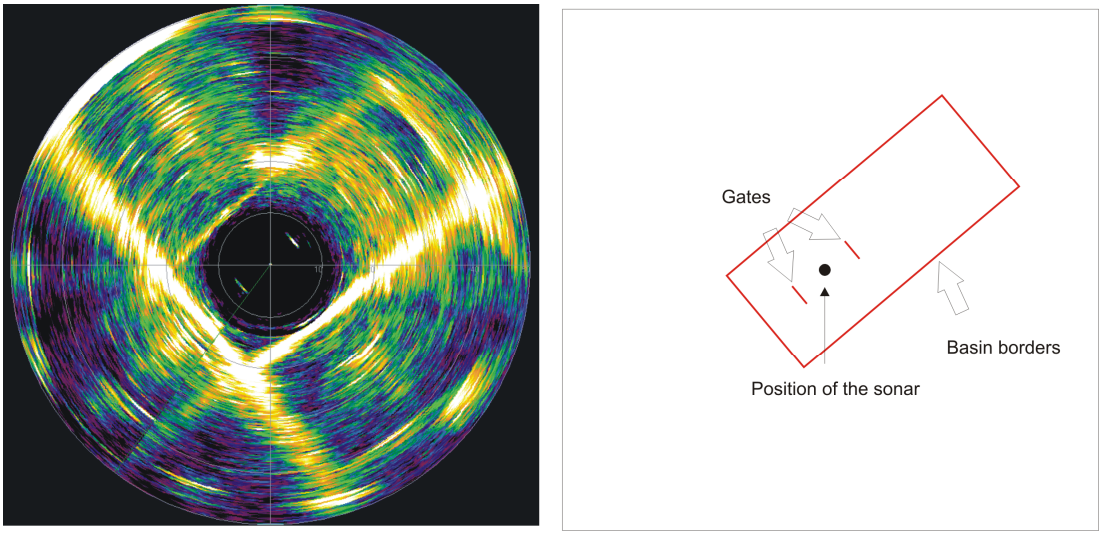


Figure 8: Image sonar de la zone de compétition de SAUC-E 2009

Si le bassin est composé de bords verticaux, l'équation d'observation du système peut alors s'écrire :

$$\mathbf{y}(k) = \mathbf{g}_k(\mathbf{x}(k), \mathbf{u})$$

avec $\mathbf{y} = (d, \theta, \dot{\theta})$ et

$$\mathbf{g}_k\left(\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}, \begin{matrix} u_1 \\ u_2 \end{matrix}\right) = \begin{pmatrix} g_{1,k}(\mathbf{x}) \\ \theta \\ \dot{\theta} \end{pmatrix}$$

où $g_{1,k}$ est une fonction donnée par un algorithme de calcul de distance entre 2 segments.

L'une des tâches les plus difficiles pour le sous-marin est de se localiser dynamiquement dans le bassin. La plupart des solutions existantes pour résoudre ce problème sont basées sur des techniques probabilistes (filtres de Kalman, filtres particulaires [1][4]…). Dans notre sous-marin, un observateur d'état utilisant le calcul par intervalles et robuste par rapport aux données aberrantes venant parfois du sonar permet au robot d'estimer sa position de manière relativement précise. De plus, une méthode originale utilisant une notion de contracteur sur image (voir [5] pour plus d'informations sur les contracteurs et [6] pour le contracteur sur image) et une notion d'accumulateurs représentés par des polynômes à coefficients intervalles (voir [6]) a été élaborée et testée sur notre robot SAUC'ISSE dans la piscine de ENSTA Bretagne et au concours SAUC-E 2011.

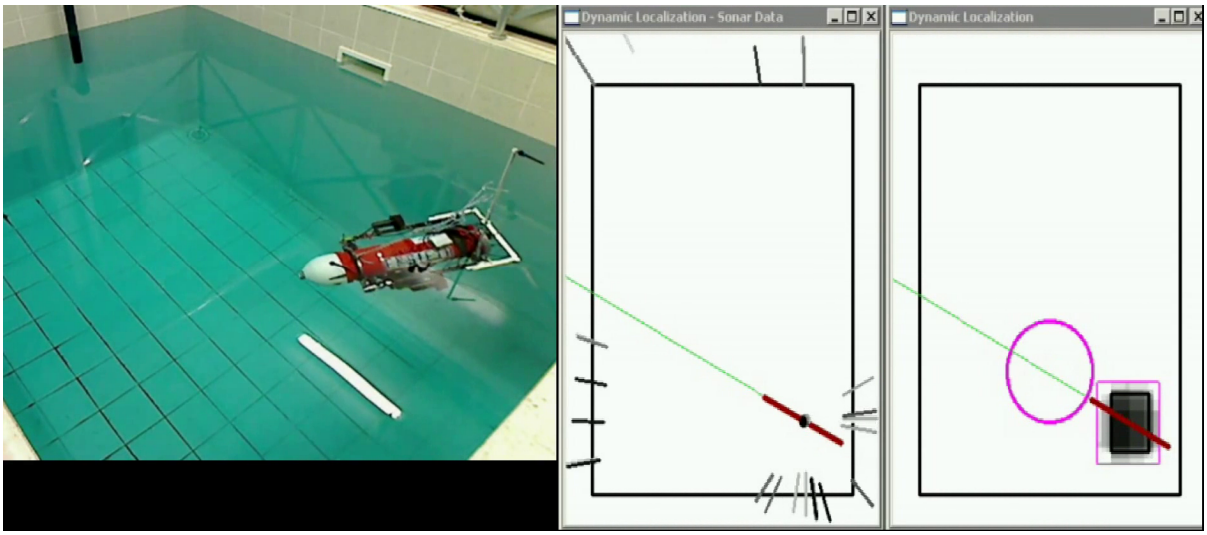


Figure 9: Localisation dynamique avec données aberrantes avec l'approche des accumulateurs: la boîte la plus foncée correspond à celle compatible avec le plus de données sonar. Ici, le sous-marin SAUC'ISSE effectue des allers-retours de manière autonome (2 waypoints dans la piscine ont été fixés, le 1er est dessiné sous forme de rond rose) dans la piscine de l'ENSTA Bretagne (rectangulaire, de dimensions 3m*4m*3m)

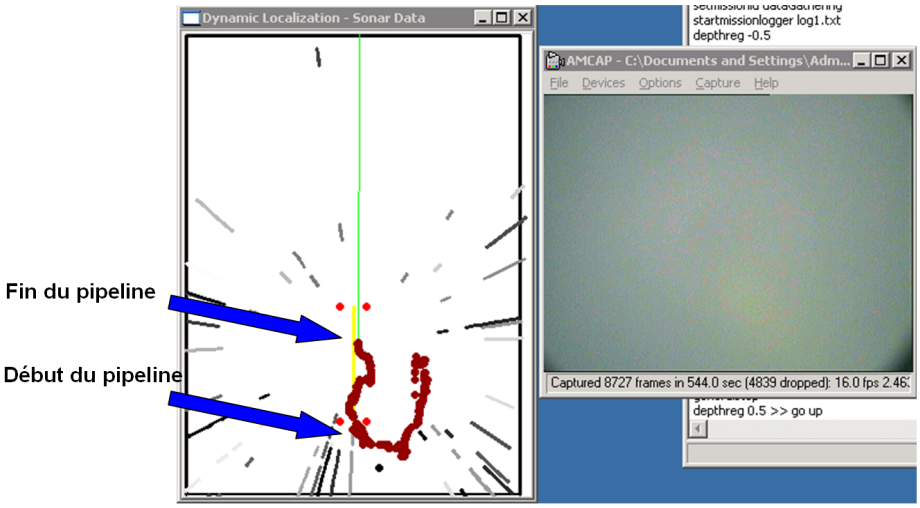


Figure 10: Utilisation de la localisation dynamique pour trouver la forme du pipeline jaune du concours SAUC-E 2011. Ici, le sous-marin SAUC'ISSE est utilisé en mode téléopéré pour explorer la zone de compétition. En suivant le pipeline grâce à la caméra inférieure du sous-marin et en dessinant les positions successives du robot calculées par l'algorithme de localisation dynamique (points rouges foncés), nous avons pu constater que le pipeline avait une forme de «S»

Déroulement du concours SAUC-E 2011

Comme tous les ans, nous avons participé au concours SAUC-E (Student Autonomous Underwater Challenge - Europe), qui a eu lieu cette année du 4 au 10 Juillet à La Spezia en Italie. Nous avons terminé 4ème sur 10 équipes, les 1ers étant l'Université de Luebeck (Allemagne), les 2èmes l'Université de Girona (Espagne) et les 3èmes l'Université de Bremen (Allemagne). 10 équipes étaient présentes.

Figure 11 : L'équipe de l'ENSTA Bretagne pour SAUC'E 2011 avec les sous-marins SAUC'ISSE et SARDINE, ainsi que notre hovercraft (robot de surface)

Les épreuves se sont déroulées dans les mêmes conditions que l'année précédente : dans l'eau de mer d'une marina du NURC (NATO Undersea Research Centre).

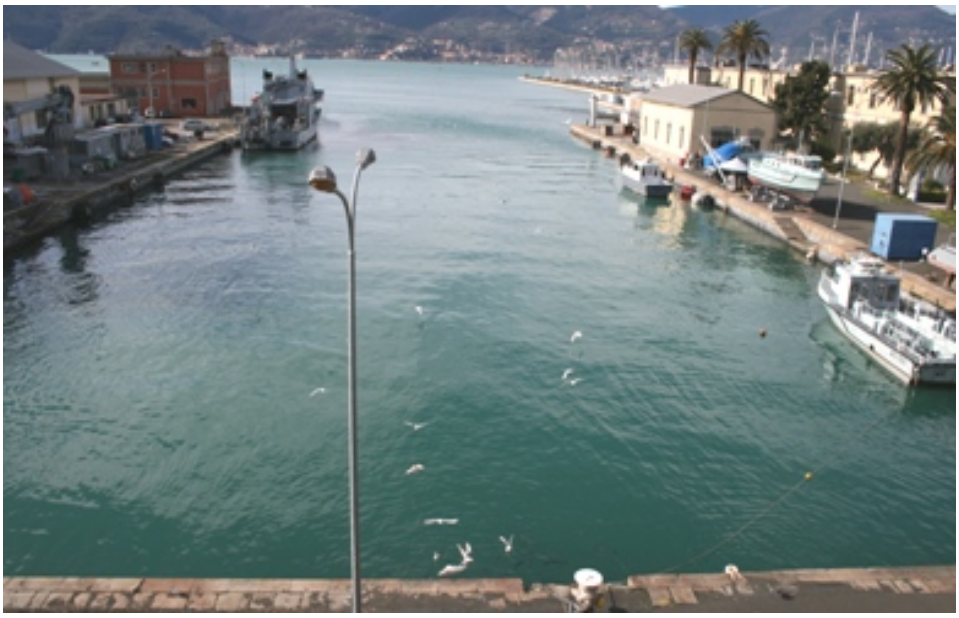


Figure 12 : Zone de compétition de SAUC-E 2011

Les épreuves que les AUV (Autonomous Underwater Vehicle) devaient effectuer étaient les suivantes :

1) Choisir le point de départ du sous-marin. 2 points de départs étaient possibles. Le premier dit « Start 1 » se trouvait sur le bord Nord ou Sud de la marina et l'autre dit « Start 2 » se trouvait au milieu.
2) Passer ensuite par la porte de validation (nommée Validation Gate, aussi utilisée comme épreuve de qualification).
3) Faire demi-tour et suivre un pipeline non rectiligne (en « S ») de couleur jaune à 50 cm de distance.
4) Contourner puis libérer une bouée blanche se trouvant à proximité de la fin du pipeline en coupant le fil qui la maintenait.
5) Suivre les murs formant l'un des coins de la marina à une distance supérieure à 2 m.
6) Suivre le véhicule de surface autonome du NURC (par le dessous). Celui-ci était muni d'un pinger (générateur de signaux acoustiques) émettant toutes les secondes à 12 kHz.
7) Faire surface sous le véhicule de surface.

Un fichier contenant les positions et actions du sous-marin en fonction du temps, ainsi que des preuves de détection (images du pipeline…) devaient être fournis aux organisateurs à la fin des épreuves.

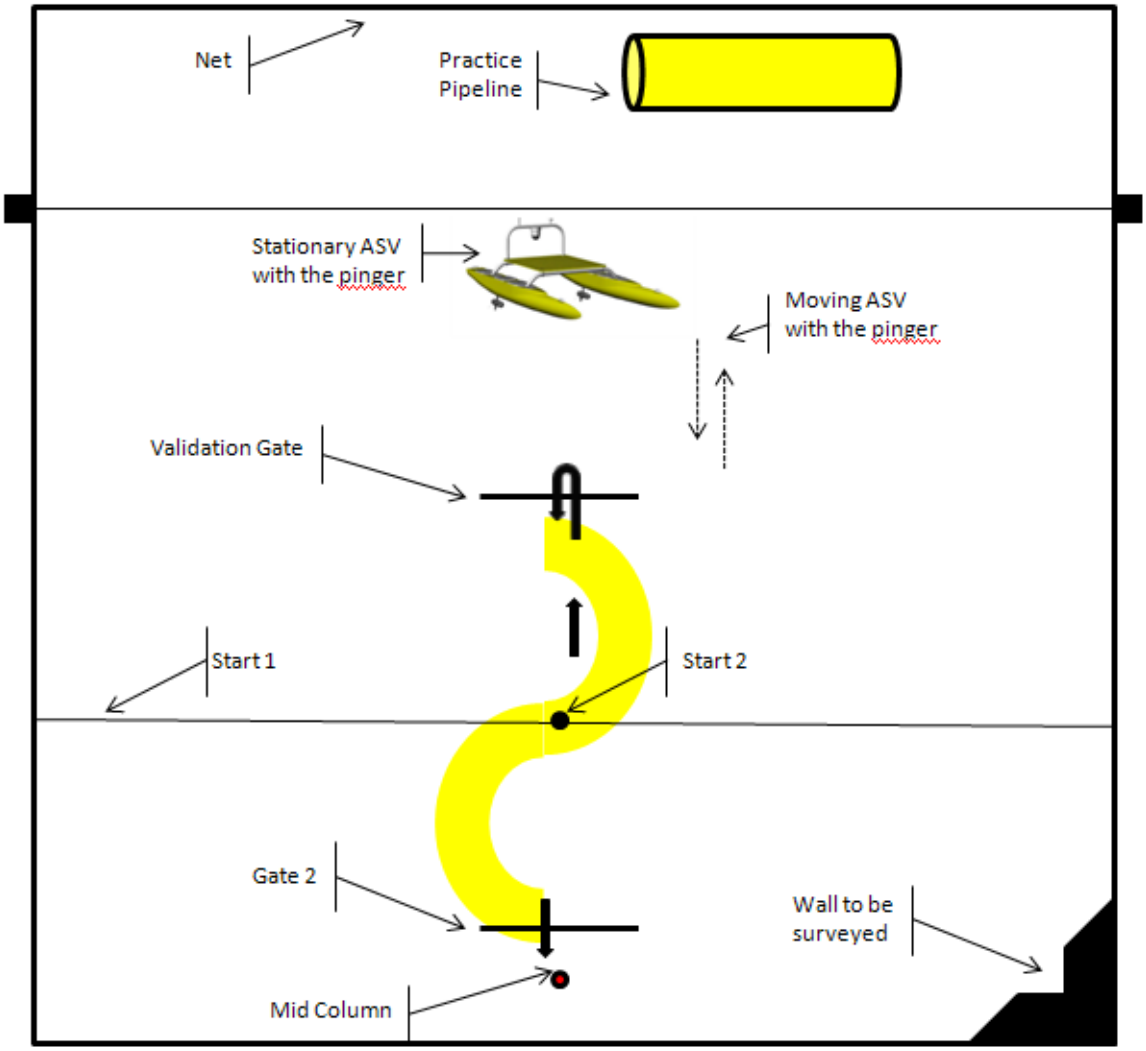


Figure 13 : Epreuves du concours SAUC-E 2011

Durant la compétition, nous avons eu plusieurs problèmes :

- L'alimentation des caméras de SAUC'ISSE a cessé de fonctionner de la même façon que l'année dernière, puis après remplacement les caméras ont elles-mêmes montré des signes de faiblesses. Nous avons dû les remplacer et modifier rapidement l'alimentation pour tout faire pour que le problème se reproduise. Cependant, nous n'avons pas encore clairement identifié sa cause.
- Lors d'un transport, l'un des connecteurs étanches de SAUC'ISSE a été endommagé. Il a été rapidement remplacé.
- Un fil (pour retenir le sous-marin) s'est coincé dans le propulseur vertical de SARDINE alors qu'il était en fonctionnement, ce qui l'a détruit. Il a fallu le remplacer aussi.
- Un brouilleur WIFI était présent sur la zone de compétition, sans que les organisateurs le sachent. Nous avons pu détecter sa présence et prévenir les organisateurs pour qu'ils le configurent pour autoriser nos réseaux WIFI. Toutes les équipes étaient affectées par ce problème.
- Un problème de gestion des temps de calculs a empêché de réussir correctement plusieurs missions lors de la finale.

Malgré ces problèmes, nous étions 3ème aux qualifications, puis 2ème après la demi-finale. Nous avons réussi (au moins en partie, individuellement ou au cours d'un enchainement pour certaines) les épreuves 1, 2, 3, 5. Malheureusement, le dernier problème cité nous a fait perdre des places au classement final.

Figure 14 : SAUC'ISSE, le robot sous-marin principal

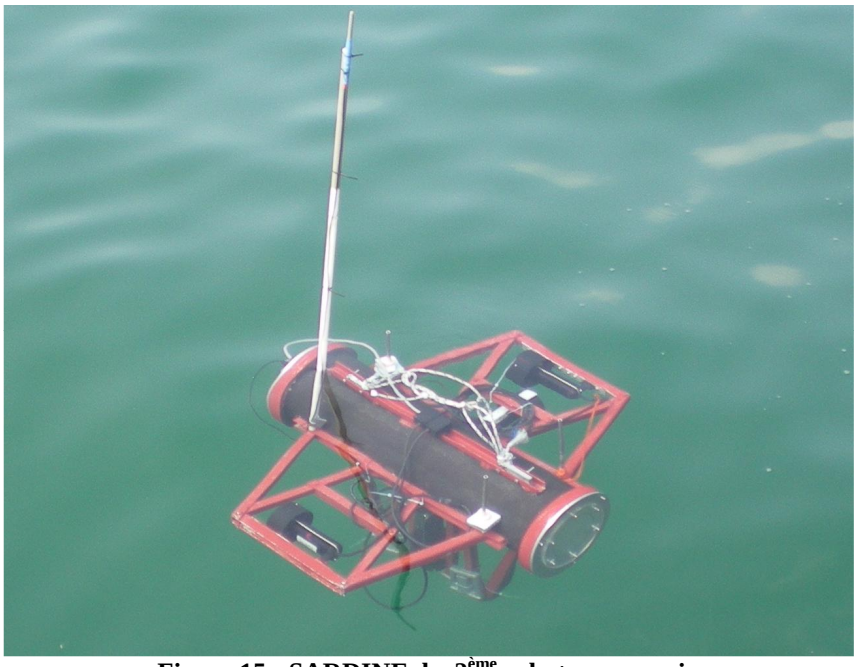


Figure 15 : SARDINE, le 2ème robot sous-marin

Logiciel d'aide à la détection de mines sous-marines

Présentation

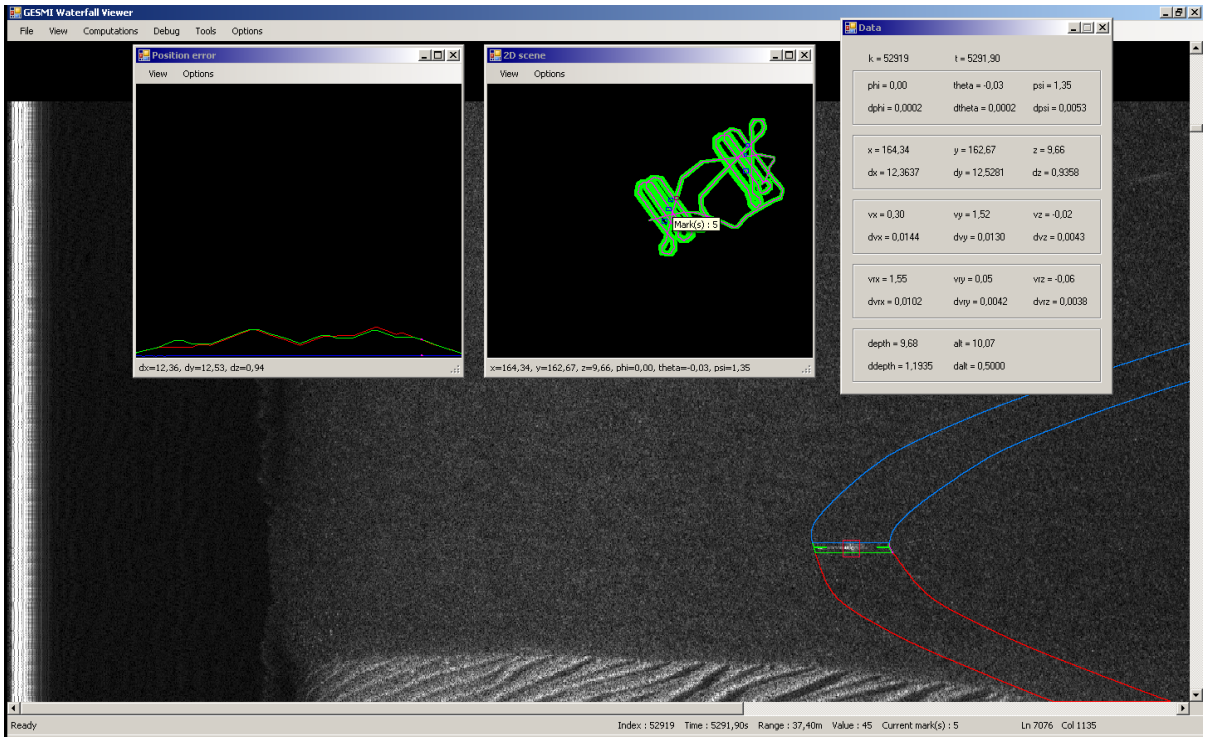


Figure 16 : Capture d'écran de GESMI, programme qui calcule et affiche la position des mines, l'enveloppe de la trajectoire, l'erreur en position d'un sous-marin lors d'une expérience de quadrillage de zone inconnue, et qui aide à identifier les mines parmi les différentes détections

Dans le cadre d'un contrat avec le GESMA sur le traitement des données de leur sous-marin Daurade, un logiciel d'aide à la détection de mines par SLAM (Simultaneous Localization And Mapping) par intervalles a été réalisé. Ce logiciel, nommé GESMI (Guaranteed Estimation of Sea Mines with Intervals) est basé sur celui déjà commencé en 2007 et qui traitait les données du Redermor (voir [7] et [8]). C'est un programme permettant d'estimer la trajectoire d'un robot sous-marin et la position d'objets particuliers (préalablement détectés au moins 1 fois par un opérateur humain en parcourant les données sonar) en utilisant le calcul par intervalles. Il permet en même temps de vérifier la cohérence entre les données qui lui sont fournies en entrée et d'aider à la détection des objets sur l'image sonar.

Les données dont il a besoin pour fonctionner sont les suivantes :

- Données de navigation du sous-marin (angles du sous-marin, altitude, profondeur, vitesses dans le repère mobile, quelques positions GPS connues)
- La distance sur l'image sonar où un amer a été détecté et s'il est sur tribord ou bâbord.
- Les temps et estimation d'erreur maximale pour chacune des données.
- La portée et l'erreur en distance du sonar.
- Les images sonar.

Les données qu'il produit ou affiche sont les suivantes :

- La trajectoire du sous-marin et la position des amers (sous forme d'enveloppe et centre de l'enveloppe ainsi que d'intervalles).
- L'erreur en position au cours du temps.
- La reconstitution de l'image sonar indiquant les distances sous-marin à amers, montrant les endroits où le sous-marin est passé à proximité des amers.

Le programme a été repris et amélioré notamment pour permettre son utilisation avec les données de la Daurade (rajout de la gestion de la partie bâbord des images sonar...), faciliter l'interprétation des résultats (affichage des erreurs en position...) et aider au maximum l'opérateur humain dans sa recherche de mines (amélioration des indications de positions estimées de mines directement sur l'image sonar...).

Concepts

GESMI utilise à la fois les données de navigation (vitesses, angles...) et les détections d'amers vus par un opérateur humain sur les images sonar pour estimer le plus précisément possible une enveloppe de la trajectoire du sous-marin et de la position réelle de ces amers. La cohérence entre les données de navigation et les détections est vérifiée en même temps.

Dans le programme, toutes les données sont représentées par des intervalles, pavés ou tubes. Par exemple, si à t = 308,0 s la centrale inertielle du sous-marin indique un angle de -0,0258 rad et que la documentation du constructeur de la centrale inertielle précise que son erreur maximale est de 0,0001 rad, l'angle à t = 308,0 s sera représenté dans le programme par l'intervalle [-0,0258-0,0001;-0,0258+0,0001]=[-0,0259;-0,0257]. Tous les intervalles obtenus à partir des fichiers de données chargés par le programme sont ensuite utilisés dans des calculs mettant en jeu les équations d'état du sous-marin ainsi que d'autres relations géométriques. Ces équations et relations sont discrétisées (par rapport au temps) et mettent en jeu des additions, soustractions, cos, sin... d'intervalles de la même façon qu'avec des nombres réels, ainsi que des intersections et unions d'intervalles. Si un intervalle vide est trouvé lors d'un calcul, cela signifie en général qu'il y a une incohérence dans les données de navigation ou détections. Le plus souvent, elle est due à des données capteur moins précises que l'on croyait. Il faut donc augmenter l'erreur avec laquelle on considère ces données. Des détections imprécises d'objets sur l'image sonar peuvent aussi provoquer des incohérences. Dans ce cas il faut vérifier les détections et éventuellement éliminer celles qui sont trop approximatives pour pouvoir les corriger par la suite à l'aide de la reconstitution de l'image sonar.

Les 3 principales étapes de calcul pour la reconstitution de la trajectoire du sous-marin sont les suivantes :

- Propagation : les équations d'état du sous-marin donnent une relation entre les positions et données de navigation à t et t + dt. Des calculs sont donc faits en partant de t = 0 s vers la …n. Par exemple, la position GPS du sous-marin à t = 0 s va permettre d'estimer ses positions suivantes avec une erreur qui augmente au cours du temps et qui dépend aussi de l'erreur avec laquelle on connait ses différentes données de navigation (vitesses, angles...).
- Retro-propagation : les équations d'état du sous-marin peuvent donner aussi une relation entre les positions et données de navigation à t + dt et t. Des calculs sont donc faits en partant de la fin vers t = 0 s. Par exemple, la position GPS finale du sous-marin va permettre d'améliorer l'estimation de ses positions précédentes.
- Mise en cohérence avec les objets particuliers détectés sur l'image sonar : des relations géométriques entre les positions du sous-marin et celles des objets à leurs temps de détection sur l'image sonar permettent d'une part d'estimer la position des objets dans le repère de navigation et d'améliorer l'estimation de la trajectoire du sous-marin d'autre part (principe du SLAM).

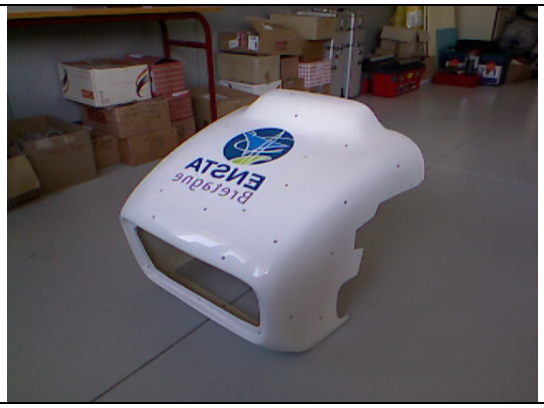
Ces 3 étapes doivent en général être réexécutées plusieurs fois pour atteindre la meilleure précision (on s'arrête lorsqu'on voit que l'estimation de l'erreur et de la position ne change plus).

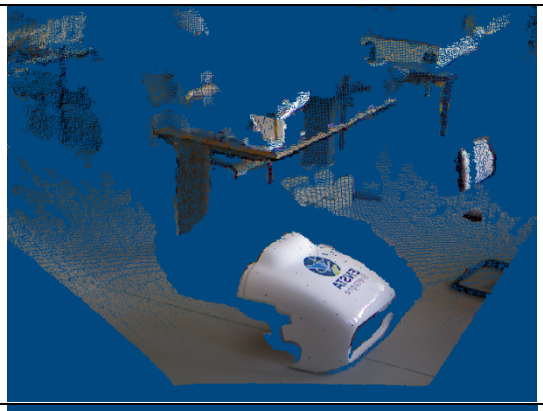
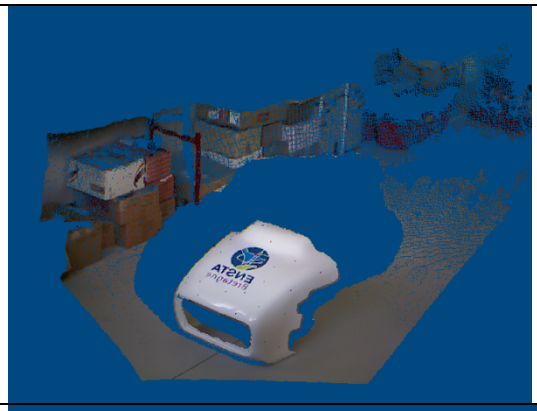
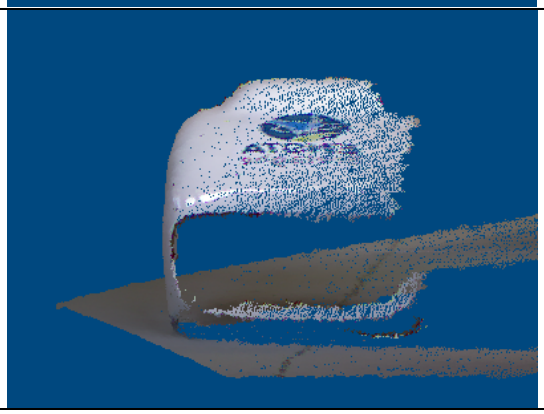
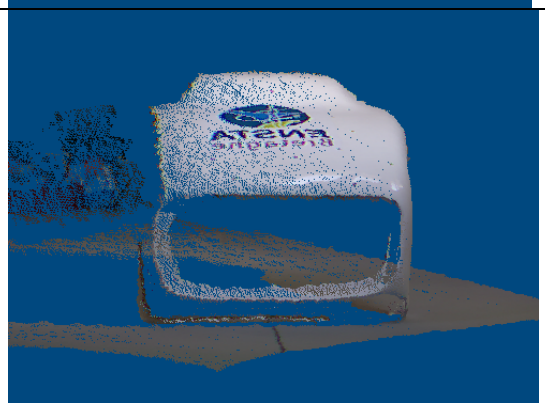
Une fois la trajectoire reconstituée avec la meilleure précision qu'on pouvait, GESMI peut indiquer à l'utilisateur les éventuelles détections d'objets sur l'image sonar qu'il aurait oubliées. En effet, connaissant la trajectoire du sous-marin et au moins une détection sur l'image sonar d'un objet, le logiciel peut indiquer les autres moments où le sous-marin a pu repasser devant l'objet et le voir sur le sonar. Ceci est affiché sous forme de reconstitution de l'image sonar qui affiche les enveloppes de la distance objet-sous-marin, superposées aux données sonar réelles.

Reconstruction 3D par méthodes ensemblistes avec une Kinect associée à une centrale inertielle

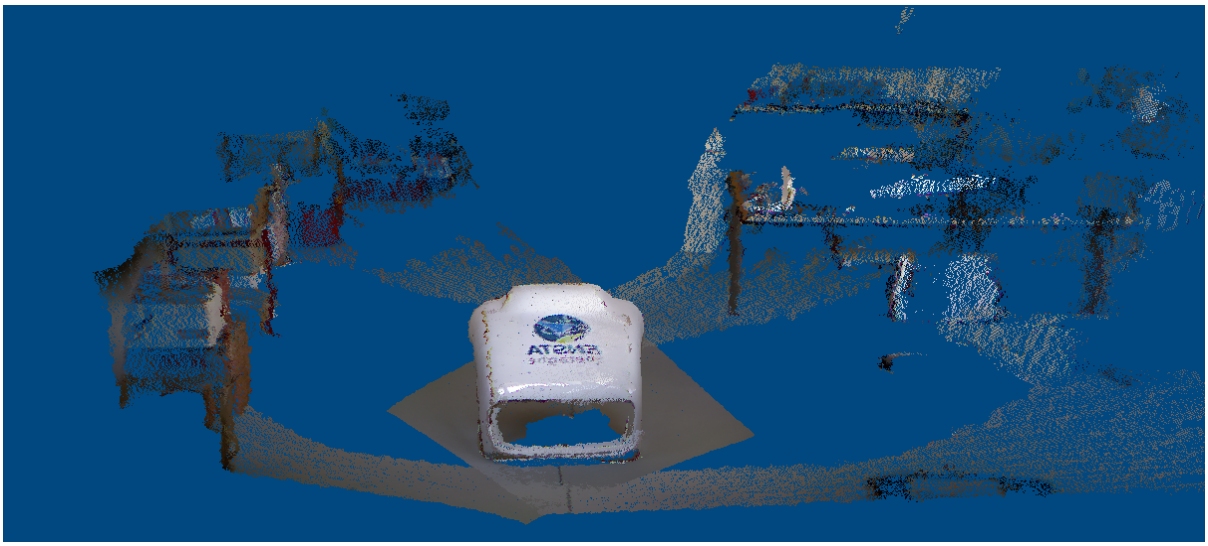
La disponibilité de caméras RGB-D à bas coût comme la Microsoft Kinect a récemment permis d'augmenter l'intérêt pour les application de VSLAM (Visual Simultaneous Localization And Mapping) utilisées par exemple pour la modélisation ou reconstruction 3D d'environnements intérieurs. Pour estimer la transformation entre deux positions, la plupart des techniques existantes utilisent l'algorithme ICP (Interest Closest Point) combiné à RANSAC sur le nuage de points des deux vues. Une nouvelle approche basée sur le calcul par intervalles pour calculer de manière robuste les paramètres de transformation a été développée.

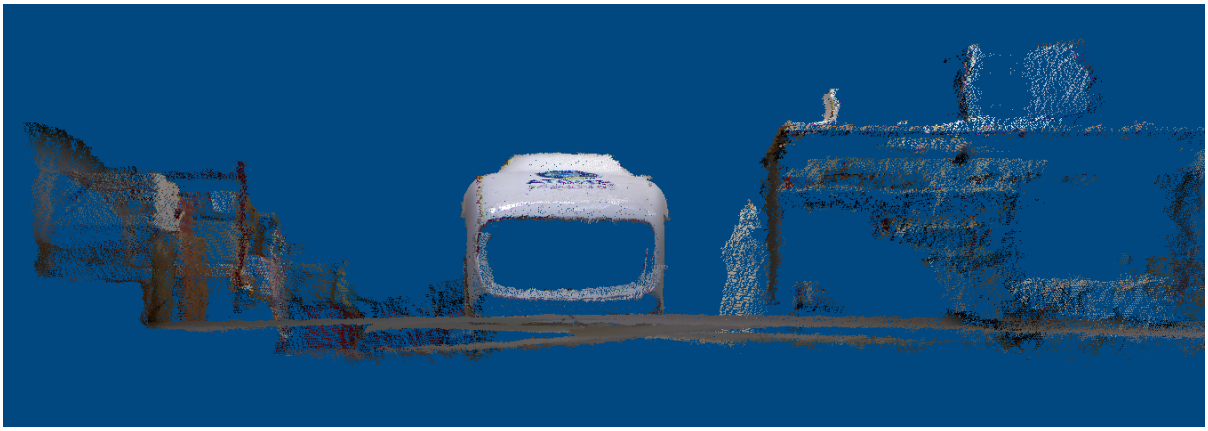
Pour cela nous avons tout d'abord pris deux photos 3D d'une structure mécanique avec la Kinect :

POSE N°	1	2
Vue 2D		

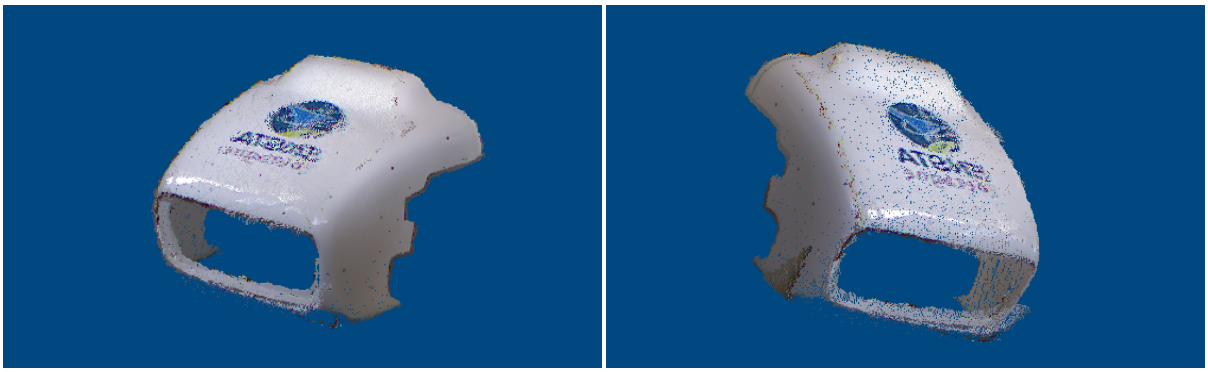
Vue 3D N°1		
Vue 3D N°2		

Nous avons utilisé SIFT, un algorithme de mise en correspondance de points d'intérêt dans les deux images 2D. Nous avons ensuite récupéré l'information de profondeur des points en question afin d'obtenir une liste de points 3D en correspondance dans les deux nuages de points. Enfin, nous avons écrit le système d'équation à résoudre afin d'obtenir les paramètres de transformation (rotation + translation) d'un nuage par rapport à l'autre. Pour résoudre ce système par méthodes ensemblistes, nous avons écrit un algorithme « forward-backward » contractant successivement les intervalles autour de la solution. Nous avons obtenu une largeur d'intervalle de 0.1rad pour les angles de rotation et de 0.05m pour les translations avec en moyenne un temps de calcul de 0.2ms. Voici le résultat obtenu sur notre exemple :





En supprimant les points inutiles, nous obtenons l'objet complet en 3D :



Enfin nous avons ajouté une centrale inertielle à notre système afin de connaitre la rotation effectuée immédiatement, nous permettant ainsi de réduire le temps de calcul et d'améliorer la précision.

Figure 17 : Kinect montée avec une centrale inertielle

Suivi de route pour un robot voilier

Pour pouvoir effectuer des mesures océanographiques, l'IFREMER a réalisé un robot voilier de 3.65m basé sur une coque de Mini-J et un gréement de type balestron : VAIMOS (Voilier Autonome Instrumenté pour Mesures Océanographiques de Surface). Ce robot

possède une sonde permettant de mesurer divers paramètres de l'eau, une carte informatique embarquée ARMADEUS (utilisée dans des cours de robotique à l'ENSTA Bretagne), une station météo (qui donne la direction et la force du vent ainsi que la position GPS), une centrale inertielle, un dispositif de communication Wifi et les actionneurs nécessaires au contrôle de sa voile et du gouvernail. L'ENSTA Bretagne participe à son automatisation pour qu'il soit capable de réaliser de manière autonome un quadrillage de zone aussi précis que possible, tout en consommant le moins d'énergie possible. Pour cela, un algorithme de suivi de ligne a été mis au point. L'objectif est de garantir que le robot reste toujours dans un couloir prédéterminé d'une largeur de 100 m par exemple, malgré les manœuvres liées aux changements de cap, remontée au vent… L'idée est d'être capable à terme d'envisager la cohabitation d'une multitude de robots voiliers en reconstituant en mer tout ce qui a été développé pour la circulation routière (carrefours, feux... par exemple), même si les éléments de circulation n'auraient qu'une existence virtuelle.

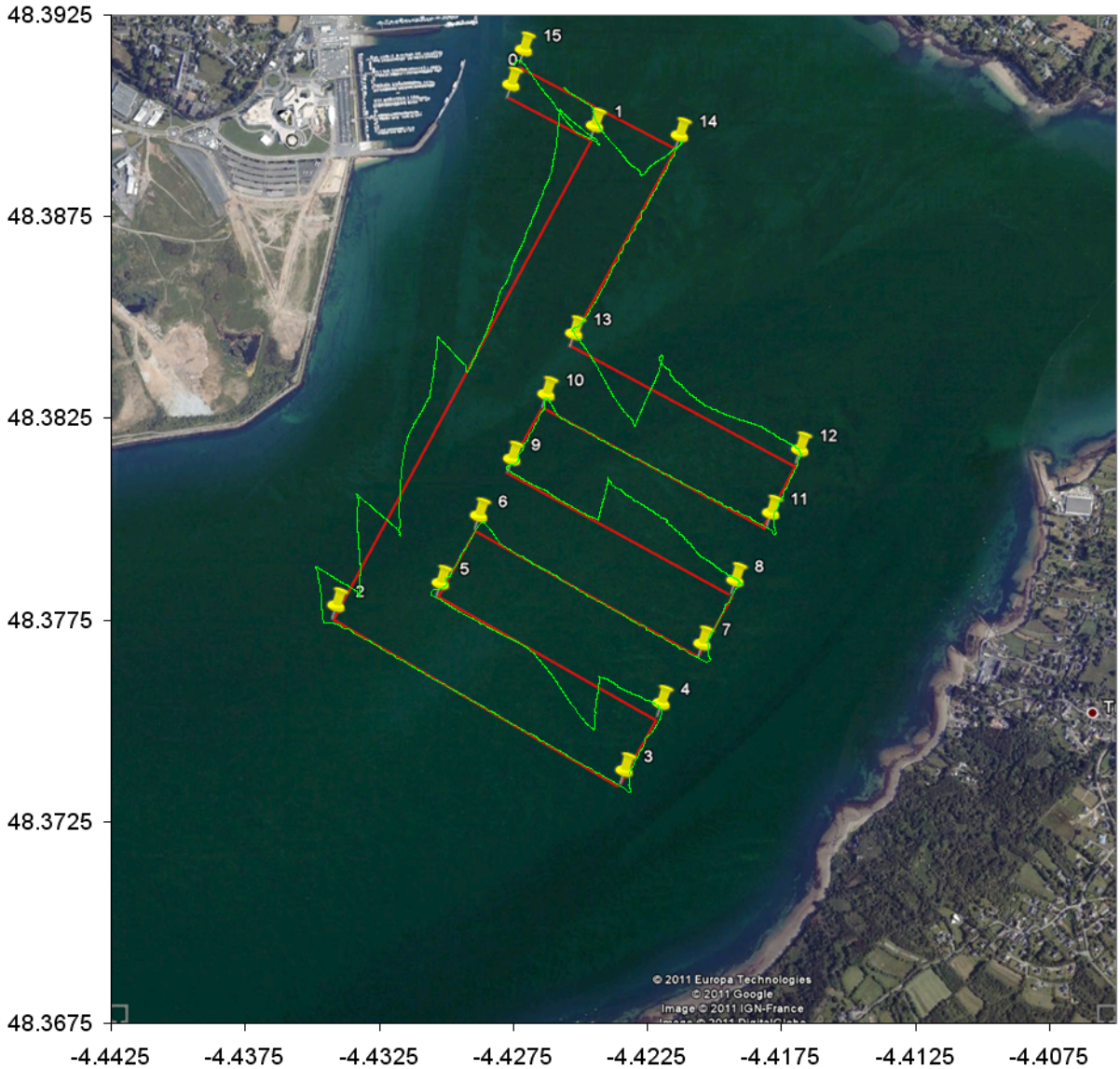


Figure 18 : Expérience faite dans la rade de Brest avec le voilier VAIMOS: le trajet prévu est indiqué par les lignes rouges (formées par les waypoints de 0 à 15), le trajet réellement effectué est en vert. On voit ici parfois des sortes de zigzags autour de la ligne voulue : ceux-ci sont dus aux manœuvres de remontée au vent du bateau (ce jour-là, le vent venait du sud-ouest en moyenne). On constate que le robot reste bien dans un couloir à 100 m près à tout moment

Bibliographie

[1] Ribas D., Ridao P., Tardós J.D., Neira J., *Underwater SLAM in man made structured environments*, Journal of Field Robotics, 2008.

[2] Moore R. E., *Methods and Applications of Interval Analysis*. SIAM, Philadelphia, PA, 1979.

[3] Jaulin L., Kieffer M., Didrit O., Walter E., *Applied Interval Analysis, with Examples in Parameter and State Estimation, Robust Control and Robotics*, Edition Springer-Verlag, London, United Kingdom, 2001.

[4] Thrun S., Bugard W., Fox D, *Probabilistic Robotics*, MIT Press, Cambridge, M.A., United Kingdom, 2005.

[5] Chabert G., Jaulin L., *"QUIMPER, A Language for Quick Interval Modelling and Programming in a Bounded-Error Context"*, Artificial Intelligence, 173:1079-1100, 2009.

[6] Sliwka J., Le Bars F., Reynet O., Jaulin L., *"Using interval methods in the context of robust localization of underwater robots"*, submitted to NAFIPS, El Paso, 2011.

[7] Jaulin L., *"A Nonlinear Set-membership Approach for the Localization and Map Building of an Underwater Robot using Interval Constraint Propagation"*, IEEE Transaction on Robotics, Vol 45, 25(1):88-98, 2009.

[8] Le Bars F., Bertholom A., Sliwka J., Jaulin L., *"Interval SLAM for underwater robots; a new experiment"*, NOLCOS, Bologna, 2010.

[9] Jaulin L., *"Robust set membership state estimation; Application to Underwater Robotics"*, Automatica, Vol 45, Issue 1, pp. 202-206, 2009.

[10] Sliwka J., Le Bars F., Jaulin L., *"Calcul ensembliste pour la localisation et la cartographie robustes"*, JD-JN-MACS, Angers, 2009.