Propagation de contraintes sur les intervalles pour la planification d'expérience en SLAM

Gilles Chabert, Luc Jaulin
ENSIETA,
2 rue François Verny 29806 Brest Cedex 9, FRANCE
gilles.chabert@ensieta.fr, luc.jaulin@ensieta.fr

Résumé

Les techniques d'analyse par intervalles ont été employées avec succès pour des problèmes d'estimation dans un contexte à erreurs bornées. En revanche, leur faculté à plus ou moins contracter les domaines semble difficile à estimer a priori, c'est-à-dire lorsque les mesures n'ont pas encore été effectuées. Cet article s'attaque à ce problème et propose une approche nouvelle capable d'estimer a priori la qualité que l'on peut espérer obtenir par une méthode classique : celle de propagation de contraintes sur les intervalles. Notre discussion s'appuie sur un problème de SLAM (*Simultaneous Localization and Mapping*) en robotique mobile sous-marine. Nous traitons tout d'abord le problème sur un plan théorique puis montrons ensuite comment adapter notre approche au cas du SLAM a priori.

Mots clefs : estimation à erreurs bornées, propagation de contraintes, analyse par intervalles, SLAM, estimation ensembliste, planification d'expérience.

1 Introduction

La propagation de contraintes sur les intervalles [3, 6, 10, 17, 4, 1, 8] s'est révélée être un outil efficace pour traiter des problèmes d'estimation dans un contexte à erreurs bornées (voir par exemple [12, 14, 9, 5]) et notamment le problème du SLAM (*simultaneous localization and mapping*) (cf. [13]).

Les méthodes par intervalles apportent deux avantages essentiels qui font défaut aux autres méthodes (e.g., le filtre de Kalman [15]) : d'une part, la possibilité de traiter un système non-linéaire sans recourir à des linéarisations (et donc d'en éviter les effets), et, d'autre part, la garantie de ne pas perdre de solutions, c.a.d., de produire un résultat prenant en compte toutes les sources d'incertitudes possibles (erreurs de mesure, arrondis, etc.). Le résultat est fourni sous forme d'intervalles dont la taille croît avec celle des incertitudes.

Lorsque l'expérience n'a pas encore eu lieu et donc que les mesures ne sont pas encore disponibles, il n'existe actuellement pas de méthode capable de nous dire quelle précision nous donnera la propagation. Le but de cet article est de répondre à ce problème, c.a.d., de pouvoir estimer la taille des intervalles obtenus par propagation en connaissant uniquement

l'incertitude liée aux mesures (et non leurs valeurs). Nous parlerons d'analyse d'erreur *a priori*. Nous illustrerons notre approche sur le problème de la planification d'une expérience de type SLAM [19] dans un contexte de robotique sous-marine.

1.1 Plan de l'article

Nous commençons par décrire le problème du SLAM (section 2) et illustrons à travers cet exemple en quoi l'analyse d'erreur a priori est un problème fondamental.

Nous étudions à la section suivante cette problématique sous un angle théorique et proposerons une approche générique.

Nous adapterons à la section 4 cet algorithme générique au problème du SLAM, mais cependant pour un *scénario* fixé. Nous montrerons enfin à la section 5 comment traiter l'erreur a priori dans le cas général.

2 SLAM a posteriori et SLAM a priori

Le problème du SLAM peut se définir ainsi [16, 7]. Un robot se déplace dans un environnement inconnu avec deux buts : se localiser (déterminer sa trajectoire) et établir une carte de son environnement (c.a.d, repérer certains types d'objets appelés *amers* et établir leur position). Les deux buts sont intriqués (ce que traduit le terme "*simultané*") car une carte aide à mieux se localiser et seule une bonne connaissance de sa position observé sur la carte, donc de construire cette dernière. La figure 1 illustre le principe du SLAM. (a) A l'instant initial, le robot sait où il se trouve. (b) Le robot avance et connaît sa position avec de plus en plus d'incertitude. Il rencontre un arbre dont il peut déduire approximativement sa position. (c) Le robot rencontre un deuxième arbre. Le robot connaît de moins en moins sa position : on dit qu'il dérive. (d) Le robot rencontre un troisième arbre mais ne peut déterminer s'il s'agit d'un nouvel arbre ou d'une réapparition du deuxième. (e) Le robot retrouve le premier arbre. Il peut recaler sa position. (f) En remontant le temps, il est alors capable d'améliorer la précision de sa trajectoire et d'affiner la position du deuxième et troisième arbre perçus. Il peut alors en conclure qu'il s'agissait bien d'arbres distincts.

Nous allons maintenant décrire l'approche classique pour traiter le problème du SLAM. Il nous faut tout d'abord représenter le robot par une équation d'état du type

$$\left\{ \begin{array}{rcl} \dot{\mathbf{x}} & = & \mathbf{f}(\mathbf{x}, \mathbf{u}) \\ \mathbf{y} & = & \mathbf{g}(\mathbf{x}) \end{array} \right.$$

On définit ensuite le vecteur $\mathbf{p}$ contenant les coordonnées de tous les amers (en supposant leur nombre constant) et par $\mathbf{z}$ le vecteur des sorties qui dépendent de ces amers (c.a.d., les pixels où apparaissent les amers sur les images sonar et à partir desquels la distance qui les séparent du robot est estimée). Si on suppose que les amers sont immobiles, on a bien sûr $\dot{\mathbf{p}} = \mathbf{0}$. On définit alors l'état étendu par $\mathbf{x}_e = (\mathbf{x}, \mathbf{p})$ et la sortie étendue par

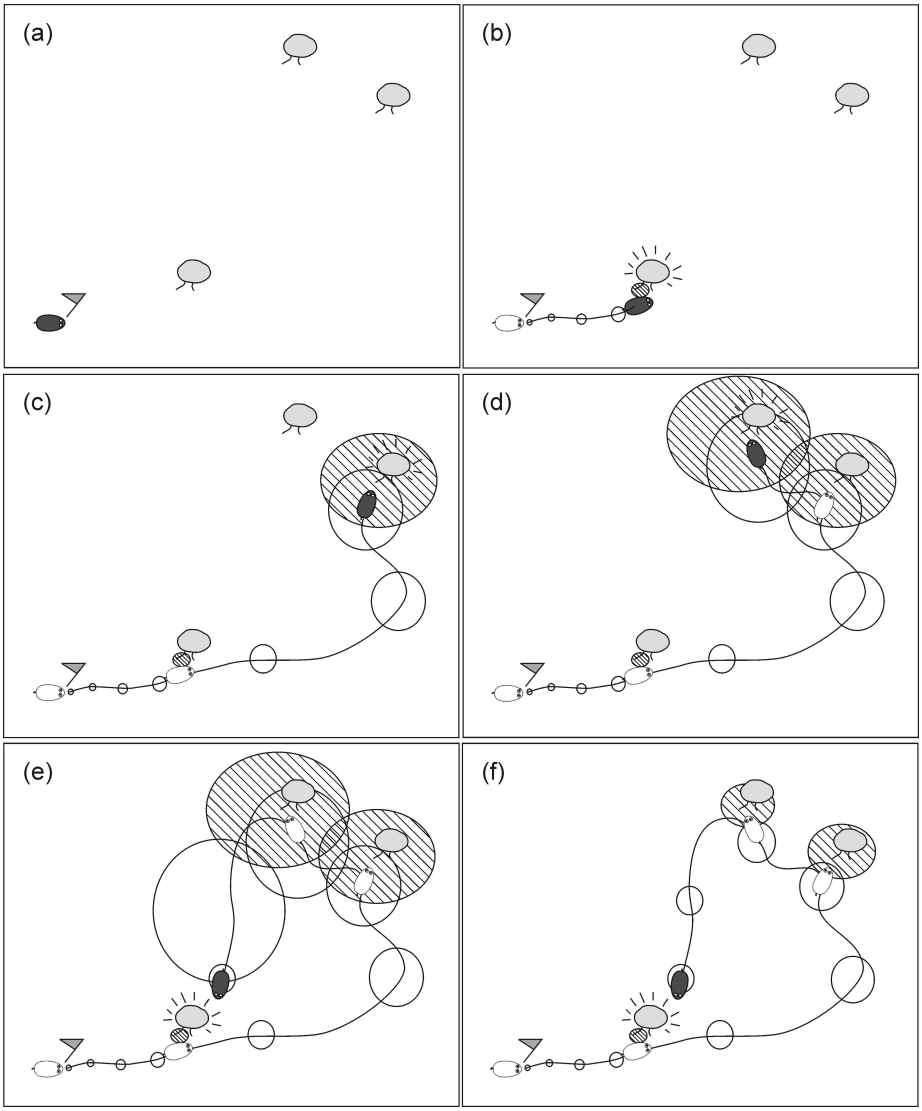

FIG. 1 – Illustration du principe de la localisation et de la cartographie simultanée (SLAM).

$\mathbf{y}_e = (\mathbf{y}, \mathbf{z})$. Le système amers-robot peut alors être décrit par une équation d'état du type

$$\begin{cases} \dot{\mathbf{x}}_e &= \mathbf{f}_e(\mathbf{x}_e, \mathbf{u}) \\ \mathbf{y}_e &= \mathbf{g}(\mathbf{x}_e). \end{cases}$$

Un observateur d'état bayésien (de type Kalman, particulaire, ensembliste, ...) devrait alors nous permettre de reconstituer l'état étendu $\mathbf{x}_e$ et donc à la fois localiser le robot et les amers. Malheureusement, en pratique, le nombre d'amer est grand, le robot est décrit par des équations d'état non-linéaires. Des approches plus sophistiquées doivent alors être considérées.

Le problème que nous venons de présenter peut être qualifié de *SLAM a posteriori* dans le sens où il ne peut se faire qu'après avoir récolté quelques mesures. Dans cet article, nous nous intéressons au *SLAM a priori* ce qui revient à considérer une planification d'expérience pour un problème de SLAM. Le problème qui sera traité sera influencé par celui traité dans l'article [11] qui illustre une méthode de propagation de contraintes a posteriori dans un contexte de robotique sous-marine. Le lecteur pourra s'en référer pour une description plus détaillée des équations et de la façon dont la propagation d'intervalles y est appliquée. Nous nous contentons ici d'une brève description des capteurs disponibles pour notre robot sous-marin.

Hors de l'eau, le robot se localise précisément à l'aide du système GPS. Sous l'eau, le robot possède pour se localiser :

- une centrale inertielle donnant ses 3 angles d'Euler ϕ, θ et ψ dans un repère fixe par rapport à la terre,
- un Loch-Doppler donnant son vecteur vitesse v_r relativement à un observateur fixe par rapport à la terre et exprimé dans le repère du robot,
- un baromètre donnant sa profondeur a du robot,
- et un sonar latéral à partir duquel il est possible d'estimer la distance r qui sépare le robot d'un éventuel amer.

Toutes les 0.1 seconde, chaque capteur produit une mesure. Toutes les données sont enregistrées puis dépouillées à l'issue de la mission.

Comme nous le verrons au §4.1, le problème est modélisé par un système d'équations $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ où certaines composantes du vecteur $\mathbf{x}$ représentent les « entrées » (les variables dont les domaines d'appartenance sont fixés, comme les mesures), et d'autres les « sorties »(les variables dont on cherche à déterminer le domaine, comme les positions du robot à chaque pas de temps et celles des amers observés). Remarquons que la position initiale et finale du robot est déterminée grâce au GPS, leur incertitude est donc liée uniquement à cet équipement.

Le système $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ est construit en discrétisant une équation différentielle, ce qui amène à considérer un nombre de variables très grand (plusieurs centaines de milliers). Dans ces conditions, la méthode la plus adaptée pour calculer de façon garantie les sorties est la propagation de contraintes où les domaines sont approximés par des intervalles (parfois raccourci en « propagation d'intervalles »[1]). Les méthodes de bissection sont donc

[1]Dans la littérature française, on trouve le terme « 2B-cohérence ». Dans la littérature anglaise : « hull

à proscrire si on souhaite un temps de calcul raisonnable.

2.1 SLAM a priori

La propagation d'intervalles a montré son efficacité pour calculer à partir de données une enveloppe de la trajectoire et de l'emplacement des amers. Cependant, la précision obtenue n'est connue qu'a posteriori, une fois le programme exécuté pour une mission donnée.

Bien entendu, l'influence de certains paramètres sur la précision du résultat est évidente : le robot se perd d'autant plus facilement que la zone couverte est large, qu'il y a peu d'amers, ou que les erreurs en entrée sont grandes. En revanche, si tous ces paramètres sont fixés, la question demeure : en quoi la précision obtenue pour quelques expériences particulières nous renseigne-t-elle sur la faculté du robot à remplir sa mission *en général* ? On ne peut savoir si le SLAM est réalisable de façon satisfaisante avant utilisation. Nous avons donc à traiter ici un problème de planification d'expériences [20] pour lesquelles, il semblerait que les méthodes par intervalles n'aient jamais été utilisées.

Dans cet article, nous cherchons donc à établir une méthode capable de calculer *a priori* une borne supérieure sur l'erreur maximale qu'une méthode ensembliste de propagation d'intervalles pourrait nous donner *a posteriori*.

Remarque 1 (Erreur réelle et erreur calculée) *L'incertitude sur la position/localisation est d'un point de vue théorique une grandeur dépendant uniquement de l'incertitude sur les capteurs. C'est cette grandeur que nous désignons par « erreur réelle » : elle peut être vue comme l'erreur minimale productible en appliquant n'importe quelle méthode de résolution garantie (avec une arithmétique à précision infinie, etc.).*

Nous ne nous intéressons pas à l'erreur réelle, mais seulement à l'erreur obtenue par la méthode ensembliste de propagation d'intervalles. Ceci se justifie dès lors que les expériences concrètes seront traitées par cette méthode. Il n'y a pas d'intérêt, en effet, à considérer l'erreur réelle si celle-ci ne peut pas être calculée en pratique sur les expériences. Quoi qu'il en soit, la propagation d'intervalles étant garantie, l'erreur réelle est forcément plus petite que l'erreur calculée.

2.2 Notation

Terminons cette introduction par quelques notations. Le milieu et le rayon d'un intervalle $[x]$ seront notés respectivement $\text{mid}[x]$ et $\text{rad}[x]$. Le milieu ou le rayon d'un vecteur d'intervalles (ou *pavé*) est défini comme le vecteur des milieux/rayons de chaque composante.

Pour appliquer la propagation d'intervalles, les domaines des variables sont représentés par des intervalles. Remarquons qu'un intervalle $[x]$ peut être rigoureusement vu comme la donnée d'un réel $\text{mid}[x]$ et d'une incertitude (ou erreur) $\text{rad}[x]$. L'erreur que l'on cherche à estimer est donc simplement le rayon du pavé contracté par propagation.

consistency ».

Pour toute expression arithmétique $\mathbf{f}(\mathbf{x})$, nous écrirons $\mathbf{f}([\mathbf{x}])$ cette même expression interprétée avec l'arithmétique d'intervalles (c.a.d, avec des arguments intervalles).

3 Formalisme

Dans cette section, nous définissons formellement le problème. Étant donné un système d'équations $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ où $f : \mathbb{R}^n \to \mathbb{R}^m$, rappelons que notre but est de déterminer le rayon maximal que l'on peut obtenir en contractant un pavé initial $[\mathbf{x}]$ inconnu mais dont le rayon de certaines composantes est borné.

Le pavé $[\mathbf{x}]$ est donc soumis à des contraintes. Autrement dit, il appartient à ensemble de pavés possibles. Nous commençons au §3.1 par introduire la structure utilisée pour représenter un ensemble de pavés. Nous montrerons ensuite au §3.2 comment passer du système d'équations initial à un CSP (*Constraint Satisfaction Problem*) dont les variables prennent leurs valeurs dans ces ensemble de pavés. Enfin, un algorithme est décrit au §3.3 pour pouvoir résoudre ce CSP.

3.1 Ensemble de pavés, ressorts

Nous introduisons dans cette section une structure permettant de représenter des ensembles de pavés. Tout d'abord, il est préférable pour des raisons évidentes d'efficacité que cette structure, notons la $\mathcal{S}$, vérifie les propriétés suivantes :
- *Finesse* : Pour tout pavé $[\mathbf{x}]$, le singleton $\{[\mathbf{x}]\}$ appartient à $\mathcal{S}$.
- *Bissectibilité* : A tout élément $\langle\mathbf{x}\rangle$ de $\mathcal{S}$ qui n'est pas un singleton, il existe deux éléments $\langle\mathbf{a}\rangle$ et $\langle\mathbf{b}\rangle$ de $\mathcal{S}$ qui ne se recouvrent pas en dehors de leurs frontières et tels que $\langle\mathbf{a}\rangle \cup \langle\mathbf{b}\rangle = \langle\mathbf{x}\rangle$.

Deux structures « canoniques » respectent ces propriétés :

Les pavés de pavés. Un *intervalle d'intervalles*, noté $[[a^-, a^+], [b^-, b^+]]$, est défini comme suit

$$\left[\left[a^-, a^+\right], \left[b^-, b^+\right]\right] = \left\{[a, b] \in \mathbb{IR},\, a \in \left[a^-, a^+\right], b \in \left[b^-, b^+\right]\right\}.$$

Les ressorts [2]**.** Un *ressort* de $\mathbb{IR}$, $\langle x\rangle = \langle[m], [r]\rangle$, avec $[m] \in \mathbb{IR}$ et $[r] = [m], [r]$ est un ensemble d'intervalles défini comme suit

$$\langle[m], [r]\rangle = \left\{[x^-, x^+] \in \mathbb{IR},\ \frac{x^- + x^+}{2} \in [m] \text{ et } \frac{x^+ - x^-}{2} \in [r]\right\}.$$

Un ressort de $\mathbb{R}^n$ est un ensemble $\langle[\mathbf{m}],[\mathbf{r}]\rangle$ de pavés $[\mathbf{x}]$ de $\mathbb{IR}^n$ défini comme suit :

$$\langle[\mathbf{m}],[\mathbf{r}]\rangle = \{[\mathbf{x}] \in \mathbb{IR}^n, \forall i \in \{1, \ldots, n\}, [x_i] \in \langle[m_i], [r_i]\rangle\}.$$

L'ensemble des ressorts sera noté $\langle\mathbb{IR}\rangle^n$. La figure 2 illustre la définition d'un ressort de $\mathbb{IR}$. Sur cette figure, le réel $x = 4$ appartient à l'intervalle $[x] = [3, 6]$. L'intervalle $[x]$ appartient au ressort $\langle x\rangle = \langle[3, 6], [1, 2]\rangle$. Si on définit

$$\sqcap \langle x\rangle = \cap \{[x] \in \langle x\rangle\} \text{ et } \sqcup \langle x\rangle = \cup \{[x] \in \langle x\rangle\},$$

le résultat obtenu est nécessairement un intervalle (qui peut être vide). Ces notions sont illustrées par les sous-figures (c) et (d).

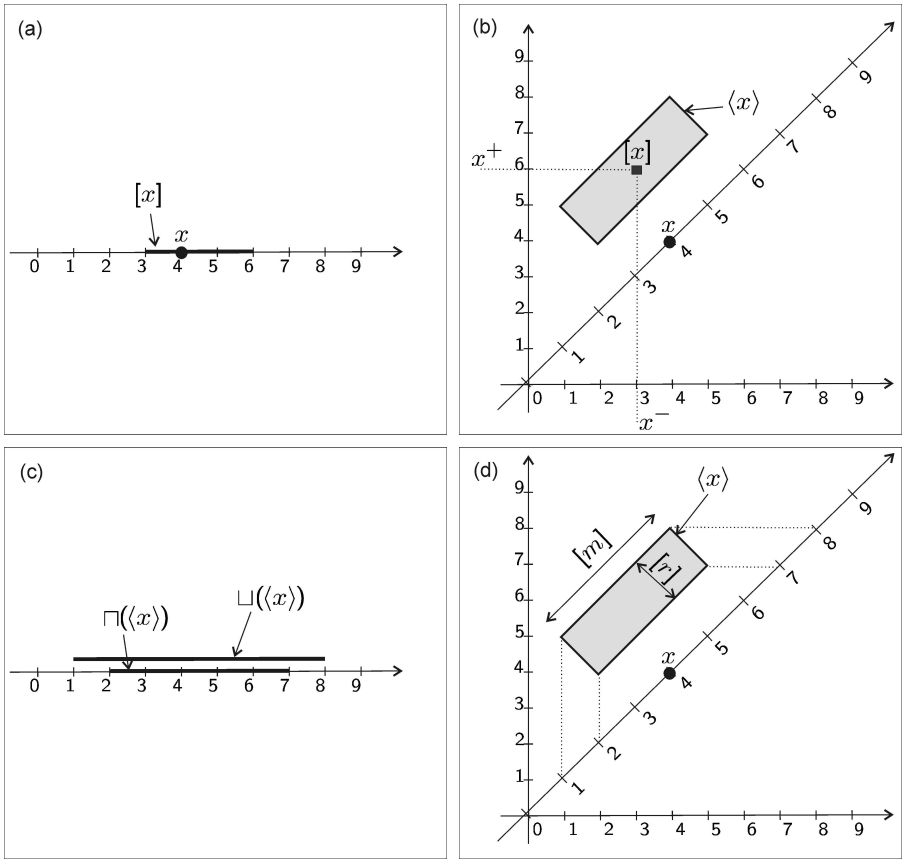

FIG. 2 – Un ressort est un ensemble d'intervalles.

Dans ce qui suit, nous choisissons de représenter les domaines des pavés par des ressorts. Deux raisons expliquent ce choix :

1. Dans le cadre de l'estimation a priori, les intervalles de mesures sont plus naturellement représentés par des ressorts que par des pavés de pavés. Par exemple, dans le cadre d'un robot équipé d'une boussole, nous savons à l'avance que la mesure qui sera faite sera un réel dans l'intervalle $[0, 2\pi]$ et que l'intervalle erreur aura un rayon connu a priori, suivant la précision de la boussole. Il est donc facile de représenter l'ensemble de tous les intervalles de mesure a priori possibles par un ressort.
2. L'arithmétique que nous définirons au §3.1 sur les ressorts est beaucoup plus naturelle et simple à établir pour les fonctions élémentaires (log, sin,exp, etc.) que celle des pavés de pavés.

3.2 CSP intervalle

Considérons tout d'abord à titre d'exemple le système d'équations suivant :

$$\begin{cases} x_1 - x_2 + x_3 &= 0 \\ x_1 x_2 - x_3 &= 0 \end{cases} \tag{1}$$

où $\mathbf{x} = (x_1, x_2, x_3)$ est supposé appartenir au pavé initial $[\mathbf{x}](0)$. Une méthode de propagation aboutit au plus grand pavé $[\mathbf{x}] = [x_1] \times [x_2] \times [x_3]$ inclus dans $[\mathbf{x}](0)$ et satisfaisant les contraintes suivantes :

$$\begin{cases} [x_1] &\subseteq [x_2] - [x_3] \\ [x_2] &\subseteq [x_1] + [x_3] \\ [x_3] &\subseteq [x_2] - [x_1] \\ [x_1] &\subseteq [x_3]/[x_2] \\ [x_2] &\subseteq [x_3]/[x_1] \\ [x_3] &\subseteq [x_1] * [x_2] \end{cases} \tag{2}$$

La généralisation est immédiate. Nous pouvons associer à tout système d'équations $\mathbf{f}([\mathbf{x}]) = \mathbf{0}$ une relation d'inclusion

$$[\mathbf{x}] \subseteq [\phi]([\mathbf{x}]), \tag{3}$$

telle que le point fixe de la propagation appliqué au système $\mathbf{f}([\mathbf{x}]) = \mathbf{0}$ sur un pavé initial $[\mathbf{x}](0)$ soit le pavé maximal (c.a.d., le plus grand au sens de l'inclusion) inclus dans $[\mathbf{x}](0)$ et vérifiant (3)[2].

Puisque l'on cherche à considérer l'ensemble des pavés $[\mathbf{x}]$ pouvant résulter de cette propagation et appartenant initialement à un ressort $< [\mathbf{m}], [\mathbf{r}] >$, alors ces pavés $[\mathbf{x}]$ sont précisément les solutions maximale du système (3), vu comme CSP de variable *intervalle* $[\mathbf{x}]$ dont le domaine est un *ressort* $< [\mathbf{m}], [\mathbf{r}] >$.

Soit x_i une variable du système. Parmi toutes les solutions maximales de ce CSP intervalle (ou ICSP), le pavé $[\mathbf{x}]$ dont le rayon $\mathrm{rad}[x_i]$ est le plus grand représente le cas le pire en terme de propagation d'erreur pour cette variable, et ce rayon est donc précisément l'une des bornes que l'on souhaite calculer.

Ainsi, en remarquant que le maximum de $\mathrm{rad}[x_i]$ est forcément atteint pour un pavé maximal, nous sommes ramenés au problème d'optimisation suivant :

Problème : Maximiser $\mathrm{rad}[x_i]$, pour $[\mathbf{x}] \in < [\mathbf{m}], [\mathbf{r}] >$ satisfaisant (3).

3.3 Arithmétique des ressorts

Nous montrons maintenant comment le problème précédent peut être résolu grâce à un contracteur sur les ressorts, de la même manière qu'une optimisation sous contraintes à variables réelles est résolu par une contraction sur les intervalles.

Chaque contrainte du ICSP (3) s'exprime comme des opérations entre intervalles. Or, une opération arithmétique $[x] \diamond [y]$ entre deux intervalles produit un intervalle $[z]$ dont le milieu

[2] Un tel pavé existe car l'ensemble des solutions de (3) forme un treillis.

mid([z]) et le rayon rad([z]) peuvent être calculés formellement en fonction des rayons et des milieux des intervalles arguments [x] et [y]. Par exemple,

$$[z] = [x] + [y] \Leftrightarrow \begin{cases} \text{rad}[z] = \text{rad}[x] + \text{rad}[y], \\ \text{mid}[z] = \text{mid}[x] + \text{mid}[y]. \end{cases} \tag{4}$$

De même, dans le cas particulier où $[x] \subset [0, \infty[$ et $[y] \subset [0, \infty[$, alors

$$[z] = [x] * [y] \Leftrightarrow \begin{cases} \text{rad}[z] = \text{rad}[x] * \text{mid}[y] + \text{rad}[y] * \text{mid}[x], \\ \text{mid}[z] = \text{mid}[x] * \text{mid}[y] + \text{rad}[y] * \text{rad}[x]. \end{cases} \tag{5}$$

Il en va de même pour des situations où $[x] \not\subset [0, \infty[$ ou $[y] \not\subset [0, \infty[$, pour des opérations impliquant une fonction élémentaires (exp, cos, etc.), une intersection ou une réunion. Ainsi, pour toute expression arithmético-ensembliste $[f]([\mathbf{x}])$ impliquant des intervalles, il est possible d'exprimer de façon analytique le rayon et le milieu du résultat $[z]$ englobant $[f]([\mathbf{x}])$ en fonction du milieu $\mathbf{m}$ et du rayon $\mathbf{r}$ du pavé $[\mathbf{x}]$. Il suffit donc, a priori, de remplacer dans ce calcul $\mathbf{m}$ et $\mathbf{r}$ par les pavés $[\mathbf{m}]$ et $[\mathbf{r}]$ pour obtenir une approximation extérieure de l'ensemble des intervalles-résultats possibles. Autrement dit, en étendant aux intervalles les formules telles que (4) et (5), il est possible d'établir une arithmétique des ressorts.

Malheureusement, les occurrences multiples de mid[**x**] et rad[**x**] dans les formules rendent l'évaluation pessimiste, y compris pour une simple multiplication. C'est pourquoi une arithmétique de ressort plus précise a été proposée dans [2]. Remarquons que, contrairement à la multiplication d'intervalles, la multiplication de ressorts introduit nécessairement une surestimation (voir figure 3).

Finalement, le problème d'optimisation peut être résolu en combinant bissection[3], élagage (grâce à la contrainte $\text{rad}[x_i] \geq$ borne courante) et contraction du ICSP. Le contracteur associé à la relation intervalle $[\mathbf{x}] \subset [\mathbf{f}]([\mathbf{x}])$ s'écrit comme suit :

$$\langle \mathbf{x} \rangle := \langle \mathbf{x} \rangle \langle \cap \rangle \langle \mathbf{f} \rangle (\langle \mathbf{x} \rangle), \tag{6}$$

où $\langle \mathbf{f} \rangle$ désigne *l'extension aux ressorts* de la fonction d'intervalles $[\mathbf{x}] \mapsto [\mathbf{f}]([\mathbf{x}])$, et $\langle \cap \rangle$ *l'extension aux ressorts* de l'intersection d'intervalles.

4 Application

Nous traitons dans cette section l'analyse d'erreur a priori pour le problème du SLAM. Toutefois, le problème ne sera pas pris dans sa généralité, et nous commençons par justifier cette restriction. Nous verrons à la section 5 comment nous en affranchir.

Le fait qu'un amer soit détecté à un instant se modélise par une contrainte, ou équation (voir paragraphe suivant). Si, par exemple, la distance entre cet amer et le robot change, la valeur d'un des paramètres de l'équation change. En revanche, si l'instant de détection

[3] Un ressort est bissecté en bissectant soit l'intervalle du milieu, soit l'intervalle du rayon.

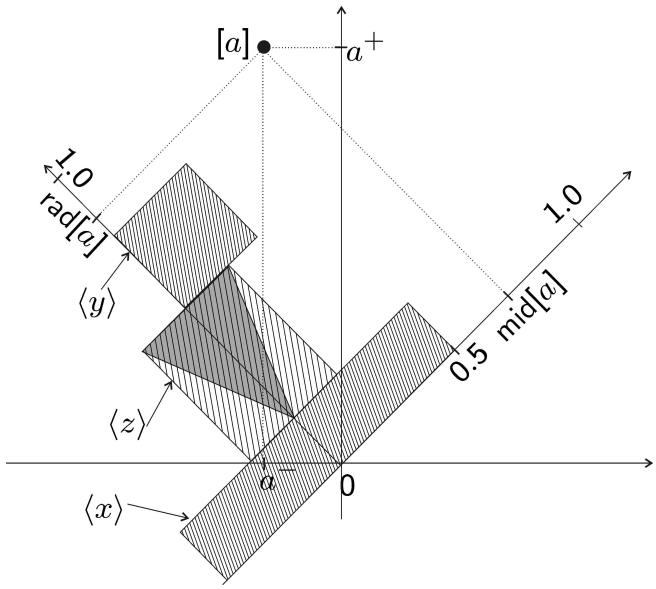

FIG. 3 – Arithmétique des ressorts. Le plan représente ici l'ensemble des intervalles Un ressort est un rectangle dans le repère milieu/rayon. La multiplication des ressorts $\langle X \rangle :=< [-0.5, 0.5], [0, 0.2] >$ et $\langle Y \rangle :=< [0, 0.3], [0.7, 1.0] >$ n'est pas un ressort mais la zone en forme de triangle. Le plus petit ressort incluant cette zone est $\langle Z \rangle :=< [-0.21, 0.21], [0, 0.7] >$.

ou le numéro de l'amer est modifié, il ne s'agit plus de la même contrainte (dans le sens où elle n'implique plus les mêmes variables).

Cela signifie que pour manipuler un ensemble de contraintes fixées, nous devons établir à l'avance à quels instants les amers sont perçus. Nous désignerons par *scénario* la donnée de ces couples ⟨instant,amer⟩. Il sera supposé tout au long de cette section un scénario fixé.

4.1 Modélisation

Le déplacement du véhicule (voir figure 4) est régi par l'équation d'état

$$\dot{\mathbf{p}}(t) = \mathbf{R}(\phi(t), \theta(t), \psi(t)).\mathbf{v}_r(t), \tag{7}$$

où

$$\mathbf{R}(\phi, \theta, \psi) = \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi & \cos\varphi \end{pmatrix}$$

et $\mathbf{v}_r$ représente la vitesse du robot mesurée par le Loch-Doppler. Après discrétisation nous sommes capables d'exprimer la position $\mathbf{p}(k)$ à un instant k en fonction de la position à l'instant d'échantillonage précédent et des mesures prélevées par les capteurs à ce même instant, en l'occurrence ϕ, θ, ψ et $\mathbf{v}_r$. Nous pouvons considérer l'ensemble de ces données

comme un seul vecteur de variables $\mathbf{u} = (\phi, \theta, \psi, v_r^x, v_r^y, v_r^z)$. L'équation d'état discrétisée se met donc sous la forme

$$\mathbf{p}(k) = \mathbf{p}(k-1) + \mathbf{g}(\mathbf{u}(k-1)) \tag{8}$$

où $\mathbf{g}$ est une fonction de $\mathbb{R}^6$ dans $\mathbb{R}^3$.

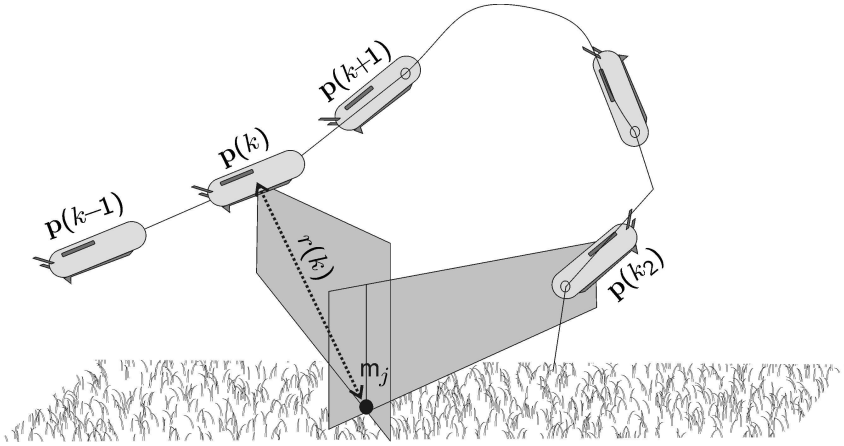

FIG. 4 – Principe du SLAM en milieu sous-marin.

En cas de détection d'un amer $\mathbf{m}_j$, le vecteur $\mathbf{p}(k) - \mathbf{m}_j$ est entièrement déterminable à partir de la distance r lue sur le sonar, la profondeur a du véhicule et les angles ϕ et θ. Ces deux angles influent en réalité très peu sur le résultat[4] et peuvent être décorrélés de leur implication dans le vecteur $\mathbf{u}$. Ainsi, nous avons de nouveau un vecteur de variables $\mathbf{w} = (r, a, \phi, \theta, \psi)$, indépendant de $\mathbf{u}$, et la contrainte suivante en cas de détection :

$$\mathbf{p}(k) = \mathbf{m}_j + \mathbf{h}(\mathbf{w}(k)), \tag{9}$$

où $\mathbf{g}$ est une fonction de $\mathbb{R}^4$ dans $\mathbb{R}^3$. Dans notre problème, les sorties sont les positions du robot à chaque pas de temps ainsi que la position des amers observés. Les entrées sont les vecteurs $\mathbf{u}$ et $\mathbf{w}$.

4.2 ICSP

Dans un contexte d'analyse d'erreur *a priori*, les données capteurs ne sont pas connues. Nous ne possédons, a priori, que deux types d'information : les domaines de variation relativement larges et une borne sur les erreurs de mesure. Pour notre robot, si on omet la possibilité d'un "dévissage" (roulis ou tangage de 90°), les domaines des angles d'Euleur sont : $\phi \in [-\pi/2, \pi/2]$, $\theta \in [-\pi/2, \pi/2]$ et $\psi \in [-\pi, \pi]$. La puissance des propulseurs permet également de borner les composantes du vecteur vitesse. La distance r est limitée

[4] Ces angles correspondent au tangage et au roulis, qui sont toujours faibles pour ce type de sous-marin. Quoi qu'il en soit, cette décorrélation ne peut pas introduire d'erreur ; elle rend au pire le résultat moins précis.

par la portée du sonar et la profondeur maximale a est une donnée topographique supposée connue. Pour toutes ces entrées, les constructeurs fournissent des bornes garanties sur les erreurs possibles. Les vecteurs $\mathbf{u}$ et $\mathbf{w}$ appartiennent donc à des pavés $[\mathbf{u}]$ et $[\mathbf{w}]$, appartenant eux-même à deux ressorts fixés $\langle \mathbf{u} \rangle$ et $\langle \mathbf{w} \rangle$.

La propagation de contraintes aboutit à des pavés qui satisfont les contraintes d'inclusion suivantes :

$$\begin{array}{lcll} [\mathbf{p}](k) & \subseteq & [\mathbf{p}](k-1) + [\mathbf{g}]([\mathbf{u}](k-1)) & , \ k \in \{1, \ldots, k_{\max}\} \\ [\mathbf{p}](k-1) & \subseteq & [\mathbf{p}](k) - [\mathbf{g}]([\mathbf{u}](k-1)) & , \ k \in \{1, \ldots, k_{\max}\} \\ [\mathbf{p}](k_j) & \subseteq & [\mathbf{m}_j] + [\mathbf{h}]([\mathbf{w}](k_j)) & , \ k_j \in K_j \\ [\mathbf{m}_j] & \subseteq & [\mathbf{p}](k_j) - [\mathbf{h}]([\mathbf{w}](k_j)) & , \ k_j \in K_j \end{array} \tag{10}$$

où K_j est l'ensemble des instants k où l'amer $\mathbf{m}_j$ est détecté. Le contracteur sur les ressorts associé s'écrit donc

$$\begin{array}{lcll} \langle \mathbf{p} \rangle (k) & := & \langle \mathbf{p} \rangle (k) \langle \cap \rangle (\langle \mathbf{p} \rangle (k-1) + \langle \mathbf{g} \rangle (\langle \mathbf{u} \rangle (k-1))) & , \ k \in \{1, \ldots, k_{\max}\} \\ \langle \mathbf{p} \rangle (k-1) & := & \langle \mathbf{p} \rangle (k-1) \langle \cap \rangle (\langle \mathbf{p} \rangle (k) - \langle \mathbf{g} \rangle (\langle \mathbf{u} \rangle (k-1))) & , \ k \in \{1, \ldots, k_{\max}\} \\ \langle \mathbf{p} \rangle (k_j) & := & \langle \mathbf{p} \rangle (k_j) \langle \cap \rangle (\langle \mathbf{m}_j \rangle + \langle \mathbf{h} \rangle (\langle \mathbf{w} \rangle (k_j))) & , \ k_j \in K_j \\ \langle \mathbf{m}_j \rangle & := & \langle \mathbf{m}_j \rangle \langle \cap \rangle (\langle \mathbf{p} \rangle (k_j) - \langle \mathbf{h} \rangle (\langle \mathbf{w} \rangle (k_j))) & , \ k_j \in K_j \end{array} \tag{11}$$

Les domaines initiaux $\langle \mathbf{p} \rangle (k)$ et $\langle \mathbf{m}_j \rangle$ étant $\langle]-\infty, +\infty[, [0, +\infty[\rangle$. Nous pouvons apporter un certain nombre de simplifications à ces contracteurs. Par exemple, en supposant que les vecteurs $\mathbf{w}(k_1), \ldots, \mathbf{w}(k_{n_j})$ sont indépendants[5], les pavés $[\mathbf{u}](k)$ et $[\mathbf{w}](k)$ décrivent toujours les mêmes ressorts quel que soit k. On peut donc précalculer à l'avance les ressorts $\langle \mathbf{g} \rangle (\langle \mathbf{u} \rangle)$ et $\langle \mathbf{h} \rangle (\langle \mathbf{w} \rangle)$.

5 Statistique de l'erreur maximale

Il pourrait être envisageable de chercher à calculer une borne sur les erreurs pour l'ensemble des scénarios possibles, en imaginant d'autres méthodes. Mais un tel objectif ne présente en réalité aucun intérêt, notamment parce que nous connaissons déjà la réponse. En effet, il existe toujours le scénario suivant : celui où le robot ne rencontre jamais plus d'une fois le même amer. Cela signifie que le robot ne peut jamais se recaler, autrement dit, qu'il est en « dérive pure ». Il suffit donc de considérer ce scénario pour obtenir immédiatement la borne maximale sur l'erreur.

Le calcul ensembliste n'est plus adapté à ce stade. En revanche, une analyse statistique de l'erreur maximale est pertinente. Connaître l'espérance et l'écart-type sur l'erreur maximale permet en effet de bien estimer a priori la faisabilité du SLAM.

5.1 Modèle probabiliste

Pour effectuer une analyse statistique sur l'ensemble des scénarios, il faut donc savoir comment modéliser la distribution de ces scénarios. Nous devons recourir pour cela aux

[5] Ce qui n'est pas tout à fait le cas puisque la distance à laquelle, à un instant donné, le robot se trouve d'un amer peut être calculée à partir de la distance à laquelle il s'y trouvait à un instant précédent et de la trajectoire parcourue entre-temps.

processus stochastiques.

Un scénario est la donnée d'apparition d'amers au cours du temps. Fixons tout d'abord un amer $\mathbf{m}_j$. Il s'agit de déterminer un processus stochastique qui permette de probabiliser le nombre de fois où cet amer est perçu dans un intervalle de temps.

En l'absence de propriétés sur la trajectoire et sur la répartition des amers dans la zone de mer, nous pouvons faire les hypothèses suivantes :

1. La probabilité d'avoir plusieurs détections dans un intervalle $[t, t+\delta_t]$ devient négligeable devant la probabilité d'en avoir qu'une seule lorsque δ_t tend vers 0.
2. Le processus est *sans mémoire* : si $[t_1, t_2]$ et $[t_3, t_4]$ sont deux intervalles de temps disjoints, le nombre de détection dans $[t_1, t_2]$ est indépendant du nombre de détection dans $[t_3, t_4]$.
3. Le processus est *homogène* : le nombre de détection dans un intervalle $[t, t+\delta_t]$ ne dépend que de sa longueur δ_t, et non sa position t dans l'axe des temps.
4. La probabilité d'avoir une seule détection sur un intervalle $[t, t+\delta_t]$ est en $\Omega(\delta_t)$ lorsque δ_t tend vers 0. Cela signifie que la probabilité de voir apparaître l'amer varie quasi-proportionnellement avec le temps d'attente. Cette hypothèse est typique des événements survenant de façon aléatoire et pour lesquels une fréquence d'apparition est connue.

Il est prouvé mathématiquement [18] que sous ces conditions, le processus $X_j(t)$ qui associe à t le nombre de fois où l'amer $\mathbf{m}_j$ est détecté dans l'intervalle de temps $[0, t]$ ne peut être qu'un processus de Poisson. Ce processus est donc caractérisé par une *fréquence* λ, correspondant au nombre moyen de fois où l'amer est perçu par unité de temps (par exemple, 1.7/heure). Il s'avère que la notion de fréquence est présente dans la culture des ingénieurs travaillant sur cette application et qu'ils savent la fixer empiriquement. Ceci valide le modèle choisi.

5.2 Génération de scénarios

Lorsque $X(t)$ suit un processus de Poisson de paramètre λ, la variable aléatoire Y donnant le temps d'attente à partir d'un instant quelconque avant l'apparition d'un amer suit une loi exponentielle de densité $\lambda e^{-\lambda t}$. Nous avons donc

$$Pr(Y \leq t) = \int_0^t \lambda e^{-\lambda \tau} d\tau = 1 - e^{-\lambda t}.$$

Les détections de l'amer $\mathbf{m}_j$ sont alors générées en échantillonnant des temps d'attente. Cette échantillonnage se fait par le biais de la fonction *quantile*, c.a.d., l'inverse de la fonction de répartition de Y :

$$Q(x) = -\frac{1}{\lambda} \log(1 - x).$$

En tirant de façon aléatoire un nombre x dans l'intervalle $[0, 1]$, $Q(x)$ retourne un temps d'attente. Si les tirages x sont uniformément répartis dans $[0, 1]$, les temps $Q(x)$ auront pour distribution la loi exponentielle.

Finalement, l'algorithme est le suivant : un compteur de temps t est initialisé à 0. Un temps d'attente δ_t est généré par le procédé décrit juste avant, puis ajouté au temps t courant (t est remplacé par $t + \delta_t$). A l'instant t, il est décidé que l'amer est détecté et la contrainte correspondante est ajoutée. L'algorithme boucle ainsi jusqu'à ce que t dépasse la durée de la mission.

Les processus de Poisson peuvent être *superposés* : pour chaque amer $\mathbf{m}_j$, la boucle précédente est exécutée.

Une fois les scénarios générés, le calcul de l'espérance et de l'écart-type se font de façon classique : deux variables S_1 et S_2 cumulent respectivement les erreurs maximale et le carré des erreurs maximales de chaque scénario, à un instant t (échantillonné). Une fois les N scénarios générés, l'espérance $E[err](t)$ et l'écart-type $\sigma(err)$ de l'erreur maximale sont estimés ainsi :

$$E[err] \sim S_1/N$$
$$\sigma(err) = \sqrt{E[err^2] - E[err]^2} \sim \sqrt{S_2/N - (S_1/N)^2}$$

6 Conclusion

Nous avons tenté de mener deux objectifs : d'une part, dégager une problématique nouvelle en programmation par contraintes et d'autre part, traiter précisément cette problématique dans le cadre du SLAM, application qui a motivé l'ensemble de ces travaux.

Sur le plan théorique, la problématique est celle de l'estimation a priori de la qualité d'un contracteur. Ce besoin apparaît clairement sur certains problèmes de nature continue, où la résolution ne peut être menée jusqu'à une précision souhaitée pour des raisons évidentes de complexité. Il est vrai qu'en domaine discret, les contracteurs ne sont souvent qu'une étape intermédiaire menant au bout du compte à des solutions isolées, mais nous pouvons opposer à cela deux faits. D'une part, il existe certains problèmes où la taille des domaines exclut également le recours à l'instanciation. D'autre part, mesurer la qualité d'un contracteur apporte toujours une information, qu'elle soit partielle ou non.

Nous avons montré dans un premier temps pour un contracteur par propagation d'intervalles (une approximation classique de l'arc-cohérence sur les domaines continus) comment modéliser ce problème sous forme de CSP, où les domaines des variables sont des « intervalles d'intervalles », ou *ressorts*. Cette modélisation permet de répondre au problème au moins en théorie, puisqu'il suffit dès lors de raisonner (calculer) dans cette structure de ressorts, où une arithmétique a pu être définie.

Dans un second temps nous avons montré comment adapter cette méthodologie au cas du SLAM. Pour cette application, nous avons montré la nécessité de travailler à deux niveaux. Le premier niveau consiste à fixer les contraintes (mais pas les domaines). Nous avons alors successivement simplifié le CSP-erreur pour en venir à un système d'inégalités linéaires de petite taille et simple à résoudre. Le second niveau gère tous les ensembles de contraintes possibles. Nous avons alors vu que seule une approche statistique était pertinente.

En conclusion, ce travail prouve que des techniques telles que la programmation par

contraintes, le calcul ensembliste et les probabilités peuvent intervenir dans une même application et s'intégrer les unes aux autres de façon cohérente.

Références

[1] F. Benhamou, F. Goualard, L. Granvilliers, and J-F. Puget. Revising Hull and Box Consistency. In *ICLP*, pages 230–244, 1999.

[2] G. Chabert and L. Jaulin. A priori Error Analysis with the help of Intervals. *submitted to SIAM Journal on Scientific Computing*.

[3] J.G. Cleary. Logical Arithmetic. *Future Computing Systems*, 2(2) :125–149, 1987.

[4] H. Collavizza, F. Delobel, and M. Rueher. Comparing Partial Consistencies. *Reliable Computing*, 5(3) :213–228, 1999.

[5] D. Daney, N. Andreff, G. Chabert, and Y. Papegay. Interval Method for Calibration of Parallel Robots : A Vision-based Experimentation. *Mechanism and Machine Theory*, pages 929–944, 2006.

[6] E. Davis. Constraint Propagation with Interval Labels. *Artificial Intelligence*, 32(3) :281–331, 1987.

[7] C. Drocourt, L. Delahoche, E. Brassart, B. Marhic, and A. Clerentin. Incremental Construction of the Robot's Environmental Map using Interval Analysis. *Global Optimization and Constraint Satisfaction : Second International Workshop, COCOS 2003*, 3478 :127–141, 2005.

[8] L. Granvilliers and F. Benhamou. Progress in the Solving of a Circuit Design Problem. *Journal of Global Optimization*, 20(2) :155–168, 2001.

[9] L. Granvilliers, J. Cruz, and P. Barahona. Parameter Estimation Using Interval Computations. *SIAM Journal on Scientific Computing*, 26(2) :591–612, 2005.

[10] E. Hyvönen. Constraint Reasoning Based on Interval Arithmetic—The Tolerance Propagation Approach. *Artificial Intelligence*, 58 :71–112, 1992.

[11] L. Jaulin. A Nonlinear Set-membership Approach for the Localization and Map Building of an Underwater Robot using Interval Constraint Propagation. *submitted to IEEE Transaction on Robotics*.

[12] L. Jaulin. Interval Constraint Propagation with Application to Bounded-Error Estimation. *Automatica*, 36(10) :1547–1552, 2000.

[13] L. Jaulin. Localization of an Underwater Robot using Interval Constraint Propagation. In *Proceedings of the Twelfth International Conference on Principles and Practice of Constraint Programming, Nantes, France, September 2006 (CP 2006)*, Nantes (France), 2006.

[14] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis*. Springer, 2001.

[15] R.E. Kalman. Contributions to the Theory of Optimal Control. *Bol. Soc. Mat. Mex.*, 5 :102–119, 1960.

[16] J.J. Leonard and H.F. Durrant-Whyte. Dynamic Map Building for an Autonomous Mobile Robot. *International Journal of Robotics Research*, 11(4), 1992.

[17] O. Lhomme. Consistency Techniques for Numeric CSPs. In *IJCAI*, pages 232–238, 1993.

[18] H.M. Taylor and S. Karlin. *An Introduction to Stochastic Modeling (3rd edition).* Academic Press, 1998.

[19] S. Thrun, W. Bugard, and D. Fox. *Probabilistic Robotics.* MIT Press, Cambridge, M.A., 2005.

[20] E. Walter and L. Pronzato. *Identification of Parametric Models from Experimental Data.* Springer-Verlag, London, UK, 1997.