



HAL
open science

Reliable robot localization: a constraint programming approach over dynamical systems

Simon Rohou

► **To cite this version:**

Simon Rohou. Reliable robot localization: a constraint programming approach over dynamical systems. Robotics [cs.RO]. Lab-STICC; UBO Brest; ENSTA Bretagne; University of Sheffield, 2017. English. NNT: . tel-03010085

HAL Id: tel-03010085

<https://hal.science/tel-03010085v1>

Submitted on 17 Nov 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



université de bretagne
occidentale

UNIVERSITE
BRETAGNE
LOIRE

THESIS / UNIVERSITY OF WESTERN BRITANNY
under the seal of the Brittany-Loire University
to obtain the title of

DOCTOR OF THE UNIVERSITY OF WESTERN BRITANNY
Branch: Robotics

Doctoral School Maths-STIC

presented by

Simon Rohou

Prepared at the ENSTA Bretagne (Lab-STICC, FR)
and at the University of Sheffield (UK)

Reliable robot localization: a constraint programming approach over dynamical systems

Thesis defended on 11th December 2017
before the jury composed of:

- **Hisham Abou-Kandil**, president of the jury
Professor, École Normale Supérieure, Cachan, FR
- **Philippe Bonnifait**, rapporteur
Professor, Heudiasyc, Compiègne, FR
- **Gilles Trombettoni**, rapporteur
Professor, LIRMM, Montpellier, FR
- **Gilles Chabert**, reviewer
Associate Professor, LS2N, Nantes, FR
- **Benoit Zerr**, reviewer
Professor, Lab-STICC, Brest, FR
- **Luc Jaulin**, thesis co-supervisor
Professor, Lab-STICC, Brest, FR
- **Lyudmila Mihaylova**, thesis co-supervisor
Professor, the University of Sheffield, UK
- **Fabrice Le Bars**, thesis co-supervisor
Associate Professor, Lab-STICC, Brest, FR
- **Sandor M. Veres**, thesis co-supervisor (invited)
Professor, the University of Sheffield, UK



The
University
Of
Sheffield.

**Reliable robot localization:
a constraint programming approach
over dynamical systems**

Simon Rohou

2017

Acknowledgments (mostly in French)

Que de chemin parcouru pendant ces trois années de thèse... et voilà déjà le moment de poser un point final sur la synthèse de ce doctorat. Je ne saurais finir ce travail sans avoir une pensée pour les nombreuses personnes qui ont croisé mon chemin tout au long de cette aventure.

Je remercie en premier lieu mes encadrants pour l'énergie déployée au cours de ces trois années. First, a special thank to Pr. Lyudmila Mihaylova and Pr. Sandor M. Veres for their hospitality in Sheffield and the discussions around this work, making this Franco-British collaboration a memorable experience. J'adresse ma sincère reconnaissance à Fabrice Le Bars pour sa grande disponibilité, son soutien technique et ses nombreux retours sur ce travail. Enfin, je remercie chaleureusement Luc Jaulin pour son encadrement exceptionnel, ses «*Alors cette thèse ?*» quotidiens et la confiance qu'il m'a accordé, depuis mes premiers jours de thèse jusqu'à l'affrontement final dans l'arène de la soutenance.

Aussi je souhaite naturellement remercier les membres de mon jury pour le temps consacré à la lecture de ce manuscrit et leur présence lors de la soutenance. En particulier, merci à Philippe Bonnifait pour son enthousiasme constructif et l'intérêt porté aux applications de ces travaux en robotique, ainsi qu'à Gilles Trombettoni pour son optimisme constant sur ces problématiques de tubes. Je remercie également Hisham Abou-Kandil d'avoir présidé ce jury ainsi que Gilles Chabert et Benoit Zerr qui ont endossé les rôles d'examineurs. J'ai été honoré d'avoir un jury si hétérogène : vos nombreuses remarques et observations me donnent déjà du grain à moudre pour le futur.

Mes pensées s'adressent évidemment à mes amis doctorants et chercheurs de tous horizons qui ont contribué, chacun à leur manière, à un environnement de travail sain et captivant dans ce bout du monde brestois. Je pense bien entendu à mes deux co-bureaux du M025, si souvent surnommé *La conciergerie* : Thomas Le Mézo, pour les innombrables bavardages, les coups de main jusqu'au lac de Polytechnique et Piombino, et son dynamisme bien souvent trop débordant qui aura profité à tant de projets. Merci également à Benoît Desrochers, alias *Benoïd*, pour sa générosité, son amitié et tout ce temps passé à traiter les données de *Daurade*.

Et comment oublier les *débriefings* du vendredi soir en compagnie des roboticiens. Merci à Benoit Zerr, alias *Benoiz*, pour son humour quotidien, sa bonne humeur, son expérience et ses conseils pertinents, qu'ils soient d'ordre scientifique, logistique

ou caféique. Je tire mon chapeau à Michel Legris, qui communique si facilement sa grande passion pour les sciences : un véritable plaisir que de découvrir toutes ces anecdotes captivantes, de discuter de problèmes *sordides* ou de débattre de l'intérêt d'être pessimiste. Merci également à Rodéric Moitié pour sa gentillesse et son amitié, ses coups de main et sa passion pour la photographie. J'espère que nous aurons d'autres occasions de sortir les 70D pour capturer quelques belles lumières.

Mes remerciements s'étendent naturellement à l'ensemble de l'ancienne équipe OSM de l'ENSTA Bretagne. Les innombrables litres de café partagés avec chacun d'entre vous ont été une grande source de motivation et de savoir. Merci à Guillaume Sicot pour son insatiable curiosité et ses *petites* questions, à Pierre Bosser pour son humour et son expérience en positionnement, à Amandine Nicolle pour sa présence pendant la rédaction, à Jordan Ninin pour ses coups de main lors de mes débuts avec *IBEX*, à Christophe Osswald pour sa sagesse, à Benoit Clément alias *Benoic* et Gilles Le Chenadec pour leurs conseils, à Nathalie Debese, Isabelle Quidu et Hélène Thomas pour leur bonne humeur permanente.

Je n'oublie pas les anciens doctorants et post-doctorants Clément Aubry, Vincent Drevelle, Saad Ibn Seddik, Jérémy Nicola, Khadimoullah Vencatasamy, Laurent Picard, Guillaume Jubelin, et les nombreux échanges sympathiques autour de cette fameuse table Véléda qui aura vu passer tant d'idées... J'en profite pour souhaiter un bon courage aux prochains, avec un clin d'œil à Gaspard Minster, Dominique Monnet, Alexandre Lefort, Julien Ogor, Thibaut Nico, Guilherme Schvarcz Franco, Juan Luis Rosendo, Vincent Myers, Yoann Sola, Auguste Bourgois.

Ces dernières années ont été l'occasion de découvrir les parapheurs et toutes les procédures administratives que l'on imagine, et je remercie tout particulièrement Annick Billon-Coat et Michèle Hofmann pour leur soutien et leur bonne humeur malgré des demandes de dernière minute. Vos anecdotes de voyages me donnent déjà envie de filer à l'aéroport.

Merci à Gilles Le Maillot, Pierre Simon, Thierry Ropert, Irvin Probst, Yvon Gallou, Olivier Reynet, Olivier Ménage, Patrick Rousseaux et Maxime Bouyssou. Vos coups de mains et conseils techniques en électronique et robotique marine sont bien souvent tombés à pic. De même, j'adresse ma reconnaissance à Alain Bertholom ainsi qu'à tout l'équipage de l'*Aventurière II* (DGA-TN Brest), sans qui les illustrations des théories de cette thèse n'auraient pas eu la même classe. Je n'oublierai pas ce barbecue sur la plage arrière de l'*Aventurière* pendant les manœuvres autonomes de *Daurade*.

Je remercie plus généralement la DGA pour le financement de ce projet dans le cadre de son programme de thèses franco-britanniques avec le DSTL anglais. J'ai été honoré de bénéficier d'un tel mécénat, si propice aux collaborations internationales. I thank my DGA/DSTL advisors Véronique Serfaty, Calum Meredith and Timothy Clarke for the meetings in Paris and Porton Down. In addition, I extend my thanks to Peter Franek for the fruitful collaboration we started and for his hospitality during my stay at the Institute of Science and Technology of Austria. I do hope we will have further opportunities to work together.

J'ai également une pensée pour les Montpelliérains Michel Benoit et Vincent Creuze et pour les discussions qui m'ont progressivement amené sur la voie de la recherche, appliquée à ma passion pour la robotique sous-marine.

Merci à tous mes amis pour les évasions aux quatre coins de la France et d'ailleurs, retardant ma transformation en robot autonome régi par de drôles d'équations d'état. Je remercie tout particulièrement ceux qui, régulièrement, sont venus prendre des nouvelles. Une attention très touchante sur ces derniers mois tendus... Merci à Anthonin pour les nombreux *MBS* et autres variantes cinématographiques. Tes distractions au quotidien ont été salutaires. À Martin, avec qui je comptabilise tant de voyages ressourçants et d'anecdotes qui me donnent encore le sourire. À Audrey et Simon, pour vos messages poignants et votre soutien. À Irène et Romain, pour votre présence que je n'oublierai pas.

Enfin, à mon frère et à mes parents : *merci* est un bien faible mot pour vous exprimer ma gratitude, mon affection, et je regrette que vous ayez eu à subir les effets secondaires de cette thèse. Votre confiance et votre présence ont été une force. Vous êtes mon socle.

Pour finir, je remercie le lecteur d'avoir eu la sagesse d'apprécier ces quelques pages de remerciements comme un échauffement avant une longue immersion dans les chapitres suivants. Puissent-ils occuper ses pensées et susciter l'intérêt, la passion, qui m'ont animé pendant ces trois premières années de recherche.

Simon.

Notations

To facilitate the understanding of this document, the mathematical notations that will be used are listed hereinafter. All of these will be introduced throughout the chapters. Vectors, matrices and vectorial functions will be represented in bold while intervals will be denoted by brackets []. The blackboard bold convention is used to represent other classical sets, *e.g.* \mathbb{X} , \mathbb{Y} .

Modelisation

- \mathbf{x} : state vector, $\mathbf{x} \in \mathbb{R}^n$
: (or an arbitrary variable)
- \mathbf{p} : 2D position vector, $\mathbf{p} = (x_1, x_2)^\top$
- \mathbf{u} : input vector, $\mathbf{u} \in \mathbb{R}^m$
- \mathbf{f} : evolution function, $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$
: (or an arbitrary function)
- \mathbf{z} : vector of observations, $\mathbf{z} \in \mathbb{R}^p$
- \mathbf{g} : observation function, $\mathbf{g} : \mathbb{R}^n \rightarrow \mathbb{R}^p$
- \mathbf{h} : drifting function (clock problem, Chapter 5)
: configuration function (SLAM method, Chapter 7)
- τ : drifting time reference
- ϕ, θ, ψ : roll, pitch, yaw (heading)

Intervals and sets

- \emptyset : empty set
- $\mathbb{I}\mathbb{R}$: set of all intervals of \mathbb{R}
- $\mathbb{I}\mathbb{R}^n$: set of all boxes of \mathbb{R}^n
- $[x]$: interval $[x^-, x^+]$, $[x] \in \mathbb{I}\mathbb{R}$
- x^- : lower bound of the interval $[x]$
- x^+ : upper bound of the interval $[x]$
- x^* : actual (unknown) value enclosed by $[x]$
- $[\mathbf{x}]$: box or interval-vector, $[\mathbf{x}] \in \mathbb{I}\mathbb{R}^n$
- $[f]$: inclusion function of f
- $[f]^*$: minimal inclusion function of f
- \sqcup : squared union, envelope of the following terms
- \mathcal{L}_f : constraint related to a function f
- \mathcal{C}_f : contractor related to \mathcal{L}_f
- $[\mathbb{X}]$: box enclosing the set \mathbb{X}
- $\partial\mathbb{X}$: boundary of the set \mathbb{X}
- $\#\mathbb{E}$: cardinality (number of items) of the set \mathbb{E}

Trajectories and tubes

- t : time variable
- (\cdot) : (dot) system independent variable
- $a(\cdot)$: trajectory, $\mathbb{R} \rightarrow \mathbb{R}$
- $a(t)$: evaluation of $a(\cdot)$ at t
- $\dot{a}(\cdot)$: derivative of $a(\cdot)$
- $[a](\cdot)$: tube of trajectories, $\mathbb{R} \rightarrow \mathbb{IR}$
- $[a](t)$: interval value of $[a](\cdot)$ at t
- $\emptyset(\cdot)$: empty tube
- $\mathbf{p}(\cdot)$: horizontal robot trajectory, $\mathbb{R} \rightarrow \mathbb{R}^2$
- $\mathcal{C}_{\frac{d}{dt}}$: differential tube contractor
- $\mathcal{C}_{\text{eval}}$: evaluation tube contractor
- \mathcal{C}_{t_1, t_2} : inter-temporal evaluation tube contractor
- $\mathcal{C}_{\mathbf{p} \Rightarrow \mathbf{z}}$: inter-temporal implication tube contractor
- d : thickness function, diagonal of a slice, $d : \mathbb{IR}^2 \rightarrow \mathbb{R}$
- δ : time discretization of a tube

Loops

- \mathbf{t} : t -pair defining a loop, also denoted by (t_1, t_2)
- \mathbb{T}^* : set of all \mathbf{t}
- \mathbb{T} : set of feasible \mathbf{t} in a bounded-error context
- \mathbb{T}_i : compact and connected subset of \mathbb{T}
- Ω : outer approximation of \mathbb{T} made of subpavings
- Ω_i : compact and connected subset of Ω
- \mathcal{N} : Newton test
- \mathcal{T} : topological degree test
- λ : number of loops along a trajectory $\mathbf{p}(\cdot)$

Other notations

- ε : precision of a SIVIA algorithm
- $\text{deg}(\mathbf{f}, \Omega)$: topological degree of \mathbf{f} over Ω
- $\mathbf{J}_{\mathbf{f}}$: Jacobian matrix of \mathbf{f}
- $\text{det}([\mathbf{J}])$: enclosure of interval matrix's determinant

Contents

1	Introduction	1
1.1	Underwater challenges	2
1.1.1	In the vastness of the unknown	2
1.1.2	Hostile environments	4
1.1.3	Autonomous Underwater Vehicles	7
1.2	The localization problem	11
1.2.1	State equations	12
1.2.2	Dead-reckoning drawbacks	13
1.2.3	Underwater acoustic positioning systems	15
1.2.4	SLAM: a standalone solution	20
1.3	PhD thesis context	24
1.3.1	New localization approach in very poor environments	24
1.3.2	Estimation methods	26
1.3.3	Constraint programming approach over dynamical systems	29
1.3.4	Thesis outlines and contributions	31
I	Interval tools	35
2	Static set-membership state estimation	37
2.1	Introduction	38
2.2	Interval analysis	41
2.2.1	Once upon a time	41
2.2.2	Intervals	42
2.2.3	Inclusion functions	47
2.2.4	Pessimism and wrapping effect	49
2.3	Constraints propagation	52
2.3.1	Constraint networks	52
2.3.2	Contractors	54
2.3.3	Application to static range-only robot localization	57
2.4	Set-inversion <i>via</i> interval analysis	60
2.4.1	Subpaving	60
2.4.2	SIVIA algorithm for set-inversion	61
2.4.3	Illustration involving contractions	65
2.4.4	Kernel characterization of an interval function	65
2.5	Discussions	69
2.5.1	From sensors to reliable results	69
2.5.2	Numerical libraries	70
2.5.3	Reliable tool for proof purposes	71

2.6	Conclusion	71
3	Constraints over sets of trajectories	73
3.1	Towards dynamic state estimation	74
3.1.1	Overall motivations	74
3.1.2	The approach defended in this thesis	76
3.2	Tubes	76
3.2.1	Definitions	76
3.2.2	Tube analysis	78
3.2.3	Contractors	81
3.3	Implementation	83
3.3.1	Data structure	85
3.3.2	Build a tube from real datasets	87
3.3.3	<i>TubeX</i> , dedicated tube library	89
3.4	Application: dead-reckoning of a mobile robot	90
3.4.1	Test case	90
3.4.2	Constraint Network	91
3.4.3	Resolution	91
3.5	Discussions	92
3.5.1	Limits	92
3.5.2	Extract the most probable trajectory from a tube	93
3.5.3	Application to path planning	95
3.6	Conclusion	95
II	Constraints-related contributions	97
4	Trajectories under differential constraints	99
4.1	Introduction	100
4.1.1	The differential problem	100
4.1.2	Attempts with set-membership methods	100
4.1.3	Contribution of this thesis	103
4.2	Differential contractor for $\mathcal{L}_{\frac{d}{dt}} : \dot{x}(\cdot) = v(\cdot)$	104
4.2.1	Definition and proof	104
4.2.2	Contraction of the derivative	109
4.2.3	Implementation	110
4.3	Contractor-based approach for state estimation	114
4.3.1	Constraint network of state equations	114
4.3.2	Fixed-point propagations	117
4.3.3	Theoretical example of interest $\dot{x} = -\sin(x)$	118
4.4	Robotic applications	121

4.4.1	Causal kinematic chain	121
4.4.2	Higher order differential constraints	123
4.4.3	Kidnapped robot problem	125
4.4.4	Actual experiment with the <i>Daurade</i> AUV	126
4.5	Conclusion	130
5	Trajectories under evaluation constraints	131
5.1	Introduction	132
5.1.1	Contribution of this thesis	132
5.1.2	Motivations to deal with time uncertainties	133
5.2	Generic contractor for trajectory evaluation	136
5.2.1	Tube contractor for the constraint $\mathcal{L}_{\text{eval}} : z = y(t)$	136
5.2.2	Implementation	142
5.2.3	Application to state estimation	144
5.3	Robotic applications	145
5.3.1	Range-only robot localization with low-cost beacons	145
5.3.2	Reliable correction of a drifting clock	153
5.4	Conclusion	160
III	Robotics-related contributions	161
6	Looped trajectories: from detections to proofs	163
6.1	Introduction	164
6.1.1	The difference between detection and verification	164
6.1.2	Proprioceptive <i>vs.</i> exteroceptive measurements	165
6.1.3	The two-dimensional case	165
6.2	Proprioceptive loop detections	166
6.2.1	Formalization	166
6.2.2	Loop detections in a bounded-error context	167
6.2.3	Approximation of the solution set \mathbb{T}	168
6.3	Proving loops in detection sets	171
6.3.1	Formalism: zero verification	171
6.3.2	Topological degree for zero verification	173
6.3.3	Loop existence test	175
6.3.4	Reliable number of loops	179
6.4	Applications	181
6.4.1	The <i>Redermor</i> mission	182
6.4.2	The <i>Daurade</i> mission	187
6.4.3	Optimality of the approach	187
6.5	Conclusion	194

7	A reliable temporal approach for the SLAM problem	195
7.1	Introduction	196
7.1.1	Motivations	196
7.1.2	SLAM formalism	198
7.1.3	Inter-temporalities	199
7.2	Temporal SLAM method	202
7.2.1	General assumptions	202
7.2.2	Temporal resolution	203
7.2.3	$\mathcal{L}_{\mathbf{p} \Rightarrow \mathbf{z}}$: inter-temporal implication constraint	204
7.2.4	The $\mathcal{C}_{\mathbf{p} \Rightarrow \mathbf{z}}$ contractor	208
7.2.5	Temporal SLAM algorithm	218
7.3	Underwater application: bathymetric SLAM	221
7.3.1	Context	221
7.3.2	<i>Daurade</i> 's underwater mission, 20 th October 2015	224
7.3.3	<i>Daurade</i> 's underwater mission, 19 th October 2015	227
7.3.4	Overview of the environment	232
7.4	Discussions	233
7.4.1	Relation to the state of the art	233
7.4.2	About a Bayesian resolution	234
7.4.3	Biased sensors	235
7.4.4	Fluctuating measurements	236
7.5	Conclusion	238
8	General conclusions and prospects	241
8.1	Conclusions	241
8.2	Summary of the contributions	243
8.3	Overall prospects	244
	Bibliography	245
	List of Figures	259
	List of Tables	263
	List of Algorithms	264
	List of Abbreviations	265
	Index	266

Contents

1.1 Underwater challenges	2
1.1.1 In the vastness of the unknown	2
1.1.2 Hostile environments	4
1.1.3 Autonomous Underwater Vehicles	7
1.2 The localization problem	11
1.2.1 State equations	12
1.2.2 Dead-reckoning drawbacks	13
1.2.3 Underwater acoustic positioning systems	15
1.2.4 SLAM: a standalone solution	20
1.3 PhD thesis context	24
1.3.1 New localization approach in very poor environments . .	24
1.3.2 Estimation methods	26
1.3.3 Constraint programming approach over dynamical systems	29
1.3.4 Thesis outlines and contributions	31

1.1 Underwater challenges

«*On peut braver les lois humaines,
mais non résister aux lois naturelles.*»

“*We may brave human laws, but we cannot resist natural ones.*”

Twenty Thousand Leagues Under the Sea, Jules Verne

1.1.1 In the vastness of the unknown

95%. This striking figure, stated¹ by the American National Oceanic and Atmospheric Administration (NOAA) tells how little we know about oceans: about 95% of this underwater realm remains unseen by human eyes. And yet, it covers two-thirds of the Earth’s surface. It is even said that we best know the Moon’s surface than our oceans’ depths. Nevertheless, marine technologies have changed dramatically since the last hundred years, allowing ways to explore the bodies of water that would have been unimaginable before.



Figure 1.1: The *HMS Challenger*, a British corvette that took part in the first global marine research expedition: the *Challenger Expedition*, 1872–1876. Painting by William Frederick Mitchell.

¹<http://www.noaa.gov/oceans-coasts>

One could say that the underwater exploration started with the *Challenger Expedition* (1872, Figure 1.1), by probing the depths from the surface with lead lines. The *Challenger Deep*, deepest known point on Earth², has been discovered during this expedition. And yet, it was not until the start of the sixties that this spot has been visited by humans, during the dive of the manned submersible *Trieste*, Figure 1.2. And ever since, the place has been reached by very few expeditions, mainly unmanned descents.

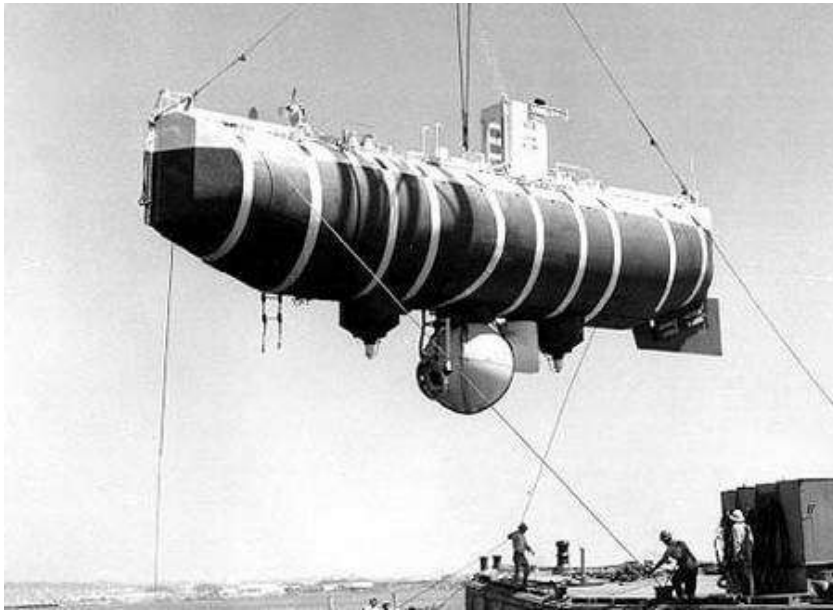


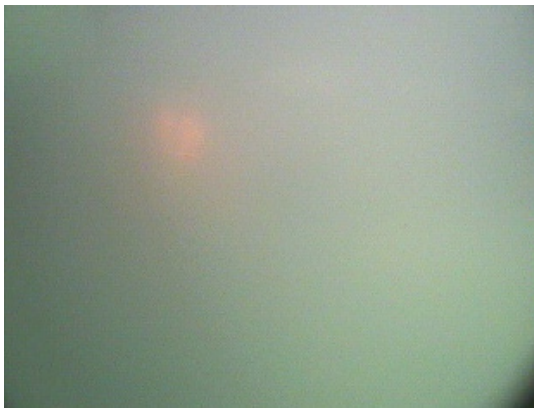
Figure 1.2: *Trieste*, a Swiss-designed and Italian-built deep-diving research bathyscaphe. It was able to reach any point of Earth's abysses such as the Mariana Trench in 1960. *Photo: U.S. Naval Historical Center.*

The dive of the *Trieste* revealed the capacity to build vehicles able to resist colossal pressures. However, the costs of this endeavor is huge compared to the range of the explored area: only few square meters around the submersible. And if exploration techniques have evolved considerably over the years, the ratio exploration/cost or exploration/time remains a major impediment in the discovery of our oceans.

²*Challenger Deep*: depth estimated at 10916m *in situ* by submersibles.

1.1.2 Hostile environments

Withstand the high pressures of the column water, corrosive salinity, unpredictable currents, *etc.*, is one thing. However, perceive the environment is another matter. Figure 1.3 provides an example of poor visibility that can be encountered under the surface. Strong opacities in shallow waters, or lack of light in the deepest ones, make it difficult to gather information from cameras. Other conventional means of exploration or communication suffer from strong attenuations of their electromagnetic waves through the water column.



(a) An orange buoy dimly visible at 3m.



(b) Unstructured environments.



(c) A lost wireless router.



(d) Sea life, leading to outliers.

Figure 1.3: In the shallow waters of La Spezia (Italy) during the SAUC-E competitions in the NATO Centre for Maritime Research and Experimentation (CMRE, formerly NURC), 2013–2014. These are images taken by the ENSTA Bretagne’s autonomous robot *Vici*. Design algorithms to automatically analyze these observations remains a challenging task.

Underwater acoustics

Underwater acoustics is about the only technology left with sufficient performances to increase the range of visibility. A telling experiment is the *Heard Island* test performed in 1991 [Munk et al., 1994] and planned in order to test the emission of a man-made acoustic signal throughout the world's oceans. A special phase modulated signal of 57Hz, emitted from an island located in the southern Indian Ocean, has been received by sixteen sites around the world, some of them based on both coasts of North America. This experiment demonstrated that great distances are reachable by acoustics.

Considering an estimation of the sound celerity profile along the propagation, an acoustic wave is even well suited to perceive distances between the emitter and any obstacle in the environment. In practice, ranges of a few dozen meters are affordable to maintain precision at reasonable energy cost. However, one should note that an acoustic signal rarely propagates in straight line. This impacts distances' estimations and may even generate blind zones³. Underwater acoustics remains nonetheless the most suited approach for wide explorations, but the related solutions are far from being straightforward.

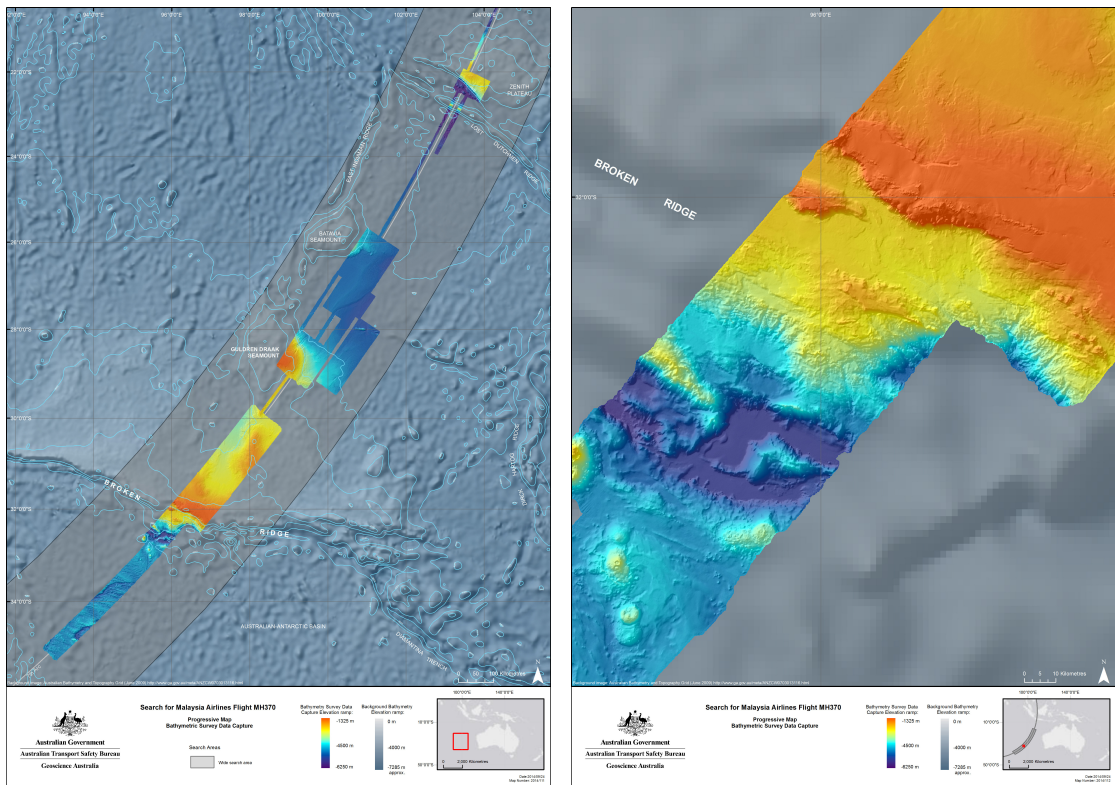
A needle in a haystack

The work on this thesis started on the very same day as the beginning of the underwater search for the lost MH370 aircraft operated by Malaysia Airlines, that presumably disappeared in the southern Indian Ocean in 2014. Despite a tremendous deployment of maritime means, making this multinational search effort the largest and most expensive in aviation history, the aircraft remains unfindable. From October 2014 to January 2017, an overall survey of 120000km² of the seafloor was performed, with unsuccessful results. Given the vast areas involved, this search sadly reveals the difficulty we still have to explore the extent of the seabed.

The unfruitful research allowed nonetheless to improve the knowledge we had on this part of the oceans, providing a level of details rarely reached before in the deep environment [Picard et al., 2017]. Figure 1.4 illustrates a comparison between the previous mapping of the seabed, that had an average spatial resolution

³In the Atlantic Ocean for instance, due to the physical properties of the environment, two vehicles on the same layer of water and separated by 60 meters may not be able to perceive each other.

of about 5km^2 , and the new Digital Elevation Model (DEM) obtained with a resolution of less than 0.01km^2 . During the search, the vessels equipped with acoustic means such as side-scan sonars or multibeam echosounders were not able to scan the entire extent of the search area. Indeed, the seabed parts with the most complex and challenging topography were only reachable by Autonomous Underwater Vehicles (AUVs), equipped with similar technology and specifically designed for high resolution survey operations in remote deep water locations. These vehicles lend a helping robotic hand in such exploration efforts.



(a) Overview of the survey.

(b) Zoomed area.


Figure 1.4: Extract from the bathymetric survey conducted during the search for MH370 aircraft off the west coast of Australia. Gray areas correspond to the bathymetry indirectly estimated using satellite-derived gravity data. In contrast, colored data have been acquired by marine means, highlighting the need to undertake surveys *in situ* for higher precisions. ©Copyright 2014, Commonwealth of Australia.

1.1.3 Autonomous Underwater Vehicles

Because of difficulties due to complex environments and vast areas still uncovered, the use of autonomous vehicles appears to be a durable solution to face these conditions and push the boundaries of the oceans knowledge. Indeed, even with efficient methods such as underwater acoustics, the footprint of marine sensors is still modest in view of the extent of what has to be explored. Multiply the number of vessels equipped with sensors costs a lot, due to the involvement of crews. On top of that, surface vehicles are not sufficient to provide details of deep waters. Marine robots [Creuze, 2014] are an attractive alternative to increase the exploration means at reasonable costs.

Furthermore, a global supervision of an underwater robot performing an exploration task is rarely affordable due to the opacities of the environment mentioned before. The low-rate of underwater communications and the latency during the propagation of messages require the robot a full degree of autonomy. For these reasons, new marine robots are designed to make unsupervised decisions in order to achieve a given task. They can be involved in several marine applications such as hydrography, oceanography, climate change monitoring, military operations in mine hunting [Toumelin and Lemaire, 2001], wrecks search [L'Hour and Creuze, 2016], to name but a few.

As they sail underwater without receiving orders from the surface, they have to sense their environment and act accordingly. AUVs are then equipped with sensors such as sonars or cameras. In addition, they estimate their own position by themselves [Leonard et al., 1998], which is a complicated task as always in the underwater world. The localization problem will be presented in Section 1.2 and is the main motivation of this thesis. The contributions of this work will be presented through actual experiments involving two AUVs⁴, *Redermor* and *Daurade*, introduced hereinafter.

⁴The main characters of this document will be drawn by the following  as reference to the MOOS-IvP middleware [Benjamin et al., 2010] from which this symbol comes from. MOOS-IvP is a set of open source modules for providing autonomy on robotic platforms, in particular autonomous marine vehicles. This framework has been used during this work as basis of actual experiments.

The *Redermor* AUV

The *Redermor*⁵ AUV, pictured in Figure 1.5, was an experimental robot designed during the Franco-British collaborative project *Remote Mine Hunting System*. Built during the nineties at DGA Techniques Navales Brest (formerly GESMA), it served as platform for several studies [Quidu et al., 2007]. The main characteristics of the vehicle are summarized in Table 1.1, [Toumelin and Lemaire, 2001].

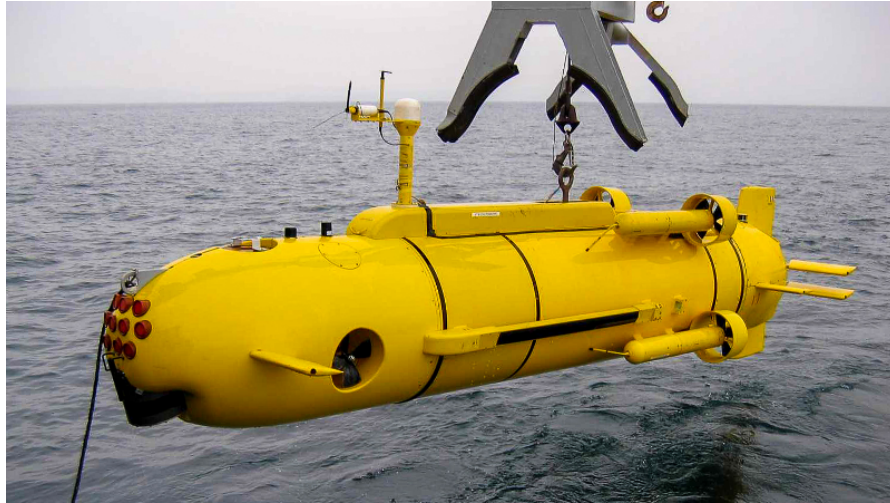


Figure 1.5: The *Redermor* AUV before a sea trial. The thrusters' layout allows it to circumnavigate a point such as a mine to identify, its front looking sonar providing different viewing angles of the target. *Photo: DGA-TN Brest.*

Table 1.1: *Redermor*'s main characteristics.

weight	: 3400kg
length	: 6.40m
speed	: up to 10 knots (5.14m/s)
max depth	: 200m

During a mission, the position of the robot is provided by an Inertial Navigation System (INS) coupled with a Doppler Velocity Log (DVL) sensing robot's speed. The positioning error is estimated at some meters per hour. It is difficult to provide the reader with accurate figures about this error as it is related to the pattern followed by the vehicle, its altitude or its speed⁶.

⁵*Redermor* means *rider of the seas* in the Breton language.

⁶The DVL accuracy depends among other things on its distance from the seabed and the sensed velocity. For a 1200kHz Teledyne DVL, the errors are given as: $\pm 0.3\text{cm/s}$ at 1m/s, $\pm 0.4\text{cm/s}$ at 3m/s, $\pm 0.5\text{cm/s}$ at 5m/s.

The *Daurade* AUV

Redermor is today retired and left its place to the new *Daurade* AUV, see Figure 1.6. This vehicle has been built by the ECA group and performed many experiments since 2005 on the shores of France. It is still used today by DGA-TN Brest, in collaboration with the Service Hydrographique et Océanographique de la Marine (SHOM) for survey purposes or mine hunting applications. Its main characteristics are given in Table 1.2.



Figure 1.6: *Daurade* AUV managed by the crew of the *Aventurière II*, during an experiment in the Rade de Brest, October 2015. Photo: S. Rohou.

Table 1.2: *Daurade*'s main characteristics.

weight	: 1010kg
length	: 5m
speed	: up to 8 knots (4.11m/s)
max depth	: 300m
autonomy	: 10h at 4 knots, 2h at 8 knots
sonar coverage range	: 150m

It is equipped with an INS Phins from iXblue, connected to a DVL⁷ as for the *Redermor*. Its positioning accuracy is 3m/h at 2 knots, or 0.1% of the traveled distance, based upon a hybridization INS/DVL. On the other hand, 20 meters of positioning error are obtained after 5 minutes of navigation in pure inertial mode.

⁷The vehicle is configurable with either a 300kHz or a 1200kHz Workhorse Teledyne RDI DVL.

Redermor and *Daurade* are heavy vehicles with high costs of handling and maintenance. Furthermore, the embedded navigation systems cannot be easily changed, which is a limitation when it comes to try new algorithms for autonomous navigation. This motivated the design of smaller and cheaper units.

The *Toutatis* AUVs project

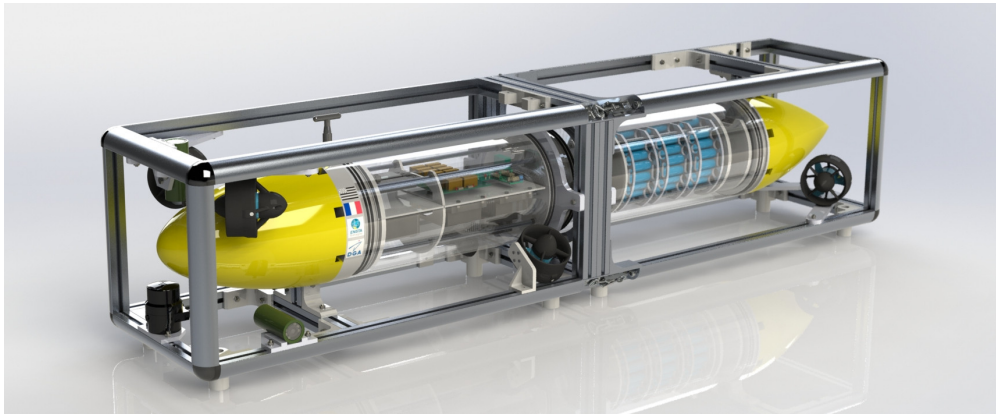
A new class of autonomous underwater vehicles has been designed during this PhD thesis. The *class* term refers to a group of several units of the same type. The *Toutatis*⁸ project, as Team Of Underwater robots for Autonomous Tasks of Inspection and Survey, was aimed at applying the tools presented in this document in realistic scenarios. The project has been paused and will be resumed later.

Figures 1.7 picture some modeling views of the vehicles. The units are modular in order to fit with the mission requirements. The aluminium cage protects the tube, the sensors and the thrusters. It is also convenient to arrange devices everywhere on the frame without difficulty. In addition, the cage is useful to carry, transport and store the vehicles; then it will be possible to stow all AUVs on top of each other in a reduced place. Finally, landing on the seabed will not present any risk.

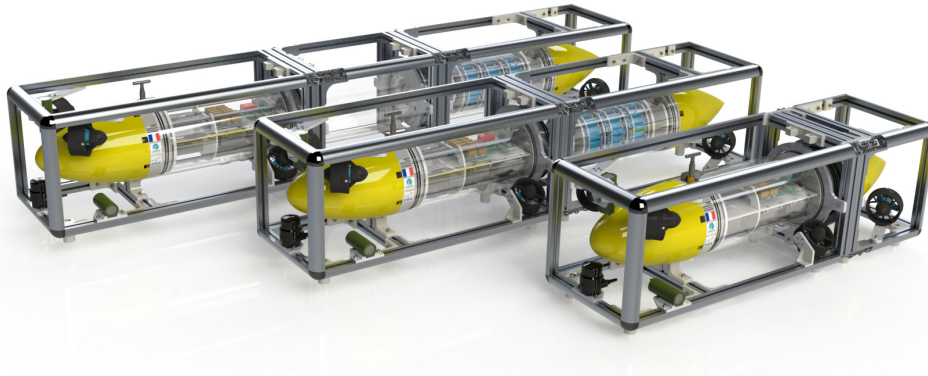
Being powered by six thrusters, the AUVs will be omni-directional which is of interest to orient sensors according to the needs. The vehicles should be equipped with cameras, sonars, echosounders, an acoustic modem, a low-cost Inertial Measurement Unit (IMU), a DVL and a pressure sensor.

If it appears clear that AUVs have the potential to revolutionize the means of exploring our oceans, several challenges still remain before considering their active use, the primary of which is the localization problem.

⁸*Toutatis* is a Celtic god in ancient Gaul and Britain culture. It was seen as the tribe's leader: this name illustrates the future behaviour of these robots: they will act as members of a team, based on communication and collaboration.



(a) One unit.



(b) A stack of modular vehicles.

Figure 1.7: An overview of the *Toutatis* AUVs project.

1.2 The localization problem

A robot localization is the process of estimating the vehicle's position in a given reference frame. This is a key point of mobile robotics as it conditions the success of other processes such as sensing, actuation, manipulation or mapping. In the latter case, a poor positioning estimation will directly lead to datasets acquisitions with meaningless spatial distribution. On top of that, a good localization is mandatory to ensure the safety of the vehicle: for instance when moving in the vicinity of some offshore construction or during the recovery procedure.

In the case of autonomous vehicles, the localization process has to be embedded. Indeed, as soon as a robot dives under the surface, it does not receive electromagnetic

waves anymore. Global Navigation Satellite Systems (GNSSs)⁹, well used in terrestrial and aerial applications, cannot be considered in the underwater case. This raises an important amount of work in the community of underwater robotics, in order to investigate new localization techniques. It has led to the design of dedicated sensors and algorithms.

This thesis is a contribution to the localization problem. The current section formalizes the problem and briefly presents already existing positioning approaches in order to place our work with respect to the state of the art and see its main added values. Our motivation being the exploration of wide and unknown underwater areas, we will only consider the generic case of long range navigations without any prior knowledge¹⁰ on the environment.

1.2.1 State equations

The localization algorithms make use of data collected by a set of sensors that we can divide into two categories: *proprioceptive* measurements and *exteroceptive* ones. The first one gathers information related to the robot's state, such as its acceleration, heading, speed, while the second is related to the environment: temperature, distance from a beacon, camera images, *etc.*

Mathematics provide a way to convert the world into equations. For the localization problem, the following state equations are generally used:

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), & (1.1a) \\ \mathbf{z}(t) = \mathbf{g}(\mathbf{x}(t)). & (1.1b) \end{cases}$$

Here, $\mathbf{x} \in \mathbb{R}^n$ depicts the state of the robot: position, heading, speed, *etc.* We then speak about *state estimation* as the localization problem amounts to estimating \mathbf{x} based on these equations and both proprioceptive and exteroceptive measurements.

⁹At the time of writing, GPS, GLONASS and Galileo are available terrestrial positioning systems respectively handled by the United States, Russia and the European Union.

¹⁰Otherwise, when a given initial map of the environment is available, a process of data matching between robot data and the map leads to map-based navigation approaches [Tuohy et al., 1996, Tyrén, 1982].

Equation (1.1a) is differential and depicts the state evolution. $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$ is called *evolution function*. The input vector $\mathbf{u} \in \mathbb{R}^m$ represents the control applied on \mathbf{x} . Measurements are depicted by a vector $\mathbf{z} \in \mathbb{R}^p$ related to \mathbf{x} through the *observation function* $\mathbf{g} : \mathbb{R}^n \rightarrow \mathbb{R}^p$, Equation (1.1b). We emphasize that in practice, both functions \mathbf{f} and \mathbf{g} may be uncertain or non-linear. These constraints will be carefully taken into account in this document.

For instance, an underwater robot may be described as $\mathbf{x} = (x_1, x_2, x_3, \psi, \vartheta)^\top$ where x_1, x_2, x_3 are respectively the east, north and vertical positions of the robot, ψ its heading and ϑ its speed. An onboard pressure sensor will easily provide proprioceptive data about the vertical position of the robot¹¹. However, the estimation of the horizontal location (x_1, x_2) is a lot more challenging. We summarize in the next sections several useful localization methods [Leonard et al., 1998]. Note that in this document, the horizontal position will sometimes be denoted $\mathbf{p} = (x_1, x_2)^\top$ to simplify the reading.

1.2.2 Dead-reckoning drawbacks

Proprioceptive approach

The simplest way to localize one-self is *dead-reckoning*. From successive proprioceptive measurements, a system will estimate its own evolution, step by step. A blind walker would proceed in the same manner by counting its footsteps and then roughly estimating its move. This is the most common localization approach in mobile robotics, as it only requires inner sensors and runs in most environments.

An embedded IMU will provide information on linear accelerations and rotation speeds of the system. Coupled with a magnetometer, the system will also be able to assess its *Euler angles*: the *bank* ϕ , the *elevation* θ and the *heading* ψ . The terms *roll* ϕ , *pitch* θ , *yaw* ψ are usually employed to depict these orientations. Then, a dedicated unit called Inertial Navigation System (INS) will provide an estimate of the robot's state based on these measurements and some algorithms such as a Kalman filter [Kalman, 1960].

External references are not involved in this process. However, in the field of underwater robotics, it is important to mention DVLs that provide information

¹¹The estimated depth depends on pressure and water salinity. In the oceans, each 10 meters of depth adds another 1.025 bar to the surface pressure.

about the vehicle speed. From the emission of acoustic beams, the device will measure velocities within the water column using the Doppler effect. When the beams reach the bottom, measured velocities can be used to compute displacements relative to the seabed. Today, DVLs are well hybridized with marine INSs, which greatly improve performances regarding pure inertial navigations.

Drifting effects

From a known initial position \mathbf{p}_0 , an INS will filter the measurements and estimate the successive poses of the robot. This is achieved by integrating the motion data in time, sometimes twice in the case of acceleration measurements, which mathematically leads to quadratic errors. For instance, a position estimated by means of biased acceleration measurements $\mathbf{a}_b(t) = \mathbf{a}^*(t) + \mathbf{b}$ is expressed by

$$\mathbf{p}_b(t) = \iint_{t_0}^t \mathbf{a}_b(\tau) d\tau + \mathbf{p}_0. \quad (1.2)$$

The bias \mathbf{b} is cumulated over time in such a way that the position error is:

$$\mathbf{e}(t) \approx \mathbf{b} \frac{t^2}{2}. \quad (1.3)$$

This effect is pictured in Figure 1.8. Unfortunately the drift cannot be bounded and is even substantial when using second order measurements such as accelerations.

The errors can have various causes: noise from sensors, wrong calibrations of the units, a misalignment between the magnetometer and the IMU, *etc.* In the case of underwater robotics, one must also consider the impact of ocean currents on the vehicle which adds another velocity component poorly sensed by these sensors. Furthermore, without mentioning power consumptions, the cost of an accurate INS may be too excessive for small AUVs. An essential link between an INS and conventional exteroceptive sensors such as sonars has to be contemplated [Dillon, 2016].

Therefore, dead-reckoning methods are not suited for underwater long range navigations. The AUV will have to surface on a regular basis in order to fix its positioning estimation thanks to GNSS signals. This entails risks related to discretion, safety, or even collision with surface vehicles. Furthermore, when AUVs have to operate in very deep waters – as for the MH370 aircraft search – the process of surfacing takes time and energy. Finally, other applications such as exploring ice-covered oceans or karst environments [Lasbouygues et al., 2014] will necessarily require other approaches to perform a long-term localization.

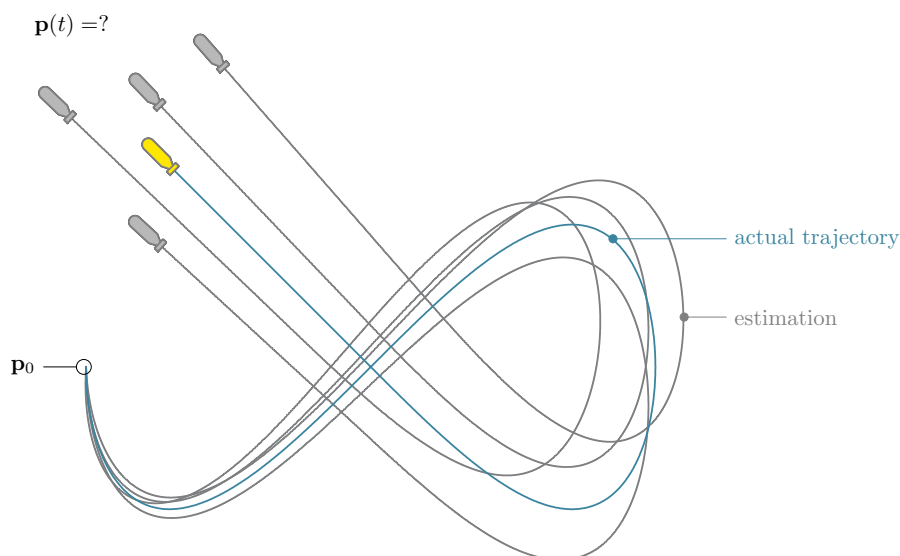


Figure 1.8: Illustration of drifting state estimations with a dead-reckoning method. From a known initial position \mathbf{p}_0 , a dead-reckoning method will integrate speed and inertial measurements which lead to cumulative errors with time. Successive actual positions are plotted in blue \bullet and four arbitrary estimations are drawn in gray \bullet .

1.2.3 Underwater acoustic positioning systems

This introduction is not aimed at providing a comprehensive list of dead-reckoning alternatives, but the two following acoustic techniques are widespread enough to be referenced. They will be mentioned afterwards in this document, as part of simulated examples or as a basis of actual datasets.

Basics

Acoustic positioning systems stand on the measurement of signals' time of flight. Distances between an emitter and a receiver (or an obstacle reflecting the wave) can then be computed assuming a good estimation of the sound velocity profile through the water column¹². The *Heard Island* experiment demonstrated that great distances can be reached by acoustics, and even assessed if the sound celerity

¹²The sound celerity profile mainly depends on pressure, salinity and temperature: parameters not always well known and that have to be measured *in situ*. In salt water, the sound travels at about 1500m/s.

profile is sufficiently known along the propagation. However, estimations get complicated in the vicinity of interfaces of two different media, namely water/seabed or water/surface, due to the reflection of waves and multipath interferences. Robust filters have to be used in order to overcome these outliers [Vaganay et al., 1996].

Positioning systems involve acoustic beacons in different configurations, two of which are presented hereinafter. The reader interested in this topic can refer to the literature for additional information [Jensen et al., 2011, Milne, 1983].

Long baseline acoustic positioning systems

A Long BaseLine (LBL) system is made of an array of acoustic transponders deployed on the seabed and precisely geolocalized. The vehicle is also equipped with a transponder in order to trigger the emission of signals from the beacons and receive the feedback. Figure 1.9 illustrates a typical LBL installation.

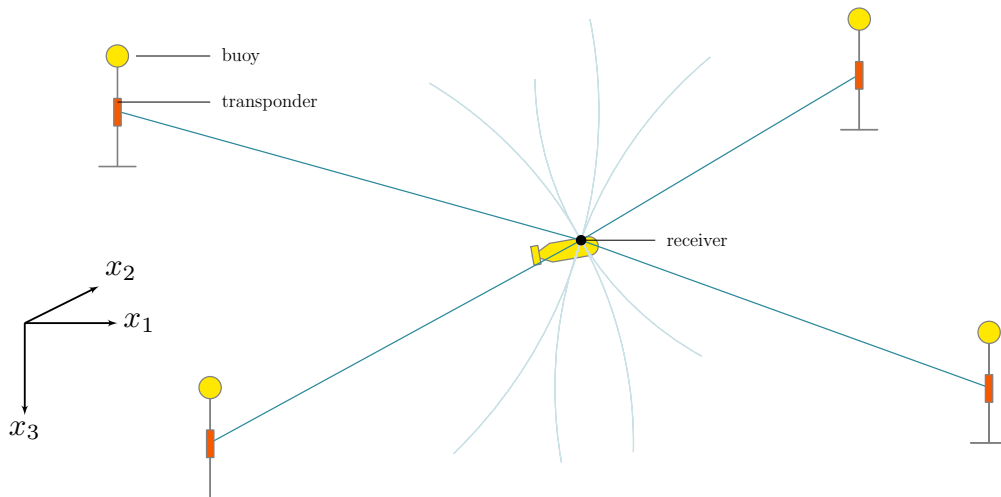


Figure 1.9: A LBL navigation system made of four hydrophones. The vehicle receives range-only signals and then estimates its position.

A received signal is *range-only* as it consists in a wave emission that propagates spherically from the emitter. The *bearing* information (direction-of-arrival data) is not assessed with a single beacon. The position of the vehicle is then estimated at the intersection of the spheres centered on the beacons, each of which with a radius measured from the sound time of flight. The transponder installation can be up to few kilometers wide while providing a positioning accuracy of a few meters.

Of course the deployment of such installation may be expensive, not to mention the duration of the calibration phase that can involve a complete crew. Furthermore, the installation is not suited for wide explorations of several tens or hundreds of kilometers.

Ultra-short baselines

A more mobile approach is the Ultra-Short BaseLine (USBL): a concentrated array of transceivers mounted on the very same device [Pennec, 2010], see for instance the system pictured in Figure 1.10a that has been used during some of the experiments presented in this document.

The device includes a set of acoustic transceivers providing both range and bearing measurements between the USBL and the receiver unit embedded in the vehicle, Figure 1.10b. In this approach, the calibration is made once by the manufacturer, allowing a straightforward use of the device. Furthermore, it can be settled under a boat localized with an accurate GNSS¹³. The combination of these two positioning systems enables the estimation of vehicle's absolute positions with an accuracy of one meter or below. However, due to the proximity of the transceivers, the angular accuracy may not be sufficiently good to localize a distant vehicle. In addition, the transceivers will usually be located near the surface, suffering from several outliers as pictured in Figure 1.11.

In practice, these devices are well suited for local underwater operations such as docking procedures or when a boat has to monitor an AUV. They cannot be considered as a standalone solution for pure autonomous navigation.

Towards dynamical positioning systems

Recent years have witnessed a new approach for underwater localization involving groups of vehicles and collaborative positioning. Range-only beacons or USBL can be mounted on AUVs in order to follow the exploration progress.

Autonomous vehicles can make use of acoustics to communicate low-rate data and exchange information such as state estimations. Then, new algorithms [Bahr

¹³Some devices such as the *GAPS*, Figure 1.10a, also include a fiber-optic INS to take into consideration the attitude of the boat whatever the ocean surface conditions.



(a) USBL mounted on the mission boat. This device is made of four transceivers.



(b) The receiver unit is embedded on top of the AUV among other sensors.

Figure 1.10: A USBL transceiver *GAPS* from iXblue used during sea experiments with the *Daurade* AUV. The device is mounted under the boat and provides an estimation of the actual AUV trajectory. Photos: *S. Rohou*.

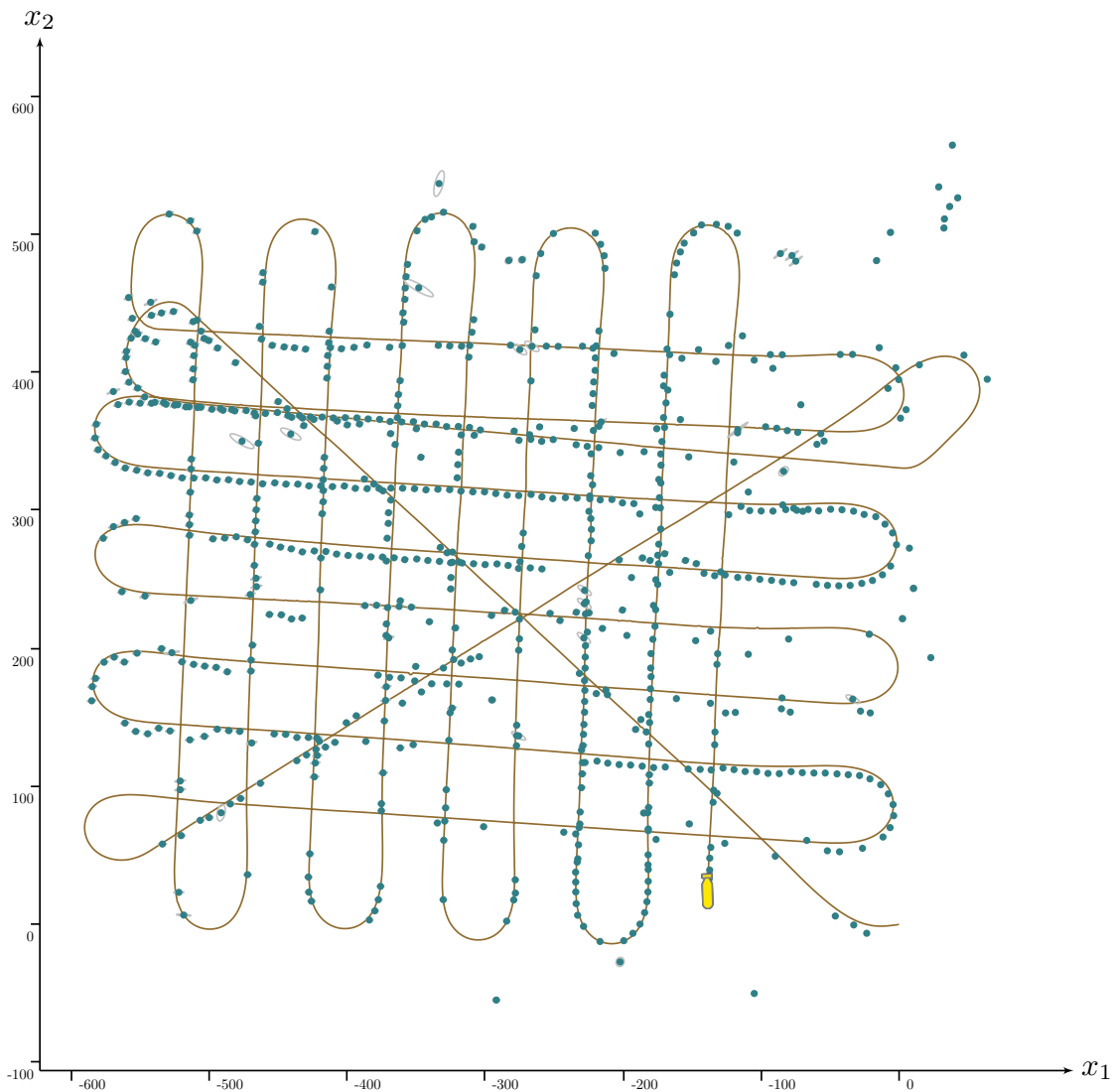


Figure 1.11: An overview of 2D positioning results obtained with a USBL system during an experiment involving the *Daurade* AUV. Non-filtered acoustic data are pictured by dots \bullet while the filtered trajectory, obtained with a Kalman filter and proprioceptive measurements, is plotted with a continuous line \bullet . Numerous outliers and low ping frequency are the main drawbacks of such system. Note that only a part of the received signals are pictured on this figure, other points are serious outliers located out of the survey area.

et al., 2009, Paull et al., 2014, Seddik, 2015] are employed to perform a decentralized localization of the group items. In this context, AUVs may even play specific roles in order to diversify the data obtained by the group. Namely, some items can stay at the surface to benefit from GNSS signals and assist the deeper vehicles that localize themselves and explore the seabed. In the same approach, a reconfigurable LBL has also been studied in [Matsuda et al., 2012, Matsuda et al., 2015], introducing the concept of alternating landmark navigation: a technique for which USBLs lay on the seabed as fixed points or explore while being localized by the motionless vehicles. This can be seen as a step by step approach where each step is performed by a vehicle laid on the seabed.

While these works achieve promising results, they also raise other difficulties such as the saturation of the acoustic channel, as messages are broadcasted in the same environment.

1.2.4 SLAM: a standalone solution

Another approach for robot localization has received large attention since the early stages in this field [Smith et al., 1990]. The Simultaneous Localization And Mapping (SLAM)¹⁴, is an approach that ties together the problem of state estimation and the one of mapping an unknown environment.

A chicken and egg problem

We have seen that a dead-reckoning localization necessary leads to uncertain positioning estimations with time. A robot exploring its surroundings will associate these uncertainties to the observed features, assigning their location with some error. However, a scene of the environment may be seen several times during the exploration, which leads to an *inter-temporal* measurement which could benefit both localization and mapping procedures. Indeed, a robot that recognizes a part of the environment will deduce to be close to a previous position. This is highlighted by the example presented in Figure 1.12.

Hence, these methods consider that positioning and mapping errors are closely related: a concurrent resolution may apply [Leonard and Durrant-Whyte, 1991].

¹⁴In the literature, SLAM is sometimes referenced as CML: Concurrent Mapping and Localization.

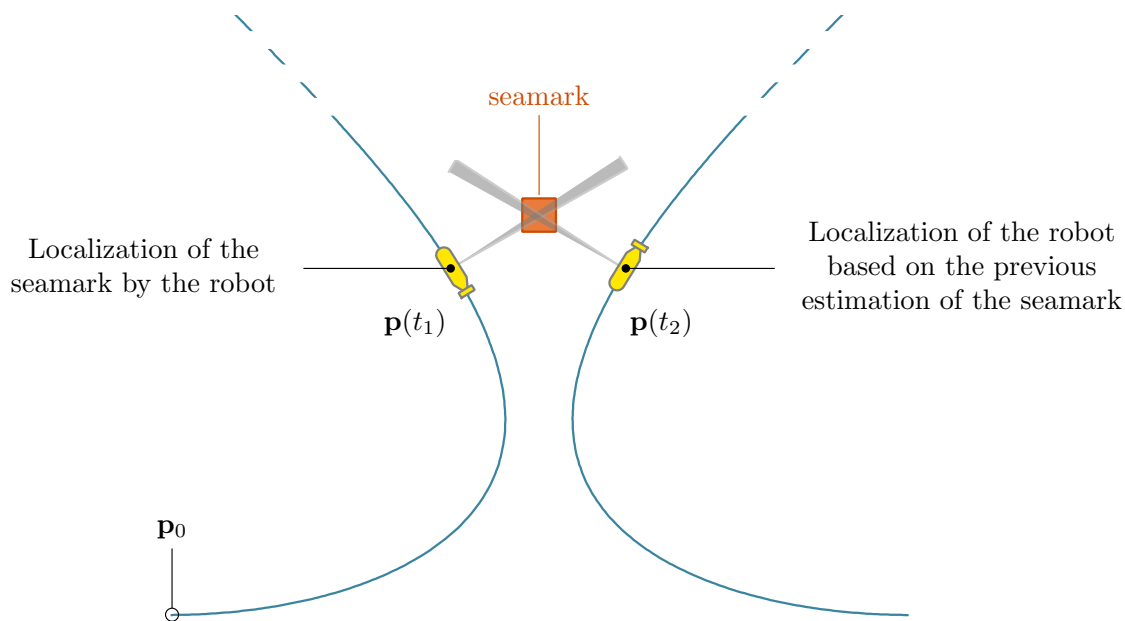


Figure 1.12: A simple SLAM illustration presenting robot's positions at two different times t_1 and t_2 . In pure dead-reckoning, the left part of the image would depict a precise estimation of the robot's trajectory while the positioning uncertainties would be strong on the right one. Considering a SLAM approach, a robot coming back to a previous place and perceiving again an object, such as a seamark, will be able to refine its state estimation.

A significant amount of work has been completed around this topic, still subject to further research [Newman and Leonard, 2003, Lemaire et al., 2007].

Loop closure

The key point of SLAM methods is to detect that a place has been previously visited. This problem is known in the literature as *loop closure*.

It may be difficult for a robot to detect a closure, due to poor estimations on both its position and map-matchings. Worse still, two different objects of same shape may be considered as unique by algorithms standing on too uncertain positioning estimations. Figure 1.13 highlights the case of two identical objects and uncertain trajectory estimates.

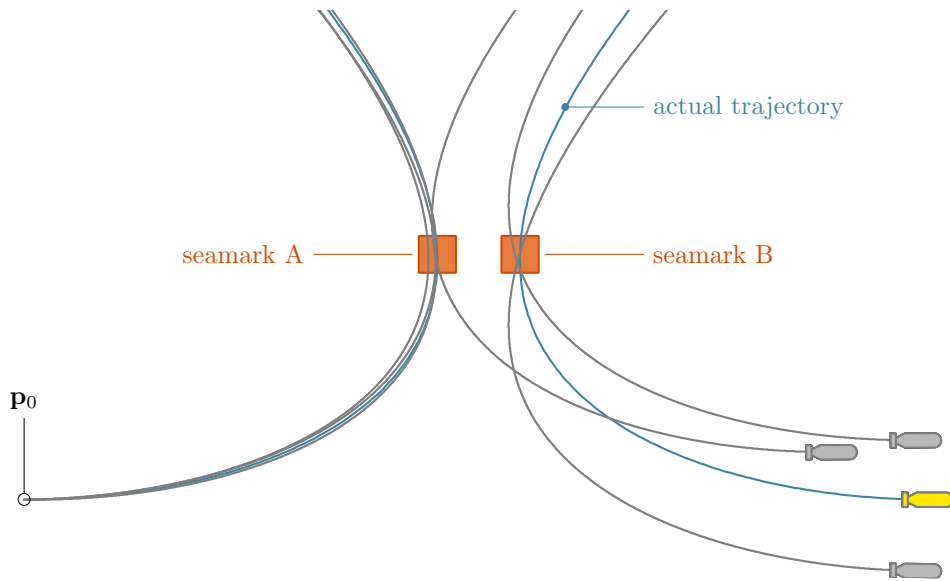


Figure 1.13: A robot flying over two different but same-looking seamarks. The actual trajectory is plotted in blue • while several dead-reckoning estimations are drawn in gray •. All the trajectories are consistent with the observations. A well-known map would not prevent from wrong detections.

This problem is not fully resolved yet. One of the contributions of this thesis is to provide a test to prove that a robot performed a looped trajectory, considering uncertainties from a set of proprioceptive measurements. This work will be presented in Chapter 6 and illustrated in the underwater case through actual experiments involving the previously introduced AUVs. The tool is well suited to prevent from false loop detections in similar environments. It has also other uses such as the reduction of the computational burden of SLAM algorithms.

Computational burden

The complexity of SLAM algorithms quickly increases with the exploration of wide environments, as it implies lots of loop closures to identify among a dense set of data. To this day, the execution of SLAM programs in 3D environments is often not affordable for classical embedded systems powering the robots. A part of the community hence focuses on light-weight solutions, sometimes at the expense of the loss of data associations.

Algorithms involving proprioceptive measurements only are now able to detect

loops without going into a costly analysis of observation datasets [Aubry et al., 2013]. This approach significantly reduces the complexity of SLAM algorithms.

Homogeneous environments

Another challenging issue is the identification of points of interest. Aside from the problem of confusable similar scenes, it is relatively straightforward to recognize artificial objects in terrestrial environments, exploiting high definition cameras and clear visibility of the scenes. A set of ready-to-use image processing libraries already provides efficient results for such applications.

However, points of interest are less identifiable in natural environments. It then requires raw-data approaches that do not stand on the identification of objects. Dealing with underwater environments, globally homogeneous in shape and observable with poor visibility, the only SLAM methods left to the roboticist are raw-data approaches. This thesis provides an original method considering these constraints.

1.3 PhD thesis context

“The necessary knowledge is that of what to observe.”

Complete Tales & Poems, Edgar Allan Poe

What characterizes the underwater case is the paucity of relevant information. This thesis focuses on a new localization approach in such poor environments. The presented method can be characterized as a raw-data SLAM approach, but we propose a temporal resolution – which differs from usual methods – by considering time as a standard variable to be estimated. This concept raises new opportunities for state estimation, under-exploited so far.

However, such temporal resolution is not straightforward and requires a set of theoretical tools in order to achieve the main purpose of localization. This thesis is thus not only a contribution in the field of mobile robotics; it also provides new perspectives in the areas of constraint programming and set-membership approaches.

This section briefly presents the proposed localization method and highlights the intermediate steps investigated during this work, providing the reader an overview of the document structure.

1.3.1 New localization approach in very poor environments

Assumptions

By *very poor*, we mean environments that do not present land/sea marks or any visible object that could be used as reference. A wide seabed area without recognizable objects such as anchors or wrecks corresponds to such poor environment. By going further, we will even assume that the observation function \mathbf{g} is unknown due to too much uncertainties about the environment. Furthermore, we will only consider a static environment which does not evolve during the exploration mission in a non-deterministic manner. Any measurably change of it must be formally known, for instance based on physical models.

Inter-temporal approach

Standing on the static environment assumption, a robot coming back to a previous position $\mathbf{p} = \mathbf{x}_{1,2}$ must sense the same observation \mathbf{z} as the first time. Formally,

$$\mathbf{p}(t_1) = \mathbf{p}(t_2) \implies \mathbf{z}(t_1) = \mathbf{z}(t_2). \quad (1.4)$$

We propose the following generic formalism:

$$\left\{ \begin{array}{l} \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \\ \mathbf{z}(t) = \mathbf{g}(\mathbf{x}(t)), \\ \underbrace{\mathbf{h}(\mathbf{x}(t_1)) = \mathbf{h}(\mathbf{x}(t_2))}_{\text{same state configurations}} \implies \underbrace{\mathbf{z}(t_1) = \mathbf{z}(t_2)}_{\text{same observations}}, \end{array} \right. \quad \begin{array}{l} (1.5a) \\ (1.5b) \\ (1.5c) \end{array}$$

introducing the *configuration* function $\mathbf{h} : \mathbb{R}^n \rightarrow \mathbb{R}^{n'}$ that depicts a singular configuration of a state. When same configurations are encountered twice at times t_1 and t_2 , then $\mathbf{x}(t_1)$ and $\mathbf{x}(t_2)$ present a singular relation leading to identical measurements. In this document, we will only focus on the positioning components $\mathbf{p} = (x_1, x_2)^\top$ of the state vector, but Equation (1.5c) allows wider applications¹⁵. We emphasize that the observation function \mathbf{g} is not involved in this formalism.

Our illustrations will stand on scalar bathymetric measurements acquired from a single beam echosounder measuring the altitude of the AUV over the seafloor. Figure 1.14 provides a synthesis image of an underwater robot crossing back a previous position. Here, the robot is regulated at constant depth and senses its altitude with a sonar. We will see that other kind of observations can be considered, such as the passive electric sense.

Equation (1.5c) provides a new relation between robot states and observations, as the function \mathbf{g} did. The difference is, however, that it links information that can be temporally very distant, even if an analytical expression between \mathbf{x} and \mathbf{z} is not at hand. According to the resolution method employed to solve the problem, the time references t_1 and t_2 could also present uncertainties.

To keep things simple, errors are not represented in Equations (1.5). It is a simple formalism and each resolution method – such as a Bayesian approach or a set-membership estimation – will model the uncertainties in its own way.

¹⁵For instance, the method should also apply in environments presenting symmetry properties. This has been the object of a study *Robot localization in an unknown but symmetric environment* during this thesis, not detailed in this document.

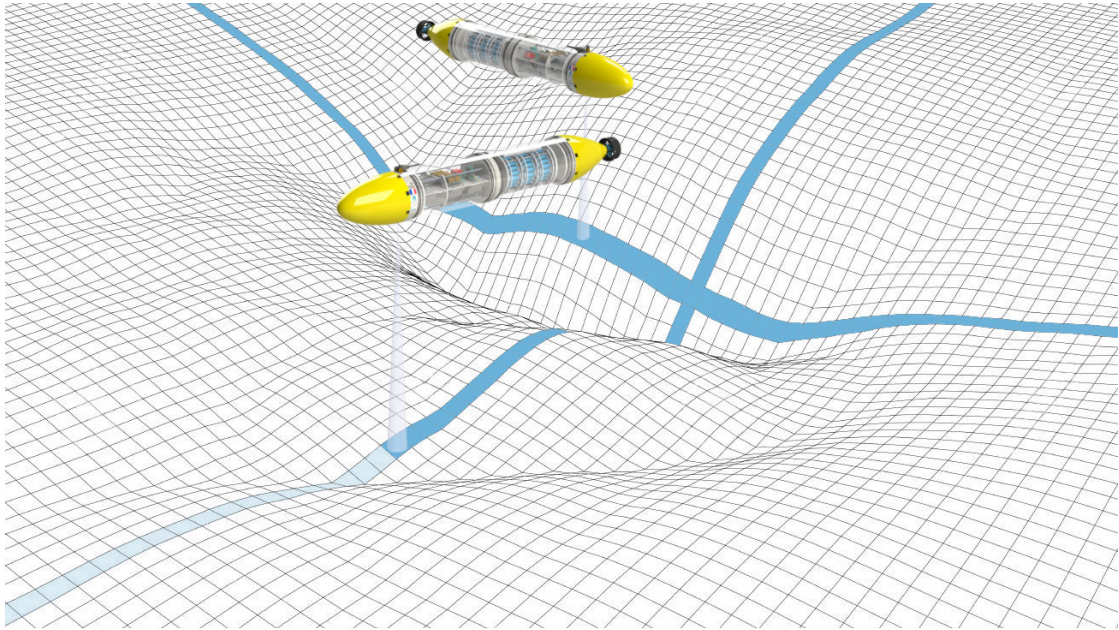


Figure 1.14: A *Toutatis* AUV exploring its surroundings with a single beam echosounder. This view presents two instants of the mission, before and after performing a loop. The localization approach defended in this document is based on the *constraint* raised during the trajectory crossing, where observations should be identical despite their temporal distancing.

1.3.2 Estimation methods

The localization problem modeled by the System (1.5) involves differential equations and functions that can be non-linear. Furthermore, uncertainties have to be propagated over the evolution of the system. As inter-temporal measurements will be the core of the issue, the effects of these propagations over the time intervals must be assessable.

Exact resolution methods cannot apply for this problem as it does not present analytical solutions. Estimation approaches have to be considered. This section presents our motivation for set-membership methods.

Probabilistic approaches

These usual methods compute a unique estimated solution with uncertainties qualified by means of covariance matrices [Papoulis and Pillai, 2002]. When dealing with linear equations and uncertainties that are Gaussian-distributed, it has been shown that the well-known Kalman filter [Kalman, 1960] is optimal. Extensions to the non-linear case have been studied afterwards, performing linearizations when necessary. However, this might be a source of estimation errors and it becomes difficult to qualify such uncertainty.

On the other hand, stochastic approaches have received a large attention from the community of automatic and control, providing significant results in the non-linear case [Thrun et al., 2005]. These methods randomly sweep across feasible inputs or parameters and generate a set of probable trajectories: samples. To increase the chances to have one estimation among the samples that is close to the actual solution, a lot of computations have to be attempted. Their probability is evaluated and further samples are then generated in the probable *areas* based on these likelihoods, allowing a convergence of the algorithm towards a relevant estimation.

However, these random-based computations may badly behave in case of strong non-linearities or few available observations. The risk is to assign a high likelihood to a wrong solution and let the algorithm converge to it. In our localization context, it is essential to address the problem with an estimation method robust to a lack of data and non-redundant information. Probabilistic methods do not seem suited for this purpose.

Set-membership methods

We will pay a specific attention to set-membership approaches that have proved their worth to deal with non-linearities and substantial uncertainties.

“I think it is much more interesting to live not knowing than to have answers which might be wrong. I have approximate answers and possible beliefs and different degrees of certainty about different things, but I am not absolutely sure of anything and there are many things I do not know anything about.”

Richard Feynman, *BBC Horizon*, 1981

The approach effectively does differ from probabilistic methods in regards of the nature of the estimated solution. Probabilistic methods compute a punctual potential solution – for instance a vector – while in set-membership ones, it is the set of all feasible solutions that is evaluated, and thus an infinity of potential solutions. Another main distinction lies in the way things are computed: with set-membership methods, estimations are not randomly performed. Computations are *deterministic*: given a set of parameters or inputs, algorithms will always output the same result.

These methods stand on reliable computations over bounds defining ranges of possibilities. Operations are guaranteed not to lose any solution. The main counterpart is that any unlikely solution will be kept in the resulting set if it is consistent with the system’s equations. As a result, algorithms provide pessimistic outcomes and sometimes even meaningless results depending on the situation: “*I am not absolutely sure of anything*”.

The maximum distance between the unknown actual solution and any point in the output set is computable and defines the quality of the approximation: it is the worst case error if any point in the set is taken as solution. Therefore, in contrast to the above-mentioned approach, these methods are well suited to reliably propagate uncertainties over the operations.

Conclusion

The problem we are dealing with presents only few observations, especially non-redundant. Capitalize upon them is the key and an accurate resolution method is necessary to never diverge from this information. This is the reason why a set-membership approach is considered in this document. Furthermore, in our underwater applications, uncertainties are strong and non-linearities omnipresent, due to range-only observations: these are situations easily managed by this approach.

In addition, using set-membership methods will provide guaranteed outcomes, which can be of main interest for the safety of robotic systems [Goubault et al., 2014, Monnet et al., 2016].

Last, but not least, this SLAM problem will be solved in a temporal way, by comparing observations that are temporally distant and considering time references as complete variables. It seems that only a strict and reliable approach can suit. However, set-membership methods do not provide the necessary theoretical tools

yet. One of the purposes of this thesis is to extend the methods in order to deal with temporal relations, and apply the proposed tools on our localization problem.

To achieve this resolution, a constraint based approach will be applied. The work motivated by our robotic problem will lead to contributions in the communities of constraint programming and robotics. The theoretical contributions related to the constraints field are briefly motivated in the following section.

1.3.3 Constraint programming approach over dynamical systems

We have seen that the aim of set-membership approaches is to define a reliable set of feasible solutions. This strategy goes well with constraint propagation techniques: another area widely explored since the 1980's by a part of the artificial intelligence community [Cleary, 1987, Sam-Haroud and Faltings, 1996]. In particular, we will concentrate on continuous constraints and propose tools to implement new ones in a differential context.

Constraint programming

The constraint programming aims at solving a complex problem by defining it in the form of elementary facts and rules among variables: so-called *constraints*. A constraint is understood as the expression of any relation that binds variables, which are known to belong to some domains. In our context, constraints may be equalities between physical values as well as non-linear equations, inequalities, or quantified parameters. For instance, the programmer addressing the problem of Equations (1.5), page 25, will list a set of mathematical constraints and will then build a *solver*. Uncertainties and spaces of solutions are specified either by other constraints or by restricting the domains of the variables. Hence, this approach is in perfect accordance with set-membership methods, benefiting from their reliable operations to apply constraints over sets of values.

In this approach, instead of thinking about *how* he can solve a problem, the developer will focus on *what* is the problem, thus leaving the computer to the question of the *how*. Indeed, each elementary constraint will then be implemented as a black box that does not require any configuration. Constraints can also be easily combined in order to increase in complexity, while preserving simplicity.

Therefore, the strength of this declarative paradigm lies in its simpleness, as it allows one to describe a problem with elementary facts that do not require the knowledge of resolution tools coming with specific parameters to choose. The second asset is its genericity: a situation is seen from a high-level point of view and this abstraction enables the resolution of a wide range of problems. The *Prolog* language [Benhamou and Touraïvane, 1995] appears to be the best known illustration of a general-purpose logic programming method. However, the main drawback of these languages may be their lack of efficiency: the simplicity of the modeling step comes at the expense of a lack of control on the resolution process. Improving the efficiency of such solver by a set of parameters to configure will always compromise simplicity and adaptability.

A major effort from the community has been undertaken around this concept, providing an extensive constraint store. However, the approach must be extended to dynamical systems in order to deal with problems such as state estimations encountered in robotics.

Extension to dynamical systems

There is a need to enrich the constraint store with new tools to manage differential equations such as $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t))$. The objective is to adapt the constraint programming approach to dynamical systems and propose a resolution in the temporal space. Further constraints such as inter-temporal equations should also be considered, in order to address relations such as $\mathbf{p}(t_1) = \mathbf{p}(t_2) \implies \mathbf{z}(t_1) = \mathbf{z}(t_2)$. Naturally, this effort has to be done while maintaining the assets of set-membership methods and constraint programming, namely: guaranteed results, non-linearities control and simplicity.

Some attempts in this direction have been proposed, but they do not provide the level of simplicity and genericity we may be looking for when dealing with concrete applications: non-analytical models, asynchronous observations, unknown initial conditions, inter-temporal measurements, backward refinements, to name but a few.

This thesis goes a step further and proposes a reliable constraint programming approach over dynamical systems, illustrated with concrete applications. The approach defended in this document consists in considering *trajectories* as variables and apply differential constraints on them. Ultimately, a set of heterogeneous tools will be available, merging classical continuous constraints together with

differential ones. The domains of feasible solutions will be represented by means of *tubes*: an interval object that recently appeared in this community [Le Bars et al., 2012, Bethencourt and Jaulin, 2014], see Figure 1.15.

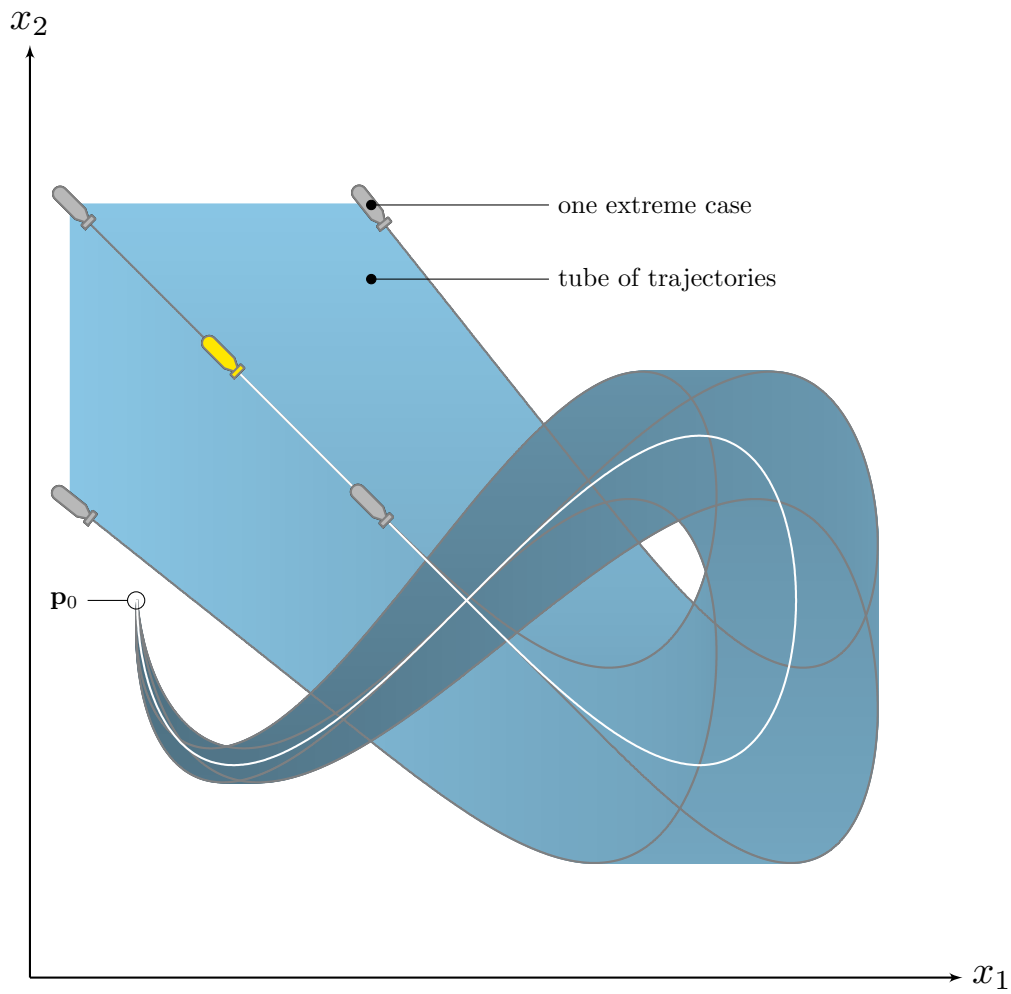


Figure 1.15: First illustration of a *tube* of robot trajectories: the building block of the contributions presented in this document. The bounds of the tube are worst-case trajectories; this representation allows the enclosure of an actual trajectory in a representable set of dynamical solutions.

1.3.4 Thesis outlines and contributions

Towards a reliable raw-data SLAM method

In this thesis, the variables of interest are both time references and trajectories. By adopting a set-membership approach, time variables will be handled by means of *intervals* introduced in Chapter 2 while trajectories will lie within *tubes* presented in Chapter 3. The aim will be to reduce these sets using proprioceptive measurements and raw-data observations such as bathymetric sensing. This will be achieved by means of two new reliable operators: *contractors* on tubes and intervals presented in the second part of this manuscript: Chapters 4 and 5.

The third part of the thesis will focus on Equation (1.5c), page 25, in the context of 2D looped trajectories:

$$\underbrace{\mathbf{h}(\mathbf{x}(t_1)) = \mathbf{h}(\mathbf{x}(t_2))}_{\mathbf{p}(t_1) = \mathbf{p}(t_2)} \implies \mathbf{z}(t_1) = \mathbf{z}(t_2). \quad (1.6)$$

The reliable assessment of $\mathbf{p}(t_1) = \mathbf{p}(t_2)$ is not straightforward in an uncertain context. Figure 1.16 illustrates the issue with wrong estimations and doubtful loops. We then need a tool to verify that a trajectory crosses itself at some point, whatever the uncertainties describing it. This problem is addressed in Chapter 6. Ultimately, the new SLAM method will be discussed in Chapter 7 with concrete applications.

We claim that all the algorithms and computations presented in this document are guaranteed not to lose solutions, from a mathematical point of view. More precisely, we will see that we do not have to consider an infinitely small integration timestep for the resolution of differential equations to make this statement.

Document structure

The contributions fields are twofold: we will develop theoretical tools related to the constraint programming approach and then propose a new localization method for mobile robots, standing on these tools. Each of the intermediate steps investigated during this work will be the subject of a chapter. The overall document structure is reminded hereinafter.

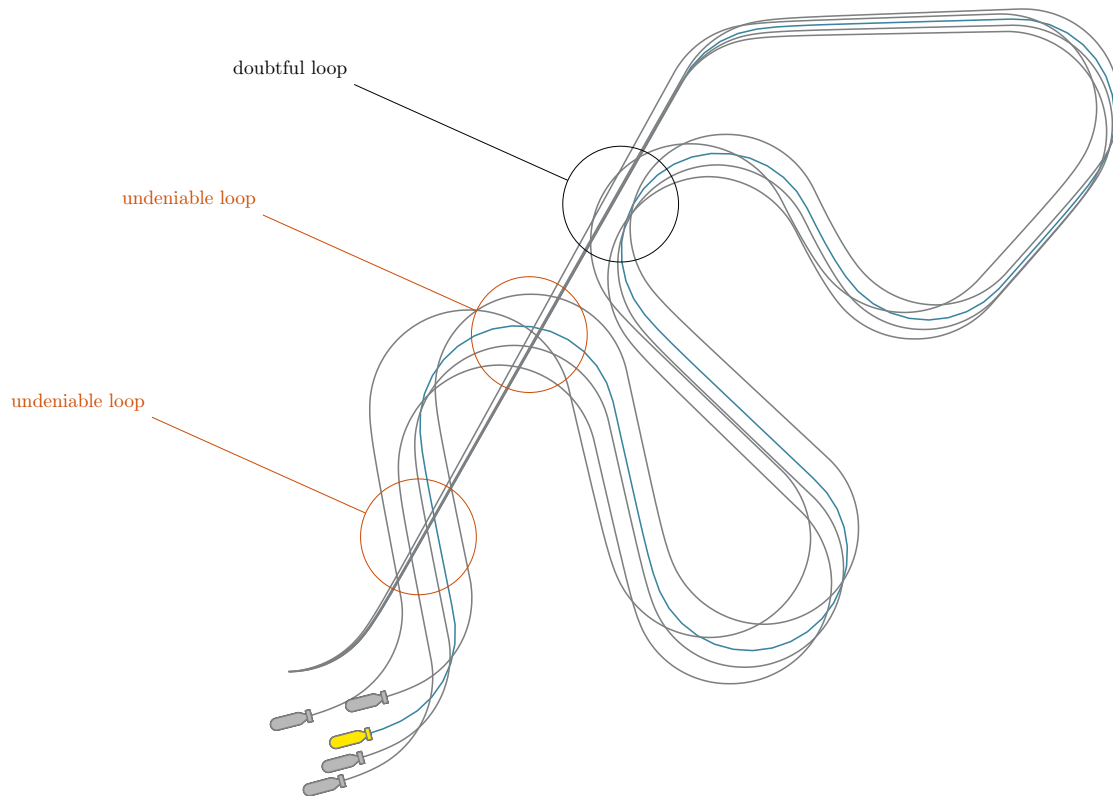


Figure 1.16: Loop detections based on uncertain state estimations. The actual trajectory involves two loops while the four estimations may present between two and four crossings. Methods to identify doubtful loops from undeniable cases have to be studied.

Part I Interval tools

Chapter 2 Its purpose is to make the reader familiar with the set-membership tools that are used later in the manuscript. A simple static range-only localization example will be used as common thread of this chapter, illustrating the concepts of interval analysis, constraints, contractors and set-inversion.

Chapter 3 It introduces the notion of tubes with their related properties, algebraic contractors and implementation choices. The previous static example is extended to the dynamical context with a reliable dead-reckoning estimation.

Part II Constraints-related contributions

Chapter 4 Our first contribution focuses on the primary differential constraint $\dot{x}(\cdot) = v(\cdot)$. We propose a new tube contractor $\mathcal{C}_{\frac{d}{dt}}$ with proofs and algorithms. This first step will allow to deal with complete state estimations based on asynchronous observations.

Chapter 5 This second theoretical contribution will complement the first one, focusing on the *evaluation* constraint $z = y(t)$. The efficiency of the new contractor $\mathcal{C}_{\text{eval}}$ will be illustrated on two examples: the dynamic localization of a mobile robot and the correction of a drifting clock.

Part III Robotics-related contributions

Chapter 6 Our third contribution is an appropriate test to prove that a robot performed a loop whatever the uncertainties in its evolution. Our approach stands on the topological degree theory applied through interval functions evaluated from tubes. Convincing results have been obtained during actual AUVs experiments.

Chapter 7 The culmination of this thesis is an original reliable raw-data SLAM method. The constraint-based resolution we propose is new and achieved based on each of the previous contributions. We show the interest of the approach through actual experiments involving the *Daurade* AUV. This temporal process is a first step towards new opportunities of localization, under-exploited so far.

Chapter 8 will finally conclude this thesis and discuss perspectives.

Part I

Interval tools

Our choice is to address robotics problems with set-membership methods. In this part, we introduce a set of already existing tools standing on interval analysis, illustrated over several simple examples of dead-reckoning and range-only state estimation. Each notion developed in this part will then be applied in the following chapters.

Chapter 2 introduces the concepts of interval analysis, constraint programming, contractors and set-inversion. Chapter 3 extends the approach to temporal systems by providing the notion of tubes with their related properties, algebraic contractors and implementation choices.

Static set-membership state estimation

Contents

2.1	Introduction	38
2.2	Interval analysis	41
2.2.1	Once upon a time	41
2.2.2	Intervals	42
2.2.3	Inclusion functions	47
2.2.4	Pessimism and wrapping effect	49
2.3	Constraints propagation	52
2.3.1	Constraint networks	52
2.3.2	Contractors	54
2.3.3	Application to static range-only robot localization	57
2.4	Set-inversion <i>via</i> interval analysis	60
2.4.1	Subpaving	60
2.4.2	SIVIA algorithm for set-inversion	61
2.4.3	Illustration involving contractions	65
2.4.4	Kernel characterization of an interval function	65
2.5	Discussions	69
2.5.1	From sensors to reliable results	69
2.5.2	Numerical libraries	70
2.5.3	Reliable tool for proof purposes	71
2.6	Conclusion	71

2.1 Introduction

A state estimation problem can be dealt with set-membership approaches, where both the uncertainties on the model and the measurements errors are known. By *known*, we mean that an unknown actual value is guaranteed to lie within bounds that delineate the uncertainties, thus defining a *solution set* [Walter and Piet-Lahanier, 1988, Cerone, 1996, Veres and Norton, 1996, Maksarov and Norton, 1996].

The estimation then consists in reducing this set of feasible values by means of operators, where other approaches would have minimized an error criterion. The computations are not performed in a probabilistic way but based on deterministic operations on the bounds of the set. This approach is significantly different from usual methods: here, we do not assume a probability distribution in the calculations. Furthermore, the bounding property of such approach is guaranteed even if the system is non-linear. This quality is of particular importance in mobile robotics where several problems present non-linearities, the case of a range-only localization being one of them [Caiti et al., 2005].

Range-only state estimation

A telling example of set-membership state estimation is the localization of a mobile robot among beacons. This kind of application has already been appropriately presented in [Drevelle, 2011] to introduce set-membership methods. We will use it as a guiding thread of this document, firstly in this chapter in a static context without considering temporal evolutions: the pose of the robot is estimated with synchronous measurements. The same example will be extended to dynamical state estimation in Chapters 3, 4 and 5 based on differential state equations and asynchronous measurements, sometimes obtained with time uncertainties.

A robot \mathcal{R} is described by its state $\mathbf{x} = (x_1, x_2)^\top$ depicting its 2D location. The robot evolves among a set of beacons \mathcal{B}_k located at (x_1^k, x_2^k) and sending synchronously acoustic signals, as illustrated by Figure 2.1. The distances ρ_k from the \mathcal{B}_k will be used to localize \mathcal{R} . The following observation function is used to this end:

$$\rho_k = g_k(\mathbf{x}) = \sqrt{(x_1 - x_1^k)^2 + (x_2 - x_2^k)^2}. \quad (2.1)$$

The robot is located on one of the intersections of the ranges circles. When at least three beacons are used, the intersection is generally a single point: the horizontal localization is done without ambiguity.

Wrappers

Now, considering uncertainties on the measurements, the circles become *thick* and the intersection then results in a set that can be of any shape, as pictured in Figure 2.1.

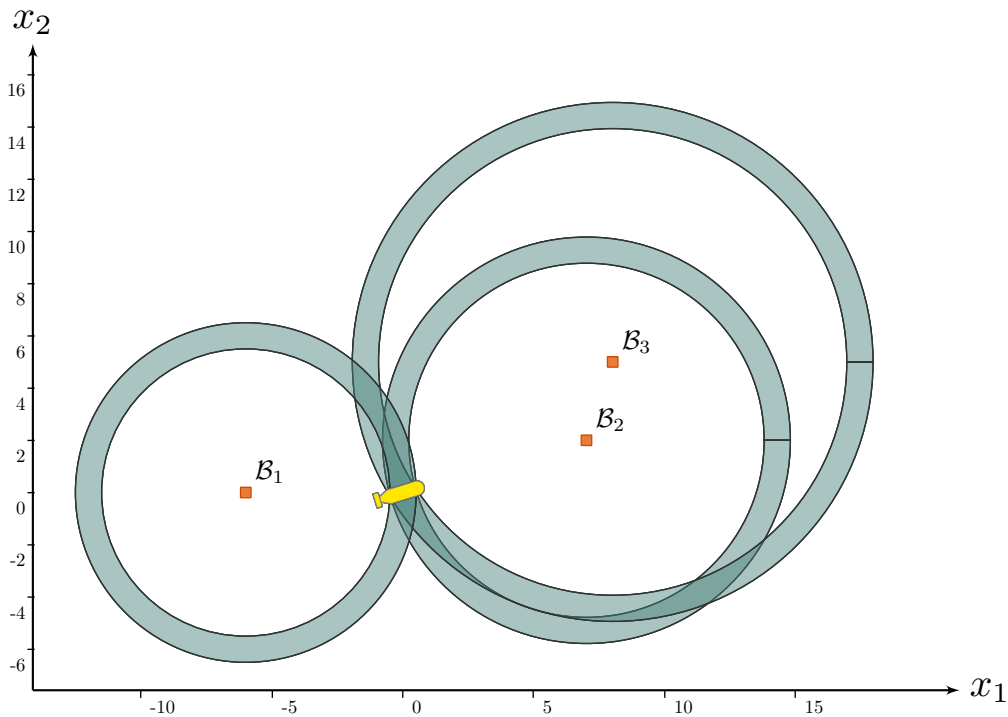


Figure 2.1: Range-only robot localization among three beacons pictured by red boxes \bullet . The position of \mathcal{R} in $(0, 0)$ is unknown. The state estimation is performed based on uncertain measurements displayed by rings \bullet . Several kinds of solution sets are shown in Figure 2.2.

The solution set may even be made of several connected subsets or include holes. Figure 2.2 illustrates different methods to represent a set, namely zonotopes or polyhedral enclosures [Combastel, 2005, Walter and Piet-Lahanier, 1989], ellipsoids [Rokityanskiy and Veres, 2005], intervals or subpavings [Jaulin and Walter, 1993a]. These latter two have been proven to be efficient when dealing with non-linear

systems or complex solution shapes. The current chapter presents their theoretical basis that will be used then throughout this document. Section 2.2 focuses on interval analysis and Section 2.3 presents tools named *contractors* that aim at reliably reducing bounds of interval sets. Subpavings (Figure 2.2d) are introduced in Section 2.4 for set-inversion, before discussions about implementation and concrete use of intervals in Section 2.5.

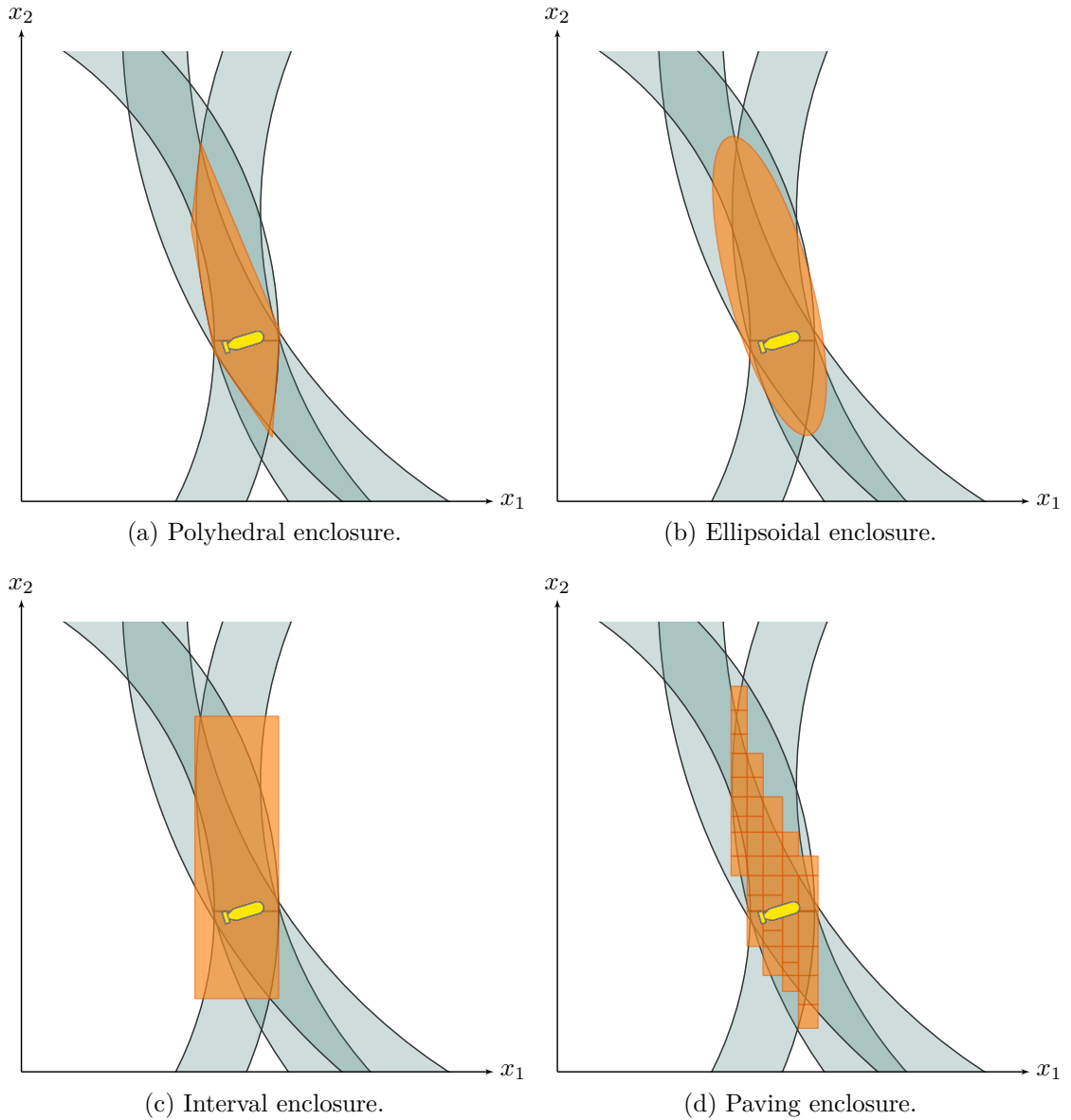


Figure 2.2: Reliable set-membership approaches to enclose the set of feasible positions of \mathcal{R} based on uncertain beacons measurements.

2.2 Interval analysis

2.2.1 Once upon a time

From the beginning of mathematics, the consideration of irrational numbers has raised the question of their decimal representation. Intervals appear to be a natural solution to provide an accurate approximation of a number with two bounds. For instance, Archimede calculated a reliable enclosure of π : $\frac{223}{71} < \pi < \frac{22}{7}$.

However, the spirit of interval analysis appeared recently with the advent of numerical computations, shining a new light on intervals. Using computers, real numbers of infinite precision are implemented by floating point values of finite precision. An approximation of the numbers is then made and can lead to increasing errors during the computations¹.

The following illustration has been given in [Rump, 1988]. Let us compute

$$f(x, y) = \frac{1335}{4}y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + 5.5y^8 + \frac{x}{2y} \quad (2.2)$$

with $x = 77617.0$ and $y = 33096.0$ that are exactly computer representable numbers. Almost *same* results are obtained using single, double and extended precisions: the first seven figures are equal.

single precision:	$f = 1.172603\dots$
double precision:	$f = 1.1726039400531\dots$
extended precision:	$f = 1.172603940053178\dots$

The exact value² $f^* = -0.827396059946821\frac{4}{3}$ shows how wrong the computations are, despite similar results among several levels of precision.

Using bounds to represent a value is of interest for computer representation, since these limits can be floating point numbers of finite precision while ensuring a guaranteed numerical representation of a real number such as π . The counterpart is pessimism, because this arithmetic handles ranges of possibilities in place of

¹Note that these errors may vary from one computer to another.

²This value has been computed by a reliable algorithm so that the displayed figures are guaranteed and the sixteenth known to yield between 3 and 4.

unique values. Hence, the evaluation of Equation (2.2) using interval arithmetic provides a large enclosure of the actual f^* , but guaranteed to contain it.

This modern need opened a new thread in mathematics. The first main book on this topic, [Moore, 1966], brought a tremendous impetus to interval analysis, putting intervals into new perspective: the uncertainties regarding floating point precision can be extended to physical uncertainties. This is of high interest for robotic applications dealing with measurements errors and unknown environments. We will show in this document that intervals can even be used to represent strong temporal uncertainties. Additionally, interval analysis is appropriate to compute a search space and lends itself for many robotic applications such as forward kinematics [Merlet, 2004], trajectory planning [Piazzi and Visioli, 1997] or workspace analysis [Chablat et al., 2002], to name but a few.

This thesis will not focus on floating point precision but strong physical uncertainties. And because all the computations are standing on rigorous interval analysis, we ensure the presented outcomes are numerically guaranteed. In this context, results can be used for proof purposes.

2.2.2 Intervals

Elementary notions about intervals are given in the following sections, all of them being used in the next chapters. For more information on interval analysis and its applications, the reader may refer to [Jaulin et al., 2001].

Basics

An interval $[x]$ is a closed and connected subset of \mathbb{R} . The set of all intervals is denoted \mathbb{IR} . An interval $[x]$ is delimited by a lower bound x^- and an upper one x^+ that can be infinite³:

$$[x] = [x^-, x^+] = \{x \in \mathbb{R} \mid x^- \leq x \leq x^+\}. \quad (2.3)$$

When $x^- = x^+$, the interval $[x]$ is said *degenerate*. In the following computations, any real number can be considered as a degenerate interval for the sake of generality.

³In this document, we will sometimes use the notation $\text{lb}([x]) = x^-$ (respectively $\text{ub}([x]) = x^+$) to denote the lower (upper) bound of $[x]$.

The same apply for the empty set \emptyset . In the problems we are dealing with, \emptyset will depict an absence of solution.

The following are interval examples:

- $[2, 3]$,
- $[5] = \{5\}$,
- $[-\infty, \infty]$,
- $[0, \infty]$,
- \emptyset .

In this document, an actual but unknown value to be approximated will be denoted by a star: x^* . Its estimation is expressed by $[x]$, the width of which depicts the uncertainty on x^* :

$$\text{width}([x]) = x^+ - x^-. \quad (2.4)$$

When some applications have to deal with a scalar value, the center of $[x]$ can be considered to represent an evaluation of the unknown x^* . The relevance of such choice will be discussed for tubes in Chapter 3, Section 3.5.2.

$$\text{mid}([x]) = \frac{x^- + x^+}{2}. \quad (2.5)$$

Set operations on intervals

The intersection of two intervals is an interval:

$$[x] \cap [y] = \{z \in \mathbb{R} \mid z \in [x] \text{ and } z \in [y]\}. \quad (2.6)$$

However, the union of two intervals may not be an interval:

$$[x] \cup [y] = \{z \in \mathbb{R} \mid z \in [x] \text{ or } z \in [y]\}. \quad (2.7)$$

The interval union is computed as the *interval hull* of $[x] \cup [y]$ so that the result of the union is a connected subset of \mathbb{R} . In this document, this union is denoted by $[x] \sqcup [y]$ and defined as

$$[x] \sqcup [y] = [[x] \cup [y]]. \quad (2.8)$$

Some specific cases are listed below:

$$[x] \sqcup \emptyset = [x], \quad (2.9)$$

$$[x] \cap \emptyset = \emptyset, \quad (2.10)$$

$$[x] \sqcup [-\infty, \infty] = [-\infty, \infty], \quad (2.11)$$

$$[x] \cap [-\infty, \infty] = [x]. \quad (2.12)$$

Interval computations

Interval analysis is based on the extension of all classical real arithmetic operators. Consider two intervals $[x]$ and $[y]$ and an operator $\diamond \in \{+, -, \cdot, /\}$. We define $[x] \diamond [y]$ as the smallest interval containing all feasible values for $x \diamond y$, assuming that $x \in [x]$ and $y \in [y]$, [Moore and Yang, 1959]:

$$[x] \diamond [y] = \{ \{x \diamond y \in \mathbb{R} \mid x \in [x], y \in [y]\} \}, \quad (2.13)$$

$$[x] \diamond \emptyset = \emptyset. \quad (2.14)$$

Dealing with closed intervals, most of the operations can rely on their bounds. It is for instance the case of addition, difference, union, *etc.*:

$$[x] + [y] = [x^- + y^-, x^+ + y^+], \quad (2.15)$$

$$[x] - [y] = [x^- - y^+, x^+ - y^-], \quad (2.16)$$

$$[x] \sqcup [y] = [\min(x^-, y^-), \max(x^+, y^+)], \quad (2.17)$$

$$\begin{aligned} [x] \cap [y] &= [\max(x^-, y^-), \min(x^+, y^+)] \text{ if } \max\{x^-, y^-\} \leq \min\{x^+, y^+\}, \\ &= \emptyset \text{ otherwise.} \end{aligned} \quad (2.18)$$

However, computing some operations is sometimes not as straightforward. Take

for instance the case of the division:

$$1/[y] = \begin{cases} \emptyset & \text{if } [y] = [0, 0] \\ [1/y^+, 1/y^-] & \text{if } 0 \notin [y] \\ [1/y^+, \infty] & \text{if } y^- = 0 \text{ and } y^+ > 0 \\ [-\infty, 1/y^-] & \text{if } y^- < 0 \text{ and } y^+ = 0 \\ [-\infty, \infty] & \text{if } y^- < 0 \text{ and } y^+ > 0 \end{cases}, \quad (2.19)$$

$$[x]/[y] = [x] \cdot (1/[y]). \quad (2.20)$$

This arithmetic extension also includes the adaptation of elementary functions such as \cos , \exp , \tan . Sometimes the image of an interval $[x]$ through a function f is not an interval, as it is the case for discontinuous functions. Then, the interval evaluation of $f([x])$, denoted $[f]([x])$, is the smallest interval containing all the images of all defined inputs through the function:

$$[f]([x]) = \{f(x) \mid x \in [x]\}. \quad (2.21)$$

When f is monotonic, $[f]([x])$ can be evaluated directly from its bounds:

$$[\exp]([x]) = [\exp(x^-), \exp(x^+)]. \quad (2.22)$$

Otherwise, other expressions or algorithms have to be used [Bouron, 2002]. The cosine function is a typical example of a non-monotonic function:

$$[\cos]([0, 2\pi]) = [-1, 1] \neq \underbrace{[\cos(0), \cos(2\pi)]}_{[1,1]}. \quad (2.23)$$

Generalization

One may consider to extend the notion of intervals of real numbers to other sets, such as intervals of functions, sets [Desrochers and Jaulin, 2017], booleans or even graphs [Jaulin, 2015b]. Besides, the main contribution of this thesis is to provide tools to deal with intervals of trajectories, that will be introduced in Chapter 3. The following section focuses on intervals of vectors.

Interval vectors

A Cartesian product of n intervals defines a *box* – also called *interval vector* – belonging to the set \mathbb{IR}^n . As for vectors \mathbf{x} , boxes will be represented in bold: $[\mathbf{x}]$.

$$\begin{aligned} [\mathbf{x}] &= [x_1] \times \cdots \times [x_n], \\ &= [x_1^-, x_1^+] \times \cdots \times [x_n^-, x_n^+], \\ &= ([x_1], \dots, [x_n])^\top. \end{aligned} \tag{2.24}$$

An interval vector $[\mathbf{x}]$ of \mathbb{IR}^n is an axis-aligned box, closed and connected subset of \mathbb{R}^n . The i -th component $[x_i]$ is therefore the projection of $[\mathbf{x}]$ onto the i -th axis, as depicted in Figure 2.3. The empty set of \mathbb{R}^n is $(\emptyset \times \cdots \times \emptyset)^\top$.

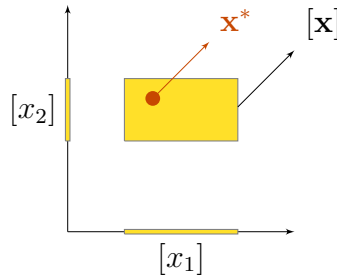


Figure 2.3: An interval vector $[\mathbf{x}] \in \mathbb{IR}^2$, its components $[x_i]$ being projections onto axes.

Most operations on intervals are easily scalable to boxes by performing computations on each component. For instance, a box $[\mathbf{x}]$ is defined by its bounds such that

$$\mathbf{x}^- = (x_1^-, \dots, x_n^-)^\top, \quad \mathbf{x}^+ = (x_1^+, \dots, x_n^+)^\top. \tag{2.25}$$

These extensions also apply to matrices of intervals, for which each component is an interval, as for boxes.

2.2.3 Inclusion functions

Definition

Considering a n -dimensional box $[\mathbf{x}]$ as input, a function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ will output a set which is not necessarily a box, as pictured in Figure 2.4 with an image made of non-connected subsets and a hole. Obtaining an accurate representation of the output set is often a complicated task, sometimes achieved with a computational burden of particular concern.

Instead, we use an *inclusion function* $[\mathbf{f}] : \mathbb{I}\mathbb{R}^n \rightarrow \mathbb{I}\mathbb{R}^m$ to enclose the image of $[\mathbf{x}]$ by \mathbf{f} in a box such that $\forall [\mathbf{x}] \in \mathbb{I}\mathbb{R}^n, \mathbf{f}([\mathbf{x}]) \subset [\mathbf{f}]([\mathbf{x}])$. Hence, a reliable enclosure of the image set can be evaluated reasonably quickly. Furthermore, inclusion functions can stand on analytical expressions or even algorithms based on datasets.

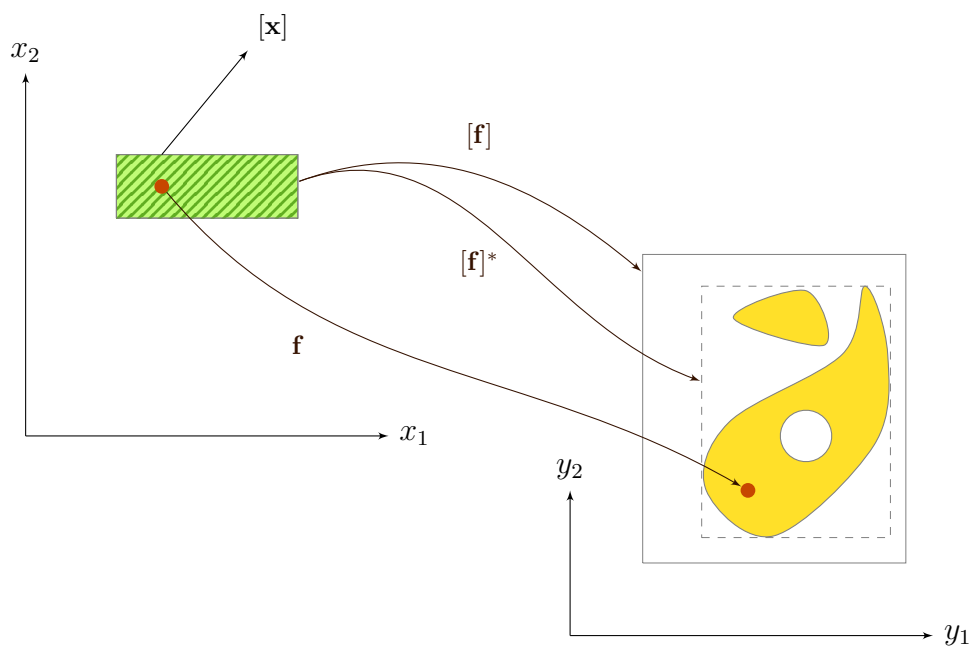


Figure 2.4: Inclusion functions: the image of a box $[\mathbf{x}]$ in green \bullet is an arbitrary set $\mathbf{f}([\mathbf{x}])$, pictured in yellow \bullet , that can be approximated with the *inclusion function* $[\mathbf{f}]$. The minimal inclusion function $[\mathbf{f}]^*$ provides the smallest enclosure of $\mathbf{f}([\mathbf{x}])$.

Properties

An inclusion function is *thin* if the image of any degenerate interval vector $[\mathbf{x}] = \mathbf{x}$ is also punctual: $[\mathbf{f}](\mathbf{x}) = \{\mathbf{f}(\mathbf{x})\}$.

$[\mathbf{f}]$ is said *inclusion monotonic* if:

$$[\mathbf{x}] \subset [\mathbf{y}] \implies [\mathbf{f}]([\mathbf{x}]) \subset [\mathbf{f}]([\mathbf{y}]). \quad (2.26)$$

An infinity of inclusion functions exist for a given \mathbf{f} but only one of them will be *minimal* and denoted $[\mathbf{f}]^*$. $[\mathbf{f}]$ is minimal if $\forall [\mathbf{x}]$, $[\mathbf{f}]([\mathbf{x}])$ is the smallest box containing $\mathbf{f}([\mathbf{x}])$. Figure 2.4 illustrates this notion. Any non-minimal inclusion function is said *pessimistic*.

Natural inclusion functions

When \mathbf{f} is a function made of a finite suite of elementary functions such as \sin , \tan , $\sqrt{\cdot}$, \min , \dots and operators $+$, $-$, $*$, $/$, then the simplest method to obtain an inclusion function of it is to replace the variables x_1, x_2, \dots by their interval representation $[x_1], [x_2], \dots$ and the functions and operators by their interval counterpart: $[\sin]$, $[\tan]$, *etc.* The obtained $[\mathbf{f}]$ is then said *natural inclusion function* of \mathbf{f} .

$[\mathbf{f}]$ is inclusion monotonic and thin. In addition, it is convergent if made of continuous functions and operators. However, a natural inclusion function may not be minimal due to dependencies between the variables and some wrapping effect, discussed in the next Section 2.2.4. $[\mathbf{f}]$ will be minimal if each variable only appears once in its expression and if only continuous functions and operators are involved in its expression.

Let us come back to the range-only beacon localization example and compute the natural inclusion of the distance function \mathbf{g} :

$$\begin{aligned} \mathbf{g} : \mathbb{R}^2 &\rightarrow \mathbb{R}, \\ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} &\mapsto \sqrt{(x_1 - x_1^k)^2 + (x_2 - x_2^k)^2}. \end{aligned} \quad (2.27)$$

Replacing items by their interval counterpart, $[\mathbf{g}]$ is given by:

$$\begin{aligned} [\mathbf{g}] : \mathbb{IR}^2 &\rightarrow \mathbb{IR}, \\ \begin{pmatrix} [x_1] \\ [x_2] \end{pmatrix} &\mapsto \sqrt{([x_1] - x_1^k)^2 + ([x_2] - x_2^k)^2}. \end{aligned} \quad (2.28)$$

As functions $\sqrt{\cdot}$, $(\cdot)^2$ and operators $+$, $-$ are continuous on their definition domain and variables x_1 , x_2 appear only once, the natural inclusion function $[\mathbf{g}]$ is minimal.

2.2.4 Pessimism and wrapping effect

We have seen that properties of basic operations on intervals may differ from their equivalent in \mathbb{R} , sometimes inducing unwanted pessimism. This section briefly presents two causes of over-estimation. Although this effect does not impact the reliability of the results, it may lead to meaningless outcomes when intervals are too wide. In most cases, it becomes important to think about how to overcome such pessimism.

Dependencies between the variables

In interval analysis, different analytical expressions of a same function often lead to significantly different performances. As an example, consider the difference between two non-degenerate intervals:

$$[x] - [x] = [\{a - b \mid a \in [x], b \in [x]\}] = [x^- - x^+, x^+ - x^-]. \quad (2.29)$$

The result is far from being thin. This issue appears when an expression involves a variable several times. Figure 2.5 provides another telling example [Ceberio and Granvilliers, 2002].

Dependencies between the variables may be dealt analytically, for instance by representing systems of equations into unique directed acyclic graphs [Schichl and Neumaier, 2005] or by exploiting and deleting common sub-expressions [Araya et al., 2008]. To this day however, no general method has been adopted in existing solvers.

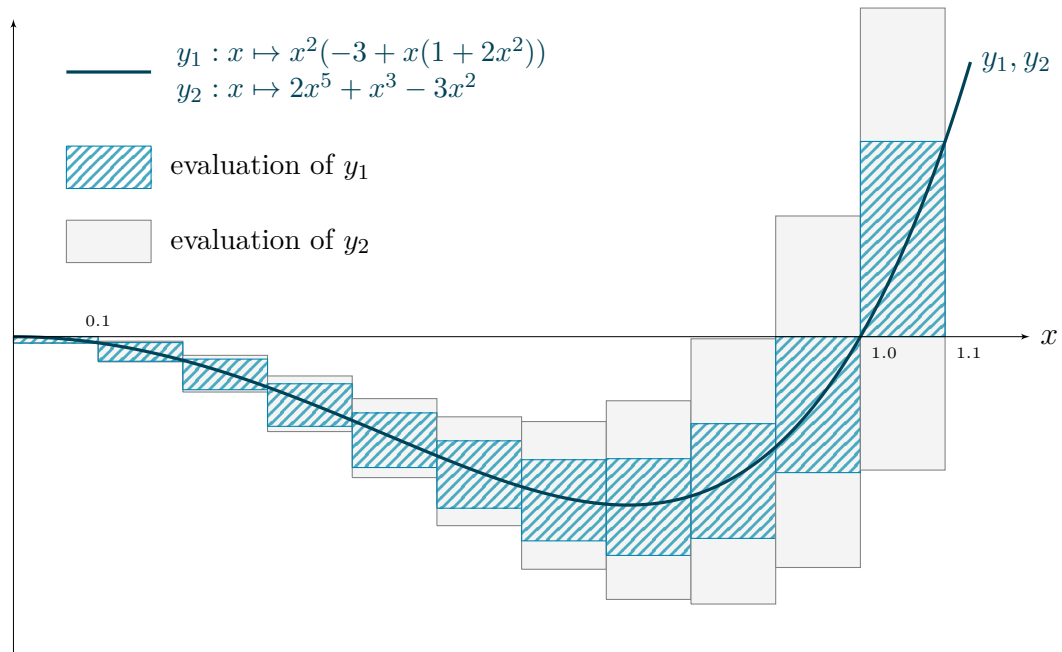


Figure 2.5: Comparison of different interval evaluations of a same function $y_1 = y_2$. Properties of interval arithmetic sometimes lead to computed intervals that can be much larger than the exact range. This *dependency* problem can be overcome by rewriting expressions and using different factorization schemes. This example shows a classical interval evaluation and its counterpart pictured in blue \bullet , using a factorization based on Horner's rule [Ceberio and Granvilliers, 2002].

Wrapping effect

Intervals and interval-vectors are axis-aligned items. Therefore, any set that is not a box made of boundaries aligned with axes will suffer from a pessimistic enclosure representation: a so-called *wrapping effect*.

This over-enclosure may quickly increase when a set is successively evaluated by a composition of functions, each of which inducing its own wrapping effect. A well-known illustration has been given in [Moore, 1966], presenting a suite of rotations of a box, depicted in Figure 2.6.

This effect can be overcome by dividing the solution space into a set of non-overlapping boxes, see Section 2.4.1, at the expense of longer computation times and increased memory space.

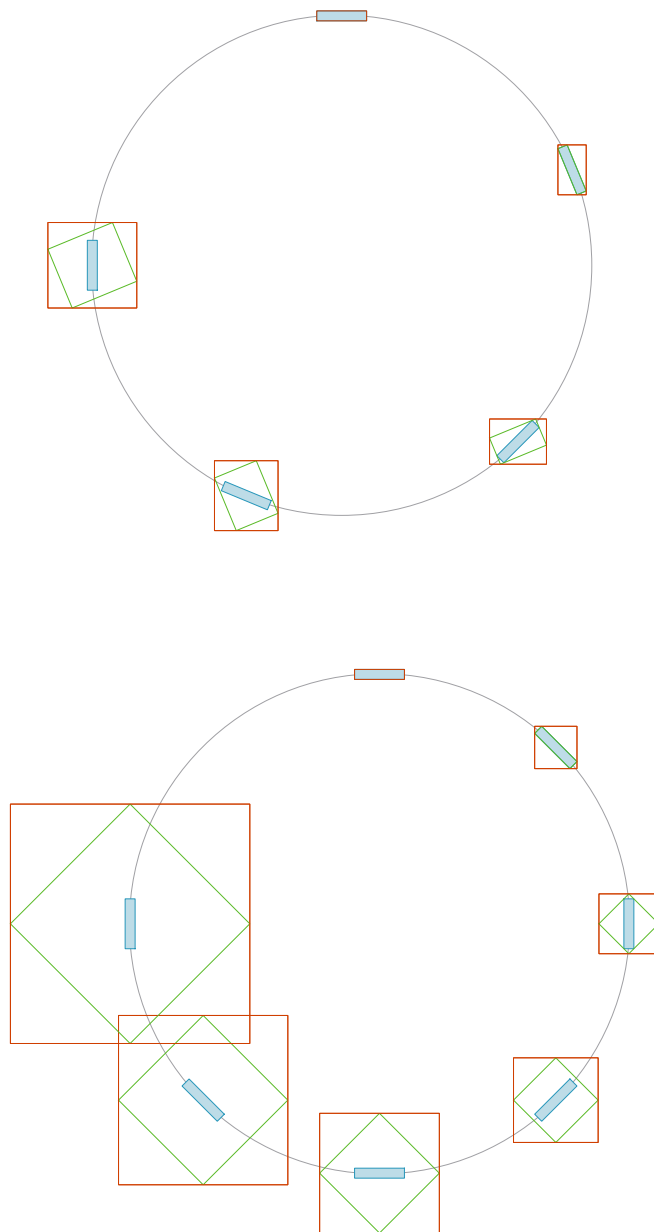


Figure 2.6: The wrapping effect revealed through several boxes rotations. A blue box \bullet is submitted to a suite of rotating angles resulting in a $\frac{3\pi}{2}$ global rotation. Red boxes \bullet depict interval enclosures, green ones \bullet are the results of rotated enclosures. The rotation is performed in four steps in the first case, six in the second one. This highlights how the effect increases with successive computations. Note that a rotation performed in one or three steps would have provided the minimal enclosure of the final blue box since each intermediate evaluation is axis-aligned.

2.3 Constraints propagation

In parallel with the emergence of interval analysis, the artificial intelligence community has developed new approaches based on constraint propagations [Bessiere, 2006]. These methods are used for systems solving involving discrete or continuous variables of real numbers [Benhamou and Older, 1997, Van Hentenryck et al., 1998] and go well with the use of intervals defining the domains of these variables.

In our applications, a state estimation problem will be depicted by a Constraint Network (CN) and solved by using *contractors* that are tools to reduce the domains of the network's variables.

2.3.1 Constraint networks

Presentation

A mathematical problem can be presented by means of a CN involving variables $\{x_1, \dots, x_n\}$ that must satisfy a set of rules or facts, called constraints and denoted $\{\mathcal{L}_1, \dots, \mathcal{L}_m\}$, over domains defining a non-empty range of feasible values $\{\mathbb{X}_1, \dots, \mathbb{X}_n\}$ [Mackworth, 1977].

The variables x_i can be symbols, real numbers [Araya et al., 2012] or vectors of \mathbb{R}^n . As presented in the introduction of this chapter, domains can be intervals, boxes, polytopes, *etc.* Finally, there are very few restrictions on the forms of the constraints, that can be non-linear equations between the variables, such as $x_3 = \cos(x_1 + \exp(x_2))$, inequalities or even quantified parameters [Goldsztein, 2006].

The estimation then consists in computing the smallest variables' domain while satisfying the defined constraints⁴. Figure 2.7 provides a simple view of this approach.

⁴We also often speak about Constraint Satisfaction Problems (CSPs) formulated as $\mathcal{H} : (\mathbf{f}(\mathbf{x}) = \mathbf{0}, \mathbf{x} \in [\mathbf{x}])$ and for which the resolution consists in computing the best approximation of \mathbf{x} . However, a set-membership state estimation is not formalized by such \mathcal{H} and it is more appropriate to speak about CNs to depict the constraints of our applications.

Decomposition

Problems involving complex equations can be broken down into a set of primitive constraints. Here, *primitive* means that the constraints cannot be decomposed anymore. For instance, in our range-only localization problem, the observation constraint \mathcal{L}_{g_k} that stands on Equation (2.1), page 38, can be decomposed into:

$$\mathcal{L}_{g_k} : \rho_k = \sqrt{(x_1 - x_1^k)^2 + (x_2 - x_2^k)^2} \iff \begin{cases} a = x_1 - x_1^k, \\ b = x_2 - x_2^k, \\ c = a^2, \\ d = b^2, \\ e = c + d, \\ \rho_k = \sqrt{e}. \end{cases} \quad (2.30)$$

where a, b, \dots, e are intermediate variables used for ease of decomposition. This constitutes a network made of the \mathcal{L}_- , \mathcal{L}_+ , $\mathcal{L}_{(\cdot)^2}$, and $\mathcal{L}_{\sqrt{\cdot}}$ elementary constraints.

Propagation

When working with finite domains, a propagation technique [Waltz, 1972] can be used to simplify a problem. The process is run several times up to a fixed point reached when domains \mathbb{X}_i cannot be reduced anymore. Interval analysis can be efficiently used for this purpose, taking advantage of interval arithmetic and its capacity to preserve any feasible solution.

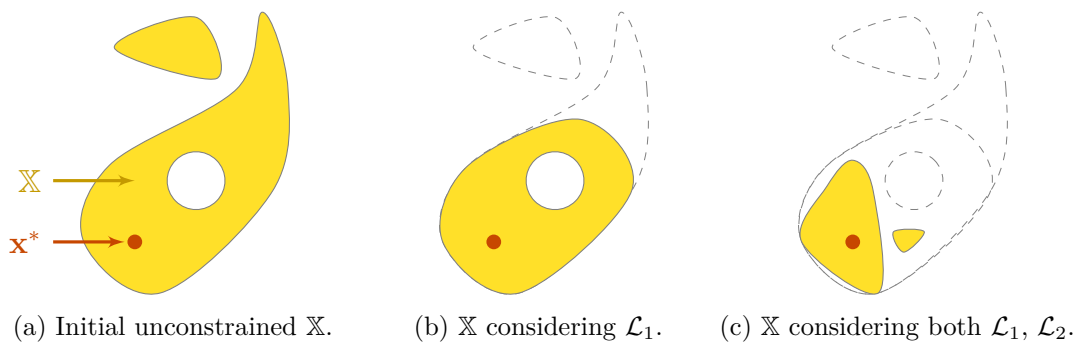


Figure 2.7: In this theoretical view, a domain \mathbb{X} depicted in yellow \bullet is known to enclose a solution \mathbf{x}^* pictured by a red dot \bullet and consistent with two constraints \mathcal{L}_1 and \mathcal{L}_2 . The estimation of \mathbf{x}^* consists in reducing \mathbb{X} while satisfying \mathcal{L}_1 and \mathcal{L}_2 .

Furthermore, using monotonous operators in iterative resolution processes, the constraints can be applied in any order [Apt, 1999]. The sequence can only impact the computation time, as it could be more interesting to apply one constraint before another.

The approach adopted in this document is to apply a given constraint on a box $[\mathbf{x}] \in \mathbb{IR}^n$ by means of a contractor \mathcal{C} .

2.3.2 Contractors

Formally, a contractor $\mathcal{C}_{\mathcal{L}}$ associated to the constraint \mathcal{L} is an operator $\mathbb{IR}^n \rightarrow \mathbb{IR}^n$ that returns a box $\mathcal{C}_{\mathcal{L}}([\mathbf{x}]) \subseteq [\mathbf{x}]$ without removing any vector consistent with \mathcal{L} . We will use the following definition, adapted from [Chabert and Jaulin, 2009]:

Definition 2.1

A contractor is a mapping $\mathcal{C}_{\mathcal{L}}$ from \mathbb{IR}^n to \mathbb{IR}^n such that

- (i) $\forall [\mathbf{x}] \in \mathbb{IR}^n, \mathcal{C}_{\mathcal{L}}([\mathbf{x}]) \subseteq [\mathbf{x}],$ (contraction)
- (ii) $\left(\begin{array}{l} \mathcal{L}(\mathbf{x}) \\ \mathbf{x} \in [\mathbf{x}] \end{array} \right) \implies \mathbf{x} \in \mathcal{C}_{\mathcal{L}}([\mathbf{x}]).$ (consistency)

Figure 2.8 gives a simple illustration of contractions.

Property (i) states that a box can only be reduced when submitted to a contractor, while the second one justifies that a solution consistent with \mathcal{L} cannot

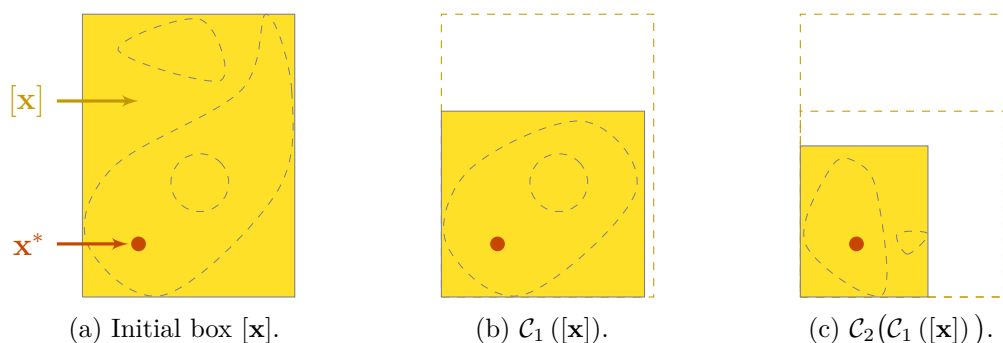


Figure 2.8: Application of the constraints \mathcal{L}_1 and \mathcal{L}_2 of Figure 2.7 by means of respective contractors \mathcal{C}_1 and \mathcal{C}_2 . On this theoretical example, the domain \mathbb{X} is now a subset of a box $[\mathbf{x}]$, easily representable and contractible.

be removed. Therefore, contractors can be applied on boxes as many times as necessary without risking losing a solution or being more pessimistic.

In practice, these operators are usually given by polynomial-time algorithms. Constructing a store of contractors such as \mathcal{C}_+ , \mathcal{C}_{\sin} , \mathcal{C}_{\exp} associated to primitive equations such as $z = x + y$, $y = \sin(x)$, $y = \exp(x)$ has been the subject of much work, see for instance [Jaulin et al., 2001, Chabert and Jaulin, 2009, Desrochers and Jaulin, 2016]. A significant part of interval analysis algorithms can also be wrapped into these contractors, as illustrated below.

Examples

Let us consider the constraint $\mathcal{L}_+(a, x, y) : a = x + y$. The related contractor \mathcal{C}_+ is defined as

$$\begin{pmatrix} [a] \\ [x] \\ [y] \end{pmatrix} \xrightarrow{\mathcal{C}_+} \begin{pmatrix} [a] \cap ([x] + [y]) \\ [x] \cap ([a] - [y]) \\ [y] \cap ([a] - [x]) \end{pmatrix}. \quad (2.31)$$

In this way, information on either $[a]$, $[x]$ or $[y]$ can be propagated to the other intervals. For instance, $\mathcal{C}_+([4, 5], [0, 3], [-2, 2])$ will produce $([4, 5], [2, 3], [1, 2])$. As another example, let us write the contractor \mathcal{C}_{\exp} for the non-linear constraint $\mathcal{L}_{\exp}(a, b) : a = \exp(b)$:

$$\begin{pmatrix} [a] \\ [b] \end{pmatrix} \xrightarrow{\mathcal{C}_{\exp}} \begin{pmatrix} [a] \cap \exp([b]) \\ [b] \cap \log([a]) \end{pmatrix}. \quad (2.32)$$

Properties considered in this document

The *minimal* contractor is obtained when a box $[\mathbf{x}]$ is contracted to be the smallest box containing the solution set.

A contractor is said *monotonous* if

$$[\mathbf{x}] \subseteq [\mathbf{y}] \implies \mathcal{C}([\mathbf{x}]) \subseteq \mathcal{C}([\mathbf{y}]). \quad (2.33)$$

Contractor programming

If the implementation of an elementary constraint such as \mathcal{L}_+ can be trivial, things are further complicated when facing complex ones. The propagation is then often affordable from a low-level point of view, but assuming a good knowledge of the used library. Furthermore, the developed solver will be dedicated to this composition and hardly scalable for non-advised users.

The concept of *contractor programming* introduced in [Chabert and Jaulin, 2009] carries the approach a step further, proposing a formalism where a contractor $\mathbb{R}^n \rightarrow \mathbb{R}^n$ can also be interpreted as subset of \mathbb{R}^n . This allows one to consider all the standard operations on sets on contractors such as:

$$\begin{aligned}
(\mathcal{C}_1 \cap \mathcal{C}_2)([\mathbf{x}]) &:= \mathcal{C}_1([\mathbf{x}]) \cap \mathcal{C}_2([\mathbf{x}]) && \textit{(intersection)} \\
(\mathcal{C}_1 \cup \mathcal{C}_2)([\mathbf{x}]) &:= \mathcal{C}_1([\mathbf{x}]) \sqcup \mathcal{C}_2([\mathbf{x}]) && \textit{(union)} \\
(\mathcal{C}_1 \circ \mathcal{C}_2)([\mathbf{x}]) &:= \mathcal{C}_1(\mathcal{C}_2([\mathbf{x}])) && \textit{(composition)} \\
\mathcal{C}_1^\infty &:= \mathcal{C}_1 \circ \mathcal{C}_1 \circ \mathcal{C}_1 \circ \dots && \textit{(iterated composition)}
\end{aligned} \tag{2.34}$$

Using this formalism allows simple combinations of primitive contractors. Combining these operators leads to a complex one that still provides reliable results, thus allowing one to deal with a wide range of problems.

The direct outcomes of such framework are genericity and simplicity: the user now focuses on the *what* instead of the *how* to build a solver, which is the essence of declarative programming. The energy is spent programming a solver based on mathematical constraints by combining contractors, rather than configuring a dedicated algorithm.

The contractor programming concept has triggered the development of several successful applications [Gning and Bonnifait, 2006, Alexandre dit Sandretto et al., 2014, Jaulin, 2011], thus demonstrating its efficiency. Very recent works proposed to extend this concept to dynamical systems, outlining good news in this line of research. The two first contributions of this thesis are to provide new contractors related to differential equations, opening doors to the estimation of dynamical systems such as those encountered in mobile robotics.

2.3.3 Application to static range-only robot localization

We will perform a concrete non-linear state estimation using the afore mentioned tools.

Problem statement

The robot \mathcal{R} is located between three beacons \mathcal{B}_k , $k \in \{1, 2, 3\}$. Respective synchronous range measurements are ρ_k^* . However, these values are not known and we shall assume the following bounded measurements:

Table 2.1: Beacons' positions and respective measurements

	(x_1^k, x_2^k)	ρ_k^*	$[\rho_k]$
\mathcal{B}_1	$(-0.5, 4.0)$	4.03	$[3.63, 4.43]$
\mathcal{B}_2	$(-2.5, -2.5)$	3.53	$[3.13, 3.93]$
\mathcal{B}_3	$(2.5, -0.5)$	2.55	$[2.15, 2.95]$

The observation constraint that links a measurement ρ_k to the state \mathbf{x} of \mathcal{R} is reminded:

$$\mathcal{L}_{g_k}(\mathbf{x}, \rho_k) : \rho_k = \sqrt{(x_1 - x_1^k)^2 + (x_2 - x_2^k)^2}. \quad (2.35)$$

The problem is synthesized with the following CN:

$$\text{CN:} \left\{ \begin{array}{l} \mathbf{Variables:} \mathbf{x}, \rho_1, \rho_2, \rho_3 \\ \mathbf{Constraints:} \\ \quad 1. \mathcal{L}_{g_1}(\mathbf{x}, \rho_1) \\ \quad 2. \mathcal{L}_{g_2}(\mathbf{x}, \rho_2) \\ \quad 3. \mathcal{L}_{g_3}(\mathbf{x}, \rho_3) \\ \mathbf{Domains:} [\mathbf{x}], [\rho_1], [\rho_2], [\rho_3] \end{array} \right. \quad (2.36)$$

The domains are intervals initialized with the bounded values provided in Table 2.1. Robot's position is considered unknown: $[\mathbf{x}] = [-\infty, \infty]^2$.

State estimation

Each constraint \mathcal{L}_{g_k} is implemented with a combination of primitive contractors, based on the decomposition detailed in Equation (2.30). Three contractors \mathcal{C}_{g_k} are built this way and applied on the domains:

1. $\mathcal{C}_{g_1}([\mathbf{x}], [\rho_1])$: contraction from \mathcal{B}_1 's measurement, Figures 2.9a, 2.9d.
2. $\mathcal{C}_{g_2}([\mathbf{x}], [\rho_2])$: contraction from \mathcal{B}_2 's measurement, Figures 2.9b, 2.9e.
3. $\mathcal{C}_{g_3}([\mathbf{x}], [\rho_3])$: contraction from \mathcal{B}_3 's measurement, Figure 2.9c.

Each contractor will reduce the domain $[\mathbf{x}]$, which may raise new contraction possibilities for the other constraints. It becomes interesting to call again the other contractors in order to take benefit from any contraction. An iterative resolution process is then used, where the contractors are called till a fixed point has been reached. By *fixed point* we mean that none of the domains $[\mathbf{x}]$ and $[\rho_k]$ has been contracted during a complete iteration. Figure 2.9 provides the synoptic of this state estimation. In this example, constraints have been propagated over 7 iterations in less than 0.01 second.

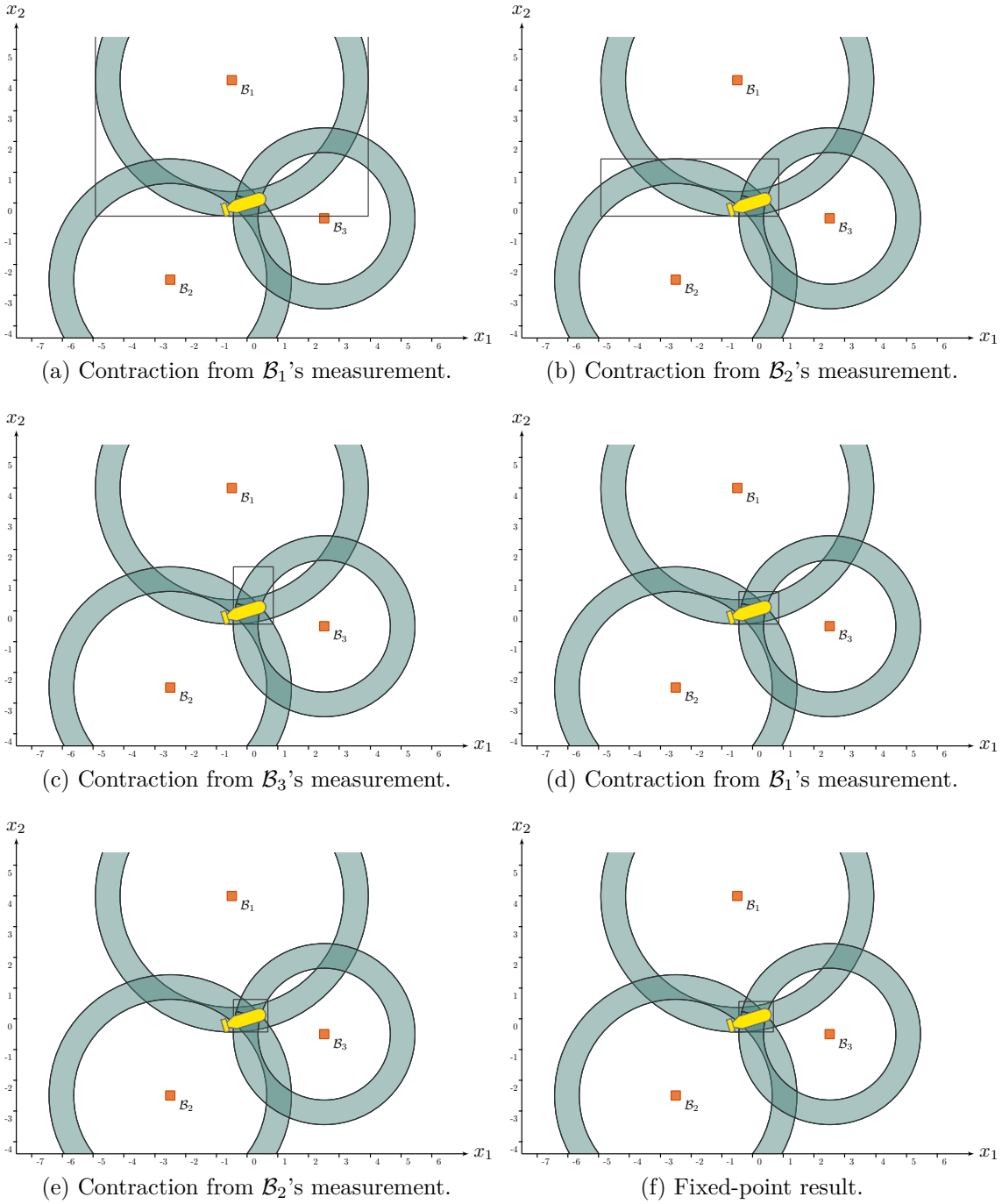


Figure 2.9: Set-membership localization with range-only measurements. Beacons pictured by red boxes \bullet emit signals received by the robot \mathcal{R} drawn in yellow \bullet . The figures illustrate successive contractions of the position box of \mathcal{R} from each range-only bounded measurement depicted by rings \bullet .

2.4 Set-inversion *via* interval analysis

When the solution set is made of holes or several non-connected subsets, its enclosure may suffer from a strong pessimism: the so-called wrapping-effect presented in Section 2.2.4. A refinement can be obtained by dividing the solution space and test for each subdivision whether it encloses a part of the solution set or not. The result constitutes a new kind of wrapper, called *subpaving*. It will be particularly useful for *set-inversion* problems, presented in this section.

2.4.1 Subpaving

A thinner estimation of a set $\mathbb{X} \subset \mathbb{R}^n$ enclosed by a box $[\mathbf{x}] \in \mathbb{IR}^n$ can be made with a union of non-overlapping boxes $[\mathbf{x}]^{(i)}$ included in $[\mathbf{x}]$. This set of boxes is called *subpaving*.

A subpaving \mathbb{K} of $[\mathbf{x}]$ covering completely $[\mathbf{x}]$ such that

$$[\mathbf{x}] = \bigcup_{[\mathbf{b}] \in \mathbb{K}} [\mathbf{b}] \quad (2.37)$$

is called *paving* of $[\mathbf{x}]$ and can be made of a collection of several subpavings.

If a thinner approximation of \mathbb{X} is affordable with a subpaving, we might also expect a qualification of such approximation. This can be done using two subpavings denoted \mathbb{X}^- and \mathbb{X}^+ such that

$$\mathbb{X}^- \subset \mathbb{X} \subset \mathbb{X}^+. \quad (2.38)$$

Figure 2.10 illustrates so-called *inner* and *outer* approximations of a set \mathbb{X} with subpavings, respectively denoted \mathbb{X}^- and \mathbb{X}^+ . An inner approximation gathers boxes that contain only solutions while an outer approximation is made of boxes in which a solution may be. The precision of the computation is given by the width of the set $[\mathbb{X}^-, \mathbb{X}^+]$ enclosing the boundary $\partial\mathbb{X}$ of the solution set. A thinner splitting of boxes will increase this precision, at the expense of longer computation times and raising memory space.

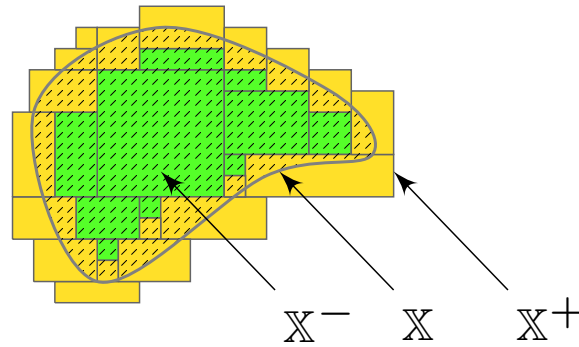


Figure 2.10: Inner and outer approximation of a set \mathbb{X} (hatched part) with two subpavings \mathbb{X}^- and \mathbb{X}^+ represented respectively by green \bullet and the union of green and yellow boxes $\bullet\bullet$. In any case, the boundary $\partial\mathbb{X}$ must remain enclosed within the visible yellow boxes.

2.4.2 SIVIA algorithm for set-inversion

Let us consider the computation of the reciprocal image $\mathbb{X} \subset \mathbb{R}^n$ such that $\mathbb{X} = \mathbf{f}^{-1}(\mathbb{Y})$ where $\mathbb{Y} \subset \mathbb{R}^m$ is the image set of \mathbb{X} by a possibly non-linear function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$. This operation is called *set-inversion* and is formalized as the characterization of:

$$\mathbb{X} = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{f}(\mathbf{x}) \in \mathbb{Y}\} = \mathbf{f}^{-1}(\mathbb{Y}). \quad (2.39)$$

A SIVIA⁵ algorithm [Jaulin and Walter, 1993b] can be used to approximate \mathbb{X} from any $\mathbb{Y} \subset \mathbb{R}^m$ and any function \mathbf{f} admitting an inclusion function $[\mathbf{f}] : \mathbb{I}\mathbb{R}^n \rightarrow \mathbb{I}\mathbb{R}^m$. The approximation is made by bracketing \mathbb{X} between two subpavings \mathbb{X}^- and \mathbb{X}^+ . Starting from an initial box $[\mathbf{x}]^{(0)} \in \mathbb{I}\mathbb{R}^n$, SIVIA will apply inclusion tests to decide whether it belongs to \mathbb{X}^+ , both \mathbb{X}^- and \mathbb{X}^+ , or none. In case of undecidability, the strategy is to bisect the box and apply again the tests on the sub-boxes.

A recursive version of SIVIA is given in Algorithm 1 in which four cases are encountered:

1. $[\mathbf{f}]([\mathbf{x}]) \cap \mathbb{Y} = \emptyset$: $[\mathbf{x}]$ does not belong to \mathbb{X} ;
2. $[\mathbf{f}]([\mathbf{x}]) \subset \mathbb{Y}$: any vector in $[\mathbf{x}]$ is solution, so $[\mathbf{x}]$ belongs to \mathbb{X} and is stored in both \mathbb{X}^- and \mathbb{X}^+ ;

⁵Set-Inversion *via* Interval Analysis (SIVIA)

3. $[f]([x])$ has a non-empty intersection with Y while not being a subset of Y . It is an undetermined case where $[x]$ may contain a solution. Therefore,
 - (a) if $\text{width}([x]) < \varepsilon$, then the box is considered small enough regarding the expected precision of the algorithm. The process stops here by storing $[x]$ into X^+ ;
 - (b) otherwise, a bisection of $[x]$ is performed, for instance along its largest dimension, and new tests are applied on each resulting box.

These cases are illustrated in Figure 2.11. Such inversion is easily affordable as it only stands on the inclusion of f : its inverse is not required.

Algorithm 1 SIVIA(in: $[f]$, $[x]$, Y , ε – inout: X^-, X^+)

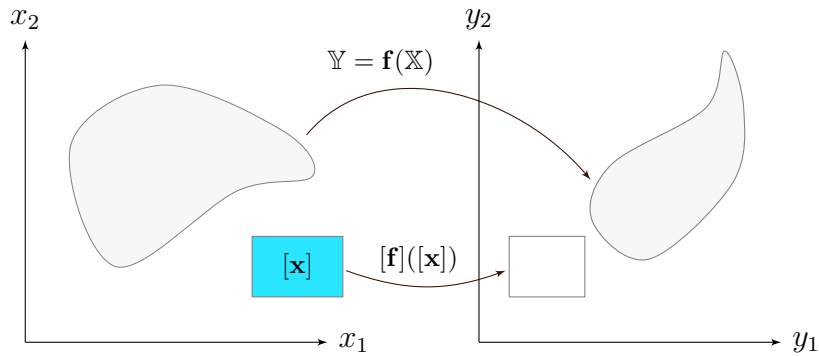
```

1: if  $[f]([x]) \cap Y \neq \emptyset$  then
2:   if  $[f]([x]) \subset Y$  then
3:      $X^+ \leftarrow X^+ \cup [x]$   $\triangleright$  outer set
4:      $X^- \leftarrow X^- \cup [x]$   $\triangleright$  inner set
5:   else if  $\text{width}([x]) < \varepsilon$  then
6:      $X^+ \leftarrow X^+ \cup [x]$   $\triangleright$  outer set only
7:   else
8:     bisect( $[x]$ ) into  $[x]^{(1)}$  and  $[x]^{(2)}$ 
9:     SIVIA( $[f]$ ,  $[x]^{(1)}$ ,  $Y$ ,  $\varepsilon$ ,  $X^-, X^+$ )
10:    SIVIA( $[f]$ ,  $[x]^{(2)}$ ,  $Y$ ,  $\varepsilon$ ,  $X^-, X^+$ )
11:   end if
12: end if

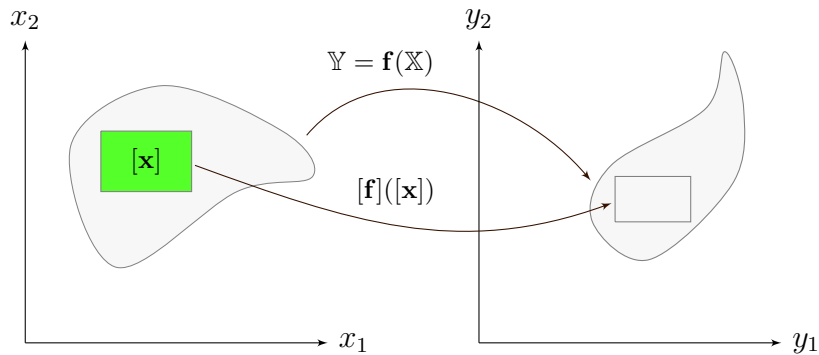
```

The precision ε of the approximation, materialized by the width of the interval $[X^-, X^+]$, is the only parameter to set with this algorithm. The thinner the interval $[X^-, X^+]$, the better the approximation of the inversion. In any case the true solution set X remains enclosed within these bounds. Figure 2.12 provides an illustration of subpavings computed by SIVIA, in various accuracy levels.

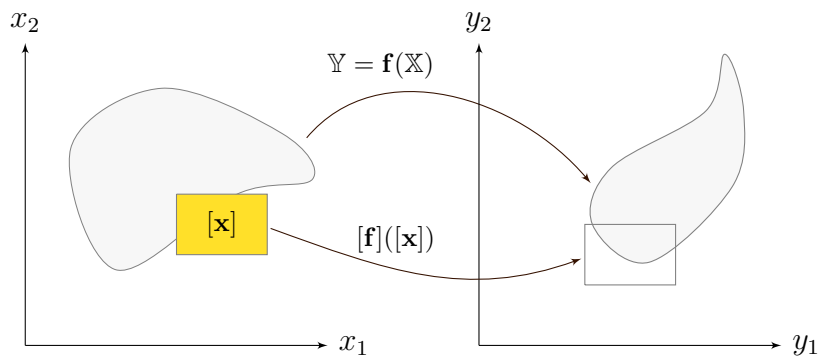
Several optimizations can be done. For instance, the SIVIA algorithm is easily parallelizable when several evaluations of $[f]([x])$ can be done simultaneously. In addition the paving of the solution space can be built as a binary tree, each node of which corresponding to a bisection of a box $[x]$. This regular representation speeds up the access to solution boxes.



(a) $[f]([x]) \cap Y = \emptyset \implies f([x]) \cap Y = \emptyset$. The box $[x]$ does not belong to the outer set X^+ and so to the inner one X^- .



(b) $[f]([x]) \subset Y \implies f([x]) \subset Y$. The box $[x]$ belongs to both the inner and outer sets, respectively X^- and X^+ .



(c) Indefinite case. $[x]$ is either subdivided or placed in the outer set X^+ .

Figure 2.11: Inclusion tests for set-inversion. The chosen color code is kept in the remainder of this document: green ● for inner solution sets, yellow ● for boxes belonging to outer sets only and blue ● for no-solution sets.

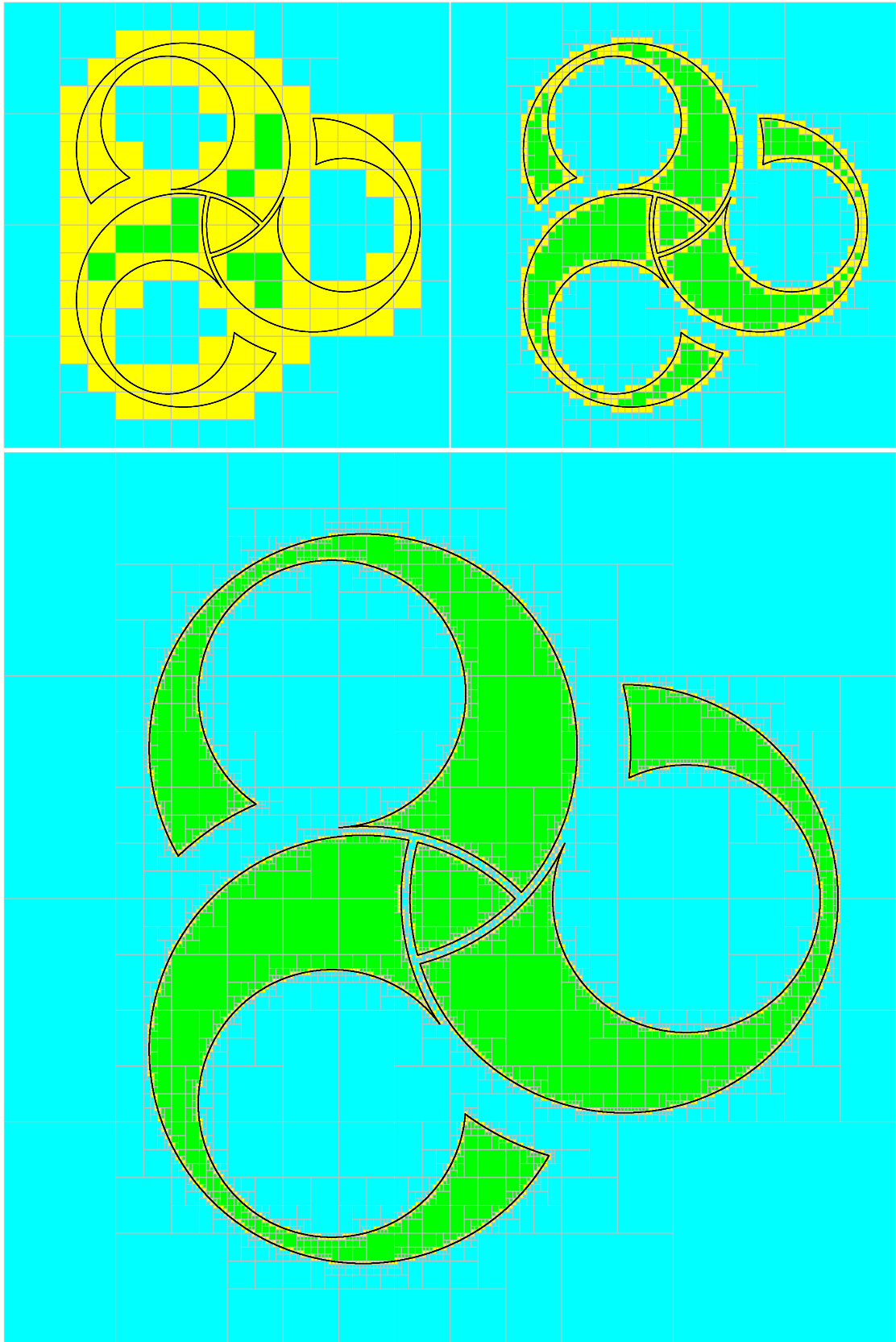


Figure 2.12: Subpavings computed by a SIVIA algorithm in various accuracy levels. The boundary $\partial\mathbb{X}$ of the true solution set is plotted by a black line. Inner and outer sets are respectively drawn by green \bullet and both yellow and green boxes $\bullet\bullet$. The part proven to not contain solutions is represented in blue \bullet . This approach enables the estimation of sets of any shape such as this triskelion [Le Gallo, 2016].

2.4.3 Illustration involving contractions

Coming back to the range-only problem, we will now voluntarily consider two beacons instead of three in order to deal with an ambiguous solution set. The robot stays located at $(0, 0)$.

Table 2.2: Beacons' positions and respective measurements

	(x_1^k, x_2^k)	ρ_k^*	$[\rho_k]$
\mathcal{B}_1	(8.0, 5.0)	9.43	[9.03, 9.83]
\mathcal{B}_2	(7.0, 2.2)	7.34	[6.94, 7.74]

SIVIA will provide a thinner approximation of the solution set while the only use of contractors would enclose it by a single box. Nonetheless, the algorithm can be coupled with contractors in order to decrease the time complexity, or equivalently the space complexity, by reducing the boxes to be evaluated in the subpavings. Figure 2.13 provides a comparison between a classical SIVIA algorithm for this range-only problem (Table 2.2) and its combination with the contractors presented in Section 2.3.3 on page 58. This coupling is simply done by contracting $[\mathbf{x}]$ before its bisection in the inconclusive inclusion test, see line 8 of Algorithm 1. The counter-part of this approach is that the paving is no more regular.

2.4.4 Kernel characterization of an interval function

The kernel characterization of a function is elementary and can be encountered in many problems under the form $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, $\mathbf{x} \in [\mathbf{x}]$. The kernel $\ker \mathbf{f}$ of a function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is a subset of the domain of \mathbf{f} defined as

$$\ker \mathbf{f} = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{f}(\mathbf{x}) = \mathbf{0}\} = \mathbf{f}^{-1}(\mathbf{0}). \quad (2.40)$$

When \mathbf{f} is known to be bounded by an interval-valued function $[\mathbf{f}]$, then the characterization of $\ker[\mathbf{f}]$ can be done with a SIVIA algorithm. This has been the object of [Aubry et al., 2014] with definitions and examples. For the sake of being self-contained, we will briefly remind these concepts that will be used afterward in this document.

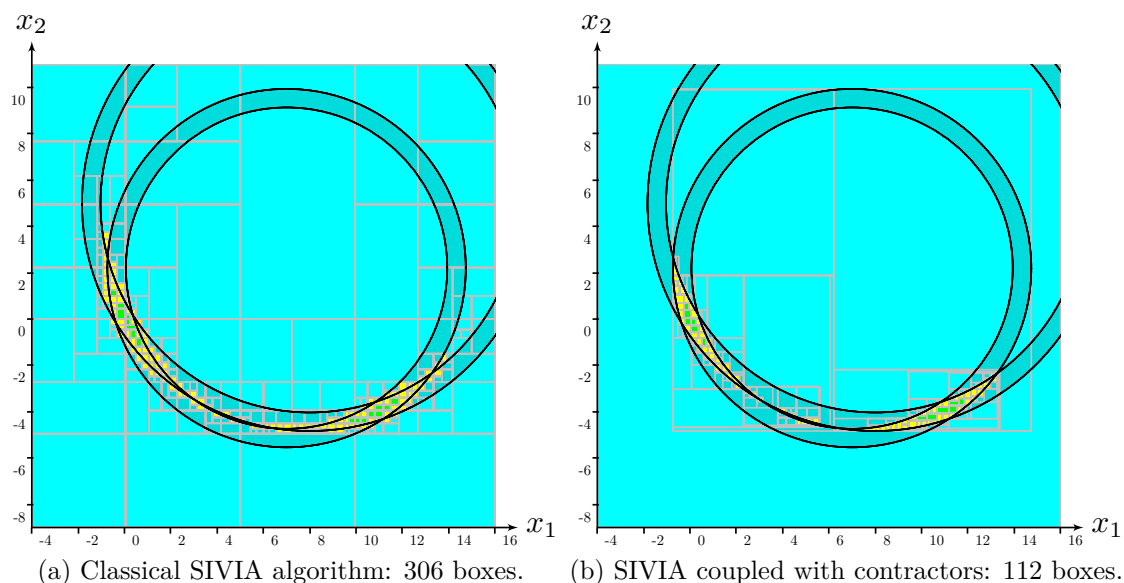


Figure 2.13: Range-only problem with two beacons. The solution set is complex and its enclosure by a box would present too much pessimism. The use of SIVIA then becomes relevant on this problem. The figures provide a comparison between a classical algorithm and its adaptation including a contraction process.

The kernel of an interval function $[\mathbf{f}]$ is defined by

$$\ker[\mathbf{f}] = \bigcup_{\mathbf{f} \in [\mathbf{f}]} \ker \mathbf{f} = \{\mathbf{x} \in [\mathbf{x}] \mid \mathbf{0} \in [\mathbf{f}](\mathbf{x})\}. \quad (2.41)$$

and is illustrated by Figure 2.14.

$\ker[\mathbf{f}]$ is a set \mathbb{X} that can be approximated by two subpavings \mathbb{X}^- and \mathbb{X}^+ . Therefore, the kernel of the actual but unknown function \mathbf{f}^* can be evaluated by $\ker \mathbf{f}^* \subset \mathbb{X}^+$. In the following, we will denote $\mathbf{f}^-(\mathbf{x})$ and $\mathbf{f}^+(\mathbf{x})$ the upper and lower bounds of $[\mathbf{f}](\mathbf{x})$, see Figure 2.15. We have:

$$\forall \mathbf{x}, [\mathbf{f}](\mathbf{x}) = [\mathbf{f}^-(\mathbf{x}), \mathbf{f}^+(\mathbf{x})]. \quad (2.42)$$

We will further assume the following two convergent inclusion functions $[\mathbf{f}^c]$ and $[\mathbf{f}^d]$ illustrated by Figure 2.16 and defined by:

$$[\mathbf{f}^c]([\mathbf{x}]) = [\text{ub}(\mathbf{f}^-([\mathbf{x}])), \text{lb}(\mathbf{f}^+([\mathbf{x}]))], \quad (2.43)$$

$$[\mathbf{f}^d]([\mathbf{x}]) = [\text{lb}(\mathbf{f}^-([\mathbf{x}])), \text{ub}(\mathbf{f}^+([\mathbf{x}]))]. \quad (2.44)$$

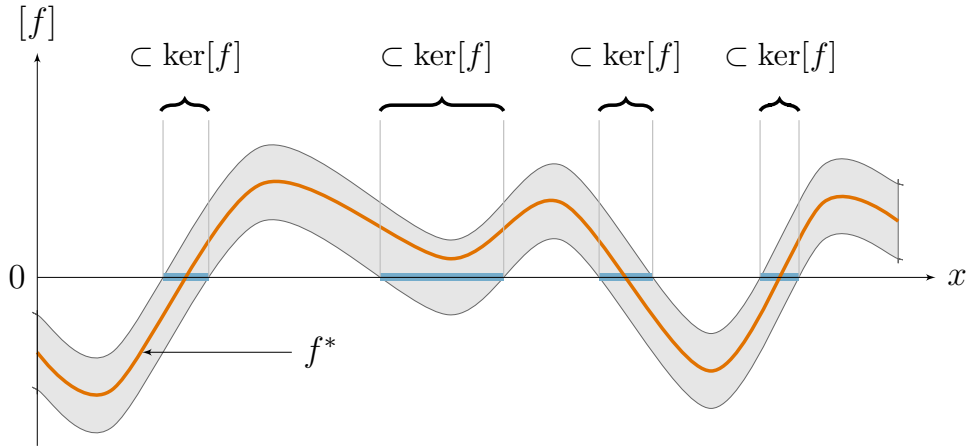


Figure 2.14: The kernel of an interval function $[f]$.

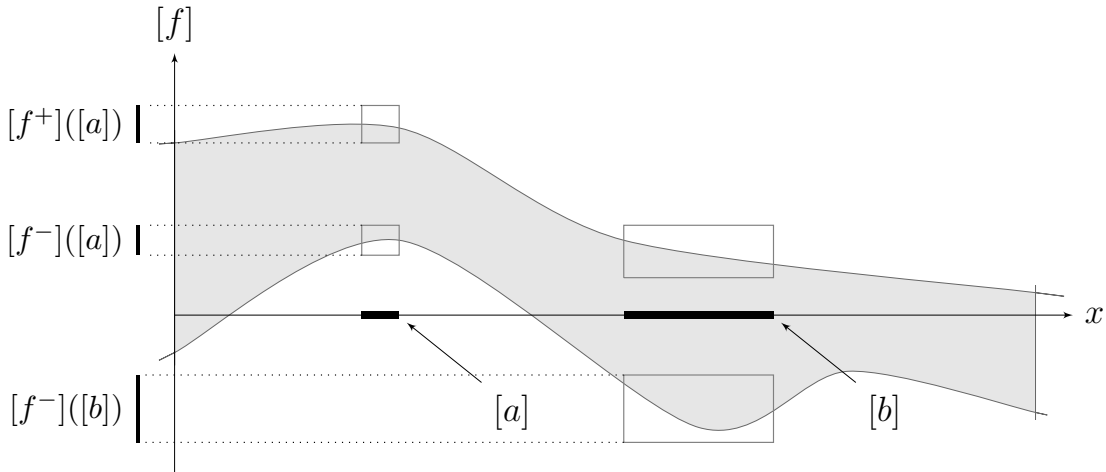


Figure 2.15: Bounds on an interval function $[f]$.

These definitions allow the following inclusion tests:

$$\mathbf{0} \in [f^c](\mathbf{x}) \implies \mathbf{x} \subset \mathbb{X}, \quad (2.45)$$

$$\mathbf{0} \notin [f^c](\mathbf{x}) \implies \mathbf{x} \cap \mathbb{X} = \emptyset. \quad (2.46)$$

Then, the approximation of $\ker[\mathbf{f}] = \mathbb{X}$ can be made based on these tests, as presented in Algorithm 2.

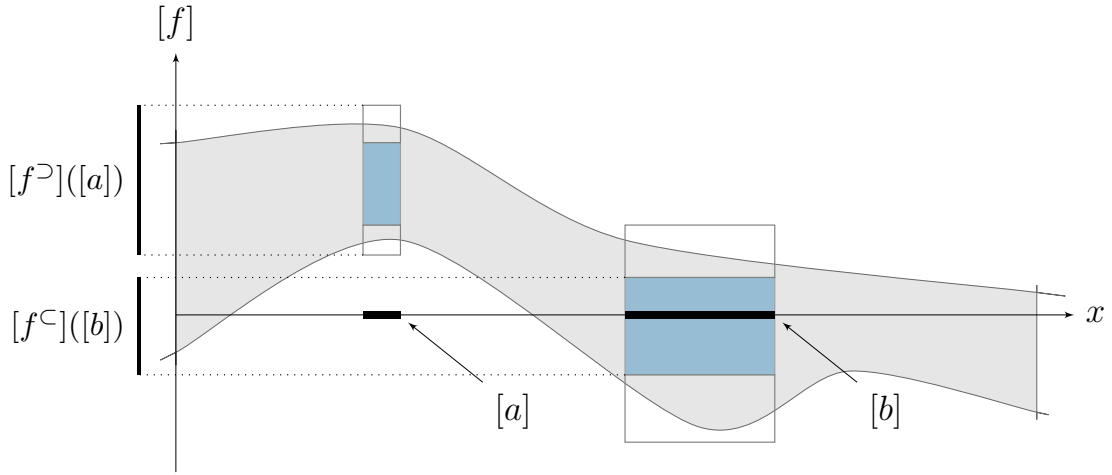


Figure 2.16: Inclusion functions $[f^C]$ and $[f^D]$.

Algorithm 2 kernelSIVIA (in : $[f]$, $[x]$, ε , inout : $\mathbb{X}^-, \mathbb{X}^+$)

```

1: if  $0 \in [f^D]([x])$  then
2:   if  $0 \in [f^C]([x])$  then
3:      $\mathbb{X}^+ \leftarrow \mathbb{X}^+ \cup [x]$   $\triangleright$  outer set
4:      $\mathbb{X}^- \leftarrow \mathbb{X}^- \cup [x]$   $\triangleright$  inner set
5:   else if  $\text{width}([x]) < \varepsilon$  then
6:      $\mathbb{X}^+ \leftarrow \mathbb{X}^+ \cup [x]$   $\triangleright$  outer set only
7:   else
8:     bisect( $[x]$ ) into  $[x]^{(1)}$  and  $[x]^{(2)}$ 
9:     kernelSIVIA( $[f]$ ,  $[x]^{(1)}$ ,  $\varepsilon$ ,  $\mathbb{X}^-, \mathbb{X}^+$ )
10:    kernelSIVIA( $[f]$ ,  $[x]^{(2)}$ ,  $\varepsilon$ ,  $\mathbb{X}^-, \mathbb{X}^+$ )
11:   end if
12: end if

```

2.5 Discussions

Today, the use of interval methods remains marginal in the communities of mobile robotics and automatic control: Bayesian approaches are clearly much more common in these fields. The reason is mainly due to the fact that research on the interval topic is new, though promising. In addition, set-membership methods work on sets while most applications expect scalar results, often assessed by some probabilities that are hardly verifiable. It is mainly a matter of how one can appropriately use one method or another according to the context. By going further, the approaches could be combined in order to keep the best of each world, but this is a topic in which only few work has been done [Abdallah et al., 2008, Neuland et al., 2014, De Freitas et al., 2016].

This section aims at providing some answers to recurring questions on this approach.

2.5.1 From sensors to reliable results

When dealing with real situations, speaking about guaranteed approaches is all based on inputs of our algorithms: the data-sets. The transition from theoretical computations to real values is a significant matter and has to be done rigorously in order to ensure further guaranteed outcomes.

In practice, a measurement error is often modeled by a Gaussian distribution which has an infinite support. Therefore, setting bounds around this measurement already constitutes a theoretical risk of losing the actual value. A choice has to be made at this step, considering such risk. After that, however, any algorithm standing on interval methods is ensured to not increase this risk.

Figure 2.17 presents the interval evaluation of a measurement μ assumed to follow a Gaussian distribution, so that we consider the real value enclosed within the interval $[x]$, centered on μ , with a 95% confidence rate. Datasheets usually give sensors specifications such as the standard deviation σ . The bounded value $[x]$ can then be inflated according to this dispersion value.

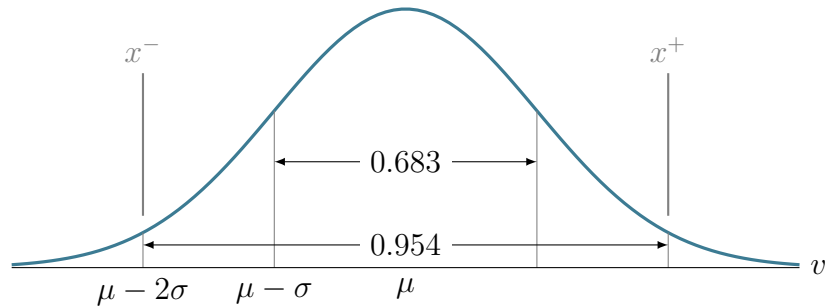


Figure 2.17: An interval $[x] = [x^-, x^+]$ computed from a Gaussian distribution to guarantee a 95% confidence rate over a measurement μ : $[x] = [\mu - 2\sigma, \mu + 2\sigma]$.

2.5.2 Numerical libraries

As stated in Section 2.2.1, page 41, the computer representation of a real number – or an interval defined by real bounds – can also induce errors. Indeed, the nearest floating point number is usually used by machines to represent a real $x \in \mathbb{R}$. The exact value of x will be lost if it does not exist among the computer numbers. As we have seen when introducing interval analysis, a rigorous containment of x will be assessed using an interval defined by representable bounds. Throughout the calculations, these bounds must be reliably represented, thus preventing from any loss of value. This procedure is called *outward rounding*, see Figure 2.18, and has to be executed for any arithmetic operation on intervals.

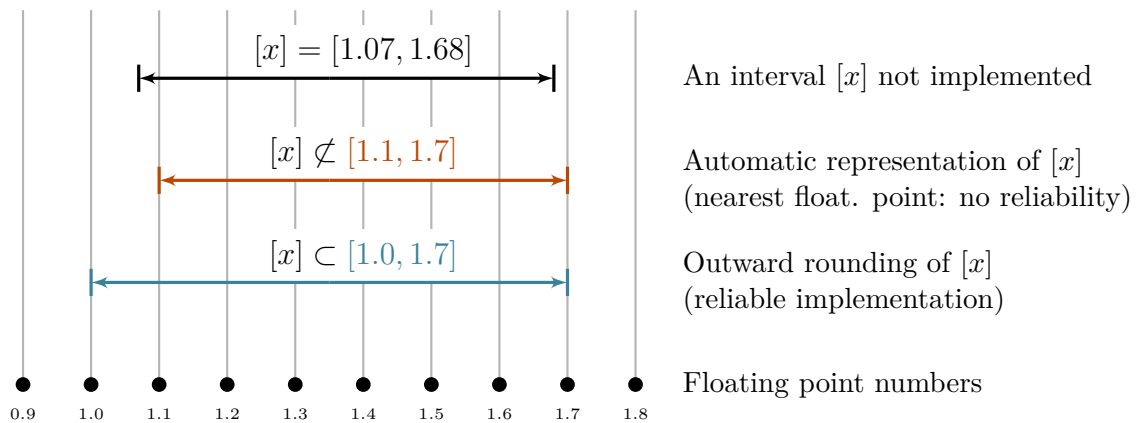


Figure 2.18: Outward rounding of an interval defined by non-representable bounds. Black dots \bullet represent floating point numbers depending on the computer precision. Its reliable approximation, represented in blue \bullet , encloses the initial range of values.

Several numerical libraries have been created to this end. For instance, computations presented in this document stand on the *GAOL*⁶ and *filib++*⁷ [Nehmeier and von Gudenberg, 2011] libraries that ensure numerical computations such as outward rounding.

Besides and at a higher level of abstraction, we use *IBEX*⁸ [Chabert, 2017]: a C++ library for constraint processing over real numbers. It provides a set of tools from the contractor programming paradigm presented in this chapter.

Finally, the contributions presented in this document come together with a dedicated open-source library called *Tubex*⁹, the aim of which is to implement constraints over sets of trajectories. This will be discussed from Chapter 3.

2.5.3 Reliable tool for proof purposes

Once any source of error is reliably handled, the following results are both computationally and mathematically guaranteed not to lose solutions. The outcomes of these algorithms can therefore be used for verified computing and so for proof purposes, see for instance: [Tucker, 1999, Goldsztejn et al., 2011].

This asset of set-membership methods will be highlighted in Chapter 6 in which we provide an original method to prove loops along an uncertain robot trajectory.

2.6 Conclusion

Interval analysis provides a reliable way to deal with uncertainties over real numbers. In this chapter, it has been shown the simplicity to address non-linear problems without having to perform any linearization nor approximation, as we do for usual methods such as the Kalman filter. Furthermore, intervals stand as a reliable solution to deal with poor datasets in which any data is of interest. Chapter 7 will highlight this point by providing a new localization method in poor environments, where other approaches would badly behave. Finally, the reliability of the results

⁶<http://frederic.goualard.net/#research-software>

⁷<http://www2.math.uni-wuppertal.de/~xsc/software/filib.html>

⁸<http://www.ibex-lib.org>

⁹<http://www.simon-rohou.fr/research/tubex-lib>

gives to this method a significant role for the development of new mathematical proofs.

In addition, when coupled with constraint propagation approaches, interval analysis allows one to deal with a wide range of problems in the most simple way. The definition of a set of constraints and their application by contractors on intervals and boxes have proved their worth and still gather the communities of constraint programming and set-membership tools. There is still a lot of lines of research to explore in this field, for instance in order to overcome wrapping effects, propose new elementary contractors or even to reliably address problems with respect to outliers outside measurements errors [Norton and Veres, 1993, Pronzato and Walter, 1996, Carbonnel et al., 2014].

The next chapter extends these interval concepts to continuous time dynamical systems. Our goal is to be able to deal with a wider class of problems such as differential equations and inter-temporal measurements. This will be the contributions of the next chapters, still with a solving process achieved by a contractor programming approach.

Constraints over sets of trajectories

Contents

3.1	Towards dynamic state estimation	74
3.1.1	Overall motivations	74
3.1.2	The approach defended in this thesis	76
3.2	Tubes	76
3.2.1	Definitions	76
3.2.2	Tube analysis	78
3.2.3	Contractors	81
3.3	Implementation	83
3.3.1	Data structure	85
3.3.2	Build a tube from real datasets	87
3.3.3	<i>TubeX</i> , dedicated tube library	89
3.4	Application: dead-reckoning of a mobile robot	90
3.4.1	Test case	90
3.4.2	Constraint Network	91
3.4.3	Resolution	91
3.5	Discussions	92
3.5.1	Limits	92
3.5.2	Extract the most probable trajectory from a tube	93
3.5.3	Application to path planning	95
3.6	Conclusion	95

3.1 Towards dynamic state estimation

3.1.1 Overall motivations

The example of the range-only robot localization presented in Chapter 2 was only a *static* state estimation problem. In practice, state observations are asynchronous and the system evolves between each measurement. Taking into account all this information distributed over time can be challenging, especially when dealing with non-linearities and strong uncertainties. Figure 3.1 illustrates an extension of the range-only problem we were considering until now.

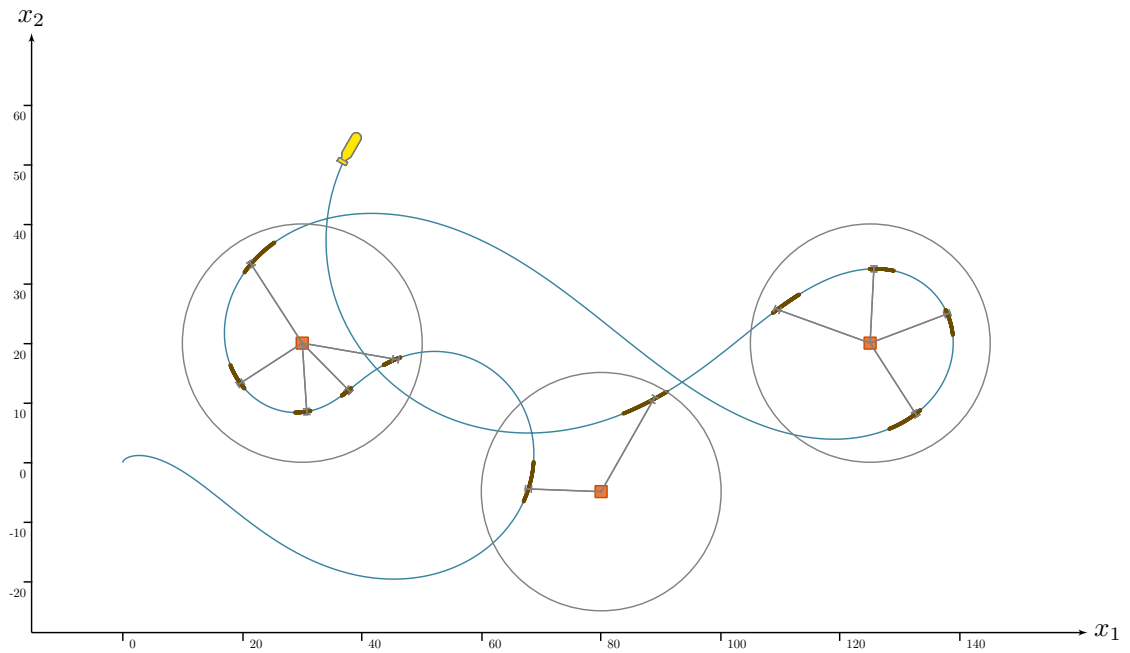


Figure 3.1: Motivating example of range-only localization with few asynchronous measurements. Emitting beacons, drawn by red boxes \bullet , send some range signals pictured by gray lines \bullet and received by the robot at uncertain times along its trajectory, plotted in blue \bullet . This application is challenging as it involves differential equations, non-linearities and uncertainties that are both spatial and temporal.

Therefore, we have to soundly deal with state estimations by considering both evolution and observation state equations:

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), & (3.1a) \\ z_i = g(\mathbf{x}(t_i)). & (3.1b) \end{cases}$$

We emphasize that any uncertainty has to be considered, for instance on the initial state \mathbf{x}_0 , on the evolution model depicted by the function \mathbf{f} and even on the measurements times t_i .

About state evolutions...

The first Equation (3.1a) is an Ordinary Differential Equation (ODE). The resolution of these equations remains an open question as they do not have analytical solutions, apart from some special cases such as in linear contexts. Of course, several numerical methods exist to afford approximations of the solutions [Hairer et al., 1993]. In particular, a considerable work has been done to solve the so-called Initial Value Problem (IVP) that consists in estimating the temporal evolution of a system, based on a given initial condition. Knowing the initial state \mathbf{x}_0 of a robot, *in which state will it be at time t ?*

Well-known methods exist such as the Kalman filter [Kalman, 1960], Particle Filters (PFs) [Montemerlo et al., 2003] or even new approaches such as Box Particle Filters (BPFs) [Abdallah et al., 2008, Gning et al., 2013, De Freitas et al., 2016]. Unfortunately, these methods show limits when facing non-linearities and/or uncertainties. PF-based methods may provide better estimations in non-linear cases but at the cost of considerable computations as they generate random systems from the knowledge of the initial condition. In any case, though, the provided outcomes are not guaranteed; this can be a significant limitation for systems safety.

...constrained by any uncertain observation

The situation is further complicated when the initial conditions are not known. Such case is typically encountered in the *kidnapped robot problem* for which a robot is carried to an unknown location. It then has to perform a full localization – which does not correspond to an IVP problem – based on a set of asynchronous observations and its own evolution since its kidnapping.

The problem becomes even much more complex when these observations are both spatially and temporally uncertain: the robot may deal with measurements where neither the value of the output z_i nor the acquisition date t_i are known exactly.

3.1.2 The approach defended in this thesis

Our strategy is to extend the constraint programming approach to differential constraint networks. We will consider trajectories as variables and implement contractors to reduce their domains, based on constraints that can be algebraic or differential. This extended approach has already been the subject of some recent work [Le Bars et al., 2012, Bethencourt and Jaulin, 2014]. The current chapter aims at introducing these new tools and see how algebraic constraints can be applied on sets of trajectories. This material will be sufficient to address a dead-reckoning problem for purposes of illustration.

The next Chapter 4 will then focus on differential contractors to completely address state estimations with asynchronous observations. Chapter 5 will finally focus on state observations with time uncertainties, providing another elementary contractor to constraint trajectories at a given time.

3.2 Tubes

Sets of trajectories will be approximated by means of tubes.

3.2.1 Definitions

Trajectories and dot notation

We will apply constraints to univariate function variables depicting so-called *trajectories*, without considering multi-variable cases since our applications only evolve with time. Hence, t is the independent evolution variable and the image vector is the trajectory value representing a state, an observation, *etc.*

In this manuscript, the notation (\cdot) is used in order to clearly distinguish a whole trajectory $\mathbf{x}(\cdot) : \mathbb{R} \rightarrow \mathbb{R}^n$ from a local evaluation: $\mathbf{x}(t) \in \mathbb{R}^n$. Indeed, in Chapters 5 and 7, time will not only be an independent variable but also a classical variable to be estimated.

Envelope of trajectories

A tube is defined over a domain $[t_0, t_f]$ as an envelope of trajectories $\mathbf{x}(\cdot) : \mathbb{R} \rightarrow \mathbb{R}^n$. The concept appeared in [Kurzhan and Filippova, 1993, Filippova et al., 1996] in the context of ellipsoidal estimations. We speak about an *envelope* as it may exist trajectories enclosed in the tube that are not solutions of our problem. Besides, tubes can also be employed to handle other signals such as discontinuous functions. We speak about a tube as a finite superset of a function graph.

In this document, we will use the definition given in [Le Bars et al., 2012, Bethencourt and Jaulin, 2014] where a tube $[\mathbf{x}](\cdot) : \mathbb{R} \rightarrow \mathbb{I}\mathbb{R}^n$ is an interval of two trajectories $[\mathbf{x}^-(\cdot), \mathbf{x}^+(\cdot)]$ such that $\forall t \in [t_0, t_f], \mathbf{x}^-(t) \leq \mathbf{x}^+(t)$. We also consider empty tubes that depict an absence of solutions, denoted $\emptyset(\cdot)$.

A trajectory $\mathbf{x}(\cdot)$ belongs to the tube $[\mathbf{x}](\cdot)$ if $\forall t \in [t_0, t_f], \mathbf{x}(t) \in [\mathbf{x}](t)$. Figure 3.2 illustrates a one-dimensional tube enclosing a trajectory $x^*(\cdot)$. Note that for the sake of simplified notations, the tubes mentioned in this chapter may be one-dimensional, without loss of generality as the methods are readily scalable to the vector case. In addition, we assume that all the tubes involved in a given resolution process share the same domain $[t_0, t_f]$.

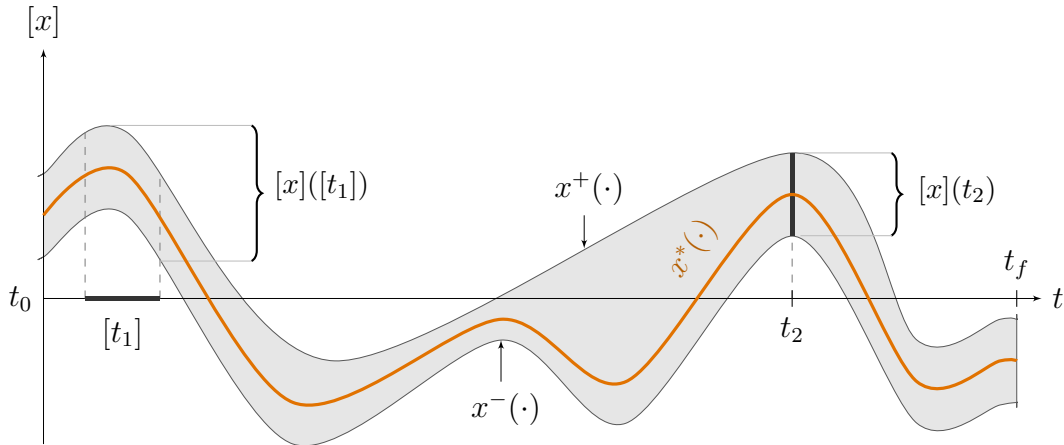


Figure 3.2: A one-dimensional tube $[x](\cdot)$, interval of two functions $[x^-(\cdot), x^+(\cdot)]$, enclosing a random signal $x^*(\cdot)$.

3.2.2 Tube analysis

Evaluations

Definition 3.1

The interval evaluation of a tube $[x](\cdot)$ over a bounded domain $[t]$ is given by [Bethencourt and Jaulin, 2014]:

$$[x]([t]) = \left[\{x(t) \mid x(\cdot) \in [x](\cdot), t \in [t]\} \right], \quad (3.2)$$

$$= \bigsqcup_{t \in [t]} [x](t). \quad (3.3)$$

$[x]([t])$ is then the smallest box enclosing all solutions for $x(t)$ such that $x(\cdot) \in [x](\cdot), t \in [t]$. See for instance $[x]([t_1])$ on Figure 3.2.

Definition 3.2

The tube inversion, denoted $[x]^{-1}([y])$, is defined by:

$$[x]^{-1}([y]) = \bigsqcup_{y \in [y]} \{t \mid y \in [x](t)\}, \quad (3.4)$$

and is illustrated by Figure 3.3. The result is the interval enclosing all preimages (fibers) of $[y]$ under $[x](\cdot)$. Solution subsets are easily assessed through binary search algorithms, not detailed here.

Arithmetics

Consider two tubes $[x](\cdot)$ and $[y](\cdot)$ and an operator $\diamond \in \{+, -, \cdot, /\}$. We define $[x](\cdot) \diamond [y](\cdot)$ as the smallest tube (with respect to inclusion) containing all feasible values for $x(\cdot) \diamond y(\cdot)$, assuming that $x(\cdot) \in [x](\cdot)$ and $y(\cdot) \in [y](\cdot)$:

$$[x](\cdot) \diamond [y](\cdot) = \left[\{x(\cdot) \diamond y(\cdot) \in \mathbb{R} \mid x(\cdot) \in [x](\cdot), y(\cdot) \in [y](\cdot)\} \right]. \quad (3.5)$$

This definition is an extension to trajectories of the interval arithmetic presented in Section 2.2.2 from page 42 onwards. If f is an elementary function such as \sin, \cos, \dots , we define $f([x](\cdot))$ as the smallest tube containing all feasible values:

$$f([x](\cdot)) = \left[\{f(x(\cdot)) \mid x(\cdot) \in [x](\cdot)\} \right]. \quad (3.6)$$

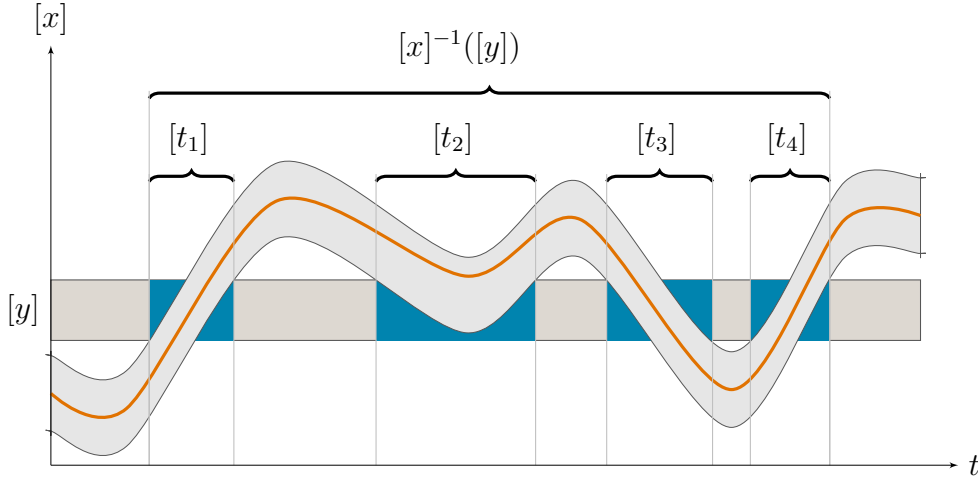


Figure 3.3: Tube inversion as defined in Equation (3.4). $[t_1]$, $[t_2]$, $[t_3]$, $[t_4]$ are preimage subsets enclosed within the inversion result $[x]^{-1}([y])$.

Integral

The integral of a tube is defined from t_1 to t_2 as the smallest interval containing all feasible integrals:

$$\int_{t_1}^{t_2} [x](\tau) d\tau = \left\{ \int_{t_1}^{t_2} x(\tau) d\tau \mid x(\cdot) \in [x](\cdot) \right\}. \quad (3.7)$$

From the monotonicity of the integral operator, we can deduce:

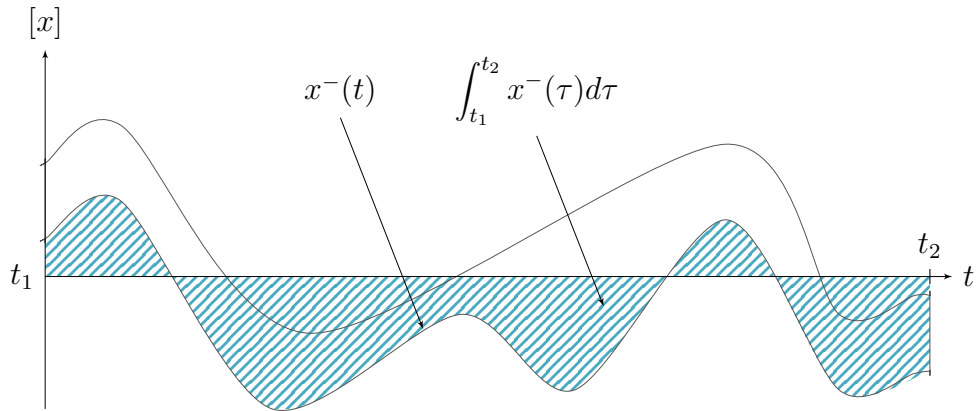
$$\int_{t_1}^{t_2} [x](\tau) d\tau = \left[\int_{t_1}^{t_2} x^-(\tau) d\tau, \int_{t_1}^{t_2} x^+(\tau) d\tau \right], \quad (3.8)$$

bearing in mind that $x^-(\cdot)$ and $x^+(\cdot)$ are the lower and upper bounds of tube $[x](\cdot) = [x^-(\cdot), x^+(\cdot)]$. The computed integral is an interval with lower and upper bounds shown on Figures 3.4. For efficiency purposes, the *interval primitive* of a tube defined by $\int_0^{\cdot} [x](\tau) d\tau$ can be computed once, from a primitive tube.

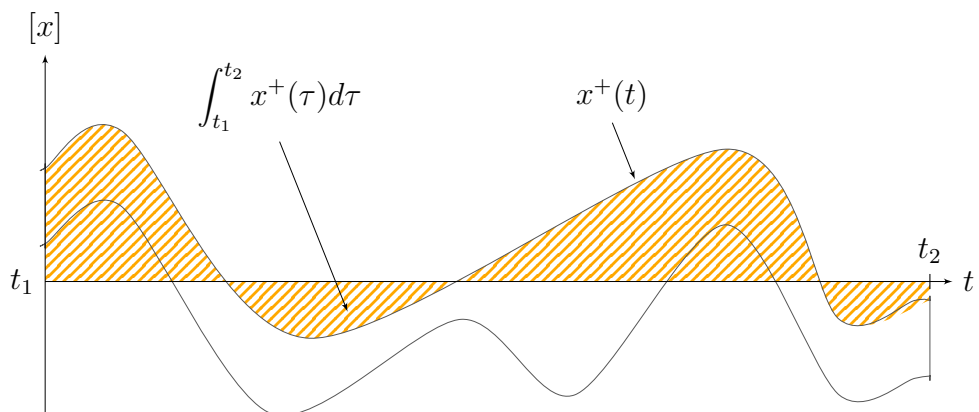
The integral can also be computed between bounded bounds $[t_1]$, $[t_2]$ by

$$\int_{[t_1]}^{[t_2]} [x](\tau) d\tau = \left[\text{lb} \left(y^-([t_2]) - y^-([t_1]) \right), \text{ub} \left(y^+([t_2]) - y^+([t_1]) \right) \right], \quad (3.9)$$

where $[y](\cdot) = \int_{t_0}^{\cdot} [x](\tau) d\tau$ is the interval primitive of $[x](\cdot)$ and $y^-(\cdot)$, $y^+(\cdot)$ are the corresponding bounds. The proof is provided in [Aubry et al., 2013, Sec. 3.3].



(a) hatched part depicts the lower bound of $\int_{t_1}^{t_2} [x](\tau) d\tau$



(b) hatched part depicts the upper bound of $\int_{t_1}^{t_2} [x](\tau) d\tau$

Figure 3.4: Lower and upper bounds of the integral of a tube.

Examples

Figures 3.5a–3.5b present two scalar tubes $[x](\cdot)$ and $[y](\cdot)$. The tube arithmetic makes it possible to compute any algebraic operation from these tubes, as illustrated by Figures 3.5c–3.5f.

3.2.3 Contractors

Definitions

The contractors presented in Section 2.3.2, page 54, can also be extended to sets of trajectories, thus allowing constraints over time such as $a(\cdot) = x(\cdot) + y(\cdot)$ or $b(\cdot) = \sin(x(\cdot))$. A *tube contractor* has been defined in [Bethencourt and Jaulin, 2014] and is recalled here.

Definition 3.3

A contractor $\mathcal{C}_{\mathcal{L}}$ applied on a tube $[x](\cdot)$ aims at removing infeasible trajectories according to a given constraint \mathcal{L} so that:

$$(i) \quad \forall t \in [t_0, t_f], \mathcal{C}_{\mathcal{L}}([x](t)) \subseteq [x](t) \quad (\text{contraction})$$

$$(ii) \quad \left(\begin{array}{c} \mathcal{L}(x(\cdot)) \\ x(\cdot) \in [x](\cdot) \end{array} \right) \implies x(\cdot) \in \mathcal{C}_{\mathcal{L}}([x](\cdot)) \quad (\text{consistency})$$

For instance, the minimal contractor \mathcal{C}_+ associated with the constraint $a(\cdot) = x(\cdot) + y(\cdot)$, is:

$$\left(\begin{array}{c} [a](\cdot) \\ [x](\cdot) \\ [y](\cdot) \end{array} \right) \mapsto \left(\begin{array}{c} [a](\cdot) \cap \left([x](\cdot) + [y](\cdot) \right) \\ [x](\cdot) \cap \left([a](\cdot) - [y](\cdot) \right) \\ [y](\cdot) \cap \left([a](\cdot) - [x](\cdot) \right) \end{array} \right). \quad (3.10)$$

In this way, information on either $[a](\cdot)$, $[x](\cdot)$ or $[y](\cdot)$ can be propagated to the other tubes. Note that in practice, for algebraic contractors, the real counterpart of the constraint is applied on the trajectories for each t of their domain.

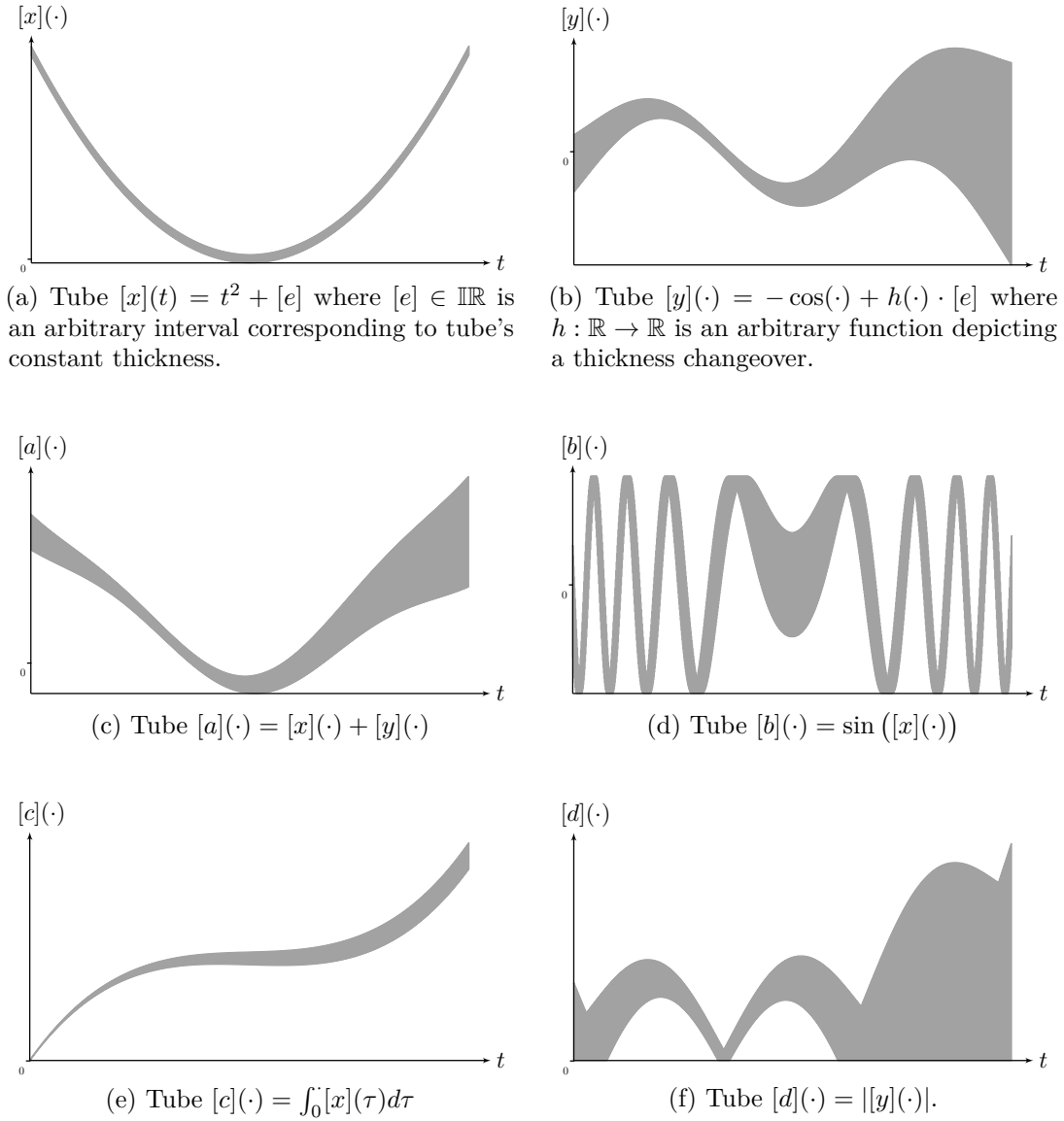


Figure 3.5: Tube arithmetics. Tubes $[a](\cdot)$, $[b](\cdot)$, $[c](\cdot)$, $[d](\cdot)$ are results obtained from algebraic operations on $[x](\cdot)$ and $[y](\cdot)$. Note that the vertical scales of these figures vary for full display.

Example

In order to illustrate constraints propagations on tubes, let us consider the following CN:

$$\text{CN: } \left\{ \begin{array}{l} \mathbf{Variables: } x(\cdot), y(\cdot), w(\cdot), a(\cdot), p(\cdot), q(\cdot) \\ \mathbf{Constraints:} \\ \quad 1. a(\cdot) = x(\cdot) + y(\cdot) \\ \quad 2. p(\cdot) = \arctan(y(\cdot)) \\ \quad 3. q(\cdot) = 2 \sin\left(\frac{a(\cdot)}{2}\right) + \sqrt{2p(\cdot)} \\ \quad 4. y(t_1) \in [i] \\ \quad 5. \dot{y}(\cdot) \in [w](\cdot) \\ \mathbf{Domains: } [x](\cdot), [y](\cdot), [w](\cdot), [a](\cdot), [p](\cdot), [q](\cdot) \end{array} \right. \quad (3.11)$$

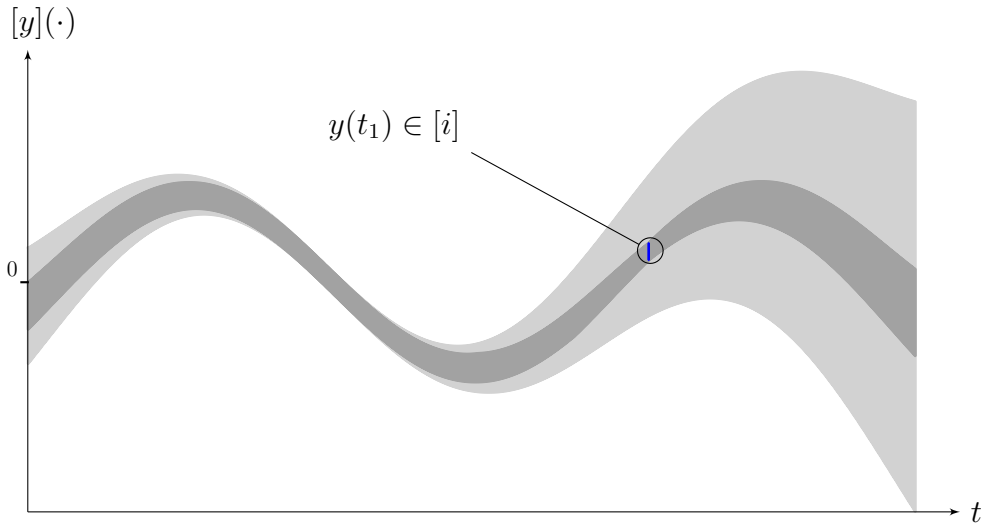
Constraints (1)–(3) are algebraic relations valid for any t . Conversely, Constraint (4) corresponds to a *local* constraint applied on the trajectory $y(\cdot)$. *Local* is understood as related to the instant t_1 . The propagation of Constraint (4) over the domain is then affordable with a differential contractor related to Constraint (5) and involving the tube $[y](\cdot)$ and the set $[w](\cdot)$ of the feasible derivatives. This is the subject of Chapter 4 and [Rohou et al., 2017].

Figure 3.6 illustrates a propagation of these constraints from an observation $(t_1, [i])$ on $y(\cdot)$. The contraction will reduce the envelope $[y](\cdot)$ in order to keep all the trajectories going through $[i]$ at t_1 . Related variables are contracted based on the above CN¹.

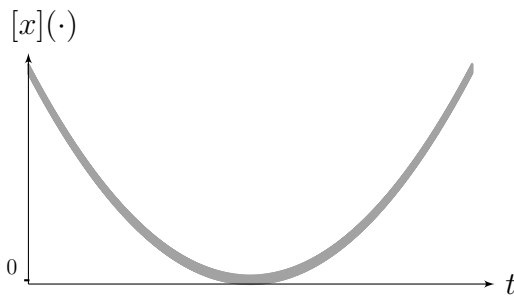
3.3 Implementation

This section gives details about the choices we made while developing an open-source library for tube programming.

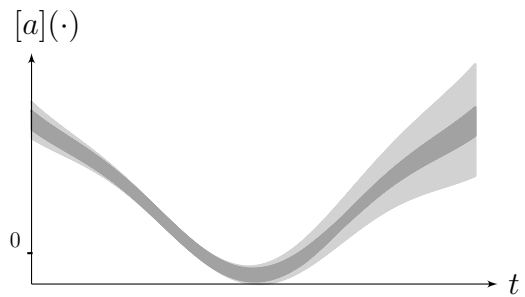
¹Note that in this example, t_1 and i could be variables of the CN. Here we do not try to estimate them but the tools presented in Chapter 5 will allow it.



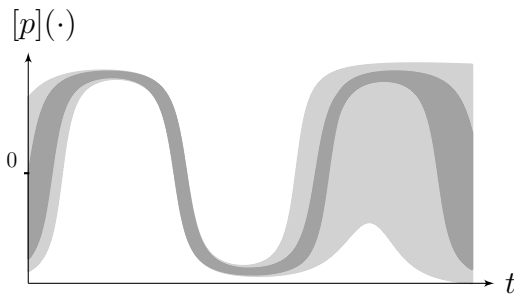
(a) Envelope of $y(\cdot)$, contracted by means of a local measurement $[i] \in \mathbb{R}$ at time t_1 . The observation is then propagated by using a differential contractor presented in Chapter 4.



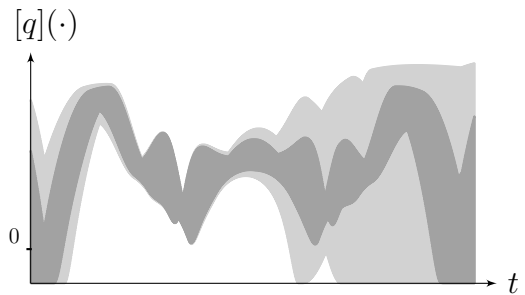
(b) $x(\cdot)$.



(c) $a(\cdot) = x(\cdot) + y(\cdot)$.



(d) $p(\cdot) = \arctan(y(\cdot))$.



(e) $q(\cdot) = 2 \sin\left(\frac{a(\cdot)}{2}\right) + \sqrt{2p(\cdot)}$.

Figure 3.6: Illustration of tubes contractions. Light gray areas represent the envelope of trajectories before contraction. Considering an improvement of the $y(\cdot)$ approximation, tubes $[a](\cdot)$, $[p](\cdot)$, $[q](\cdot)$ related to $[y](\cdot)$ by algebraic constraints can then be contracted. The final sets of solutions, obtained after applying contractors, are pictured in dark gray.

3.3.1 Data structure

Slices representation

There can be several ways to implement a tube. A computer representation based on a set of boxes that sample the tube over time has been mentioned in [Le Bars et al., 2012, Bethencourt and Jaulin, 2014, Rohou et al., 2017]. Our choice is to build a tube with a set of boxes representing slices of identical width, as depicted in Figure 3.7.

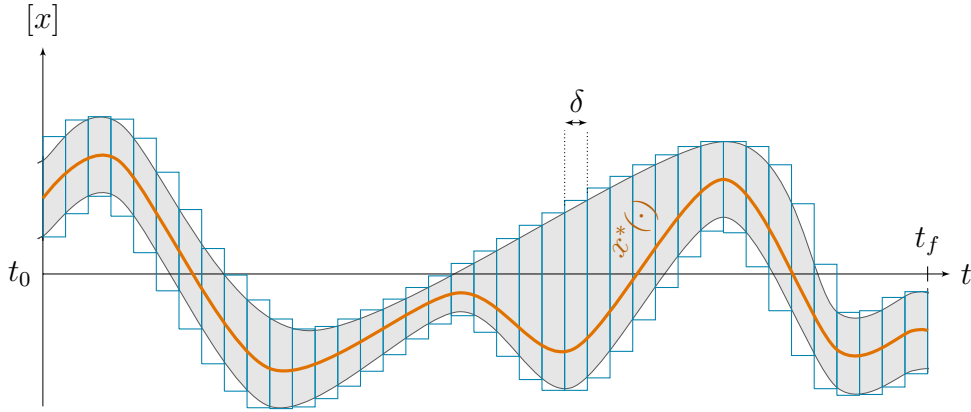


Figure 3.7: A tube $[x](\cdot)$ represented by a set of δ -width slices. This implementation can be used to enclose signals such as $x^*(\cdot)$.

More precisely, a n -dimensional tube $[\mathbf{x}](\cdot)$ with a sampling time $\delta > 0$ is implemented as a box-valued function which is constant for all t inside intervals $[k\delta, k\delta + \delta]$, $k \in \mathbb{N}$. The box $[k\delta, k\delta + \delta] \times [\mathbf{x}](t_k)$, with $t_k \in [k\delta, k\delta + \delta]$, is called the k^{th} slice of the tube $[\mathbf{x}](\cdot)$ and is denoted by $[\mathbf{x}](k)$,². The resulting approximation of a tube encloses $[\mathbf{x}^-(\cdot), \mathbf{x}^+(\cdot)]$ inside an interval of step functions $[\underline{\mathbf{x}}^-(\cdot), \overline{\mathbf{x}}^+(\cdot)]$ such that:

$$\forall t \in [t_0, t_f], \underline{\mathbf{x}}^-(t) \leq \mathbf{x}^-(t) \leq \mathbf{x}^+(t) \leq \overline{\mathbf{x}}^+(t). \quad (3.12)$$

This implementation then takes rigorously into account floating point precision when building a tube, thanks to reliable numerical libraries such as those presented in Section 2.5.2, page 70. Further computations involving $[\mathbf{x}](\cdot)$ will be based on

²When t_k belongs to the common boundary of two slices $[\mathbf{x}](k-1)$ and $[\mathbf{x}](k)$, the value $[\mathbf{x}](t_k)$ is $[\mathbf{x}](k-1) \cap [\mathbf{x}](k)$.

its slices, thus giving an outer approximation of the solution set. For instance, the lower bound of the integral of a tube, defined in Equation (3.8), is simply computed as the signed area of the region in the tx -plane that is bounded by the graph of $\underline{x}^-(t)$ and the t -axis, as pictured in Figure 3.8. The lower slice width δ , the higher the precision of the approximation.

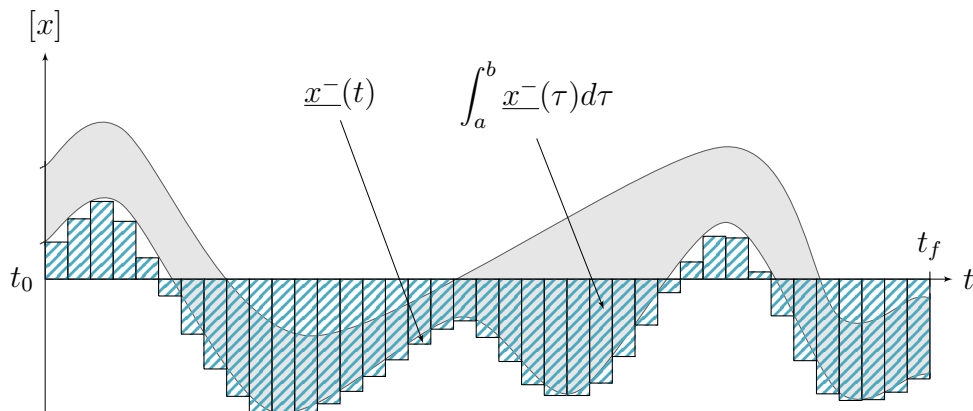


Figure 3.8: Outer approximation of the lower bound of $\int_a^b [x](\tau) d\tau$.

One should note that other kinds of implementations can be considered, such as a clever time discretization with non-constant slices width, allowing precise computations when required and an optimized representation in case of non-evolving trajectories. In this thesis however, it was decided to use a constant slice width implementation in order to keep further developments simple. On top of that, many concrete applications such as robotics are based on sensors that output values on a regular basis. The study of a custom time discretization will be the subject of future work. For instance, polynomial functions could also be used instead of slices, in order to better meet the trajectories enclosure. This technique remains to be studied.

Binary tree

A special attention can be paid to the data structure representing this list of slices. For now, our choice is to use a binary tree of height h in which each node is a synthesis of its children: the union box of the children slices. Hence, the root node summarizes the whole tube: $[t_0, t_f] \times [x] ([t_0, t_f])$. This structure allows recursive functions, fast access to the slices and quick evaluations. For instance, the test $\forall t \in [t_0, t_f], [x](t) \subset \mathbb{R}^+$ can be checked without any access to the leaves: the

information being already summarized in the root node. Figure 3.9 pictures such implementation.

Note that further information can be synthesized in this structure, such as derivatives over sub-domains that will speed up the evaluation of $\int_{[t_1]^{[t_2]}} [x](\tau) d\tau$.

3.3.2 Build a tube from real datasets

The proposed sliced representation is a straightforward solution to take into account not only analytical expressions but also datasets: any envelope of trajectories is affordable considering sufficiently accurate slice-width δ . Hence, a tube can be built to represent sensors' outputs such as altitude measurements obtained from a sonar.

In practice, off-the-shelf sensors provide data on a regular basis. However, the missing values between two pulses are unknown, which affects the reliability of our data representation. For instance, considering altitude measurements, we cannot ensure the sensing of any asperity in the ground: there may be non-assessable holes or peaks due to a too low sensor frequency f_e .

Since a tube is a time-continuous representation of trajectories, one may naturally come to the question: *what about the guaranteed representation along a complete slice?* We may tend towards a fine approximation of a trajectory if $\delta \gg \frac{1}{f_e}$, but the assessment cannot be considered as guaranteed.

There are two solutions to perform a reliable enclosure:

1. benefit from the continuous derivative of the signal – if available – which can be a tube itself such as $[\dot{x}](\cdot)$, depicting uncertainties too. In this case a reliable representation is affordable since the evolution of the trajectory between two measurements can be bounded;
2. use guaranteed sensors that output reliable values $[x]$ valid during an interval of time $[t]$ corresponding to a complete slice domain. Unfortunately, to our knowledge, such sensors do not exist yet.

We have seen in Section 2.5.1, page 69, how to build an interval enclosing a measurement with a 95% confidence rate. It is, however, difficult to evaluate such

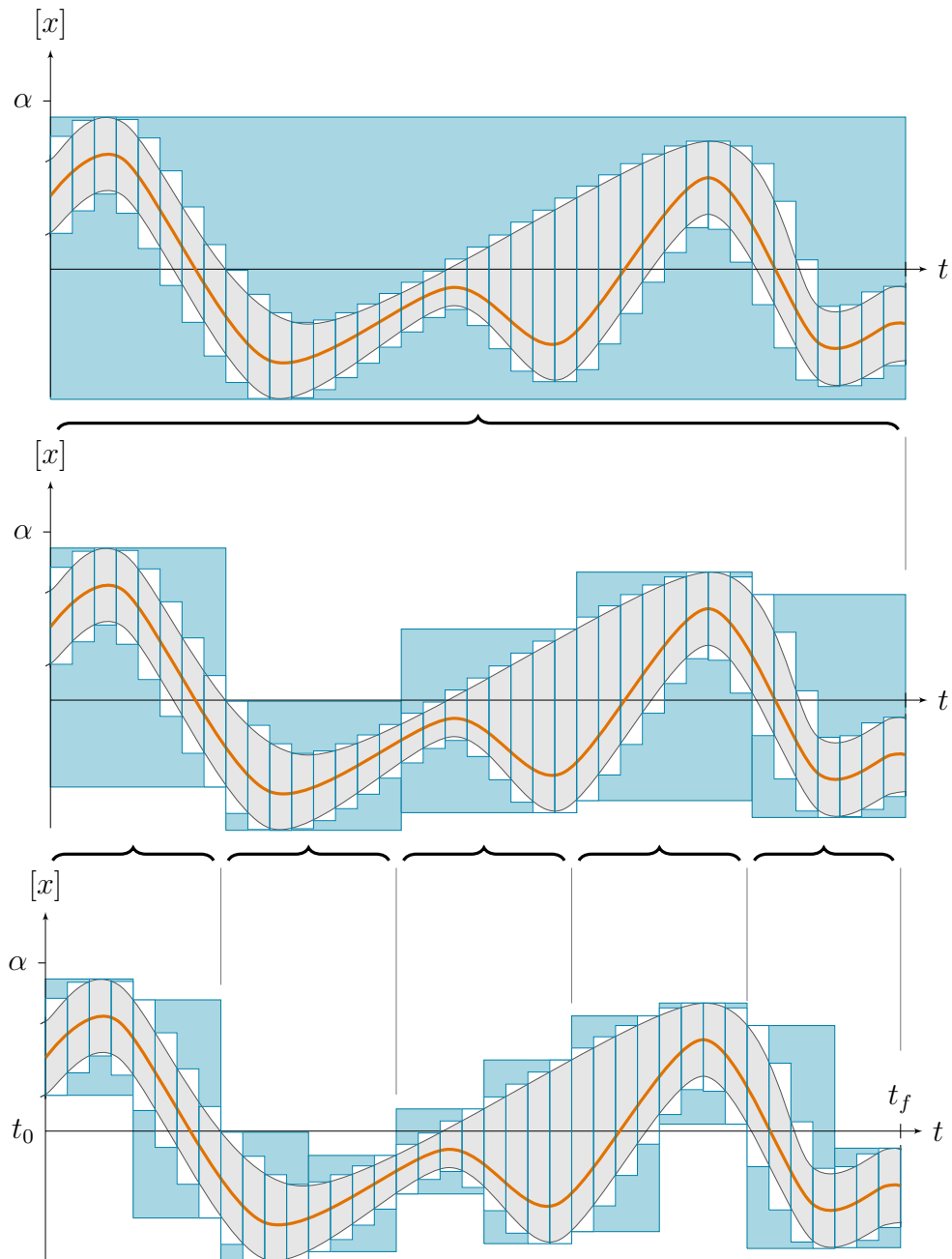


Figure 3.9: A tube implemented by means of a binary tree. Blue boxes are nodes from three non-consecutive levels of data abstraction. The root node (upper figure) summarizes by itself the whole range of feasible values covered by the trajectories, which is of interest for fast evaluations such as $\forall t \in [t_0, t_f], \forall x(\cdot) \in [x](\cdot), x(t) < \alpha$.

confidence with a tube made of these bounded measurements. A steady study on this topic would be welcomed.

For now, our choice is to build a tube from data by computing a piecewise linear interpolation $\mathbf{x}^{PL}(\cdot)$ between the measurements. We then define the k -th slice of $[\mathbf{x}](\cdot) : \mathbb{R} \rightarrow \mathbb{I}\mathbb{R}^n$ as a box:

$$[\mathbf{x}](k) \mapsto [t_k, t_{k+1}] \times \left([-2\sigma, 2\sigma]^n + \bigcup_{t=t_k}^{t_{k+1}} \mathbf{x}^{PL}(t) \right). \quad (3.13)$$

The sampling time δ of the tube is set so that each slice gathers a sufficiently robust amount of data.

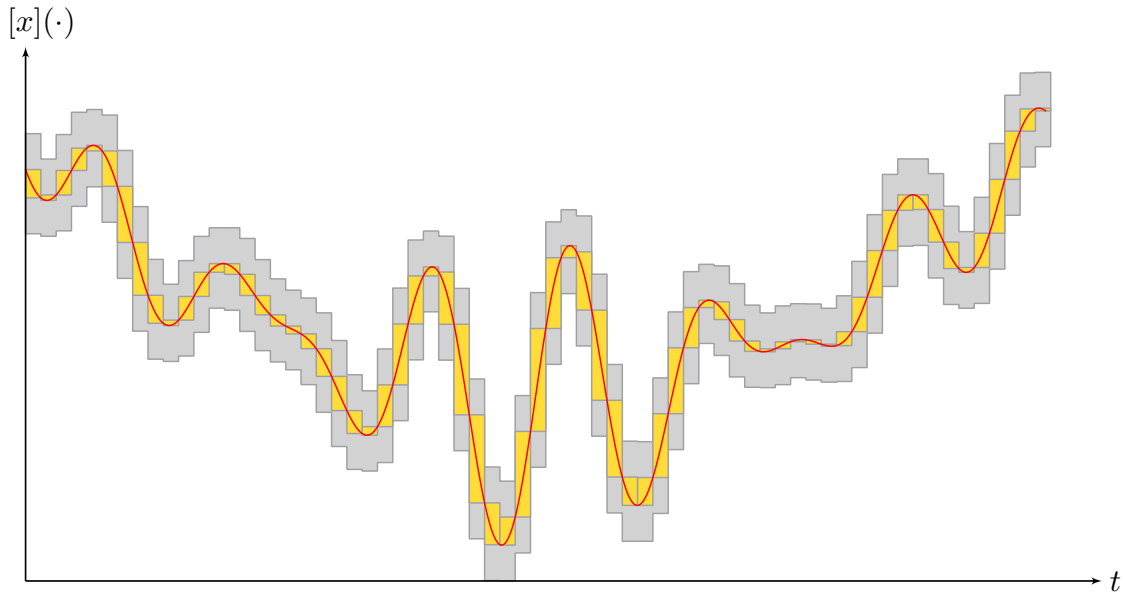


Figure 3.10: Illustration of Equation (3.13). The gray part \bullet is the final tube built from a linear interpolation plotted in red \bullet . Yellow boxes \bullet represent the minimal envelope of $\mathbf{x}^{PL}(\cdot)$ considering $\sigma = 0$.

3.3.3 *Tubex*, dedicated tube library

An optimized tube class has been implemented during this thesis and is available on <http://simon-rohou.fr/research/tubex-lib>. Most of the source code of the simulated examples presented in this document are provided in this library. The proposed framework is compatible with the contractor programming library *IBEX*³, dedicated

³<http://www.ibex-lib.org>

to constraints over real numbers [Chabert, 2017]. *TubeX* uses *VIBes*⁴ [Drevelle and Nicola, 2014] for visualizations.

3.4 Application: dead-reckoning of a mobile robot

The material presented in this chapter is sufficient to address a dead-reckoning problem⁵ based on the following evolution equation:

$$\dot{\mathbf{x}}(\cdot) = \mathbf{f}(\mathbf{x}(\cdot), \mathbf{u}(\cdot)). \quad (3.14)$$

Its differential aspect will be the main matter of the next chapter. For the moment, the resolution will be done as a simple integration of the derivatives over time, from t_0 .

3.4.1 Test case

We propose the following reproducible example in order to encourage future comparisons and criticism of the approach defended in this document. The example will gradually become more complex in the next chapters, justifying the need for new resolution tools.

A robot \mathcal{R} is described by its state $\mathbf{x} = (x_1, x_2, \psi, \vartheta)^\top$ where (x_1, x_2) depicts its location, ψ its heading and ϑ its speed. The state evolution is modeled as:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{\psi} \\ \dot{\vartheta} \end{pmatrix} \xrightarrow{\mathbf{f}} \begin{pmatrix} \vartheta \cos(\psi) \\ \vartheta \sin(\psi) \\ u_1 \\ u_2 \end{pmatrix}. \quad (3.15)$$

The state $\mathbf{x}(t)$ is submitted to the input $\mathbf{u}(t)$, bounded-valued. We propose this analytical expression to encourage comparisons, but any formula or dataset could be used.

$$\mathbf{u}(t) \in \begin{pmatrix} -9/20 \cos(t/5) \\ 1/10 + \sin(t/4) \end{pmatrix} + \frac{1}{1000} \begin{pmatrix} [-1, 1] \\ [-1, 1] \end{pmatrix}. \quad (3.16)$$

⁴<http://enstabretagnerobotics.github.io/VIBES>

⁵In the literature, this problem is also known as *simulation*.

In addition, the initial state \mathbf{x}_0 depicting the robot at t_0 is assumed to be bounded such that

$$\mathbf{x}_0 \in \begin{pmatrix} [-1, 1] \\ [-1, 1] \\ \pi/2 + [-0.01, 0.01] \\ [-0.01, 0.01] \end{pmatrix}. \quad (3.17)$$

The simulation will be run from $t_0 = 0$ to $t_f = 64$.

3.4.2 Constraint Network

As for the example presented in Section 2.3.3, we will write the following CN depicting the problem.

$$\text{CN:} \left\{ \begin{array}{l} \mathbf{Variables: } \mathbf{x}(\cdot), \mathbf{v}(\cdot), \mathbf{u}(\cdot), \mathbf{x}_0 \\ \mathbf{Constraints:} \\ \quad 1. v_1(\cdot) = x_4(\cdot) \cdot \cos(x_3(\cdot)) \\ \quad 2. v_2(\cdot) = x_4(\cdot) \cdot \sin(x_3(\cdot)) \\ \quad 3. v_3(\cdot) = u_1(\cdot) \\ \quad 4. v_4(\cdot) = u_2(\cdot) \\ \quad 5. \mathbf{x}(\cdot) = \int_{t_0}^{\cdot} \mathbf{v}(\tau) d\tau + \mathbf{x}_0 \\ \mathbf{Domains: } [\mathbf{x}](\cdot), [\mathbf{v}](\cdot), [\mathbf{u}](\cdot), [\mathbf{x}_0] \end{array} \right. \quad (3.18)$$

The components of $\mathbf{v}(\cdot)$ are intermediate variables. Their domains are the corresponding tubes $[v_1](\cdot)$, $[v_2](\cdot)$, *etc.* Note that the variable \mathbf{x}_0 cannot be refined in this process. If we prefer to let \mathbf{x}_0 appear among the variables for ease of understanding, another way would be to add further constraints such that $x_1(t_0) \in [-1, 1]$.

3.4.3 Resolution

Tubes $[\mathbf{x}](\cdot)$ and $[\mathbf{v}](\cdot)$ are initialized to $[-\infty, \infty] \forall t \in [t_0, t_f]$, contrary to $[\mathbf{u}](\cdot)$ expressed by means of tube analysis from Equation (3.16); the same for the box $[\mathbf{x}_0]$ and Equation (3.17).

The above constraints are not *primitive*: some of them can be broken down such as

$$v_1(\cdot) = x_4(\cdot) \cdot \cos(x_3(\cdot)) \iff \begin{cases} a(\cdot) = \cos(x_3(\cdot)), \\ v_1(\cdot) = x_4(\cdot) \cdot a(\cdot) \end{cases} . \quad (3.19)$$

The estimation is run by applying contractors for each primitive constraint. The considered contractors are trivial, such as \mathcal{C}_+ presented in Equation (3.10), on page 81. The integral constraint is run by intersecting $[\mathbf{x}](\cdot)$ with the primitive of the tube $[\mathbf{v}](\cdot)$, see Equation (3.8), page 79. Again, we emphasize that the order of the constraints does not impact the result of the approximation.

Figure 3.11 shows the projection of the tubes $[x_1](\cdot) \times [x_2](\cdot)$ after the resolution. Figure 3.12 depicts the amount of uncertainty resulting from the guaranteed enclosure of the actual state. As expected in dead-reckoning methods, the error is quadratic. Results are obtained in 1.51 second on a conventional computer, in 3 iteration steps, with a slice width $\delta = 0.005$. The final position vector $\mathbf{p}(t_f) = (x_1(t_f), x_2(t_f))^T$ is approximated as:

$$\mathbf{p}(t_f) \in [26.63, 50.06] \times [38.58, 67.37]. \quad (3.20)$$

3.5 Discussions

This section discusses some limits and perspectives of this approach.

3.5.1 Limits

Tubes do not appear as a straightforward solution to consider *hybrid* constraints such as: *if* $x < 1$, *then* $\dot{x} = 1$, *else*: $x^+ = 0$. These constraints implies discontinuities that may be difficult to properly handle with tubes. Other approaches, see for

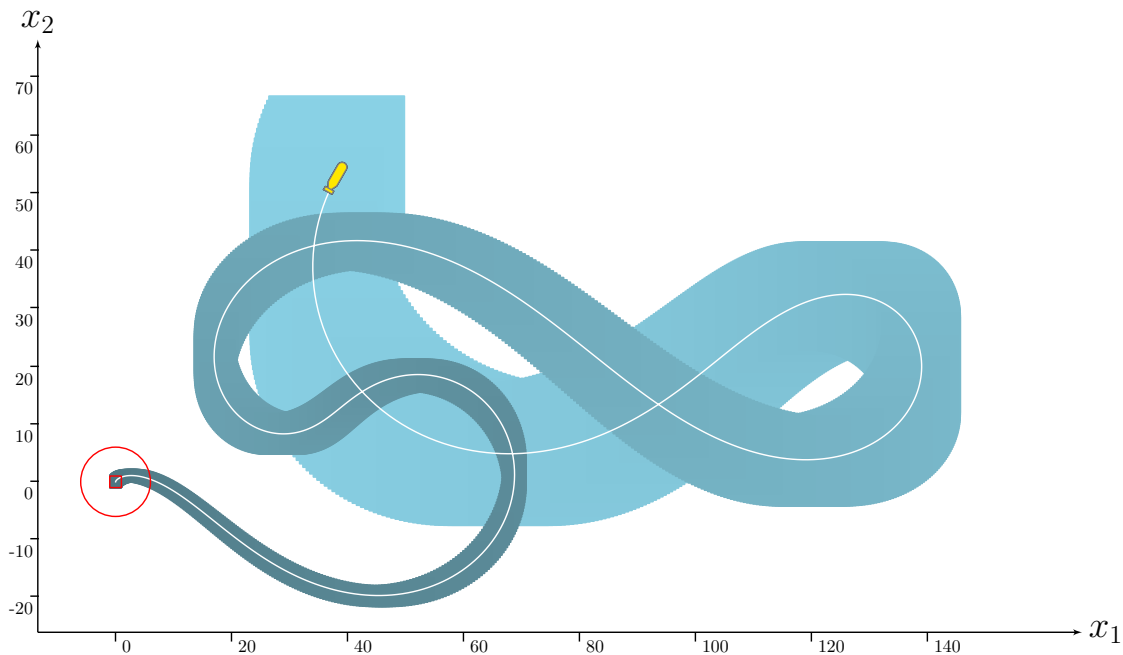


Figure 3.11: Simulation of a mobile robot from state inputs and bounded initial conditions. The initial position $(0, 0)$ is only known to belong to the red box \bullet . The true poses of the robot are plotted in white over the estimated tubes $[x_1](\cdot) \times [x_2](\cdot)$ projected in blue \bullet .

instance the *Acumen*⁶ library [Taha et al., 2015, Duracz, 2016], provide appropriate tools to address this class of conditional constraints, where they often fail to propagate information over the trajectories domain in a backward way. Both methods could be coupled to take advantage of each approach.

3.5.2 Extract the most probable trajectory from a tube

We have seen in Chapter 2 that when working with set-membership methods, it is difficult to state probability distributions over the sets. This also applies with tubes, which is among the main drawbacks of the approach especially when it comes to compare with usual approaches that deal with what we call *degenerate* solutions in our field. Hence, it is often not relevant to consider a given trajectory in the tube (*e.g.* the center of the tube) and perform a comparison with the actual value. Any trajectory is a good candidate for such comparison.

⁶<http://www.acumen-language.org>

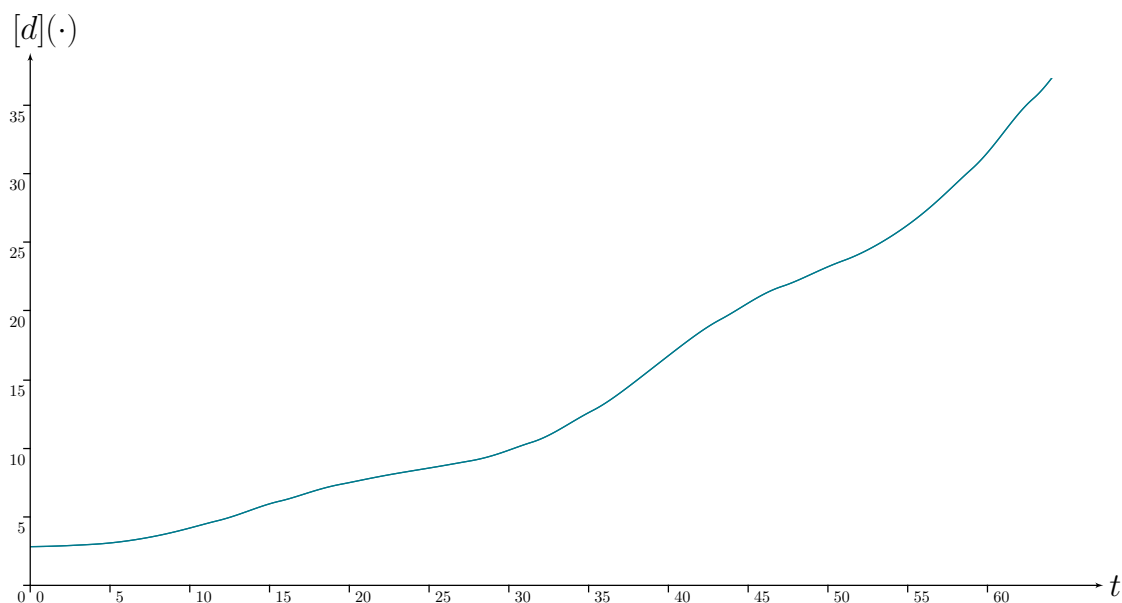


Figure 3.12: Thicknesses of the position estimation $[x_1](\cdot) \times [x_2](\cdot)$. We define $d : \mathbb{R}^2 \rightarrow \mathbb{R}$ the diagonal of a position box $[x_1] \times [x_2]$: $d([\mathbf{x}]) = \sqrt{(x_1^+ - x_1^-)^2 + (x_2^+ - x_2^-)^2}$. This depicts in the worst case the error between the unknown truth and any trajectory within the position tube. The initial non-zero uncertainty $\sqrt{8}$ is due to the bounded initial state, see the red box in Figure 3.11.

A convincing example is the following. Let us consider a mobile robot that has to move from a point A to a point B while avoiding an obstacle. It then has two optimized solutions to reach its destination: leaving the object to the left or to the right, see Figure 3.13. The center of the tube is surely not a solution to consider in this example.

We must keep in mind that a central trajectory (*e.g.* the center of the tube) may not be compliant with the constraints defining this tube. Our approach only allows one to guarantee a space where the actual solution cannot be. This remark is closely linked to the use of tubes for path planning algorithms.

3.5.3 Application to path planning

If a tube does not only contain feasible trajectories, it is however a proof that trajectories outside the tube are not compliant with the considered constraints.

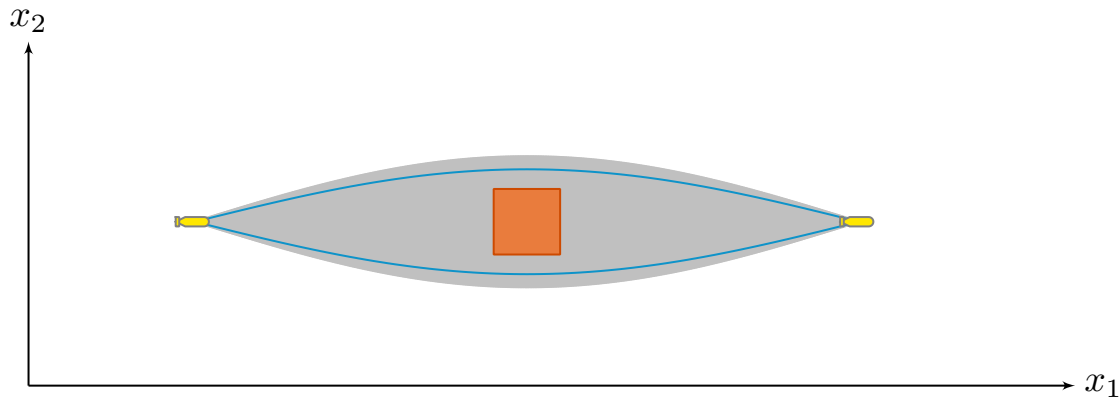


Figure 3.13: Top view of a mobile robot avoiding an obstacle \bullet . Blue lines \bullet are feasible trajectories that let the object to the left or to the right. The corresponding tube is depicted in gray \bullet .

Hence, tubes can be used to reject a path.

Our approach could then reduce the computational burden of path-planning algorithms, preventing from wrong path proposals especially in case of strong uncertainties or non-linearities [Pruski and Rohmer, 1997]. But it must be coupled with other tools to eventually extract a feasible trajectory from the tube.

Further, tubes are well suited to guarantee collision avoidance algorithms. Many applications such as the estimation of collisions [Serra et al., 2015] between an asteroid and a mobile satellite could be dealt by means of tubes.

3.6 Conclusion

When dealing with dynamical systems, we often face differential equations and non-linearities, which make solving computations difficult. Several tools exist for this purpose but most of them badly behave in case of strong non-linearities or uncertainties. In this chapter, we introduced a recent framework for which the variables of interest are trajectories submitted to both algebraic and differential equations. The philosophy is the same as the one presented in Chapter 2: formalize a problem by defining a constraint network and then apply these constraints to sets of trajectories.

However, we were only able to solve a differential equation from its initial

value (so-called IVP). Further tools have to be investigated to completely deal with differential systems, taking into account observations on trajectories at any time and propagating this information over the whole domain. This has been the object of [Le Bars et al., 2012, Bethencourt and Jaulin, 2014] but the proposed contractors were not rigorously designed, thus compromising the reliability of the results. The next Chapter 4 aims at providing a reliable contractor to deal with differential equations.

Part II

Constraints-related contributions

In order to fully expand the contractor programming approach to dynamical systems, new constraints have to be studied. The purpose of this second part is to focus on two elementary constraints related to differential equations. The proposed contractors will then be applied in Part III for specific robotic problems involving time uncertainties.

Our contributions are twofold. In Chapter 4 we propose a new tube contractor to temporally propagate an information in both forward and backward ways, considering a differential constraint $\dot{x}(\cdot) = v(\cdot)$. Another contractor is provided in Chapter 5 to evaluate a tube based on the constraint $z = y(t)$, which leads to time uncertainties considerations.

Trajectories under differential constraints

Contents

4.1 Introduction	100
4.1.1 The differential problem	100
4.1.2 Attempts with set-membership methods	100
4.1.3 Contribution of this thesis	103
4.2 Differential contractor for $\mathcal{L}_{\frac{d}{dt}} : \dot{x}(\cdot) = v(\cdot)$	104
4.2.1 Definition and proof	104
4.2.2 Contraction of the derivative	109
4.2.3 Implementation	110
4.3 Contractor-based approach for state estimation	114
4.3.1 Constraint network of state equations	114
4.3.2 Fixed-point propagations	117
4.3.3 Theoretical example of interest $\dot{x} = -\sin(x)$	118
4.4 Robotic applications	121
4.4.1 Causal kinematic chain	121
4.4.2 Higher order differential constraints	123
4.4.3 Kidnapped robot problem	125
4.4.4 Actual experiment with the <i>Daurade</i> AUV	126
4.5 Conclusion	130

4.1 Introduction

As mentioned in Chapter 2, interval-based methods effortlessly handle non-linear situations while ensuring guaranteed results. We have seen that we can also deal with trajectories by means of tubes presented in Chapter 3. However, it remains to address the differential problem.

4.1.1 The differential problem

Actually, the robotic example provided in Section 3.4 from page 90 onwards does not represent the range of state estimation problems we usually encounter, especially in mobile robotics. Indeed, a convincing approach must be able to assess observations along time while ensuring the evolution of the state. And sometimes, as for the kidnapped robot problem, the situation is even more complicated as initial conditions are not known. Hence, there is a need to fully deal with Ordinary Differential Equations (ODEs) in the most generic way.

Formally, we consider the problem of *guaranteed integration* of dynamical systems (see *e.g.* [Konečný et al., 2016]) of the form

$$\dot{\mathbf{x}}(\cdot) = \mathbf{f}(\mathbf{x}(\cdot), \mathbf{u}(\cdot)). \quad (4.1)$$

One of the main motivations of guaranteed integration methods is to develop reliable cyber-physical systems such as *Acumen*¹ [Taha et al., 2015] for dynamical systems simulation and verification.

4.1.2 Attempts with set-membership methods

Our problem corresponds to the area of *interval integration* [Moore, 1979, Berz, 1996]. Set-membership methods allow the computation of reliable bounds of sets containing the solutions for the differential equation, according to the model \mathbf{f} or its inclusion, an initial condition \mathbf{x}_0 or uncertain parameters [Raïssi et al., 2004]. Furthermore, the same methods provide a way to prove the absence of solution when the computed set is empty. This approach differs from usual

¹<http://www.acumen-language.org>

resolution techniques such as Euler, Taylor or Runge-Kutta methods, with which only punctual approximations of solutions are computed.

The guaranteed resolution of ODEs has gathered a part of the community of numerical analysis around the so-called Initial Value Problem (IVP).

Initial Value Problem

This problem consists in computing the solution of an ODE, supposed unique, over time from the knowledge of an initial condition. A guaranteed enclosure of this solution can be obtained by bounding all uncertainties, including those due to time discretization. Hence, the precision of the algorithms will only affect the accuracy of the enclosure, and not its reliability. More precisely, from an initial box $[\mathbf{x}](t_0)$ representing a bounded initial state at time $t = t_0$, the guaranteed integration provides a set of techniques to compute a box-valued function $[\mathbf{x}](\cdot)$ (or *tube*) containing all feasible solutions of the ODE. There exist several methods based on the interval extensions of usual resolution techniques such as the Euler one [Moore, 1979].

A strong effort from this community has then triggered the development of several efficient libraries for interval integration, such as *Vnode*² [Nedialkov and Jackson, 2000], *Cosy* [Revol et al., 2005], *DynIBEX*³ [Alexandre dit Sandretto and Chapoutot, 2016] or *CAPD*⁴ [Wilczak et al., 2017]. These libraries are used in robotics and automatic control to verify dynamical properties of non-linear systems [Ramdani and Nedialkov, 2011] or to compute reachable sets [Collins and Goldsztejn, 2008, Goubault et al., 2014]. They are also used by mathematicians to prove conjectures [Tucker, 1999, Goldsztejn et al., 2011].

These methods may present drawbacks such as large computation times or pessimistic enclosures, which may limit their use for specific problems of validation and system safety. Efforts are continuing to overcome these limitations, providing now tight enclosures of solutions of a wide class of ODEs [Nedialkov et al., 1999]. Nevertheless, these techniques may be difficult to configure and do not provide the level of genericity we are looking for our applications: the pure IVP does not represent by itself the diversity of state estimation problems. This limitation motivated new constraint-based approaches.

²<http://www.cas.mcmaster.ca/~nedialk/Software/VNODE/doc/webpage/main.htm>

³<http://perso.ensta-paristech.fr/~chapoutot/dynibex/>

⁴<http://capd.ii.uj.edu.pl>

Constraint-based approaches

In this context, there is a natural desire to expand the constraint programming paradigm to completely deal with ODEs, with the aim of benefiting from the simplicity and genericity aspects of this approach.

The concept appeared in [Deville et al., 1998, Janssen et al., 2001, Janssen et al., 2002] with encouraging results on the IVP. The proposed method is based on successive integration stages for which a two-step process is run: a predictor stage to compute an enclosure of the solution at a given time and a corrector one to reduce it. This approach stands on constraint techniques to perform these steps but it does not allow temporal constraints. Therefore, the level of genericity we are looking for cannot be reached with this approach.

One should also mention the work of [Hickey, 2000], providing a new language to consider constraints over variables of \mathbb{R} and functions. The so-called Analytic Constraint Logic Programming (ACLP) appeared as a new approach to solve ODEs beyond the restrictive IVP, thus addressing more generic problems. The associated language is elegant and clear which demonstrates that complex problems can be solved with a simple syntax. However, the related solver was aimed at enclosing real functions evaluations $\mathbf{x}(t)$ and was restricted to analytical functions. We still need to approximate complete trajectories, deal with actual datasets and constraint our problem with a wide range of temporal relations such as uncertain time evaluations.

A further contribution in this direction is the one of [Cruz and Barahona, 2003], introducing the concept of Constraint Satisfaction Differential Problem (CSDP) by representing an ODE linked with additional related information. Contrary to the approach of [Hickey, 2000], this new framework computes a solution set of feasible trajectories. The ODE is then restricted with constraints involving real variables, such as the *Maximum Restriction* presented in the paper. Nevertheless, the method seems less generic or more complex to use: the variable representing the ODE – what we call a *trajectory* – is not considered at the very same level as other vector or trajectory variables involved in a problem. Furthermore, the consideration of temporal constraints such as time uncertainties, delays, inter-temporalities, *etc.*, still seems unaffordable with the presented tools.

In this chapter, we will stay focused on the constraint paradigm and propose a contractor-programming approach to deal with ODEs.

4.1.3 Contribution of this thesis

A contractor-based approach

We will expand the contractor programming concept presented in Section 2.3.2, page 56, without loss of expressivity. The decomposition of any ODE will involve an elementary differential constraint denoted $\mathcal{L}_{\frac{d}{dt}}$. This chapter provides the related contractor $\mathcal{C}_{\frac{d}{dt}}$ to apply the constraint on tubes. We will see that a wide range of ODEs can be dealt with this contractor approach by combining both this differential operator and any algebraic one.

Furthermore, unlike most of other approaches, we will see that our method does not stand on the Picard-Lindelöf theorem: the assessed trajectories are not necessarily Lipschitz continuous in our framework.

Towards a guarantee that was not met before

A first step has been made in this very approach in [Le Bars et al., 2012] and [Bethencourt and Jaulin, 2014]. The introduced contractors provided the expected level of genericity, but the definitions and algorithms were only based on sliced tubes and the results were not guaranteed with respect to this time discretization.

More precisely, it was implicitly assumed that the floating points were dense enough to represent all real numbers and that the sampling time was infinitely small. The guarantee with respect to time discretization cannot be obtained without taking into account time uncertainties, as we do in this document.

To understand our statement of non-guarantee, let us take a tube $[x](\cdot)$ with a sampling time δ such that

$$[x](k_0) = [1, 5], \quad [x](k_1) = [-1, 1], \quad [x](k_2) = [1, 2]. \quad (4.2)$$

Using the integral as defined in [Le Bars et al., 2012, Section II], we obtain that if $t_1 \in [0, \delta]$, $t_2 \in [2\delta, 3\delta]$, then: $\int_{t_1}^{t_2} [x](\tau) d\tau = \delta \cdot [1, 5] + \delta \cdot [-1, 1] + \delta \cdot [1, 2] = \delta \cdot [1, 8]$. This is not correct since we may have $t_1 = \delta$ and $t_2 = 2\delta$ and thus $\int_{t_1}^{t_2} x(\tau) d\tau$ could be equal to $-\delta$. From [Aubry et al., 2013], we know that the correct result should be $\int_{t_1}^{t_2} x(\tau) d\tau \in \int_{[0, \delta]}^{[2\delta, 3\delta]} [x](\tau) d\tau = [-\delta, 8\delta]$. In practice, the discretization error has not much influence on the final result if δ is sufficiently small, but it is difficult to quantify such error.

In this chapter, we provide discretization-free and guaranteed definitions of a new contractor for differential constraints. This work has been the subject of a collaborative study between Luc Jaulin, Lyudmila Mihaylova, Fabrice Le Bars, Sandor M. Veres and the author, which led to the publication of [Rohou et al., 2017].

4.2 Differential contractor for $\mathcal{L}_{\frac{d}{dt}} : \dot{x}(\cdot) = v(\cdot)$

This section provides a new contractor to apply a differential constraint, the canonic expression of which being denoted by $\mathcal{L}_{\frac{d}{dt}}$.

4.2.1 Definition and proof

We consider the following elementary relation:

$$\mathcal{L}_{\frac{d}{dt}} : \left\{ \begin{array}{l} \mathbf{Variables:} \ x(\cdot), v(\cdot) \\ \mathbf{Constraints:} \\ \quad 1. \ \dot{x}(\cdot) = v(\cdot) \\ \mathbf{Domains:} \ [x](\cdot), [v](\cdot) \end{array} \right. \quad (4.3)$$

The related trajectories are one-dimensional functions but an extension to the multidimensional case is trivial. The variable $v(\cdot)$ depicts the derivative⁵ of $x(\cdot)$. The knowledge of both $x(\cdot)$ and $v(\cdot)$ is respectively given by the tubes $[x](\cdot)$ and $[v](\cdot)$. A new contractor denoted $\mathcal{C}_{\frac{d}{dt}}$ will aim at reducing these tubes based on the differential constraint, without losing any solution. It will imply contractions that can be propagated along the whole domain of the tubes, in both a *forward* and a *backward* way⁶.

⁵The notation $v(\cdot)$ recalls the velocity of a robot: the derivative of its position $x(\cdot)$.

⁶The reader familiar with constraint propagation techniques will note that these *forward/backward* terms are not those of this literature. Here, we only speak about temporal propagations.

Proposition 4.1

The operator $\mathcal{C}_{\frac{d}{dt}}$ is a contractor for the constraint $\mathcal{L}_{\frac{d}{dt}}$ and is defined by:

$$\begin{pmatrix} [x](t) \\ [v](t) \end{pmatrix} \xrightarrow{\mathcal{C}_{\frac{d}{dt}}} \begin{pmatrix} \bigcap_{t_1=t_0}^{t_f} \left([x](t_1) + \int_{t_1}^t [v](\tau) d\tau \right) \\ [v](t) \end{pmatrix}, \quad (4.4)$$

where $[t_0, t_f]$ is the definition domain of both $[x](\cdot)$ and $[v](\cdot)$.

Proof of Proposition 4.1

To be a contractor, $\mathcal{C}_{\frac{d}{dt}}$ needs to satisfy both the contraction and the consistency properties, given in Definition 3.3, page 81.

— **Contraction property.**

- proof for $[x](\cdot)$: after contraction, $[x](t)$ is calculated as

$$[x](t) = \bigcap_{t_1=t_0}^{t_f} \left([x](t_1) + \int_{t_1}^t [v](\tau) d\tau \right).$$

If we only focus on $t_1 = t$, then $[x](t)$ is shown to be a subset of $\left([x](t) + \int_t^t [v](\tau) d\tau \right) = ([x](t) + 0)$. Therefore, $\forall t \in [t_0, t_f]$, $[x](t)$ is at least contracted by itself, thus certifying the contraction property;

- as for $[v](\cdot)$, the demonstration is trivial: the tube is only contracted by itself.

— **Consistency property.**

The goal is to prove that a solution cannot be lost.

- proof for $[x](\cdot)$: we have to prove that for two trajectories $x(\cdot) \in [x](\cdot)$ and $v(\cdot) \in [v](\cdot)$ such that $\dot{x}(\cdot) = v(\cdot)$, we have:

$$\forall t \in [t_0, t_f], x(t) \in \bigcap_{t_1=t_0}^{t_f} \left([x](t_1) + \int_{t_1}^t [v](\tau) d\tau \right). \quad (4.5)$$

Consider a generic constraint $\mathcal{L}_f : \mathbf{a} = \mathbf{f}(\mathbf{a}, \mathbf{b})$, $\mathbf{a} \in [\mathbf{a}]$, $\mathbf{b} \in [\mathbf{b}]$. Hence, $\mathbf{a} \in [\mathbf{a}] \cap \mathbf{f}([\mathbf{a}], [\mathbf{b}])$. When combining several constraints \mathcal{L}_{f_i} , then $\mathbf{a} \in [\mathbf{a}] \cap \left(\bigcap_i \mathbf{f}_i([\mathbf{a}], [\mathbf{b}]) \right)$.

Now, from $\dot{x}(\cdot) = v(\cdot)$, let us take:

$$f_{t_1}(x(\cdot), v(\cdot)) = x(t) = x(t_1) + \int_{t_1}^t v(\tau) d\tau. \quad (4.6)$$

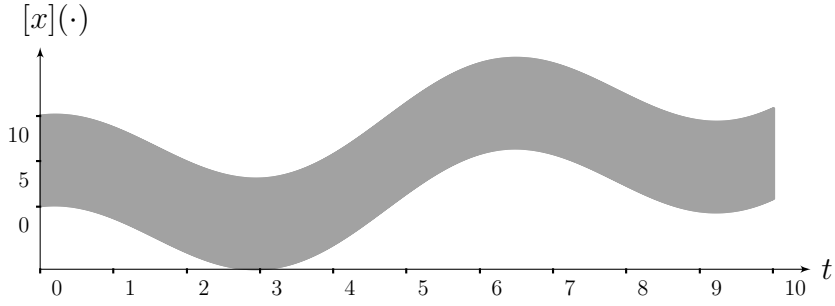
Then,

$$\begin{aligned} x(t) &\in [x](t) \cap \left(\bigcap_{t_1=t_0}^{t_f} f_{t_1}([x](\cdot), [v](\cdot)) \right) \\ &= [x](t) \cap \left(\bigcap_{t_1=t_0}^{t_f} \left([x](t_1) + \int_{t_1}^t [v](\tau) d\tau \right) \right) \\ &= \bigcap_{t_1=t_0}^{t_f} \left([x](t_1) + \int_{t_1}^t [v](\tau) d\tau \right). \end{aligned} \quad (4.7)$$

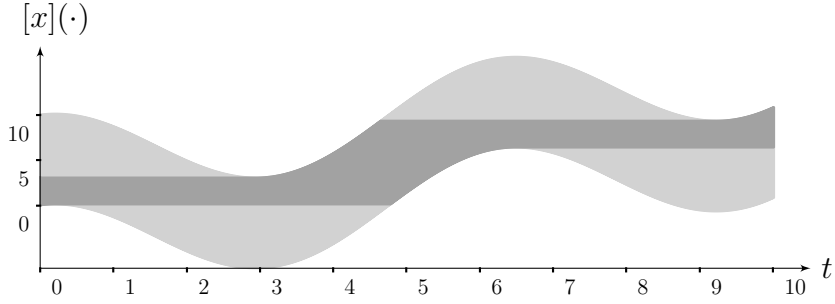
– proof for $[v](\cdot)$: trivial. ■

This contractor is useful to reach a consistency state between a tube of trajectories and its envelope of derivatives. Two examples are provided in Figures 4.1 and 4.2. More practically, when an observation constraints the set $[x](\cdot)$ at a given time t , then $\mathcal{C}_{\frac{d}{dt}}$ can be used to *smooth* the tube $[x](\cdot)$: a propagation of the differential constraint is expected in both forward (from t to t_f) and backward (from t to t_0) ways. This will be discussed in Section 4.3.

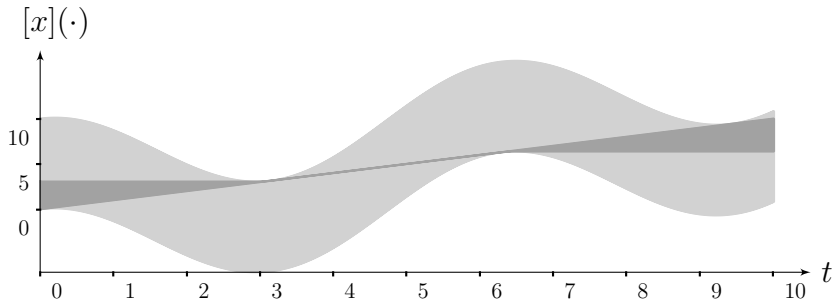
Note that a single application of $\mathcal{C}_{\frac{d}{dt}}$ is sufficient to reach a consistency state over the domains; an iterative method would not provide thinner results. One would think that $\mathcal{C}_{\frac{d}{dt}}$ is minimal but this hypothesis remains to be studied. An algorithm for $\mathcal{C}_{\frac{d}{dt}}$ will be provided in Section 4.2.3.



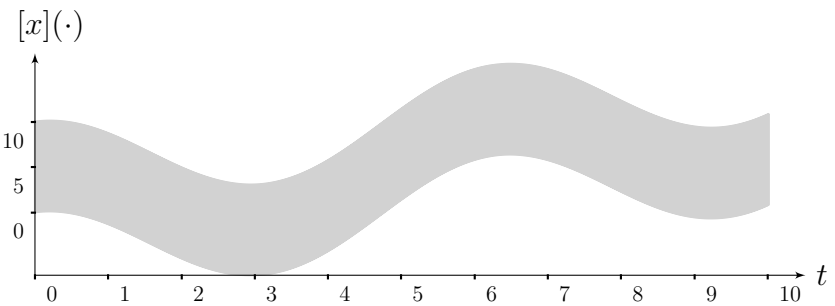
(a) Case of $[v](t) = [-\infty, \infty] \forall t$. No contraction.



(b) Case of $[v](t) = [0, \infty] \forall t$, equivalent to an *increasing* constraint.

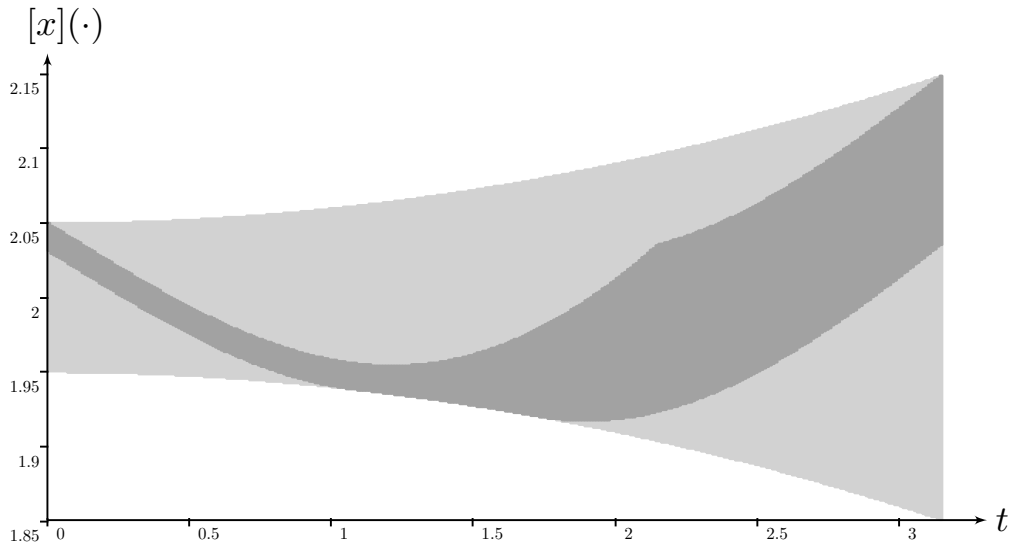


(c) Case of $[v](t) = [0, 1] \forall t$.

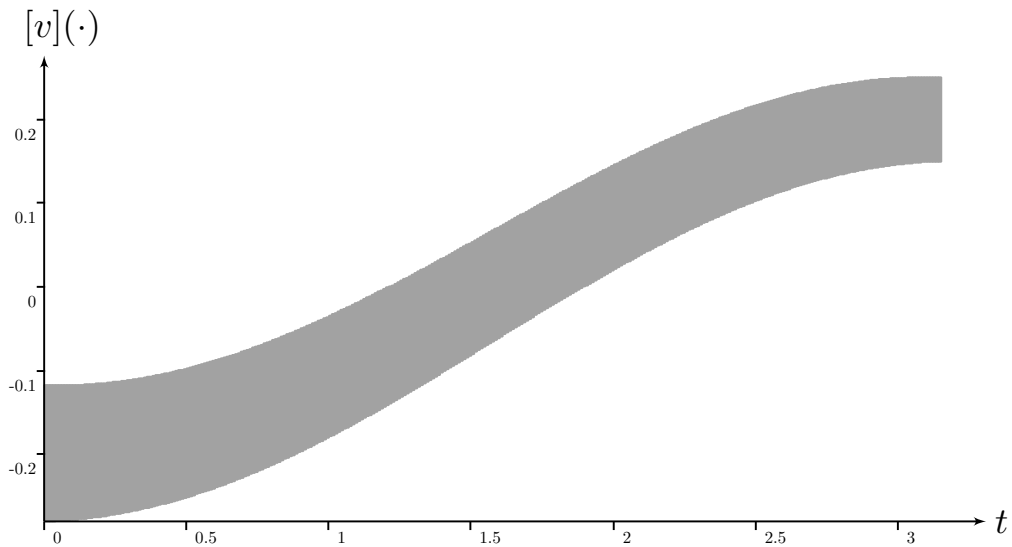


(d) Case of $[v](t) = [-\infty, 0] \forall t$, equivalent to a *decreasing* constraint, and resulting in a contraction to the empty set since the tube does not contain monotonically decreasing trajectories.

Figure 4.1: An arbitrary tube is contracted by $\mathcal{C}_{\frac{d}{dt}}$ to the envelope of trajectories $x(\cdot)$ that meet the constraint $\dot{x}(\cdot) = v(\cdot)$. The dark gray part is the tube after the contraction step. Several cases are illustrated with different derivative tubes. They are defined as constant for ease of understanding, but any set of derivatives could be considered.



(a) An arbitrary tube $[x](\cdot)$ contracted by $\mathcal{C}_{\frac{d}{dt}}$.



(b) An arbitrary tube $[v](\cdot)$ enclosing the set of feasible derivatives.

Figure 4.2: Another example of a consistency state reached with $\mathcal{C}_{\frac{d}{dt}}$ over a set of trajectories and their feasible derivatives. Only $[x](\cdot)$ is contracted. We will prove in Section 4.2.2 that $[v](\cdot)$ cannot be contracted when $[x](\cdot)$ is not a degenerate tube. Note that all the feasible derivatives in $[v](\cdot)$ are negative over $[0, 1]$ and so the contraction of $[x](\cdot)$ preserves decreasing trajectories over this part of the domain. Similarly, $[v](\cdot)$ is positive over $[2, 3]$ which corresponds to increasing trajectories kept in $[x](\cdot)$ after contraction.

Conditions on the variables $x(\cdot)$ and $v(\cdot)$

Contrary to other usual interval methods that solve the IVP, we do not need the Picard-Lindelöf theorem⁷ to perform the approximation of $x(\cdot)$ from its derivative. Indeed, the guarantee of the existence of a solution is not mandatory for our contractor, that will only remove infeasible trajectories from the domains. The contraction process may even result in an empty set for $[x](\cdot)$ for an ill-posed problem.

Therefore, we emphasize that the variables are not necessarily Lipschitz continuous in this framework, contrary to other approaches standing on the Picard-Lindelöf theorem.

4.2.2 Contraction of the derivative

One should note that the tube $[x](\cdot)$ may be contracted while the estimation of the derivative signal, represented by $[v](\cdot)$, will remain the same. Indeed, the evolution of any trajectory in $[x](\cdot)$ cannot be known, except for degenerate tubes without thickness. The derivative $\dot{x}(\cdot) \in [v](\cdot)$ could then be of any arbitrary value. Therefore, no information from $[x](\cdot)$ can be propagated back to $[v](\cdot)$. This is formalized and proved hereinafter.

Lemma 4.1

Consider the constraint $\dot{x}(\cdot) = v(\cdot)$ and two tubes $[x](\cdot)$, $[v](\cdot)$ such that there exists $c(\cdot)$ differentiable and $\varepsilon > 0$ with $c(\cdot) + [-\varepsilon, \varepsilon] \subset [x](\cdot)$. Then for all (v_1, t_1) , there exists a trajectory $x(\cdot) \in [x](\cdot)$ such that $\dot{x}(t_1) = v_1$. As a consequence, no contraction can be expected for $[v](\cdot)$ except in the cases of empty or degenerate tubes, where $[x](\cdot)$ has no uncertainty for some consecutive times.

Proof of Lemma 4.1

The function

$$a(t) = \frac{t}{1+t^2} \tag{4.8}$$

is inside the interval $[-1, 1]$ and $\dot{a}(0) = 1$. Therefore, the function

$$b(t) = \varepsilon a\left(\frac{\beta}{\varepsilon}(t - t_1)\right) \tag{4.9}$$

⁷Also known as Picard's existence theorem or Cauchy-Lipschitz theorem.

is bounded by $[-\varepsilon, \varepsilon]$ and

$$\dot{b}(t_1) = \varepsilon \frac{\beta}{\varepsilon} \dot{a}(0) = \beta. \quad (4.10)$$

We have

$$x(\cdot) = c(\cdot) + b(\cdot) \in [x](\cdot). \quad (4.11)$$

Thus,

$$\dot{x}(t_1) = \dot{c}(t_1) + \dot{b}(t_1) = \dot{c}(t_1) + \beta, \quad (4.12)$$

which is equal to v_1 if we choose $\beta = v_1 - \dot{c}(t_1)$. As a consequence, for all (v_1, t_1) , there exists a consistent trajectory that belongs to $[x](\cdot)$. ■

4.2.3 Implementation

The definition of $\mathcal{C}_{\frac{d}{dt}}$ given in Equation (4.4) has no relation with the computer representation of the tubes $[x](\cdot)$ and $[v](\cdot)$. This section provides a way to practically apply this contractor on implemented tubes as presented in Section 3.3.1 at page 85.

We will break down the implementation into two contractors $\mathcal{C}_{\frac{d}{dt}}^{\rightarrow}$ and $\mathcal{C}_{\frac{d}{dt}}^{\leftarrow}$ so that $\mathcal{C}_{\frac{d}{dt}} = \mathcal{C}_{\frac{d}{dt}}^{\rightarrow} \circ \mathcal{C}_{\frac{d}{dt}}^{\leftarrow}$.

Forward contractor $\mathcal{C}_{\frac{d}{dt}}^{\rightarrow}$

When solving an ordinary differential equation numerically such as $\dot{x}(\cdot) = v(\cdot)$, a recurrence relation is typically encountered:

$$x(t + dt) \approx x(t) + dt \cdot v(t), \quad (4.13)$$

where $dt \in \mathbb{R}$ is the resolution time step.

The corresponding contractor would be transparently obtained with bounded values and intersections:

$$[x](t + dt) := [x](t + dt) \cap ([x](t) + dt \cdot [v](t)). \quad (4.14)$$

However, this definition does not take into account the evolutions of $v(\cdot)$ during a time step. Using the chosen representation of a tube built with a set of slices of width δ , our time step dt now corresponds to δ . The feasible values of $v(\cdot)$ along a time interval $[k\delta, k\delta + \delta]$, $k \in \mathbb{N}$, is then given by $[v]([k\delta, k\delta + \delta])$.

It remains to enclose any value of $x(\cdot)$ along the same interval $[k\delta, k\delta + \delta]$:

$$\begin{aligned}
 [x]([k\delta, k\delta + \delta]) &:= [x]([k\delta, k\delta + \delta]) \cap \bigsqcup_{t=k\delta}^{k\delta+\delta} ([x](k\delta) + (t - k\delta) \cdot [v]([k\delta, k\delta + \delta])), \\
 &:= [x]([k\delta, k\delta + \delta]) \cap \left([x](k\delta) + [v]([k\delta, k\delta + \delta]) \cdot \bigsqcup_{t=k\delta}^{k\delta+\delta} (t - k\delta) \right), \\
 &:= [x]([k\delta, k\delta + \delta]) \cap ([x](k\delta) + [0, \delta] \cdot [v]([k\delta, k\delta + \delta])).
 \end{aligned} \tag{4.15}$$

Which is equivalent to:

$$[x](k + 1) := [x](k + 1) \cap \left([x](k) \cap [x](k + 1) + [0, \delta] \cdot [v](k + 1) \right), \tag{4.16}$$

where $[x](k)$ is the k^{th} slice of the tube.

Algorithm 3 performs this forward propagation.

Backward contractor $\mathcal{C}_{\frac{d}{dt}}^{\leftarrow}$

The same method is applied in backwards for the $(k - 1)^{\text{th}}$ slice:

$$[x](k - 1) := [x](k - 1) \cap \left([x](k) \cap [x](k - 1) - [0, \delta] \cdot [v](k - 1) \right). \tag{4.17}$$

Which trivially leads to Algorithm 4.

Forward/backward contractor $\mathcal{C}_{\frac{d}{dt}}$

The simple combination of the above algorithms is provided in Algorithm 5.

A step-by-step example of a forward/backward propagation is given in Figure 4.3, with an arbitrary tube $[v](\cdot)$, an uninitialized tube $[x](\cdot)$ and conditions $x(0) = 0$, $x(5) = 4$. The variable $[x_{front}]$ used in the algorithms is depicted by blue thick lines in the figure.

Algorithm 3 $\mathcal{C}_{\frac{d}{dt}}^{\rightarrow}$ (in : $[x_0]$, $[v](\cdot)$, inout : $[x](\cdot)$)

```

1: var  $[x_{front}] \leftarrow [x_0]$ 
2: var  $[x_{old}]$ 
3: for  $k = 0$  to  $\bar{k}$  do
4:    $[x_{old}] \leftarrow [x](k)$ 
5:    $[x](k) \leftarrow [x_{old}] \cap ([x_{front}] + [0, \delta] \cdot [v](k))$ 
6:   if  $k \neq \bar{k}$  then
7:      $[x_{front}] \leftarrow [x_{old}] \cap ([x_{front}] + \delta \cdot [v](k)) \cap [x](k + 1)$ 
8:   end if
9: end for

```

Algorithm 4 $\mathcal{C}_{\frac{d}{dt}}^{\leftarrow}$ (in : $[x_f]$, $[v](\cdot)$, inout : $[x](\cdot)$)

```

1: var  $[x_{front}] \leftarrow [x_f]$ 
2: var  $[x_{old}]$ 
3: for  $k = \bar{k}$  to  $0$  do
4:    $[x_{old}] \leftarrow [x](k)$ 
5:    $[x](k) \leftarrow [x_{old}] \cap ([x_{front}] - [0, \delta] \cdot [v](k))$ 
6:   if  $k \neq 1$  then
7:      $[x_{front}] \leftarrow [x_{old}] \cap ([x_{front}] - \delta \cdot [v](k)) \cap [x](k - 1)$ 
8:   end if
9: end for

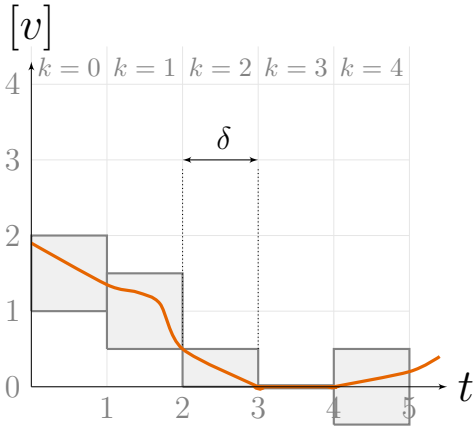
```

Algorithm 5 $\mathcal{C}_{\frac{d}{dt}}$ (in : $[x_0]$, $[x_f]$, $[v](\cdot)$, inout : $[x](\cdot)$)

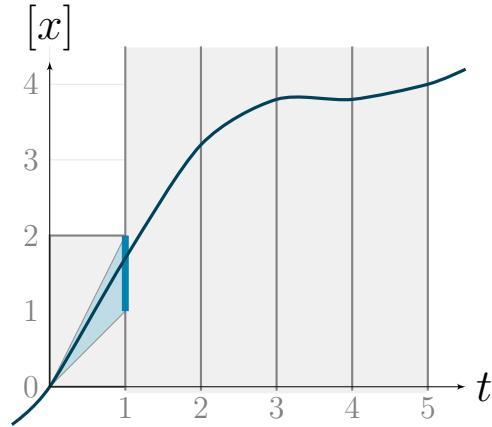
```

1:  $\mathcal{C}_{\frac{d}{dt}}^{\rightarrow}([x_0], [v](\cdot), [x](\cdot))$ 
2:  $\mathcal{C}_{\frac{d}{dt}}^{\leftarrow}([x_f], [v](\cdot), [x](\cdot))$ 

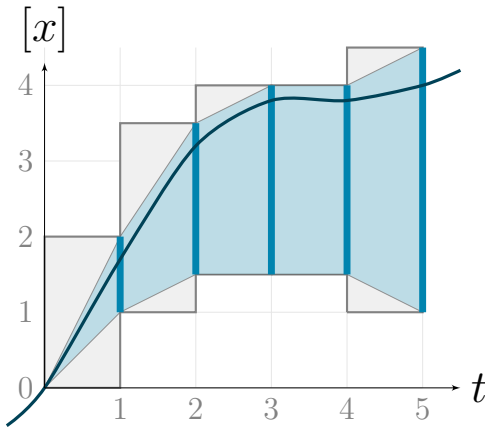
```



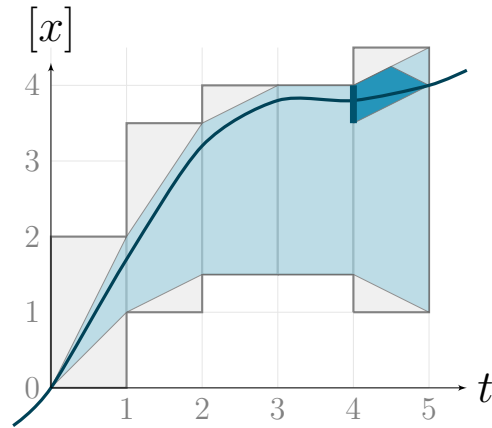
(a) Derivative tube $[v](\cdot)$ used to contract $[x](\cdot)$. The actual trajectory $x^*(\cdot)$ is plotted in orange ●.



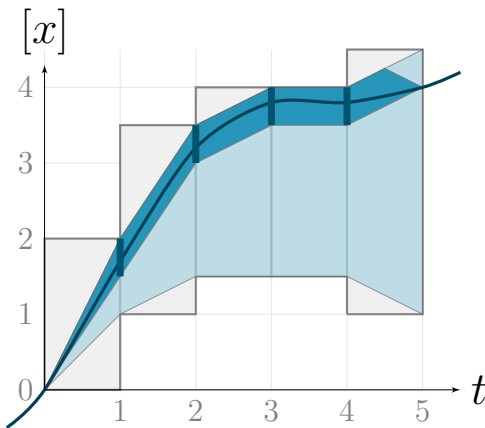
(b) First step of *forward* integration contractor: first slice k_1 is contracted from $[-\infty, \infty]$ to $[0, 2]$ with the condition $x(0) = 0$.



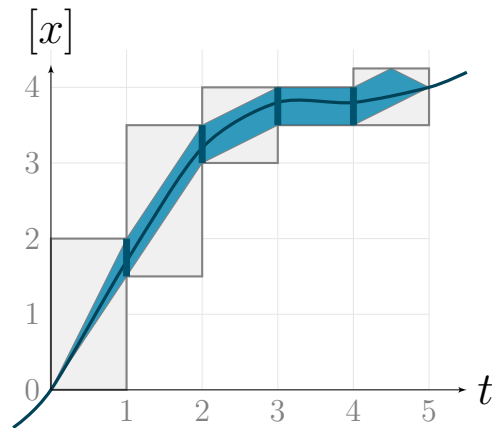
(c) End of *forward* contractions. The minimal envelope of solutions compliant with $[v](\cdot)$ is pictured in blue ●. Slices representation – outer approximation – is depicted in gray ●.



(d) First step of *backward* integration contractor, with the last condition $x(5) = 4$. A reduced envelope, pictured in dark blue ●, is computed from $t = 5$ to $t = 0$.



(e) End of *backward* processing.



(f) Tube $[x](\cdot)$ is contracted to optimally enclose the thin envelope pictured in dark blue.

Figure 4.3: A step-by-step illustration of the \mathcal{C}_d 's implementation. The contraction will propagate conditions $x(0) = 0$ and $x(5) = 4$ over $[x](\cdot)$ initialized to $[-\infty, \infty] \forall t$. Curved lines picture feasible trajectories $\dot{x}^*(\cdot)$ ● and $x^*(\cdot)$ ● that remain respectively enclosed within $[v](\cdot)$ and $[x](\cdot)$ after the contraction process.

4.3 Contractor-based approach for state estimation

A state estimation problem involving state constraints can be cast into a constraint network for which four types of uncertainty propagations are encountered:

1. the *forward* propagation;
2. the *backward* propagation;
3. the *correction*;
4. the *state constraints*.

In the literature, (1) is known as the *prediction*, (1)+(2) the *integration*, (3) the *correction*, (1)+(3) the *filter* and (1)+(2)+(3) the *smoother*. Section 4.2 provided a tool for (1)+(2), *i.e.*, the forward/backward integration. In the current section, we will show that the approach can also be extended to all types of constraints, namely (1)+(2)+(3)+(4), which is of high interest for robotic applications. The limits of the approach will be discussed after the methodology details.

4.3.1 Constraint network of state equations

We focus on the resolution of problems depicted by Equation (3.1) recalled below:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}(\mathbf{x}(\cdot), \mathbf{u}(\cdot)), & (4.18a) \\ z_i = g(\mathbf{x}(t_i)). & (4.18b) \end{cases}$$

We emphasize that in the current chapter, uncertainties on any of the variables $\mathbf{x}(\cdot)$, $\mathbf{u}(\cdot)$, z_i , and the functions \mathbf{f} and g are assessed, except for the observation times t_i that must be perfectly known. Uncertainties on these values will be the subject of Chapter 5.

All the solving process will be achieved by a contractor programming approach, making use of the new operator $\mathcal{C}_{\frac{d}{dt}}$. The above state equations are broken down into a set of primitive constraints, introducing variables $\mathbf{v}(\cdot)$, $y(\cdot)$ for ease of

decomposition:

$$\text{CN: } \left\{ \begin{array}{l} \mathbf{Variables: } \mathbf{x}(\cdot), \mathbf{v}(\cdot), \mathbf{u}(\cdot), y(\cdot), z_i \\ \mathbf{Constraints:} \\ \quad 1. v_j(\cdot) = f_j(\mathbf{x}(\cdot), \mathbf{u}(\cdot)), j \in \{1 \dots n\} \\ \quad 2. \dot{x}_j(\cdot) = v_j(\cdot) \iff \mathcal{L}_{\frac{d}{dt}}(x_j(\cdot), v_j(\cdot)) \\ \quad 3. y(\cdot) = g(\mathbf{x}(\cdot)) \\ \quad 4. z_i = y(t_i) \\ \mathbf{Domains: } [\mathbf{x}](\cdot), [\mathbf{v}](\cdot), [\mathbf{u}](\cdot), [y](\cdot), [z_i] \end{array} \right. \quad (4.19)$$

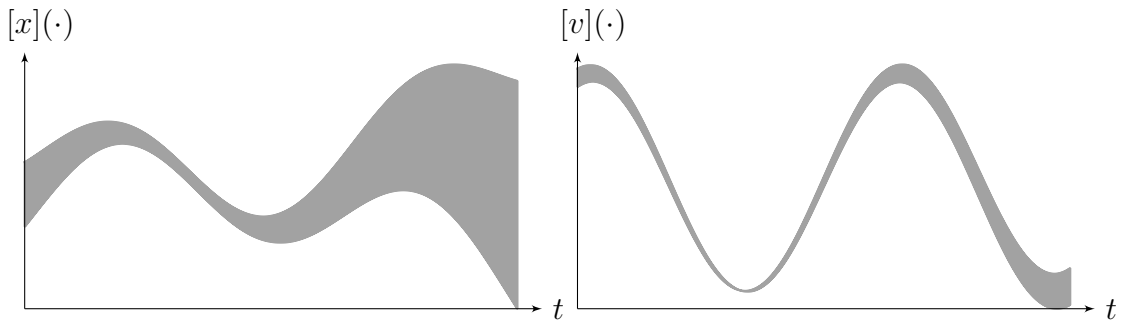
Stage (4.19):(4) is a so-called *evaluation* constraint on the trajectory $y(\cdot)$. We will consider a contractor $\mathcal{C}_{\text{eval}}$ to implement it. For now and to keep things simple, we will use the following definition while keeping in mind that it is not generic as it does not consider t_i as a variable to be estimated.

$$\left(\begin{array}{c} [z_i] \\ [y](t_i) \end{array} \right) \xrightarrow{\mathcal{C}_{\text{eval}}} \left(\begin{array}{c} [z_i] \cap [y](t_i) \\ [z_i] \cap [y](t_i) \end{array} \right). \quad (4.20)$$

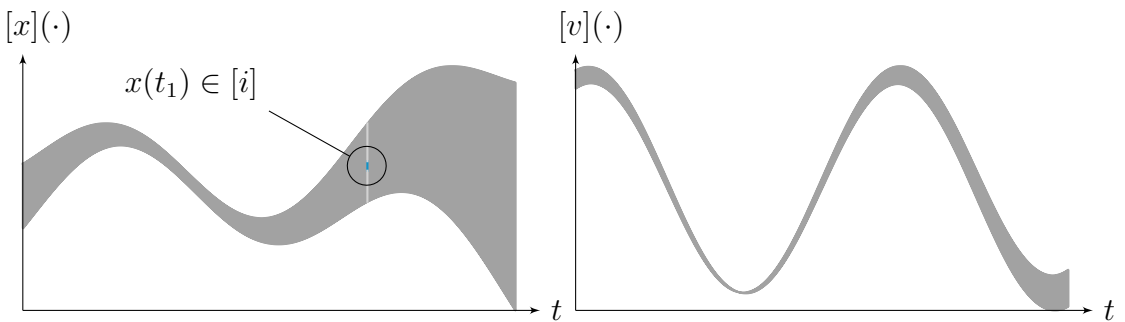
Figure 4.4 gives an illustration of $\mathcal{C}_{\text{eval}}$ coupled with $\mathcal{C}_{\frac{d}{dt}}$ in order to smooth a tube from a given observation. Steps (4.19):(1) and (4.19):(3) of CN (4.19) are implemented by related primitive contractors on tubes such as \mathcal{C}_+ defined in page 81. One is therefore well advised to use compositions of tube contractors, denoted here by $\mathcal{C}_{\mathbf{f}}$ and \mathcal{C}_g , to address complex functions \mathbf{f} and g if necessary. Finally, the differential contractor $\mathcal{C}_{\frac{d}{dt}}$ will deal with the constraint (4.19):(2). To sum up, CN (4.19) involves the following contractors⁸:

- $\mathcal{C}_{f_j}([v_j](\cdot), [\mathbf{x}](\cdot), [\mathbf{u}](\cdot)), j \in \{1 \dots n\}$
- $\mathcal{C}_{\frac{d}{dt}}([x_j](\cdot), [v_j](\cdot))$
- $\mathcal{C}_g([y](\cdot), [\mathbf{x}](\cdot))$
- $\mathcal{C}_{\text{eval}}([z_i], [y](t_i))$

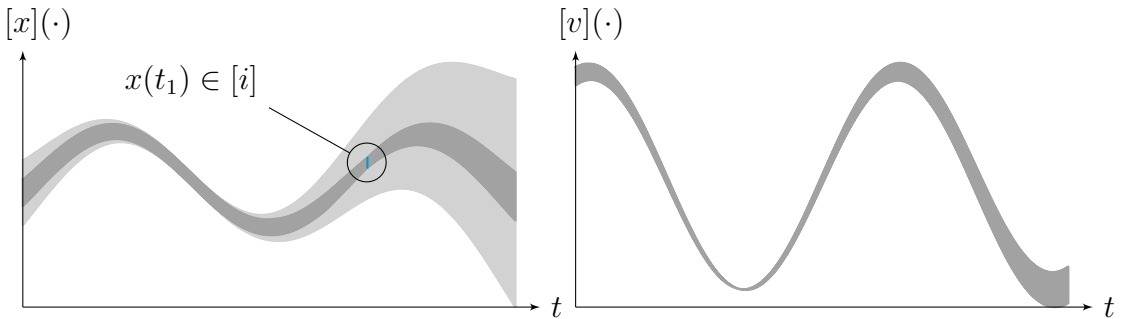
⁸A *filter* or a *smoother* procedure is respectively obtained by using $\mathcal{C}_{\frac{d}{dt}}^{\rightarrow}$ or $\mathcal{C}_{\frac{d}{dt}}$.



(a) Initial tubes $[x](\cdot)$ and $[v](\cdot)$. Consistent state.



(b) Contraction with $\mathcal{C}_{\text{eval}}$ from a bounded observation of $x(\cdot)$ at t_1 . Non consistent state.



(c) Application of $\mathcal{C}_{\frac{d}{dt}}$ resulting in a forward/backward temporal propagation of the observation. Consistent state.

Figure 4.4: An arbitrary tube $[x](\cdot)$ together with its derivative $[v](\cdot)$. The use of $\mathcal{C}_{\frac{d}{dt}}$ is presented in three steps: (a) initial tubes; (b) a given constraint $x(t_1) \in [i]$ is raised and applied thanks to $\mathcal{C}_{\text{eval}}$; (c) results of the differential constraint propagation. Contracted tubes are depicted in dark gray while the previous envelope (before contraction) is shown in light gray. It is noteworthy that only $[x](\cdot)$ has been contracted by $\mathcal{C}_{\frac{d}{dt}}$ (as explained in Section 4.2.2, page 109).

4.3.2 Fixed-point propagations

Set-membership state estimation then consists in an iterative process, each stage of which is calling these contractors. The process can be stopped when the tubes are not contracted anymore. We remind that the above contractors can be called in any order due to their monotonicity [Apt, 1999]. In this approach based on constraint propagations, the order can only impact the computation time: it could be more interesting to apply a contractor before another in order to perform the strongest contractions as soon as possible. However, this is highly specific to the considered problem.

In several robotic applications such as $\dot{\mathbf{x}}(\cdot) = \mathbf{f}(\mathbf{x}(\cdot), \mathbf{u}(\cdot))$, constraints networks form causal kinematic chains. For instance, motors generate an acceleration of the wheels rotation, driving the vehicle forwards, creating a displacement. Let us consider the following system depicting a simple mobile robot:

$$\begin{cases} \dot{x}_1(\cdot) = \cos(x_3(\cdot)), \\ \dot{x}_2(\cdot) = \sin(x_3(\cdot)), \\ \dot{x}_3(\cdot) = u(\cdot). \end{cases} \quad (4.21)$$

The propagation of information from the inputs $u(\cdot)$ toward the states $\mathbf{x}(\cdot)$ is depicted in Figure 4.5 where two linear chains are clearly visible. In such cases, the resolution is achievable in one single iteration with a smart order of contractors. Furthermore, we will show in Section 4.4 that our method appears to be efficient and competitive compared with other libraries.

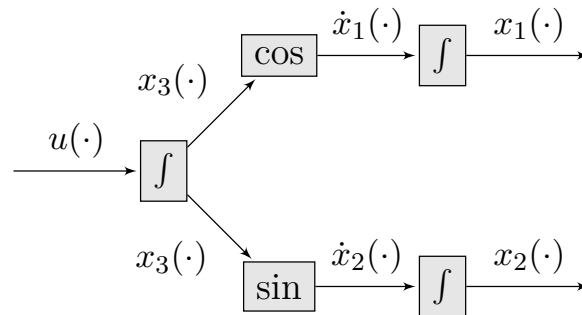


Figure 4.5: Causal kinematic chains associated to Equations (4.21).

Other cases may involve cyclic networks of constraints. This can be the case of systems such as $\dot{\mathbf{x}}(\cdot) = \mathbf{f}(\mathbf{x}(\cdot))$. The theoretical problem of $\dot{x} = -\sin(x)$ is a case in point. In this configuration, see Figure 4.6, an iterative resolution has to be processed until a fixed point is reached.

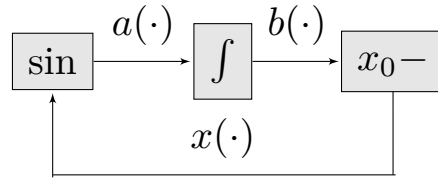


Figure 4.6: Circuit associated with the IVP defined by $(x_0, \dot{x} = -\sin(x))$. The equation can be broken down into the constraints $a(\cdot) = \sin(x(\cdot))$, $\dot{b}(\cdot) = a(\cdot)$, $x(\cdot) = x_0 - b(\cdot)$.

Our method is scalable, being able to deal with problems such as $\dot{\mathbf{x}}(\cdot) = \mathbf{f}(\mathbf{x}(\cdot), \mathbf{u}(\cdot))$ or $\dot{\mathbf{x}}(\cdot) = \mathbf{f}(\mathbf{x}(\cdot))$, but may provide pessimistic results in some cases. This is discussed in the following section with the example of $\dot{x} = -\sin(x)$.

4.3.3 Theoretical example of interest $\dot{x} = -\sin(x)$

In order to understand the limits of our approach, let us consider an unfavorable case study. The reader mainly interested in state estimation may skip this section and go directly to the applicative part at page 121.

IVP

We propose the following IVP:

$$\begin{cases} \dot{x} &= -\sin(x), \\ x_0 &= 1. \end{cases} \quad (4.22)$$

Applying the constraint programming approach, a decomposition is performed:

$$\begin{cases} a(\cdot) &= \sin(x), \\ b(\cdot) &= \int_0 a(\tau) d\tau, \\ x(\cdot) &= x_0 - b(\cdot). \end{cases} \quad (4.23)$$

We build the three associated contractors and we call them up to the fixed point, as illustrated in Figure 4.7 for the domain $[0, 10]$. Initial tubes are set to $[-\infty, \infty] \forall t$.

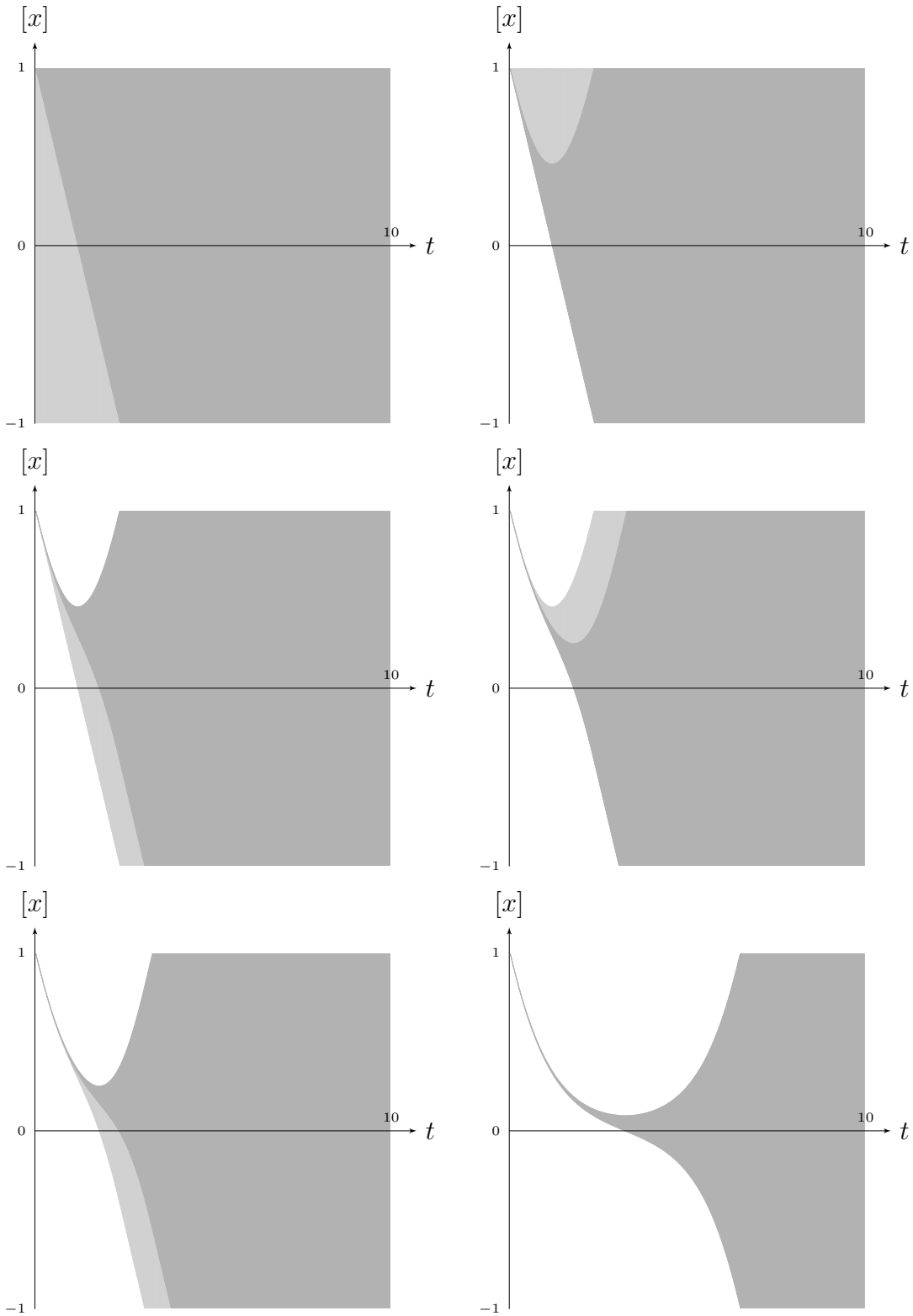


Figure 4.7: Inefficient resolution of the $\dot{x} = -\sin(x)$ problem. Successive contractions of the tube $[x](\cdot)$ are pictured: first five computations and final result when reaching a fixed point. Light gray areas represent the tube's parts that have been contracted during the contraction steps.

Divergence

Even if the propagation takes place, the obtained envelope is poor and the fixed point is reached even before the contraction of the tube over its complete domain. The same problem solved with the *CAPD* library [Wilczak et al., 2017], which can be considered as one of the most efficient libraries in this field, yields significantly better results with the following enclosure obtained in less than 1 second:

$$x(10) \in [4.96041893247 \cdot 10^{-5}, 4.96041893264 \cdot 10^{-5}]. \quad (4.24)$$

This is not surprising since *CAPD* is typically devoted to this kind of problem, where the information given on the initial state has to propagate forwards in time with the complete assessment of the analytical expression of the problem.

The principal justification for such bounds divergence, obtained with our approach, is the wrapping effect. Indeed, we first break down the IVP into some constraints involving the three trajectories $x(\cdot)$, $a(\cdot)$ and $b(\cdot)$. All the information about these variables is stored within tubes, that are pessimistic enclosures due to there sliced implementation. The contractors guarantee the reliable propagation from one slice to another, so they also propagate the wrapping effect in time, leading to a dramatic pessimism at some point, as pictured in the last iteration of Figure 4.7.

Overcome unwanted pessimism

A future work may consider to merge several approaches such as *CAPD* or *DynIBEX* [Alexandre dit Sandretto and Chapoutot, 2016] into our framework, in order to limit such divergence. Hence, the IVP could be dealt with a contractor-based strategy while taking advantage of the efficiency of dedicated libraries.

Another track to investigate is a smarter discretization of tubes: a slice width to be set according to the evolution of the trajectory along the slice, thus locally optimizing the wrapping.

Finally, a complementary strategy is the *strangle* method [Jaulin, 2002]. In the presented IVP, the approach consists in assuming $x(10) \in [a, b]$, contracting the $[x](\cdot)$ at 10 based on this assumption, and propagating it backwards towards 0. The contraction may result in an empty set, which would prove that the actual $x^*(\cdot)$ does not reach $[a, b]$ at 10. Therefore, the tube can then be surely contracted by

removing from the envelope the trajectories going through $10 \times [a, b]$. The process can be automated to perform several assumptions at a given time. At the end, a strangulation would have been performed at time 10 without providing external constraints to the problem.

This method enables thinner results equivalent to those obtained with *CAPD*, but at the cost of heavy computation times and without a guarantee of success. Indeed, the backward propagation may be too pessimistic as well, thus enclosing the initial value for each assumption. Also, random strangulations over the domain may be tried. The approach deserves further study.

4.4 Robotic applications

In this section, we provide reproducible examples involving mobile robots. It ends with a state estimation performed on an actual dataset obtained during an experiment with the *Daurade* AUV.

4.4.1 Causal kinematic chain

We will come back to the causal situation mentioned in Section 4.3.2 and apply our approach on a simple example. A comparison with the *CAPD* library will show that our method can be considered as competitive, at least for robotic applications.

Let us consider a wheeled robot \mathcal{R} with a constant velocity ϑ [Dubins, 1957] and described by the Equations (4.21) adapted below:

$$\begin{cases} \dot{x}_1(t) &= \vartheta \cos(x_3(t)), \\ \dot{x}_2(t) &= \vartheta \sin(x_3(t)), \\ \dot{x}_3(t) &= u(t). \end{cases} \quad (4.25)$$

We assume that $\vartheta = 10$ and that the initial state \mathbf{x}_0 belongs to the box:

$$\mathbf{x}_0 \in [\mathbf{x}_0] = [-1, 1] \times [-1, 1] \times [-6/5\pi - 0.02, -6/5\pi + 0.02]. \quad (4.26)$$

In order to compare with the *CAPD* library, $u(\cdot)$ is bounded by the following Equation (4.27). Note that when dealing with actual experiments, these analytical

expressions are not always at hand, which is a strong limitation for classical methods. With tubes, however, any dataset can be considered.

$$u(t) \in [u](t) = \underbrace{-\cos\left(\frac{t+33}{5}\right)}_{u^*(t)} + [-0.02, 0.02]. \quad (4.27)$$

Our state estimation method yields the tubes projected on Figure 4.8. Naturally, this dead-reckoning estimation becomes more pessimistic with time: without exteroceptive measurements, the robot progressively gets lost. The figure shows that our approach is more accurate than *CAPD* on this example.

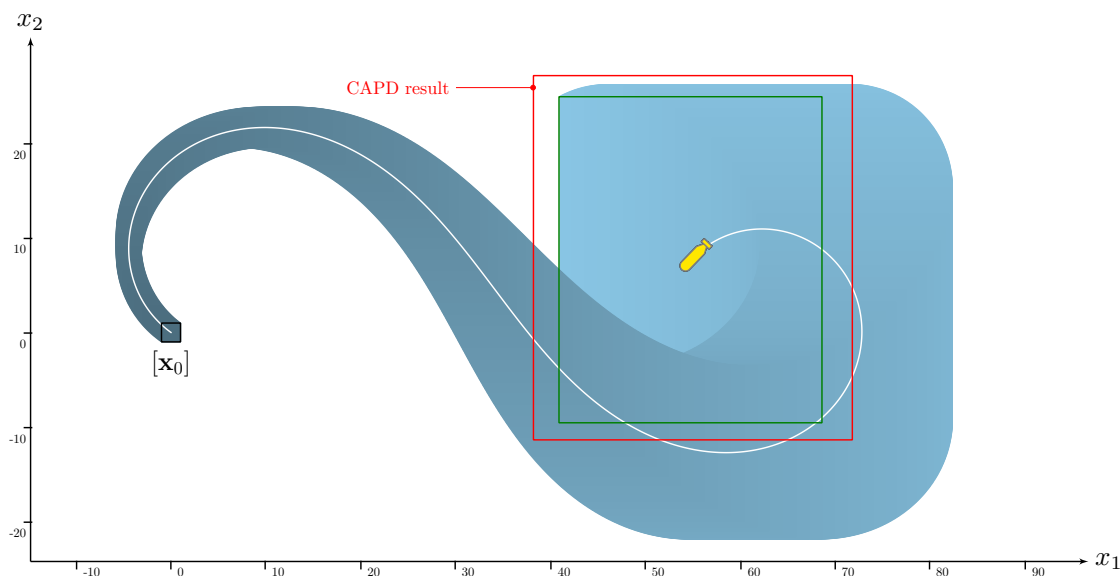


Figure 4.8: Interval simulation of the robot \mathcal{R} . The white line represents the true poses of the robot, considering the actual and unknown input $u^*(\cdot)$, while the tubes $[x_1](\cdot) \times [x_2](\cdot)$ are projected in blue \bullet . The green box \bullet is the projection of the last slice $[\mathbf{x}](t_f)$ obtained for $t_f = 14$. By comparison, the final box computed with *CAPD* is represented in red \bullet . Computation time is less than 0.1s on a conventional computer.

To illustrate the fact that our method is more general and flexible than existing guaranteed integration approaches, consider now a situation where the final state $\mathbf{x}(14)$ is known to belong to the box $[\mathbf{x}_f] = [53.9, 55.9] \times [6.9, 8.9] \times [-2.36, -2.32]$. Adding this information to the constraint network, the contractor $\mathcal{C}_{\frac{d}{dt}}$ will also perform a backward propagation. The result is illustrated by Figure 4.9.

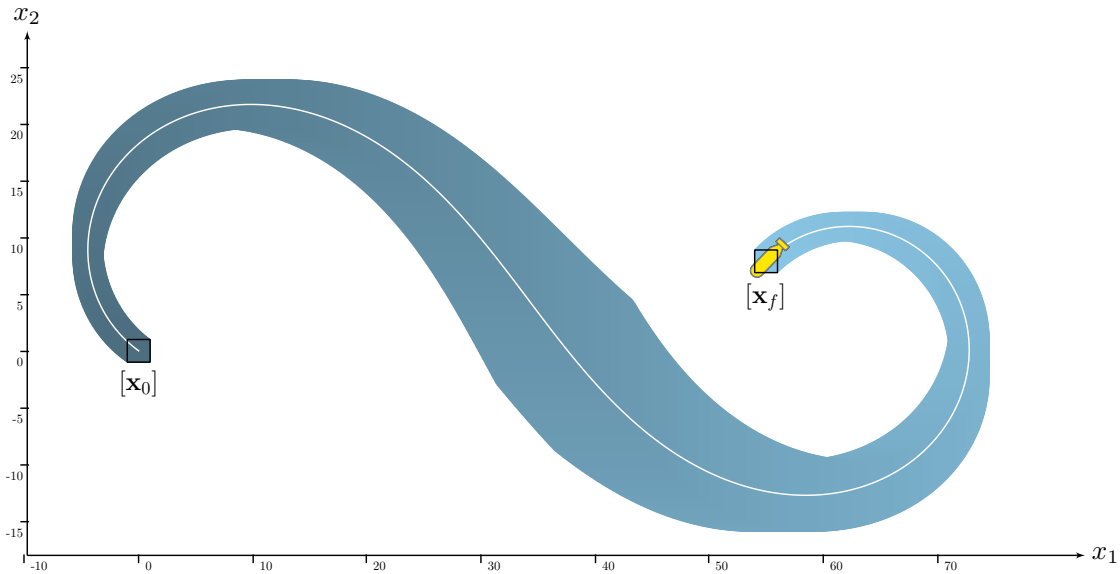


Figure 4.9: Simulation as presented in Figure 4.8. This time, the initial and final states are almost known while uncertainties are maximal in the middle of the mission.

4.4.2 Higher order differential constraints

Another brief example is the following: a 2D robot trajectory expressed as a Lissajous curve:

$$\mathbf{x}(t) = 5 \begin{pmatrix} 2 \cos(t) \\ \sin(2t) \end{pmatrix}. \quad (4.28)$$

Equation (4.28) describes the actual but unknown trajectory. To illustrate our approach, we generate differential equations satisfied by $\mathbf{x}(t)$. The initial condition is known to belong to a box $[\mathbf{x}_0]$. The associated constraint network is the following.

$$\text{CN: } \left\{ \begin{array}{l}
 \text{Variables: } \mathbf{x}(\cdot), \dot{\mathbf{x}}(\cdot), \ddot{\mathbf{x}}(\cdot) \\
 \text{Constraints:} \\
 \quad 1. \ddot{x}_1(t) \in -10 \cos(t) + [-0.001, 0.001] \\
 \quad 2. \ddot{x}_2 = -0.4 \cdot \dot{x}_1 \cdot \ddot{x}_1 \\
 \quad 3. \dot{\mathbf{x}}(0) = \begin{pmatrix} 0 \\ 10 \end{pmatrix}, \mathbf{x}(0) \in \begin{pmatrix} [9.8, 10.2] \\ [-0.2, 0.2] \end{pmatrix} \\
 \text{Domains: } [\mathbf{x}](\cdot), [\dot{\mathbf{x}}](\cdot), [\ddot{\mathbf{x}}](\cdot)
 \end{array} \right. \quad (4.29)$$

The contractor based approach provides the envelope of trajectories pictured in Figure 4.10. It shows the ease-of-use of differential constraints that are usually encountered in mobile robotics where \mathbf{x} is the position of a robot, $\dot{\mathbf{x}}$ its speed and $\ddot{\mathbf{x}}$ its acceleration.

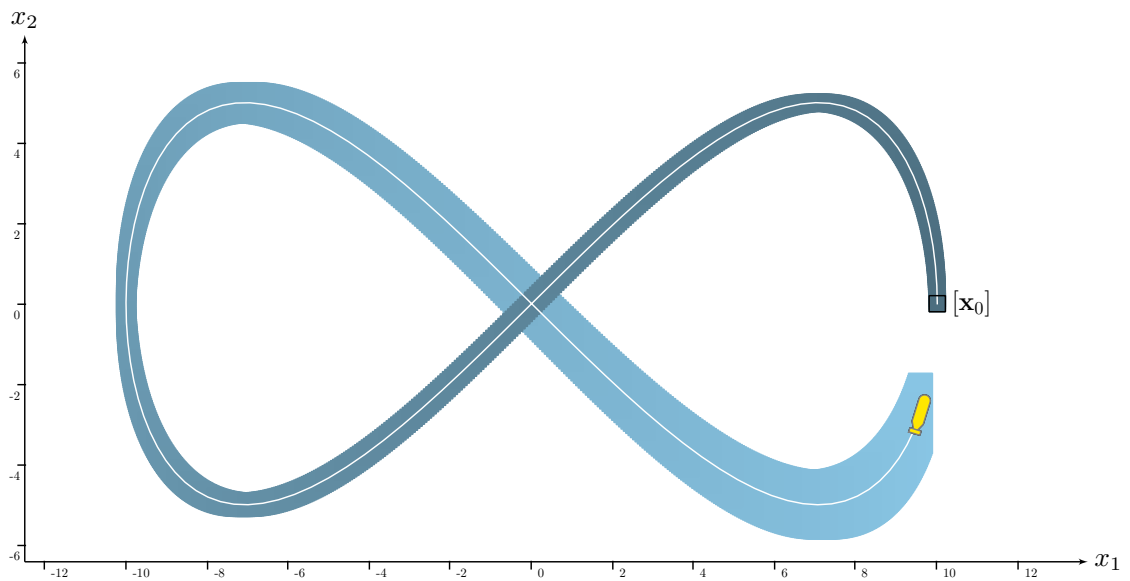


Figure 4.10: A robot following a Lissajous curve. The white line is the unknown truth given by Equation (4.28). The blue shape \bullet is the envelope of trajectories computed from CN (4.29). We emphasize that further constraints such as $x_2(t) = x_2(t + \pi)$ or $\mathbf{x}(\pi/2) = \mathbf{x}(3\pi/2)$ could be easily added to the CN.

4.4.3 Kidnapped robot problem

The last simulated example is an extension of the illustration provided in Section 3.4 from page 90 onwards. We were considering the simulation of a mobile robot from the knowledge of the system's input $u(\cdot)$ and a known initial condition \mathbf{x}_0 . Our goal is to remove this last constraint, assuming the robot has been kidnapped and placed somewhere else in an unknown state. We shall now consider the unique measurement:

$$x_1(37) \in [59.25, 61.25], \quad x_2(37) \in [36.16, 38.16]. \quad (4.30)$$

Figure 4.11 depicts the forward/backward propagation from the measurement at $t = 37$. Figure 4.12 represents the accuracy of the estimation, highlighting that state observations are easily applicable at any time without any integration from known initial conditions.

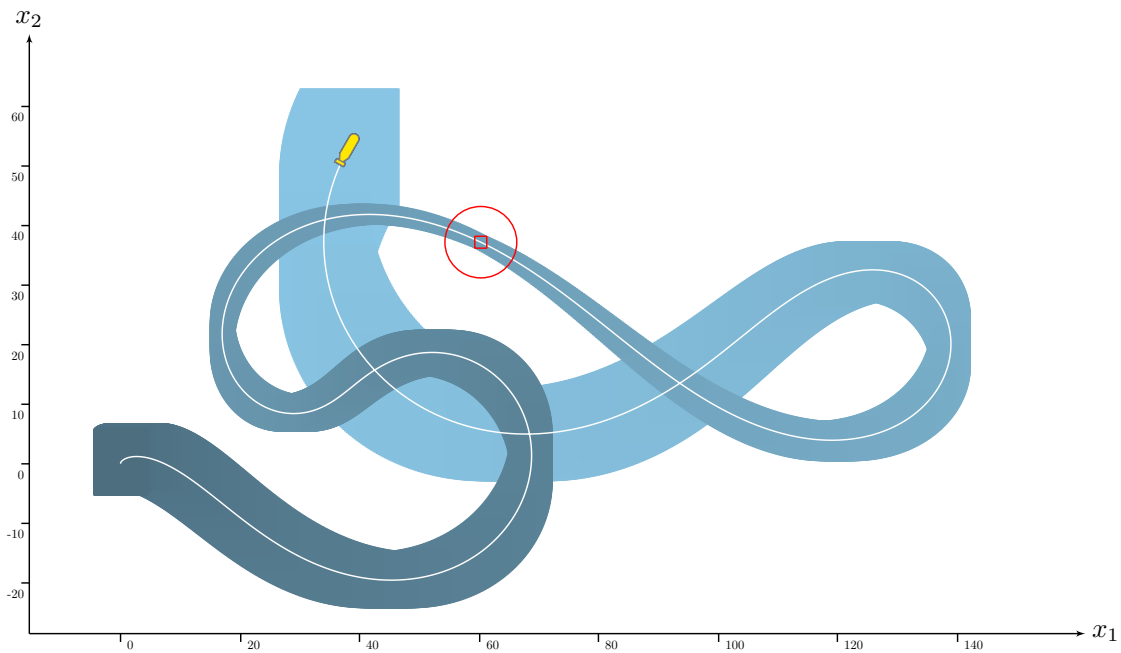


Figure 4.11: State estimation of a mobile robot without prior knowledge on its initial position. A single measurement, pictured by a red box \bullet , is assessed. The actual trajectory plotted in white remains enclosed within the estimated envelope of trajectories drawn in blue \bullet .

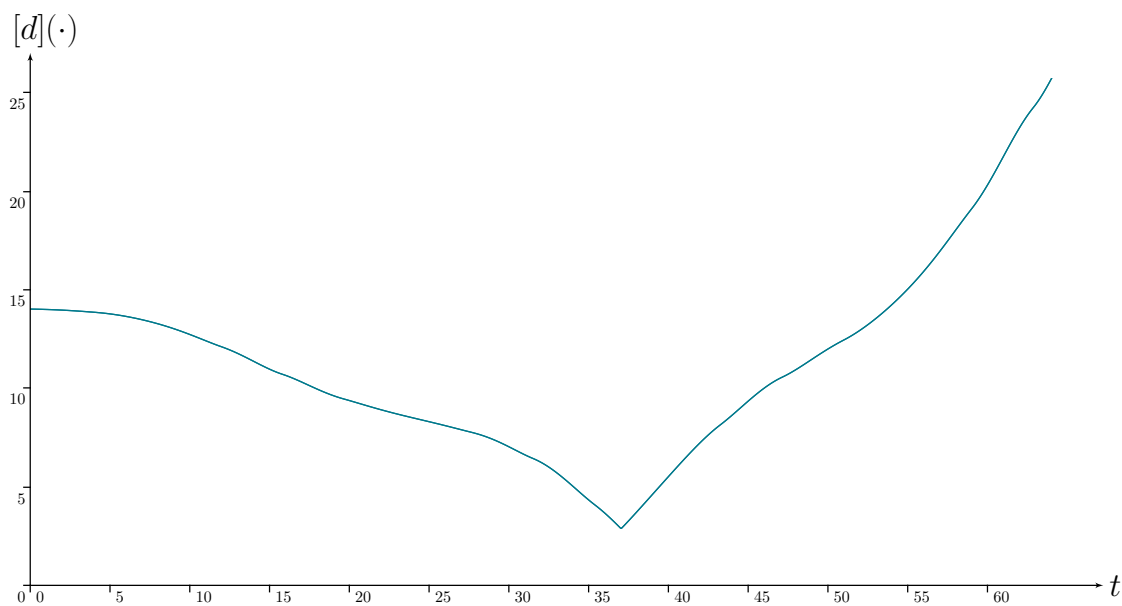


Figure 4.12: Thicknesses of the position estimation $[x_1](\cdot) \times [x_2](\cdot)$ as defined for Figure 3.12, page 94. The graph reveals the propagation of an observation at $t = 37$ in both backward and forward ways.

4.4.4 Actual experiment with the *Daurade* AUV

The changeover from analytical simulations towards actual datasets is straightforward when using tubes, since all the knowledge on trajectories is implemented as representable sets.

Let us consider a real experiment with the *Daurade* AUV, based on a classical kinematic model for an underwater robot [Fossen, 1994, Jaulin, 2015a]:

$$\begin{cases} \dot{\mathbf{p}} &= \mathbf{R}(\psi, \theta, \varphi) \cdot \mathbf{v}_r, \\ \dot{\mathbf{v}}_r &= \mathbf{a}_r - \boldsymbol{\omega}_r \wedge \mathbf{v}_r, \end{cases} \quad (4.31)$$

where $\mathbf{R}(\psi, \theta, \varphi)$ is the Euler matrix given by:

$$\begin{pmatrix} \cos \theta \cos \psi & -\cos \varphi \sin \psi + \sin \theta \cos \psi \sin \varphi & \sin \psi \sin \varphi + \sin \theta \cos \psi \cos \varphi \\ \cos \theta \sin \psi & \cos \psi \cos \varphi + \sin \theta \sin \psi \sin \varphi & -\cos \psi \sin \varphi + \sin \theta \cos \varphi \sin \psi \\ -\sin \theta & \cos \theta \sin \varphi & \cos \theta \cos \varphi \end{pmatrix}. \quad (4.32)$$

In these equations, the vector $\mathbf{p} = (p_x, p_y, p_z)^\top$ gives the coordinates of the center of the robot expressed in the absolute inertial coordinate system \mathcal{R}_0 , as well as the

three Euler angles (ψ, θ, φ) . The robot's speed vector \mathbf{v}_r and acceleration vector \mathbf{a}_r are expressed in its own coordinate system \mathcal{R}_1 . The vector $\boldsymbol{\omega}_r = (\omega_x, \omega_y, \omega_z)^\top$ corresponds to the rotation vector of the robot relative to \mathcal{R}_0 expressed in \mathcal{R}_1 . It is indeed conventional to express \mathbf{a}_r , $\boldsymbol{\omega}_r$ in the coordinate system of the robot since these quantities are generally measured by the robot itself *via* an IMU attached on it.

Now, from the inertial unit, we collect measurements for \mathbf{a}_r , $\boldsymbol{\omega}_r$, ψ , θ , φ . In addition, *Daurade* is equipped with a DVL providing measurements about its speed vector \mathbf{v}_r . Some bounded errors are established from the datasheets of these sensors, as presented in Section 2.5.1, page 69. From these data we then create initial tubes, see Section 3.3.2, page 87. With tube arithmetic and tube integration, we finally compute the speeds $[\dot{\mathbf{p}}](\cdot)$ and positions $[\mathbf{p}](\cdot)$ of the robot as expressed by Equation (4.31). One of their components is plotted in Figures 4.13 and 4.14.

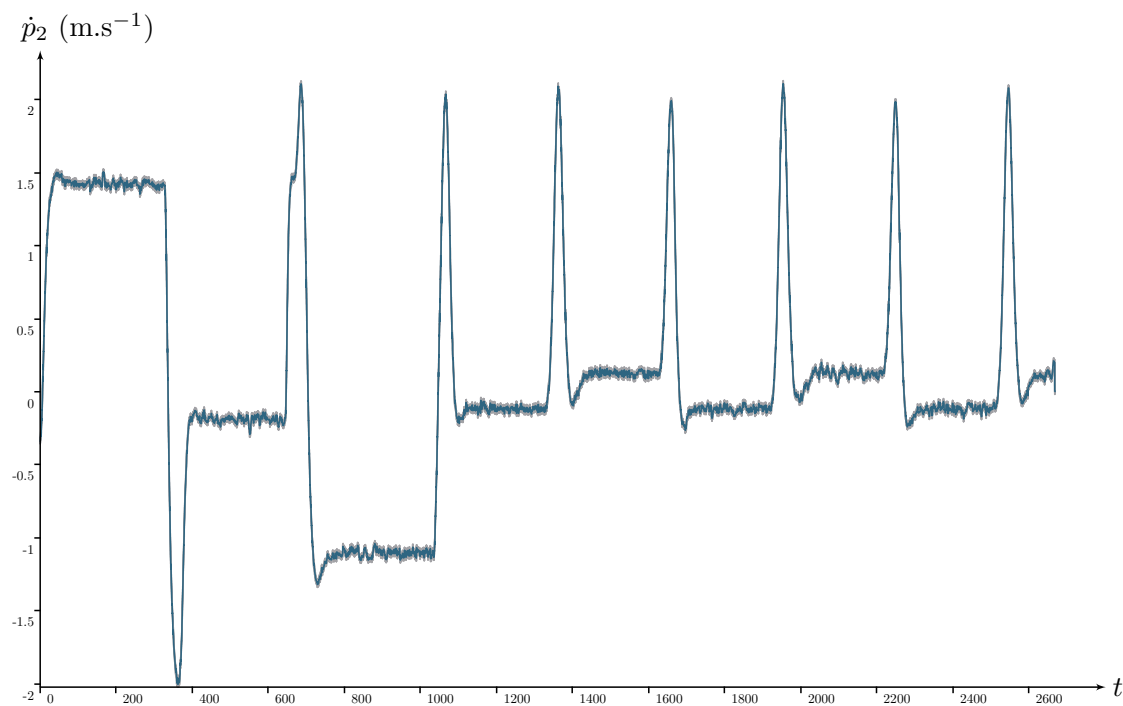


Figure 4.13: *Daurade*'s tube $[\dot{p}_2](\cdot)$ computed to represent the velocity along y in \mathcal{R}_0 . Because this tube comes from measurements without any integration process, its thickness remains almost constant.

An overview of the mission is pictured by Figure 4.15 where the first dead reckoning computation is depicted in light gray. A state estimation is then performed with the help of two trajectory measurements coming from a USBL positioning

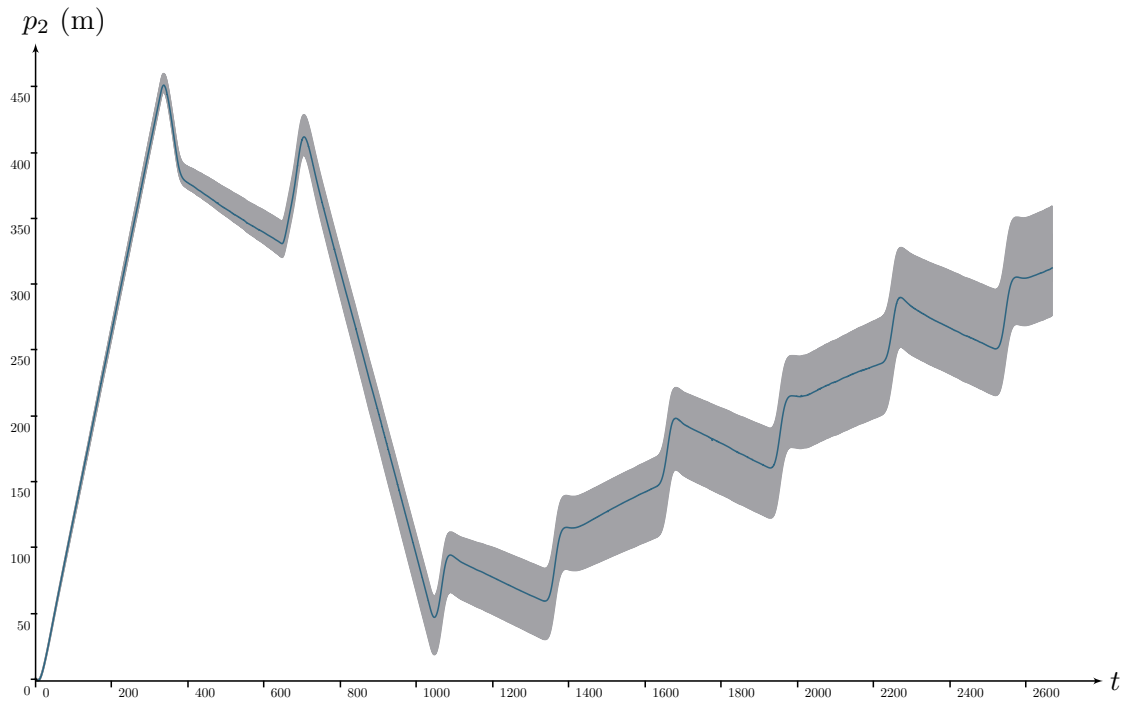


Figure 4.14: *Daurade's* tube $[p_2](\cdot)$ computed as the primitive of $[\dot{p}_2](\cdot)$. This tube becomes thicker depicting cumulative uncertainties due to the dead-reckoning method. Observations pictured in Figure 4.15 are not represented here.

system. This led to the contraction of $[\mathbf{p}](\cdot)$ over the whole mission duration, by using the contractor $\mathcal{C}_{\frac{d}{dt}}$.

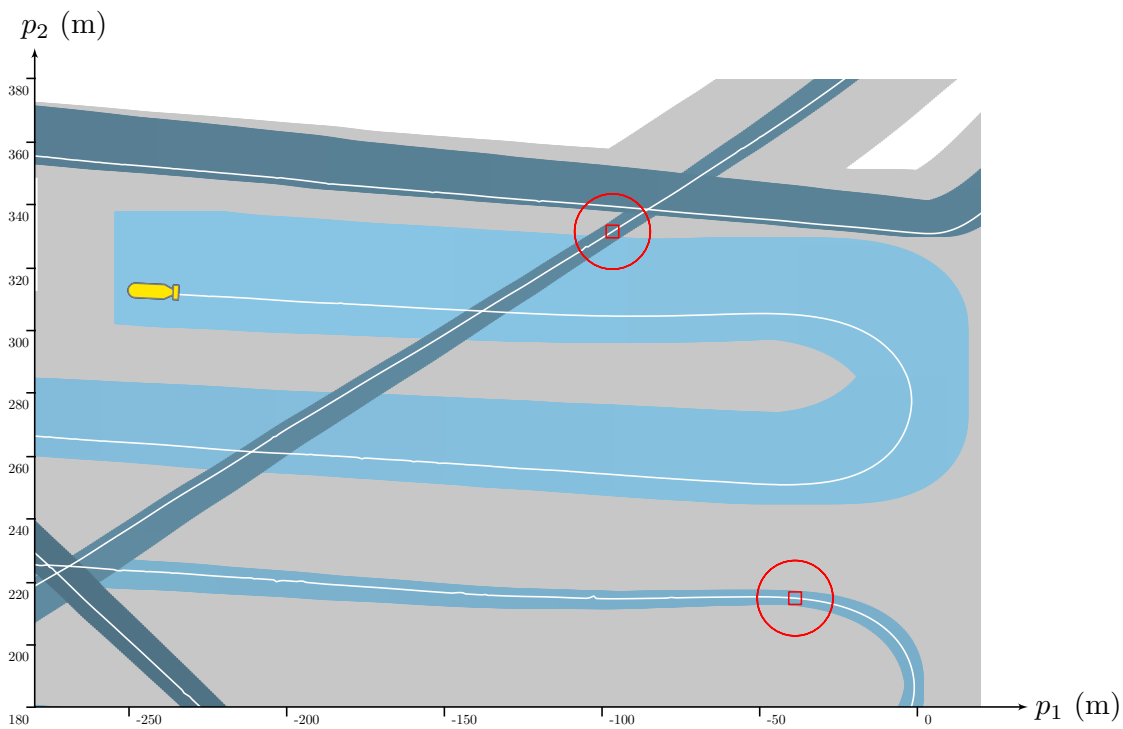
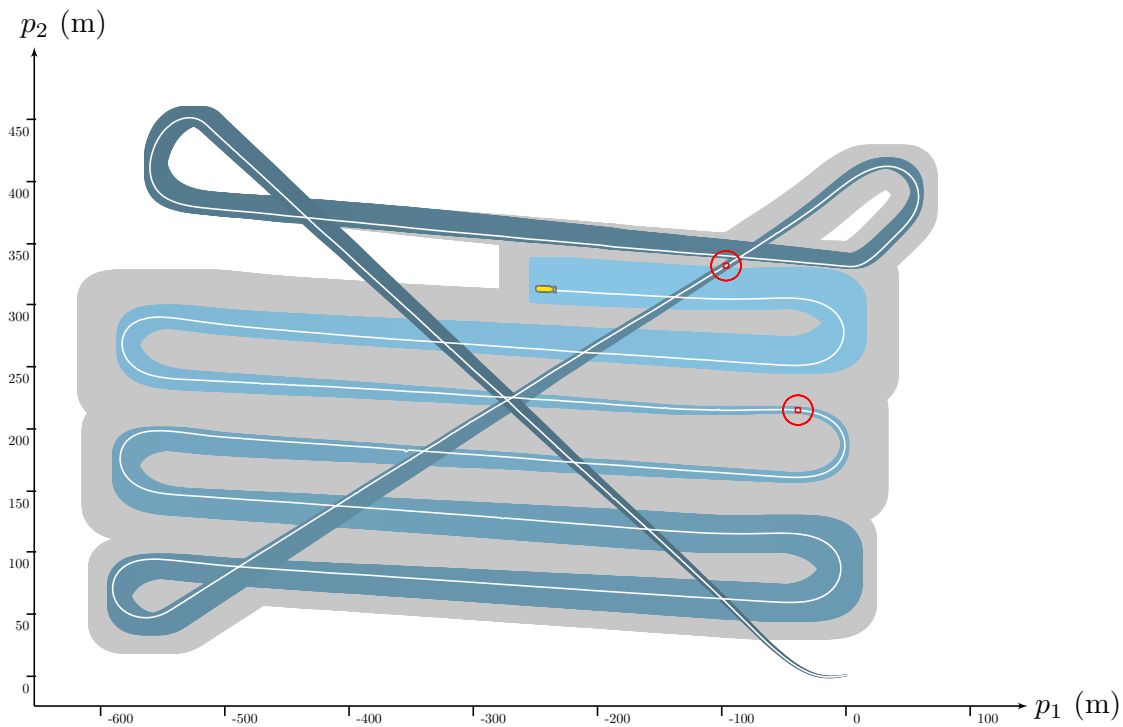


Figure 4.15: Mission map after 45 minutes, *Daurade* explored a 25 hectares area. The white line is *Daurade*'s trajectory filtered by a USBL and an INS (see Section 1.2.3, page 17). Gray background \bullet and blue shapes \bullet respectively correspond to the tubes $[p_1](\cdot) \times [p_2](\cdot)$ projected on the world frame before and after the consideration of positioning measurements, pictured by red boxes \bullet . Indeed, the only use of inertial data and velocity measurements provided by the DVL leads to a strong drift at the end of the mission. Here, thanks to two observations obtained from the USBL at $t_1 = 13\text{min}$ and $t_2 = 33\text{min}$, the drift is reduced over the whole mission.

4.5 Conclusion

In this chapter as in [Rohou et al., 2017], we have proposed a new method for guaranteed integration of state equations. The contribution is to provide a reliable framework to enclose the solutions of differential equations within tubes. We then apply a constraint programming approach in order to reduce the sets of trajectories. This is achieved by means of a new contractor $\mathcal{C}_{\frac{d}{dt}}$, the definition and proof of which being introduced for the first time in this document.

Its use is shown to be simple and more general than existing approaches dealing with guaranteed integration. Indeed, the developed framework allows one to deal with non-linear equations or differential inclusions built from datasets, while considering observations of the states of interest. Furthermore, the variables do not have to be Lipschitz continuous, which differs from other methods. In addition, the method appears even more competitive on some robotic applications expressed as $\dot{\mathbf{x}}(\cdot) = \mathbf{f}(\mathbf{x}(\cdot), \mathbf{u}(\cdot))$.

Prospects

A strong amount of work still remains in order to compare the current approach with already existing tools. It is indeed necessary to clearly state the limits in each case regarding criteria of accuracy, genericity, computation-time, *etc.* Hence, the contractor-based approach we propose should be applied on well-known benchmarks from the IVP community. Further ones could be suggested in the field of robotics, thus expanding the IVP to a wider range of applications and challenges.

Furthermore, our framework could process a user-friendly solver for ODEs and practical applications. In this context, the choice of applying $\mathcal{C}_{\frac{d}{dt}}$, or any other contractor relying on standard IVP libraries, could be automated in order to keep the best of each world.

Finally, we emphasize that optimized results could be obtained when dealing with higher order differential equations such as $\ddot{x}(\cdot) = v(\cdot)$. Indeed, we have seen that to solve this problem, a decomposition into primitive constraints has to be done: $\dot{x}(\cdot) = a(\cdot)$ and $\dot{a}(\cdot) = v(\cdot)$. Hence, the contractor $\mathcal{C}_{\frac{d}{dt}}$ will be applied twice. The drawback is the wrapping effect appearing between the calls. Contractors considering n derivatives should be investigated.

Trajectories under evaluation constraints

Contents

5.1 Introduction	132
5.1.1 Contribution of this thesis	132
5.1.2 Motivations to deal with time uncertainties	133
5.2 Generic contractor for trajectory evaluation	136
5.2.1 Tube contractor for the constraint $\mathcal{L}_{\text{eval}} : z = y(t)$	136
5.2.2 Implementation	142
5.2.3 Application to state estimation	144
5.3 Robotic applications	145
5.3.1 Range-only robot localization with low-cost beacons	145
5.3.2 Reliable correction of a drifting clock	153
5.4 Conclusion	160

5.1 Introduction

5.1.1 Contribution of this thesis

The main purpose of the second part of this thesis is to study new tools to soundly deal with state equations:

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), & (5.1a) \\ z_i = g(\mathbf{x}(t_i)). & (5.1b) \end{cases}$$

In Chapter 4, we have seen how to achieve this resolution, assuming a precise knowledge on the measurement times $t_i \in \mathbb{R}$. We are now able to compute the envelope of feasible state trajectories by using tubes as domains and the contractor programming approach expanded to trajectories and enriched with $\mathcal{C}_{\frac{d}{dt}}$ and $\mathcal{C}_{\text{eval}}$. We remind that:

- $\mathcal{C}_{\text{eval}}([z_i], [y](t_i))$ contracts the tube $[y](\cdot)$ at time t_i while keeping the set of trajectories going through $[z_i]$ at this instant;
- $\mathcal{C}_{\frac{d}{dt}}([x](\cdot), [v](\cdot))$ smooths the tube $[x](\cdot)$ in order to conserve trajectories compliant with the set of feasible derivatives enclosed in $[v](\cdot)$.

$\mathcal{C}_{\text{eval}}$ and $\mathcal{C}_{\frac{d}{dt}}$ can be used together to propagate an observation over the whole domain of a trajectory.

This chapter deals with uncertain measurement times: now, the t_i are known to belong to some interval $[t_i]$. In this context, neither the value of the output z_i nor the acquisition date t_i are known exactly. Hence, the problem becomes much more complex as the uncertainties related to the t_i are difficult to propagate through the differential equation.

Some attempts of using interval analysis have been proposed in [Le Bars et al., 2012, Bethencourt and Jaulin, 2013], but we have seen that the corresponding observers cannot be considered as guaranteed. Other works, often referred as Out Of Sequence Measurement (OOSM) [Choi et al., 2009], state problems of time delay uncertainties, which can be somehow related to our problem. However, the considered time uncertainties are tight, of the same order of magnitude as

computational time step, and treated by means of covariance matrices which do not provide guaranteed results.

In contrast, we propose a new reliable tool to deal with strong temporal uncertainties. The contractor $\mathcal{C}_{\text{eval}}$, firstly defined in Equation (4.20) at page 115, will be extended to the most generic case, assuming uncertainties on the trajectory and both the measurement value and its time reference. This work has been the subject of a collaborative study between Luc Jaulin, Lyudmila Mihaylova, Fabrice Le Bars, Sandor M. Veres and the author, which led to the publication of [Rohou et al., 2018b].

5.1.2 Motivations to deal with time uncertainties

Dealing with strong time uncertainties may appear to be irrelevant or specific to special cases. Indeed, most applications involve precise measurement dates and classical state estimation methods mainly focus on resolutions in the state and observation spaces, considering spatial rather than temporal uncertainties on the variables. However, some problems could be easily considered by dealing with time uncertainties. Indeed, some practical situations can be formulated in a different way.

As an illustration, let us consider an underwater robot \mathcal{R} performing an exploration task using a side-scan sonar. Assume that a localization of the robot is based on the perception of a wreck for which the highest point \mathbf{w} is precisely geolocalized. As pictured in Figure 5.1, the wreck image $\mathbb{W}(t)$ obtained by the sonar may be distorted, stretched and would be highly noisy in practice, depending on the robot navigation [Le Bars et al., 2012]. It is a difficult problem for image processing algorithms to detect the highest point \mathbf{w} in \mathbb{W} to be used as reference for localization. However, the problem can be dealt with in a temporal way, based on the time interval $[t]$ during which the robot has seen the wreck. This observation is related to a strong temporal uncertainty: up to several seconds or minutes. Then the state estimation amounts to a range-only problem for which

$$\exists t \in [t], \exists \rho \in [\rho] \mid \rho = g(\mathbf{x}(t)) \quad (5.2)$$

with $g : \mathbb{R}^n \rightarrow \mathbb{R}$ the distance function between \mathcal{R} and the known point \mathbf{w} .

This practical situation has been encountered by the *Daurade* AUV during the exploration of the *Rade de Brest* (France). *Daurade* overflowed the wreck of the

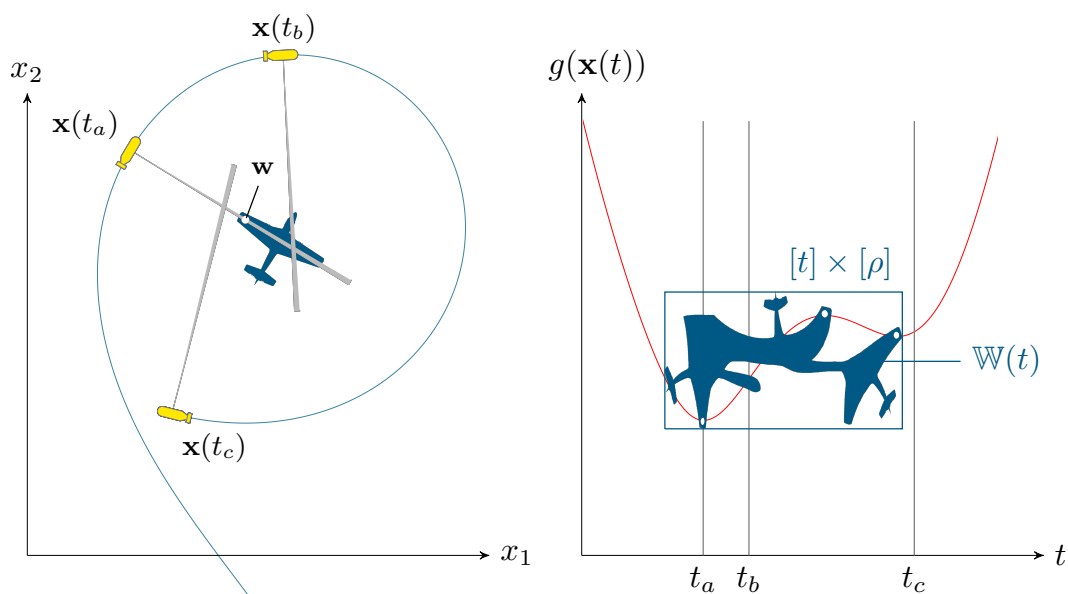
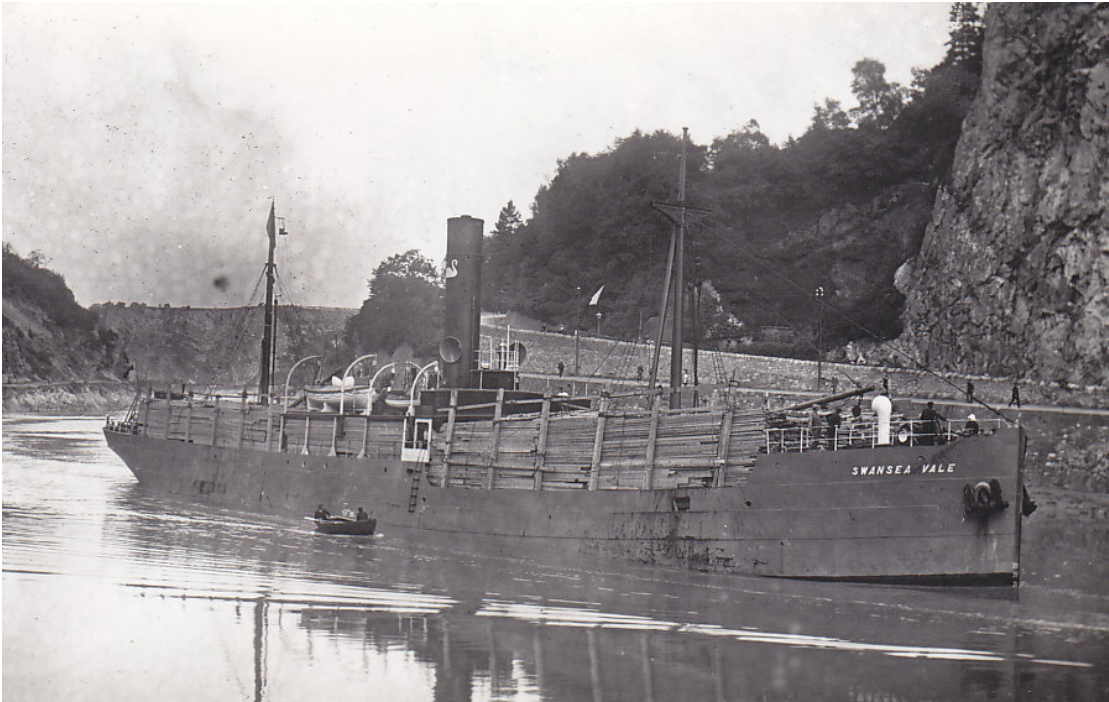


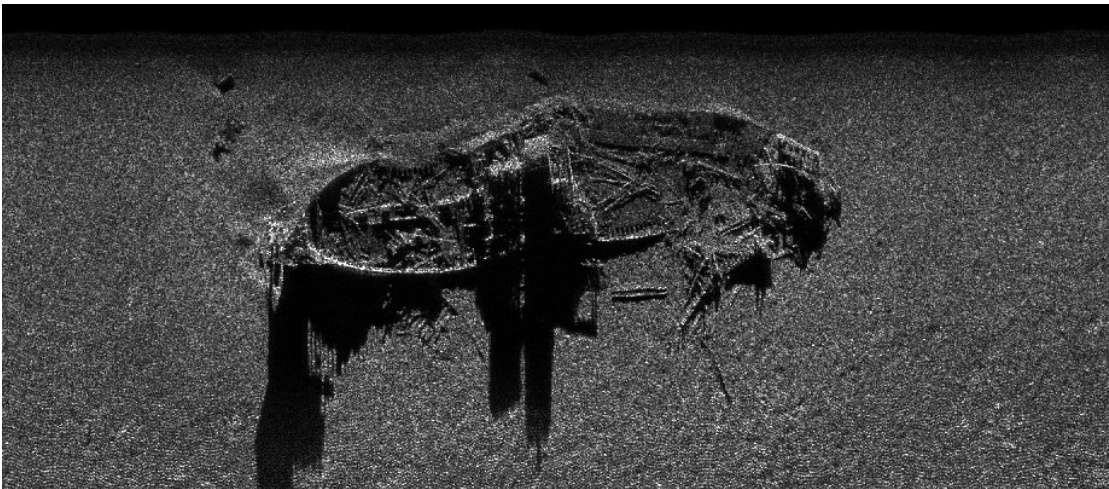
Figure 5.1: A robot \mathcal{R} perceiving a plane wreck with a side scan sonar. The observation function $g(\mathbf{x})$ represents the distance between \mathcal{R} and a point of interest \mathbf{w} on the plane, pictured by a white dot and seen at times $t_1 = t_a, t_2, t_3$. The sonar image $\mathbb{W}(t)$ is overlaid on the graph. Although \mathbf{w} has been seen three times, the t_i remain uncertain but known to belong to $[t]$. Some other robot states are illustrated at times t_a, t_b, t_c .

Swansea boat (see Figure 5.2a), for which the highest point has been precisely geolocalized by divers. The sonar images provided by the robot are distorted and strongly noised. The reader will be convinced that the image processing step, required to detect the point of interest for localization, is a highly complicated task: see Figure 5.2b. Considering the problem from a temporal point of view would allow a simpler resolution and guaranteed results. It is indeed easier to segment the wreck image from the seabed and thus to obtain a reliable envelope of the wreck. This way, we ensure to enclose the point of interest within a measurement box $[t] \times [\rho]$, where $[t]$ is the time interval during which the robot has seen the wreck.

This example shows how a classic robotic application can be related to strong time uncertainties. The current chapter is a first step towards new state estimation approaches that will focus on both the time and the state spaces. It proposes a theoretical basis to deal with the former in the most generic way and is illustrated by reproducible examples in order to highlight the interest and simplicity of the method and encourage further comparisons. Finally, the developed tools will be used in Chapter 7 for a new reliable SLAM method.



(a) The *Swansea* boat, during the First World War. *Unknown copyright.*



(b) The *Swansea* wreck perceived with a side scan sonar. The ship's funnel and superstructures cause wide shadowed areas which are the darkest parts of the sonar image. *Copyrights: SHOM, DGA-TN Brest, Michel Legris.*

Figure 5.2: Performing a so-called *wreck-based localization* is not an easy task.

5.2 Generic contractor for trajectory evaluation

This section provides a new contractor to apply an evaluation constraint $\mathcal{L}_{\text{eval}}$ to a trajectory. We recall that the dot notation $y(\cdot)$ is used to represent the whole trajectory, contrary to $y(t)$ which is the evaluation of the trajectory $y(\cdot)$ at time t .

5.2.1 Tube contractor for the constraint $\mathcal{L}_{\text{eval}} : z = y(t)$

We consider the following elementary constraint

$$\mathcal{L}_{\text{eval}} : \begin{cases} \text{Variables: } t, z, y(\cdot) \\ \text{Constraints:} \\ \quad 1. z = y(t) \\ \text{Domains: } [t], [z], [y](\cdot) \end{cases} \quad (5.3)$$

Expressed by means of quantifiers, $\mathcal{L}_{\text{eval}}$ amounts to:

$$\mathcal{L}_{\text{eval}} : \{ \exists t \in [t], \exists z \in [z], \exists y(\cdot) \in [y](\cdot) \mid z = y(t) \}. \quad (5.4)$$

The related contractor will aim at intersecting the tube by the envelope of all trajectories compliant with the bounded observation. In other words, $[y](\cdot)$ will be contracted by the tube of all $y(\cdot) \in [y](\cdot)$ going *through* the box $[t] \times [z]$, as shown in Figure 5.3. Some trajectories may partially cross the box at some point over $[t]$: the contractor must take into account the feasible propagations during the intersection process. To this end, the knowledge of the derivative $\dot{y}(\cdot)$ is required to depict the evolution of $y(\cdot)$ over $[t]$. In order to define the contractor in the most generic way, the derivative $\dot{y}(\cdot)$ will be also bounded within a tube denoted $[w](\cdot)$, thus allowing the $[y](\cdot)$ contraction even if the derivative signal is uncertain.

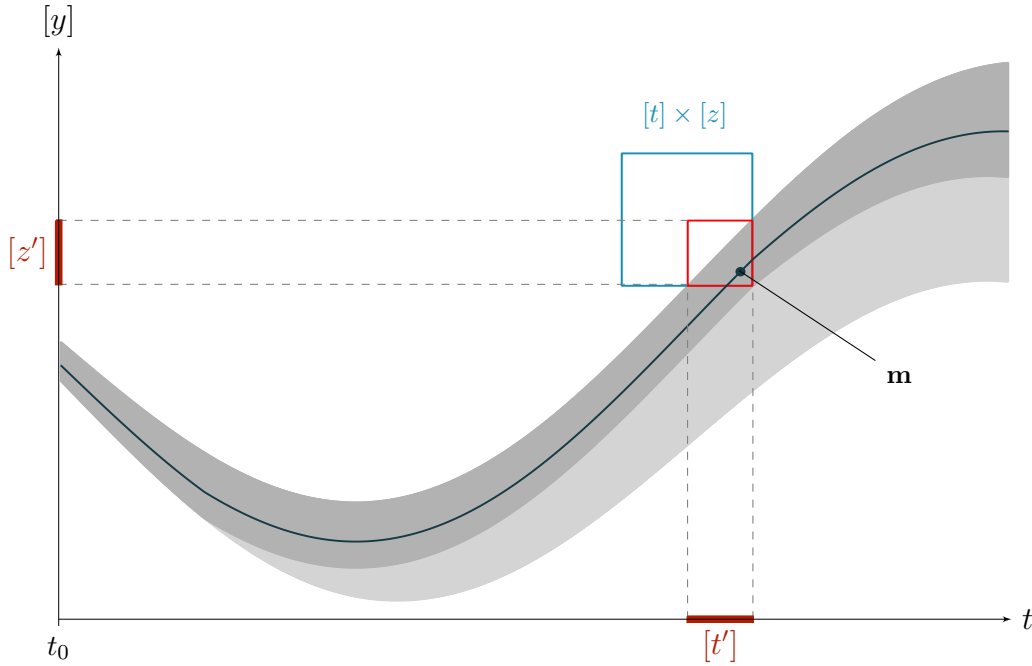


Figure 5.3: Observation on a tube $[y](\cdot)$. A given measurement $\mathbf{m} \in \mathbb{R}^2$, pictured by a black dot \bullet , is known to belong to the blue box $[t] \times [z]$ \bullet . The tube is contracted by means of $\mathcal{C}_{\text{eval}}$; the contracted part is depicted in light gray. Meanwhile, the bounded observation itself is contracted to $[t'] \times [z']$ with $[t'] \subseteq [t]$ and $[z'] \subseteq [z]$. This is illustrated by the red box \bullet . The dark line is an example of a compliant trajectory. The derivative $\dot{y}(\cdot)$, not represented here, is also enclosed within a tube.

The constraint $\mathcal{L}_{\text{eval}}$ then amounts to the following CN:

$$\mathcal{L}_{\text{eval}} : \left\{ \begin{array}{l} \mathbf{Variables: } t, z, y(\cdot), w(\cdot) \\ \mathbf{Constraints:} \\ \quad 1. z = y(t) \\ \quad 2. \dot{y}(\cdot) = w(\cdot) \\ \mathbf{Domains: } [t], [z], [y](\cdot), [w](\cdot) \end{array} \right. \quad (5.5)$$

Proposition 5.1

A contractor $\mathcal{C}_{\text{eval}}([t], [z], [y](\cdot), [w](\cdot))$ applying $\mathcal{L}_{\text{eval}}$ on intervals and tubes is defined by:

$$\begin{pmatrix} [t] \\ [z] \\ [y](\cdot) \\ [w](\cdot) \end{pmatrix} \xrightarrow{\mathcal{C}_{\text{eval}}} \begin{pmatrix} [t] \cap [y]^{-1}([z]) \\ [z] \cap [y]([t]) \\ [y](\cdot) \cap \bigsqcup_{t_1 \in [t]} \left(([y](t_1) \cap [z]) + \int_{t_1}^{\cdot} [w](\tau) d\tau \right) \\ [w](\cdot) \end{pmatrix}. \quad (5.6)$$

Proof of Proposition 5.1

To be a contractor, $\mathcal{C}_{\text{eval}}$ needs to satisfy both the contraction and the completeness properties, given in Definition 3.3, page 81.

— **Contraction property.**

The contraction property is trivial as any variable is at least contracted by itself.

— **Consistency property.**

It remains to prove that for two real numbers $t \in [t]$, $z \in [z]$ and two signals $y(\cdot) \in [y](\cdot)$ and $w(\cdot) \in [w](\cdot)$ such that $z = y(t)$, $\dot{y}(\cdot) = w(\cdot)$, we always have:

$$\begin{pmatrix} t \in [y]^{-1}([z]) & (i) \\ z \in [y]([t]) & (ii) \\ y(\cdot) \in \bigsqcup_{t_1 \in [t]} \left(([y](t_1) \cap [z]) + \int_{t_1}^{\cdot} [w](\tau) d\tau \right) & (iii) \end{pmatrix}. \quad (5.7)$$

Notation used hereafter:

Considering a generic constraint $\mathcal{L}_{\mathbf{f}} : \mathbf{b} = \mathbf{f}(\mathbf{a})$, $\mathbf{a} \in [\mathbf{a}]$, $\mathbf{b} \in [\mathbf{b}]$, the set \mathbb{B} of all vectors \mathbf{b} consistent with $\mathcal{L}_{\mathbf{f}}$ is $[\mathbf{b}] \cap \bigcup_{\mathbf{a} \in [\mathbf{a}]} \mathbf{f}(\mathbf{a})$. The closed and connected set enclosing \mathbb{B} and representable with intervals is $\bigsqcup_{\mathbf{b} \in \mathbb{B}} = [\mathbf{b}] \cap \bigsqcup_{\mathbf{a} \in [\mathbf{a}]} \mathbf{f}(\mathbf{a})$ where the symbol \bigsqcup depicts the smallest envelope containing the following terms.

Proof of Equation (5.7):

(i) the set $\mathbb{T} \subset \mathbb{R}$ of all t consistent with $\mathcal{L}_{\text{eval}}$ is:

$$\begin{aligned} \mathbb{T} &= [t] \cap \left(\bigcup_{y(\cdot) \in [y](\cdot)} \bigcup_{z \in [z]} y^{-1}(z) \right) \\ &\subset [t] \cap \left(\bigsqcup_{y(\cdot) \in [y](\cdot)} \bigsqcup_{z \in [z]} y^{-1}(z) \right) \\ &\subset [t] \cap [y]^{-1}([z]). \end{aligned} \quad (5.8)$$

An illustration of the evaluation of $[y]^{-1}([z])$ has been given in Figure 3.3, page 79.

(ii) the set $\mathbb{Z} \subset \mathbb{R}$ of all z consistent with $\mathcal{L}_{\text{eval}}$ is:

$$\begin{aligned} \mathbb{Z} &= [z] \cap \left(\bigcup_{y(\cdot) \in [y](\cdot)} \bigcup_{t \in [t]} y(t) \right) \\ &\subset [z] \cap \left(\bigsqcup_{y(\cdot) \in [y](\cdot)} \bigsqcup_{t \in [t]} y(t) \right) \\ &\subset [z] \cap [y]([t]). \end{aligned} \quad (5.9)$$

(iii) the value of $y(t)$ from t_1 is given by

$$y(t) = y_1 + \int_{t_1}^t w(\tau) d\tau, \quad \text{with } y_1 = y(t_1). \quad (5.10)$$

The set $\mathbb{Y} \subset \mathbb{R}$ of all $y(t)$ consistent with $\mathcal{L}_{\text{eval}}$ is:

$$\begin{aligned} \mathbb{Y} &= \bigcup_{t_1 \in [t]} \bigcup_{w(\cdot) \in [w](\cdot)} \bigcup_{y_1 \in [y](t_1) \cap [z]} \left(y_1 + \int_{t_1}^t w(\tau) d\tau \right) \\ &= \bigcup_{t_1 \in [t]} \left(\bigcup_{y_1 \in [y](t_1) \cap [z]} \left(y_1 + \bigcup_{w(\cdot) \in [w](\cdot)} \int_{t_1}^t w(\tau) d\tau \right) \right) \\ &= \bigcup_{t_1 \in [t]} \left(\left(\bigcup_{y_1 \in [y](t_1) \cap [z]} y_1 \right) + \int_{t_1}^t [w](\tau) d\tau \right) \\ &= \bigcup_{t_1 \in [t]} \left(([y](t_1) \cap [z]) + \int_{t_1}^t [w](\tau) d\tau \right) \\ &\subset \bigsqcup_{t_1 \in [t]} \left(([y](t_1) \cap [z]) + \int_{t_1}^t [w](\tau) d\tau \right). \end{aligned} \quad (5.11)$$

■

One should note that the tube $[y](\cdot)$ and both $[t]$ and $[z]$ may be contracted while the estimation of the derivative signal, represented by $[w](\cdot)$, will remain the same. This derives directly from Lemma 4.1 proved in Chapter 4, page 109.

Domain of contraction

$\mathcal{C}_{\text{eval}}$ will propagate the constraint as much as possible over time in a forward and backward way. Contractions may cover the whole tube domain $[t_0, t_f]$ or only a part of it, depending on the amount of uncertainties accumulated during the propagation. For instance in Figure 5.3, the contraction does not reach t_0 in backwards.

Multi-dimensions

Extension to multi-dimensional problems $\mathbf{z} = \mathbf{y}(t)$, $\mathbf{z} \in \mathbb{R}^n$, $\mathbf{y}(\cdot) \in \mathbb{R} \rightarrow \mathbb{R}^n$ amounts to applying $\mathcal{L}_{\text{eval}}$ for each component $z_j = y_j(t)$, $j \in \{1 \dots n\}$.

Inconsistency

Some inconsistency is met when the domains are not compliant with the constraint.

Lemma 5.1

If the $\mathcal{L}_{\text{eval}}$ constraint cannot be met over the domains $[t]$, $[z]$, $[y](\cdot)$, $[w](\cdot)$, then $\mathcal{C}_{\text{eval}}$ will perform a contraction to the empty set for $[t]$, $[z]$ and $[y](\cdot)$.

Proof of Lemma 5.1

We recall Equations (2.14) and (2.9) from interval arithmetic:

1. $[x] + \emptyset = \emptyset$,
2. $[x] \sqcup \emptyset = [x]$.

A non-consistent constraint $\mathcal{L}_{\text{eval}}$ is equivalent to:

$$\forall t \in [t], \forall z \in [z], \forall y(\cdot) \in [y](\cdot), \quad z \neq y(t). \quad (5.12)$$

Then we prove for each term:

$$\begin{aligned}
 & - [t] \mapsto [t] \cap [y]^{-1}([z]) = \emptyset; \\
 & - [z] \mapsto [z] \cap [y]([t]) = \emptyset; \\
 & - [y](\cdot) \mapsto [y](\cdot) \cap \bigsqcup_{t_1 \in [t]} \left(\underbrace{([y](t_1) \cap [z])}_{\emptyset} + \int_{t_1}^{\cdot} [w](\tau) d\tau \right) \\
 & = [y](\cdot) \cap \bigsqcup_{t_1 \in [t]} (\emptyset) = \emptyset. \quad \blacksquare
 \end{aligned}$$

$\mathcal{C}_{\text{eval}}$ can be used as a minimal exclusion test in order to prove that the constraint cannot apply over inconsistent domains. This can be useful for set-inversion as presented in Section 2.4, page 60. However, the minimality property of $\mathcal{C}_{\text{eval}}$ has not been studied yet.

Continuum of solutions over $[t]$

The contractor also applies when several evaluations are bounded within the same $([t], [z])$, since the union of feasible trajectories through any $t \in [t]$ is kept after contraction. As an illustration, Figure 5.1 (see page 134) presents a case of three unknown evaluations of a plane wreck, indistinguishable but enclosed within one bounded measurement $([t], [\rho])$.

Set of evaluations

When dealing with $p \in \mathbb{N}$ evaluations, a single application of $\mathcal{C}_{\text{eval}}$ for each $([t_i], [z_i])$, $i \in \{1 \dots p\}$, may not provide optimal results. Indeed, $\mathcal{C}_{\text{eval}}$ propagates an evaluation along the whole domain of $[y](\cdot)$ which may lead to new possible contractions. It is preferable to use an iterative method that applies all contractors indefinitely until they become ineffective on $[y](\cdot)$ and the $([t_i], [z_i])$'s:

$$\left(\bigcap_{i=1}^p \mathcal{C}_{\text{eval}}([t_i], [z_i], [y](\cdot), [w](\cdot)) \right)^\infty. \quad (5.13)$$

The ∞ symbol specifies an iterative application of the operators up to the fixed point. Figure 5.4 illustrates such case with a two steps iterative process.

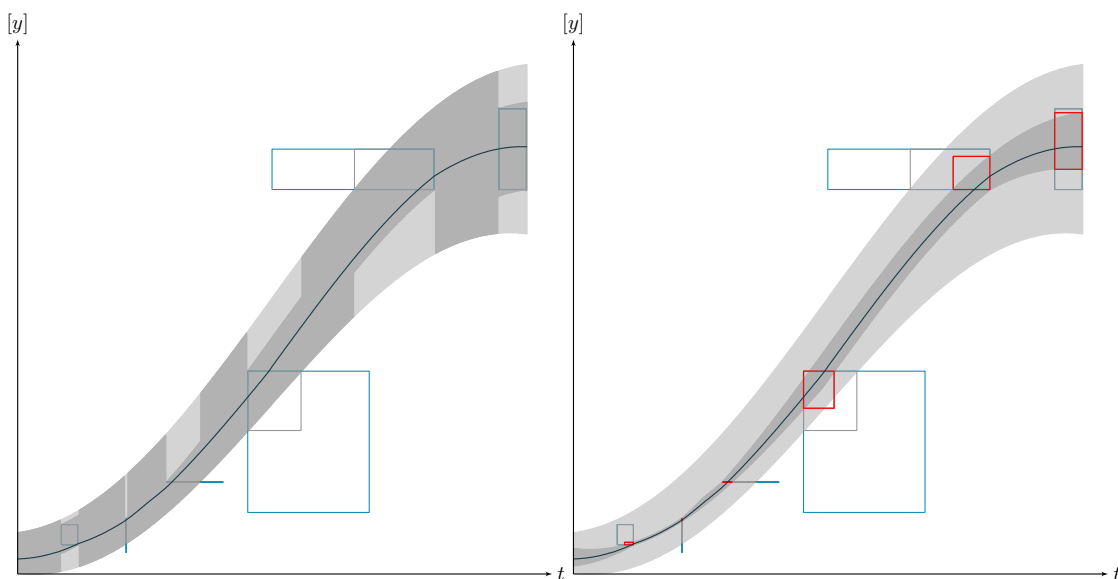


Figure 5.4: Combined $\mathcal{C}_{\text{eval}}$ contractions on a theoretical example involving a given tube $[y](\cdot)$ and some measurements. The light gray part is the set of trajectories that have been removed after contractions. Blue boxes \bullet represent the initial measurements $[t_i] \times [z_i]$. Gray boxes picture intermediate contractions of these observations, obtained from the knowledge provided by the tube. Finally, red boxes \bullet depict the contracted measurements $[t_i^*] \times [z_i^*]$ once a fixed point has been reached.

5.2.2 Implementation

Open source library

Providing the algorithm for $\mathcal{C}_{\text{eval}}$ here would take too much room. We will not detail this part in order to keep things simple and from a higher level of abstraction. The reader interested by implementation details can have a look at the *Tubex* library¹ developed during this thesis. The C++ source code of $\mathcal{C}_{\text{eval}}$ is freely available.

Compliance between discretization and observations

From an implementation point of view, contracting a tube at a given scalar t is trivial when considering an infinite number of slices, *i.e.* when $\delta \rightarrow 0$. However,

¹<http://www.simon-rohou.fr/research/tubex-lib>

the computer representation of tubes compels us to consider some discretization leading to the use of thick slices: $[t_k] \times [y_k]$, $k \in \mathbb{N}$, see Figure 5.5. In this context, the contraction of a tube at a known time t must be done without losing solutions for the considered signal to represent.

To our knowledge, there was no existing method performing this kind of contraction before. As far as we know, the contractions based on observations $[y_j](t_j)$ associated to a known date $t_j \in \mathbb{R}$ were not achieved in a guaranteed way, thus compromising the reliability of results. It is therefore fundamental to use such contractor when applying evaluation constraints on tubes, even without time uncertainties. Practically, $\mathcal{C}_{\text{eval}}$ can be used to this end by considering that the known t now belongs to $[t_k]$, the domain of the slice containing t .

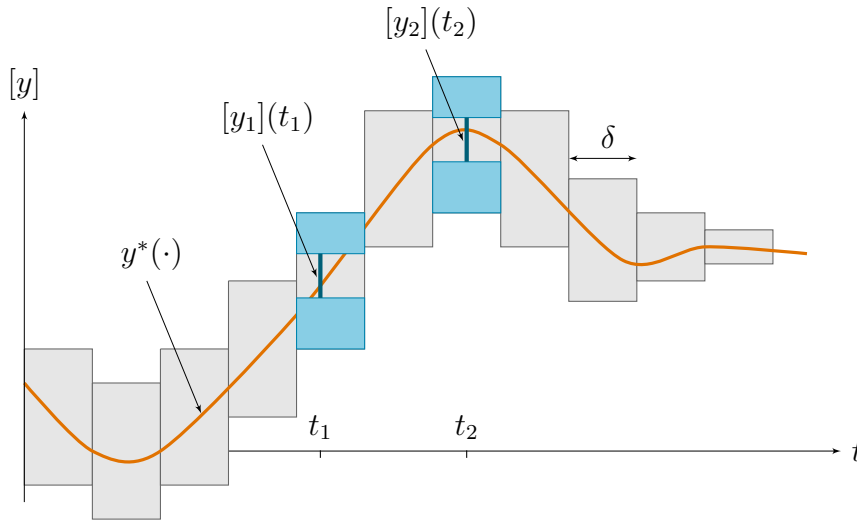


Figure 5.5: A sliced tube wrongly contracted. $[y](\cdot)$ is made of thick slices and encloses a signal $y^*(\cdot)$. Two bounded measurements $[y_j] \in \mathbb{IR}$ are made at known dates $t_j \in \mathbb{R}$. The tube is wrongly contracted at t_1 and t_2 . Fortunately the signal is not lost with the $[y_2](t_2)$ contraction, contrary to the one of $[y_1](t_1)$. A clean contraction has to be done from the knowledge of the feasible derivatives. This is achievable with $\mathcal{C}_{\text{eval}}$.

5.2.3 Application to state estimation

We have now all the material to fully deal with Equations (5.1) while considering uncertainties on any of the variables, including time. Coming back to CN (4.19), page 115, we are now able to reformulate it as:

$$\text{CN: } \left\{ \begin{array}{l}
 \text{Variables: } \mathbf{x}(\cdot), \mathbf{v}(\cdot), \mathbf{u}(\cdot), y(\cdot), w(\cdot), z_i, t_i \\
 \text{Constraints:} \\
 1. v_j(\cdot) = f_j(\mathbf{x}(\cdot), \mathbf{u}(\cdot)), j \in \{1 \dots n\} \\
 2. \dot{x}_j(\cdot) = v_j(\cdot) \Leftrightarrow \mathcal{L}_{\frac{d}{dt}}(x(\cdot), v(\cdot)) \\
 3. y(\cdot) = g(\mathbf{x}(\cdot)) \\
 4. z_i = y(t_i) \Leftrightarrow \begin{cases} z_i = y(t_i) \\ \dot{y}(\cdot) = w(\cdot) \end{cases} \Leftrightarrow \mathcal{L}_{\text{eval}}(t_i, z_i, y(\cdot), w(\cdot)) \\
 5. w(\cdot) = \dot{g}(\mathbf{x}(\cdot)) \\
 \text{Domains: } [\mathbf{x}](), [\mathbf{v}](), [\mathbf{u}](), [y](), [w](), [z_i], [t_i]
 \end{array} \right. \quad (5.14)$$

We thus introduce the variable $w(\cdot)$ into the network and consider the t_i as variables with interval domains. $w(\cdot)$ is constrained by the derivative of the observation function g , which can be estimated even in case of uncertainties on the model.

As before, each constraint is then implemented by related primitive contractors and domains are reduced while keeping solutions compliant with the state equations. The differential contractor $\mathcal{C}_{\frac{d}{dt}}$ introduced in Chapter 4 and the evaluation contractor $\mathcal{C}_{\text{eval}}$ are respectively used for the above steps (2) and (4). Algebraic constraints (1), (3), (5) are implemented with a composition of algebraic contractors on tubes.

State estimation then consists in the following contractors calls:

1. $\mathcal{C}_{f_j}([v_j](), [\mathbf{x}](), [\mathbf{u}]())$
2. $\mathcal{C}_{\frac{d}{dt}}([x_j](), [v_j]()), j \in \{1 \dots n\}$
3. $\mathcal{C}_g([y](), [\mathbf{x}]())$
4. $\mathcal{C}_{\text{eval}}([t_i], [z_i], [y](), [w]())$
5. $\mathcal{C}_{\dot{g}}([w](), [\mathbf{x}]())$

5.3 Robotic applications

The simplicity of the method allows us to deal with a wide range of state estimation problems, including time uncertainties. This section proposes two reproducible examples to illustrate the approach. The following simulations are based on analytical expressions and simple data in order to encourage future comparisons with the method provided in this document.

5.3.1 Range-only robot localization with low-cost beacons

We are now able to address the problem presented in the introduction of Chapter 3, page 74. See the following Figure 5.6. For ease of reading, we recall the equations already introduced in Section 3.4. The simulation will be run from $t_0 = 0$ to $t_f = 64$.

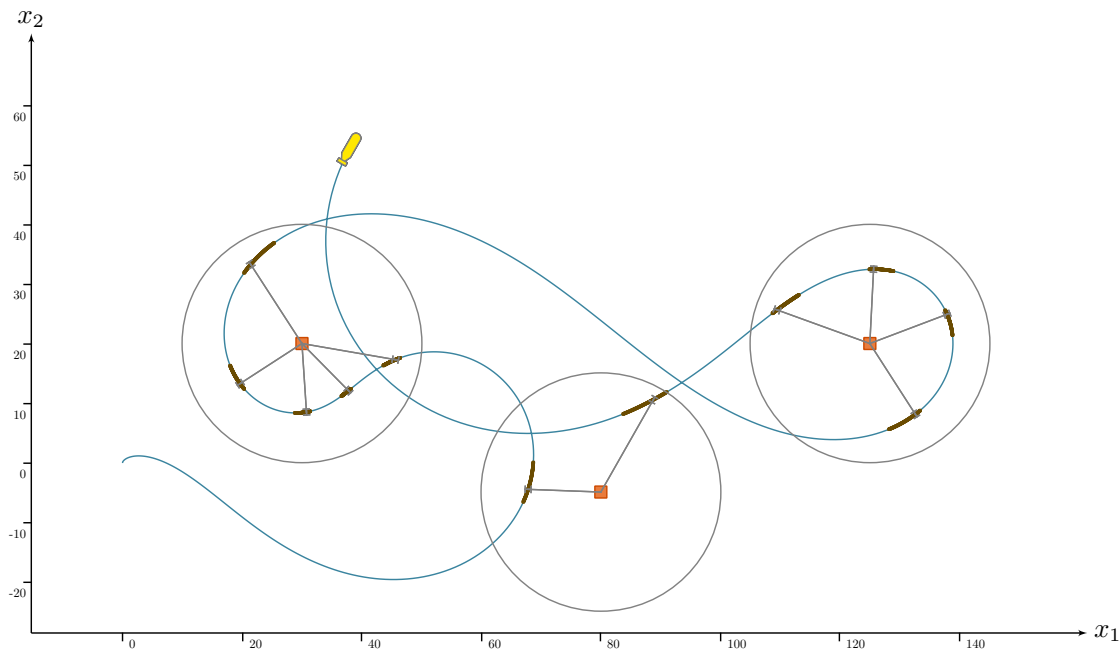


Figure 5.6: Map of the range-only localization problem based on few asynchronous measurements. Emitting beacons, drawn by red boxes \bullet , send some range signals pictured by gray lines and received by the robot at uncertain times along its trajectory, plotted in blue \bullet . This application is challenging as it involves differential equations, non-linearities and uncertainties that are both spatial and temporal.

Test case

A robot \mathcal{R} is described by its state $\mathbf{x} = (x_1, x_2, x_3 = \psi, x_4 = \vartheta)^\top$ where (x_1, x_2) depicts its location, ψ its heading and ϑ its speed. The system is modeled by the following evolution function:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 = \dot{\psi} \\ \dot{x}_4 = \dot{\vartheta} \end{pmatrix} \xrightarrow{\mathbf{f}} \begin{pmatrix} \vartheta \cos(\psi) \\ \vartheta \sin(\psi) \\ u_1 \\ u_2 \end{pmatrix}. \quad (5.15)$$

The state $\mathbf{x}(t)$ is submitted to the input $\mathbf{u}(t)$ whose value is bounded as:

$$\mathbf{u}(t) \in [\mathbf{u}](t) = \begin{pmatrix} -9/20 \cos(t/5) \\ 1/10 + \sin(t/4) \end{pmatrix} + \frac{1}{1000} \begin{pmatrix} [-1, 1] \\ [-1, 1] \end{pmatrix}. \quad (5.16)$$

Eventually, we change the initial condition to remove the knowledge on the initial position. The condition \mathbf{x}_0 depicting the robot at t_0 is now assumed to be bounded such that

$$\mathbf{x}_0 \in \begin{pmatrix} [-\infty, \infty] \\ [-\infty, \infty] \\ \pi/2 + [-0.01, 0.01] \\ [-0.01, 0.01] \end{pmatrix}. \quad (5.17)$$

The robot moves amongst low-cost beacons $\mathbf{b}_k, k \in \{\alpha, \beta, \gamma\}$, thus implying drifting clocks (strong temporal uncertainties) and measurement errors. These emitters have a maximum signal range $\rho_{\max} = 20\text{m}$ and send bounded signals $z_i \in [z_i]$ on a regular basis with time uncertainties: $t_i \in [t_i]$. The observation function g_k – Equation (5.1b) – related to a beacon \mathbf{b}_k is a distance function between \mathcal{R} and the beacon. The problem will highlight the use of $\mathcal{C}_{\text{eval}}$ based on a set of fleeting bounded measurements, see Table 5.2.

Table 5.1: Beacons' locations.

k	\mathbf{b}_k
α	(30, 20)
β	(80, -5)
γ	(125, 20)

Table 5.2: List of measurements ($[t_i], [z_i]$).

i	k	$[t_i]$	$[z_i]$
1	β	[14.75, 15.55]	[11.69, 12.69]
2	α	[20.80, 21.60]	[15.40, 16.40]
3	α	[23.80, 24.60]	[10.62, 11.62]
4	α	[26.80, 27.60]	[11.05, 12.05]
5	α	[29.80, 30.60]	[11.87, 12.87]
6	α	[32.80, 33.60]	[15.31, 16.31]
7	γ	[44.35, 45.15]	[13.65, 14.65]
8	γ	[47.35, 48.15]	[13.32, 14.32]
9	γ	[50.35, 51.15]	[12.03, 13.03]
10	γ	[53.35, 54.15]	[15.98, 16.98]
11	β	[56.75, 57.55]	[17.45, 18.45]

Resolution

The problem amounts to CN (5.18). The constraints form a network partially pictured in Figure 5.7. Tubes are initialized to $[-\infty, \infty] \forall t$ except for $[\mathbf{u}](\cdot)$, set according to Equation (5.16). Furthermore in order to apply $\mathcal{C}_{\text{eval}}$, an estimation of the feasible derivatives of $[y_k](\cdot)$, represented by a tube $[w_k](\cdot)$, has to be computed. This is easily done analytically by deriving the distance function g_k .

$$\text{CN: } \left\{ \begin{array}{l}
 \mathbf{Variables: } \mathbf{x}(\cdot), \mathbf{v}(\cdot), \mathbf{u}(\cdot), \{(t_i, z_i)\}, \{y_k(\cdot)\}, \{w_k(\cdot)\} \\
 \mathbf{Constraints:} \\
 \quad 1. \text{ Evolution function:} \\
 \quad \quad \text{--- } \mathbf{v}(\cdot) = \mathbf{f}(\mathbf{x}(\cdot), \mathbf{u}(\cdot)) \\
 \quad \quad \text{--- } \dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot) \\
 \quad \quad \text{--- } x_3(0) \in \pi/2 + [-0.01, 0.01] \\
 \quad \quad \text{--- } x_4(0) \in [-0.01, 0.01] \\
 \quad 2. \text{ Observation function:} \\
 \quad \quad \text{--- } y_k(\cdot) = \sqrt{(x_1(\cdot) - b_{k,1})^2 + (x_2(\cdot) - b_{k,2})^2} \\
 \quad \quad \text{--- } w_k(\cdot) = \frac{(x_1(\cdot) - b_{k,1}) \cdot v_1(\cdot) + (x_2(\cdot) - b_{k,2}) \cdot v_2(\cdot)}{\sqrt{(x_1(\cdot) - b_{k,1})^2 + (x_2(\cdot) - b_{k,2})^2}} \\
 \quad \quad \text{--- } \dot{y}_k(\cdot) = w_k(\cdot) \\
 \quad 3. \text{ Measurements:} \\
 \quad \quad \text{--- } z_i = y_k(t_i) \\
 \mathbf{Domains: } [\mathbf{x}](\cdot), [\mathbf{v}](\cdot), [\mathbf{u}](\cdot), \{[t_i], [z_i]\}, \{[y_k](\cdot)\}, \{[w_k](\cdot)\}
 \end{array} \right. \quad (5.18)$$

The process involving contractors is then executed. The fixed point is reached over 52 iterations in 2 minutes, but the main contractions are already obtained before the sixth iteration, as pictured in Figure 5.8: the position domain is slightly reduced during the next steps. A projection of the computed results is pictured in Figure 5.9. This example shows how the constraint satisfaction approach behaves: in an iterative way and without a necessary knowledge on the initial conditions. At the end, the true state trajectory $\mathbf{x}^*(\cdot)$ is guaranteed to lie within the tube $[\mathbf{x}](\cdot)$.

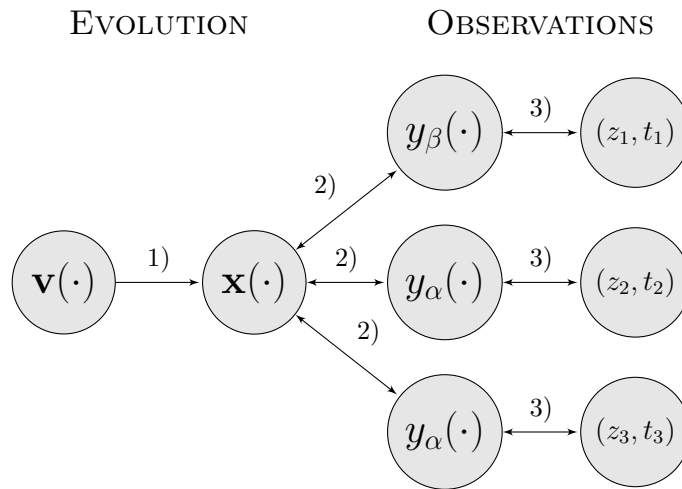


Figure 5.7: Constraint network detailing the relations of the first three measurements of Table 5.2. Arrows indicate the possible directions of information propagation. For ease of understanding, derivatives $w_k(\cdot)$ are not represented here.

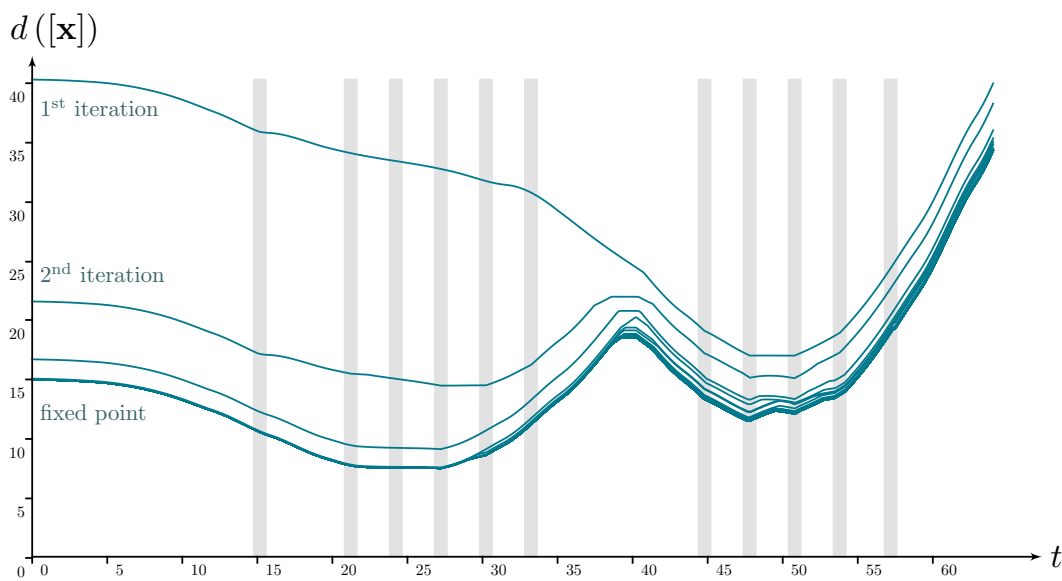
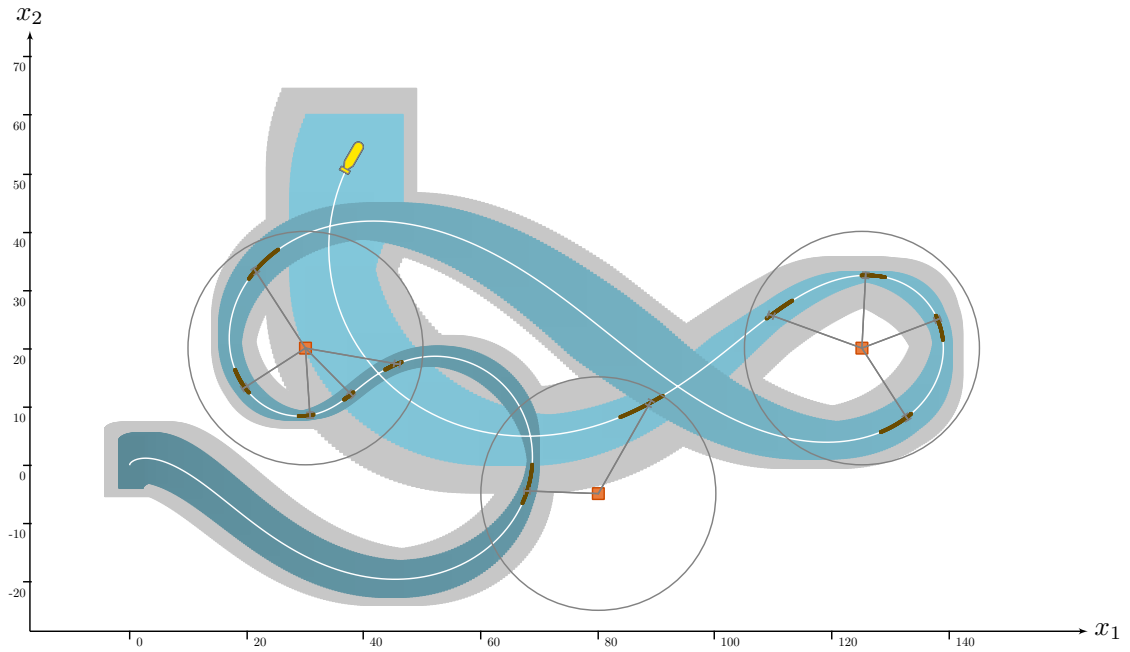
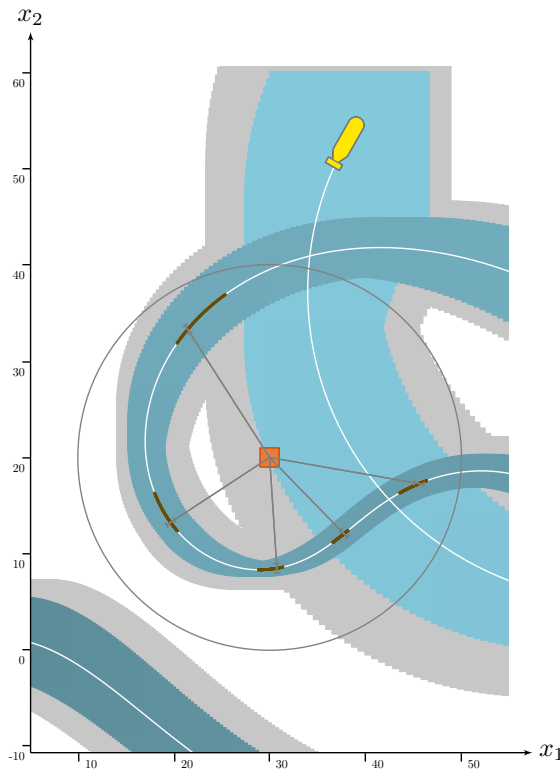


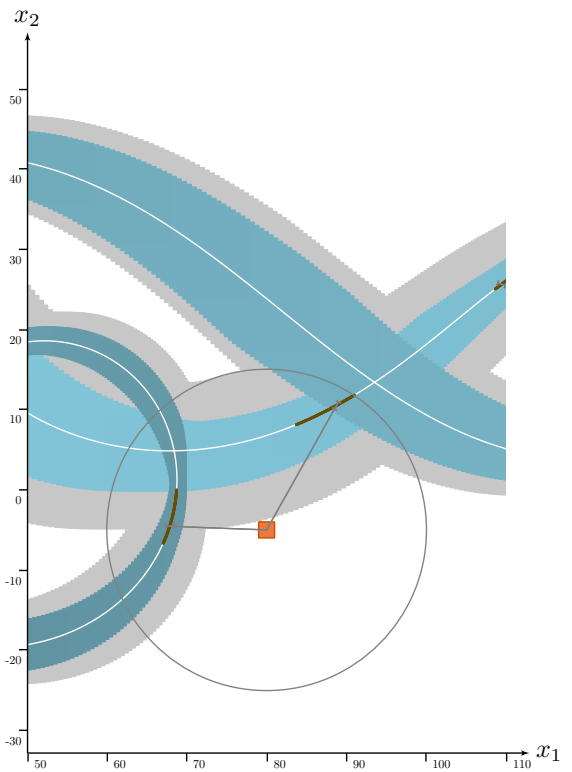
Figure 5.8: Thicknesses of the position estimation $[x_1](\cdot) \times [x_2](\cdot)$ as defined for Figure 3.12, page 94. Uncertain measurements' times $[t_i]$ are projected in light gray. The fixed point has been reached after 52 iterations while almost final results were already obtained during the first steps.



(a) Overview of the simulation.



(b) Zoom on Beacon \mathbf{b}_α .



(c) Zoom on Beacon \mathbf{b}_β .

Figure 5.9: State estimation of a mobile robot amongst a set of low-cost beacons. The initial position $(0,0)$ is not known. Beacons, pictured by red boxes \bullet , are sending signals till a range limit depicted by circles. Time uncertainties $[t_i]$ are projected along the robot path with dark thick lines. The true poses of the robot are pictured by the white line, enclosed within the estimated tubes $[x_1](\cdot) \times [x_2](\cdot)$ projected in blue \bullet and gray \bullet . The pessimism induced by time uncertainties is represented in light gray. In other words, the blue part depicts a state estimation assuming a precise knowledge on the t_i 's. In each case, these tubes are the results obtained after an iterative call of the contractors up to the fixed point.

We emphasize that results could be improved by bisecting the state space, for instance with the SIVIA algorithm presented in Section 2.4.2 at page 61. Indeed, several states $\mathbf{x}(t_i) \in [\mathbf{x}](t_i)$ may lead to the same observation $z_i = g(\mathbf{x}(t_i))$ since the function g is not injective. Then, bisections along $[x_1](\cdot)$ or $[x_2](\cdot)$ can help to consider independently several states consistent with the observation, and reject them if not consistent with the other constraints.

Analytical reformulation

One should note that the following constraint (from CN (5.18))

$$w_k = \frac{(x_1 - b_{k,1}) \cdot v_1 + (x_2 - b_{k,2}) \cdot v_2}{\sqrt{(x_1 - b_{k,1})^2 + (x_2 - b_{k,2})^2}} \quad (5.19)$$

will not apply in case of a null denominator. Its interval evaluation will be $[-\infty, \infty]$ which cannot lead to a contraction. This situation is typically met at the beginning of the resolution when $[x_1]$ and $[x_2]$ are unbounded:

$$\begin{aligned} [\mathbf{x}] = [-\infty, \infty] &\implies b_{k,1} \in [x_1], \\ &\iff 0 \in [x_1] - b_{k,1}, \\ &\iff 0 \in ([x_1] - b_{k,1})^2, \\ &\dots \\ &\iff 0 \in \sqrt{\left(([x_1] - b_{k,1})^2 + ([x_2] - b_{k,2})^2 \right)}. \end{aligned} \quad (5.20)$$

Then the resolution cannot start since the tube $[w_k](\cdot)$ will stay unbounded too, preventing from any contraction of $[y_k](\cdot)$ and so $[\mathbf{x}](\cdot)$. Indeed, a contraction cannot be expected from both $\mathcal{C}_{\frac{d}{dt}}$ and $\mathcal{C}_{\text{eval}}$ in case of unknown derivatives. A solution consists in a reformulation of Constraint (5.19) so that the interval evaluation of its denominator cannot contain zero. A possible formula is the following:

$$w_k = \frac{\text{sign}(x_1 - b_{k,1})}{\sqrt{1 + \left(\frac{x_2 - b_{k,2}}{x_1 - b_{k,1}} \right)^2}} \cdot v_1 + \frac{\text{sign}(x_2 - b_{k,2})}{\sqrt{1 + \left(\frac{x_1 - b_{k,1}}{x_2 - b_{k,2}} \right)^2}} \cdot v_2. \quad (5.21)$$

If both $[x_1]$ and $[x_2]$ are unbounded, then the interval evaluation of w_k will be

$$\begin{aligned}
 w_k &\in \frac{\text{sign}([-\infty, \infty])}{\sqrt{1 + \left(\frac{[-\infty, \infty]}{[-\infty, \infty]}\right)^2}} \cdot [v_1] + \frac{\text{sign}([-\infty, \infty])}{\sqrt{1 + \left(\frac{[-\infty, \infty]}{[-\infty, \infty]}\right)^2}} \cdot [v_2], \\
 &\in \frac{[-1, 1]}{\sqrt{1 + [0, \infty]}} \cdot [v_1] + \frac{[-1, 1]}{\sqrt{1 + [0, \infty]}} \cdot [v_2], \\
 &\in \frac{[-1, 1]}{[1, \infty]} \cdot [v_1] + \frac{[-1, 1]}{[1, \infty]} \cdot [v_2], \\
 &\in [-1, 1] \cdot [v_1] + [-1, 1] \cdot [v_2].
 \end{aligned} \tag{5.22}$$

At first, $[v_1]$ and $[v_2]$ are set to $[-\infty, \infty]$ but their contraction will be based on other variables and constraints. Once bounded values will be set for $[\mathbf{v}]$, then the contractions for $[w_k]$ will start. We emphasize that the inclusion function of Equation (5.22) is not minimal but its purpose is to sufficiently reduce the domain of $[w_k]$ in order to trigger the evaluation of related variables and enable the iterative resolution. Hence, both Constraints (5.19) and (5.22) can be considered in the same time.

5.3.2 Reliable correction of a drifting clock

A complementary illustration of this work is the situation of a drifting clock: an isolated clock that does not run at the same rate as a reference clock. This problem amounts to increasing time uncertainties that can be reduced using a collaborative method.

Test case

An underwater system, lying on the seabed at $(0, 0, -10)$, is equipped with a low-cost drifting clock. Absolute time reference is represented by t while the time value τ provided by the underwater clock is drifting² such that:

$$\tau = h(t) = 0.045t^2 + 0.98t. \quad (5.23)$$

This quadratic drift is represented in Figure 5.13. However, this information is not known: the problem consists in estimating this function. Instead, we shall assume the following bounded derivative of $h(\cdot)$, that could be obtained from the clock datasheet:

$$\dot{h}(t) \in [0.08, 0.12] \cdot t + [0.97, 1.08]. \quad (5.24)$$

The problem is constrained thanks to a localized robot \mathcal{B} evolving at the surface and a set of measured distances $z_i \in \mathbb{R}$ between the robot and the underwater clock, see Figures 5.10, 5.11 and Table 5.3.

The boat's trajectory $\mathbf{x}(\cdot) : \mathbb{R} \rightarrow \mathbb{R}^2$ is preprogrammed, forming a kind of *ephemeris* for the clock. In astronomy, an ephemeris provides the positions of astronomical objects in the sky at a given time. Here, the boat is employed in the same way as stars have been used for celestial navigation on Earth. This way, the beacon already knows where the robot must be at time t . Conversely, detecting the location of \mathcal{B} provides a temporal information to be compared with the embedded time value. Hence, the boat can be used by the underwater clock to correct this temporal drift.

²In this academic example, in order to keep things simple, we consider that the clock perfectly matches the absolute time reference at $t = 0$: $h(0) = 0$. But any unknown offset could be assumed with our resolution method.

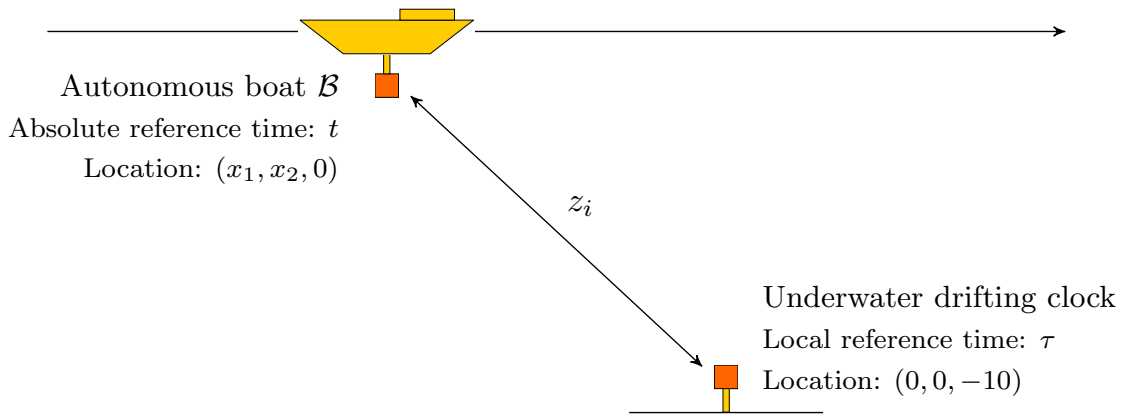


Figure 5.10: Illustrating the problem of a drifting clock corrected by ephemerides provided by an autonomous boat \mathcal{B} . The beacon holding the clock receives distance measurements from the boat once in a while.

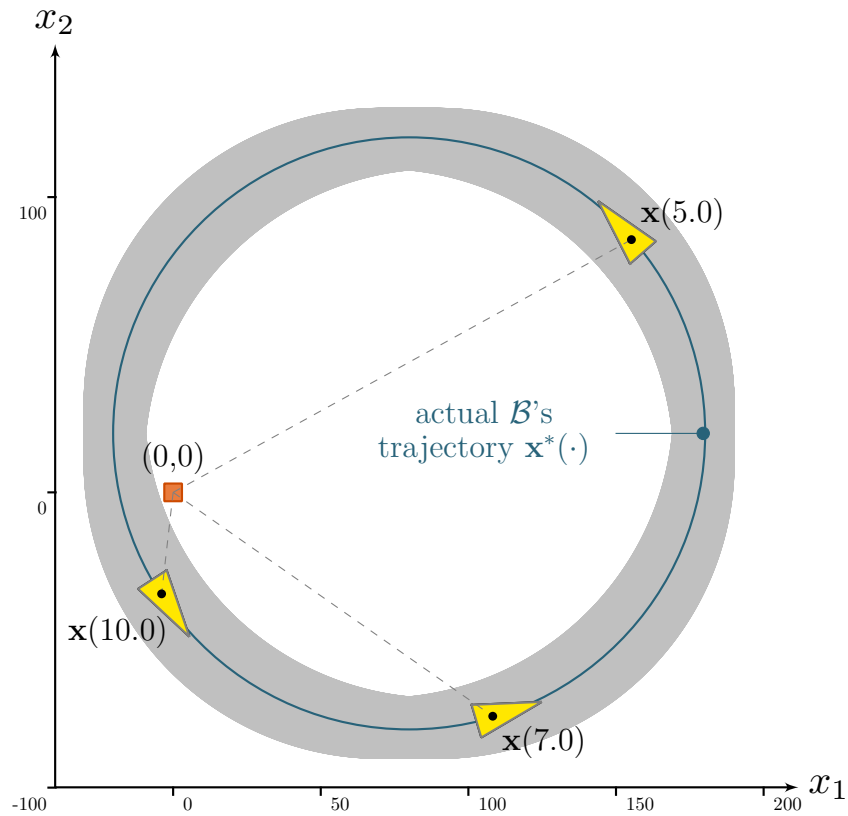


Figure 5.11: Top view of \mathcal{B} 's trajectory around the underwater beacon, depicted in red \bullet . The tube $[\mathbf{x}](\cdot)$, projected in gray \bullet , gathers the feasible positions of \mathcal{B} .

Table 5.3: List of measurements $(\tau_i, [z_i])$.

i	τ_i	$[z_i]$	i	τ_i	$[z_i]$
1	1.57	[152.47, 156.47]	5	9.88	[167.09, 171.09]
2	3.34	[34.67, 38.67]	6	12.46	[60.03, 64.03]
3	5.32	[102.38, 106.38]	7	15.25	[78.76, 82.76]
4	7.50	[184.45, 188.45]	8	18.24	[175.88, 179.88]

However, the boat may not precisely respect the defined schedule. The ephemeris thus consists in a tube $[\mathbf{x}](\cdot)$ taking into account the possible errors of the boat's locations, see Figure 5.11. The velocity $\mathbf{v}(\cdot)$ of \mathcal{B} is also bounded:

$$\mathbf{x}(\cdot) \in \begin{pmatrix} [70, 90] \\ [10, 30] \end{pmatrix} + 100 \begin{pmatrix} \cos(\cdot) \\ \sin(\cdot) \end{pmatrix}, \quad (5.25)$$

$$\mathbf{v}(\cdot) \in \begin{pmatrix} [-0.1, 0.1] \\ [-0.1, 0.1] \end{pmatrix} + 100 \begin{pmatrix} -\sin(\cdot) \\ \cos(\cdot) \end{pmatrix}. \quad (5.26)$$

Resolution

The problem amounts to the following CN (5.27). Function $y(\cdot)$ now depicts the prevision of the distances separating the boat from the beacon. Each measurement is referenced by τ_i that are temporal drifting values given by the underwater clock. The estimation of $h(\cdot)$, depicting the drift and bounded within a tube $[h](\cdot)$, will provide a reliable enclosure of the reference time t_i corresponding to each τ_i : $t_i \in [h]^{-1}(\tau_i)$.

The measurements values z_i are now referenced by $([t_i], [z_i])$ and will then be constrained by $y(\cdot)$ through $\mathcal{L}_{\text{eval}}$. In particular, the estimation $[t_i]$ will be refined. Another $\mathcal{L}_{\text{eval}}$ will constrain the $h(\cdot)$ trajectory, based on the temporal pairs $([t_i], \tau_i)$. To this end, the derivative of $h(\cdot)$, denoted $\phi(\cdot)$ and bounded by Equation (5.24), will be considered too.

$$\text{CN: } \left\{ \begin{array}{l}
 \mathbf{Variables: } \{ (t_i, z_i) \}, \mathbf{x}(\cdot), \mathbf{v}(\cdot), h(\cdot), \phi(\cdot), y(\cdot), w(\cdot) \\
 \mathbf{Constraints:} \\
 \quad 1. \text{ Ephemerides (i.e. boat's locations):} \\
 \quad \quad - \dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot) \\
 \quad 2. \text{ Beacon-boat distance function:} \\
 \quad \quad - y(\cdot) = \sqrt{x_1(\cdot)^2 + x_2(\cdot)^2 + (-10)^2} \\
 \quad \quad - w_k(\cdot) = (x_1(\cdot) \cdot v_1(\cdot) + x_2(\cdot) \cdot v_2(\cdot)) / y(\cdot) \\
 \quad \quad - \dot{y}_k(\cdot) = w_k(\cdot) \\
 \quad 3. \text{ Drifting time function:} \\
 \quad \quad - \dot{h}(\cdot) = \phi(\cdot) \\
 \quad \quad - h(0) = 0 \\
 \quad 4. \text{ Measurements:} \\
 \quad \quad - z_i = y(t_i) \\
 \quad \quad - \tau_i = h(t_i) \\
 \mathbf{Domains: } \{ ([t_i], [z_i]) \}, [\mathbf{x}](\cdot), [\mathbf{v}](\cdot), [h](\cdot), [\phi](\cdot), [y](\cdot), [w](\cdot)
 \end{array} \right. \quad (5.27)$$

As before, contractors are called on tubes in place of constraints on trajectories. Tubes $[\mathbf{x}](\cdot)$, $[\mathbf{v}](\cdot)$, $[\phi](\cdot)$ are respectively initialized according to Equations (5.25), (5.26) and (5.24). This time, the contractor of interest $\mathcal{C}_{\text{eval}}$ will be called twice, see Constraints (4) of the above CN (5.27).

Tube inversions on $[h](\cdot)$ provide the corresponding enclosures $[t_i] = [h]^{-1}(\tau_i)$ of absolute reference times t_i , see Figure 5.13. The $[t_i]$ are then used to read the ephemeris and are contracted by:

$$([t_i], [z_i], [y](\cdot), [w](\cdot)) \xrightarrow{\mathcal{C}_{\text{eval}}} ([t_i], [z_i], [y](\cdot), [w](\cdot)). \quad (5.28)$$

The contracted $[t_i]$ can then be used to reduce the tube $[h](\cdot)$ using the same contractor:

$$\left([t_i], \tau_i, [h](\cdot), [\phi](\cdot)\right) \xrightarrow{\mathcal{C}_{\text{eval}}} \left([t_i], \tau_i, [h](\cdot), [\phi](\cdot)\right). \quad (5.29)$$

An iterative resolution process is executed up to a fixed point. Indeed, the first contraction of $[h](\cdot)$ – Equation (5.29) – raises new constraints for the contraction of the $[t_i]$, Equation (5.28). In this example, constraints have been propagated over 5 steps of computation in less than 2 seconds.

Finally, the contracted tube $[h](\cdot)$ reflects the clock drift correction, see Figure 5.13. We emphasize that the real drift $h(t)$, unknown of the resolution, remains enclosed in its final envelope $[h](t), \forall t$.

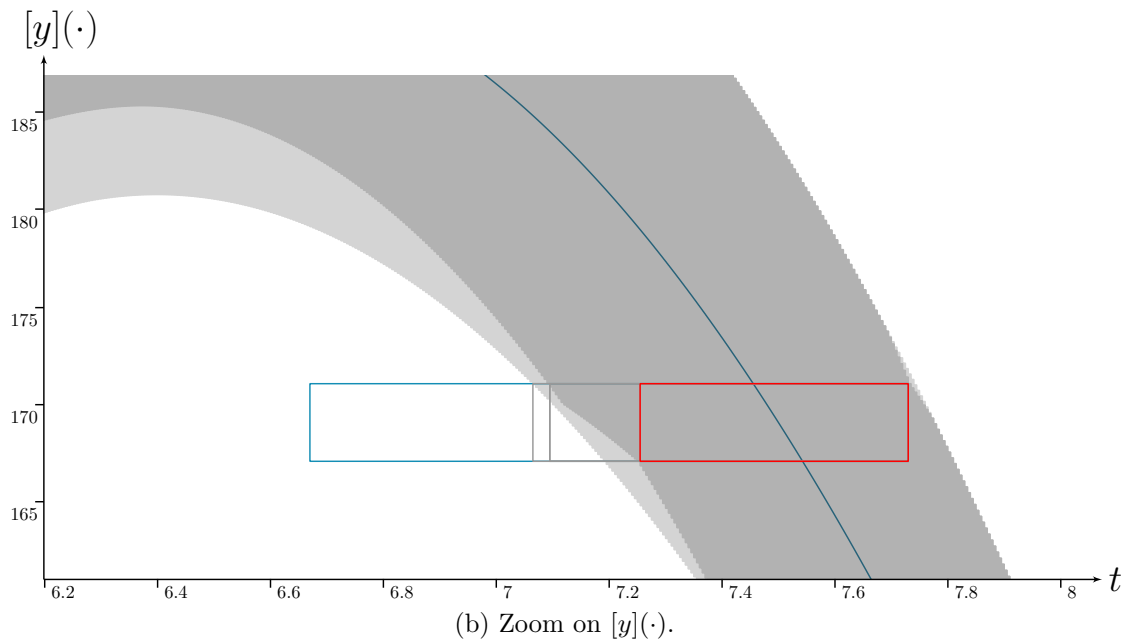
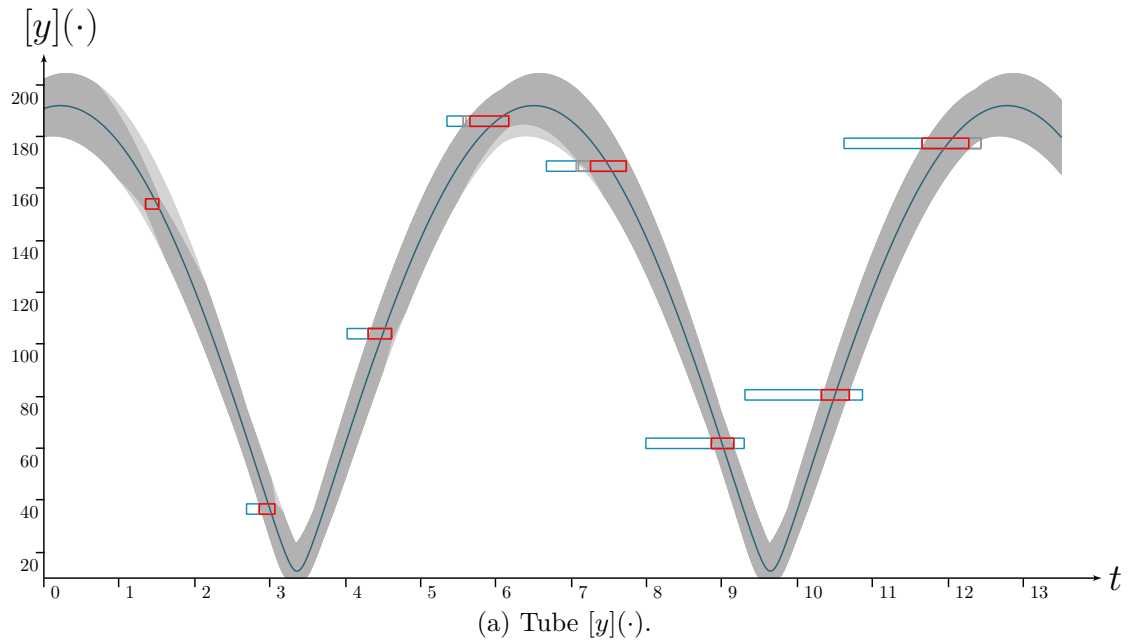


Figure 5.12: Tube $[y](\cdot)$ representing the reliable prevision of the distances between the boat and the beacon (so-called *ephemeris*). $[y](\cdot)$ is submitted to a set of measurements pictured by blue boxes \bullet , before their final contraction in red \bullet . This demonstrates the contraction of strong time uncertainties by $\mathcal{C}_{\text{eval}}$ thanks to the knowledge provided by the tube itself.

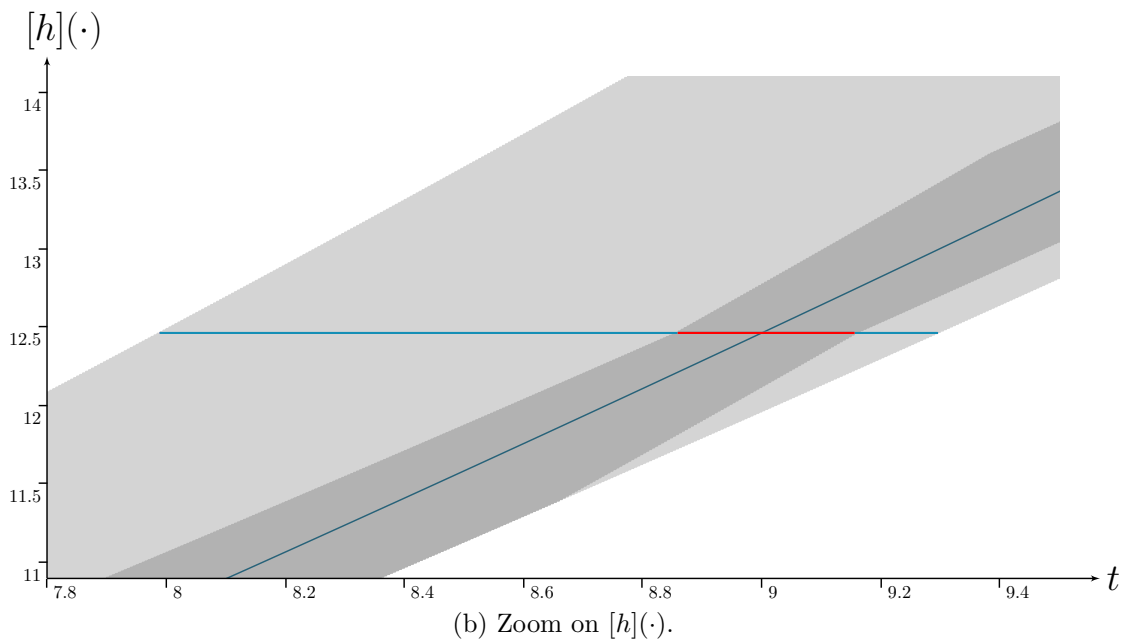
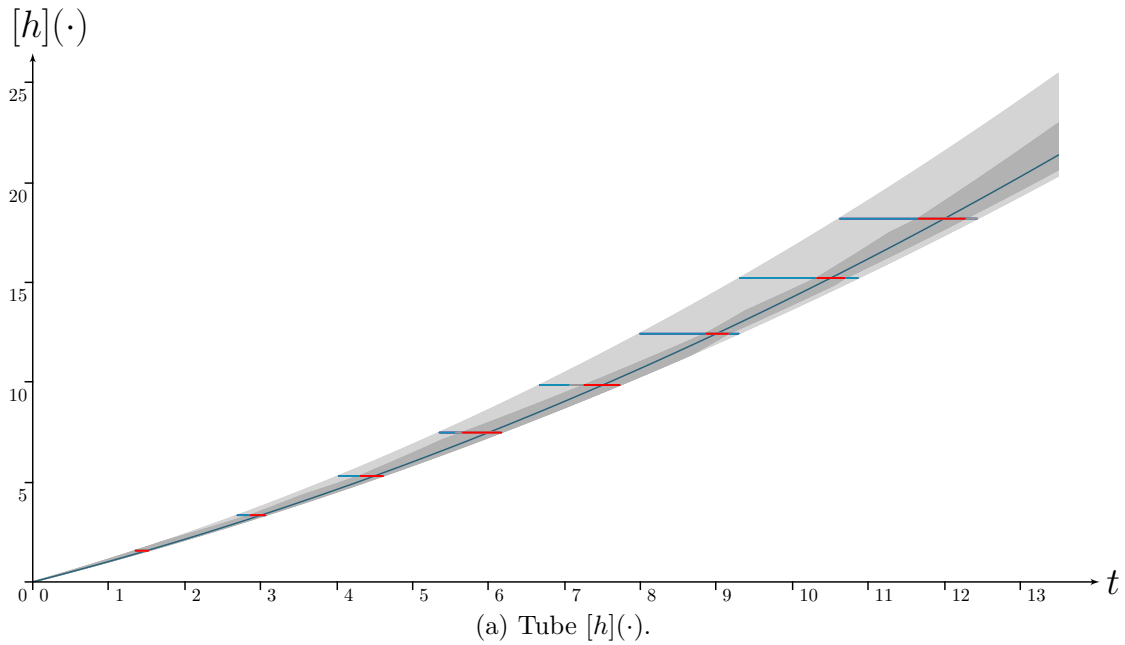


Figure 5.13: Tube $[h](\cdot)$ representing the clock drift. For a given time τ_i , $[h]^{-1}(\tau_i)$ provides an enclosure $[t_i]$ of the time reference t_i . When $[t_i]$ is contracted by means of ephemeris $[y](\cdot)$ and $\mathcal{C}_{\text{eval}}$ (see Figure 5.12), the information can be propagated back to $[h](\cdot)$. The tube's contracted part is pictured in light gray while the real drift expressed by Equation (5.23) is plotted in blue \bullet .

5.4 Conclusion

This chapter and [Rohou et al., 2018b] provide an original method to deal with time uncertainties in non-linear and differential systems. As for Chapters 3 and 4, a contractor programming approach is applied over trajectories. This chapter focuses on the elementary constraint of a trajectory evaluation, allowing to deal with uncertainties on any of the variables such as time. This is achieved by means of a new contractor $\mathcal{C}_{\text{eval}}$.

From a practical standpoint, this new contractor allows one to consider state estimation problems from a temporal point of view, where the time t becomes an unknown variable to be estimated. This novel approach, here introduced over simple examples of mobile robotics, opens the way to further applications in which the consideration of time uncertainties is relevant. This will be the cornerstone of the following Part III, presenting robotic contributions related to temporal uncertainties.

Prospects

Future work will concentrate on the wreck-based localization problem presented in the beginning of this chapter. $\mathcal{C}_{\text{eval}}$ could provide a reliable way to deal with sonar images presenting noise and distortions, in the most simple way. A reciprocal approach would consist in reducing the uncertainties in the perception of the wreck, assuming an accurate positioning of the robot obtained by other means. This approach could simplify the understanding of sonar acquisitions by automatically and reliably removing inconsistent parts of the images such as shadowed areas, see Figure 5.2b, page 135.

Another field to investigate is the automatic reformulation of analytical expressions in case of unbounded results. This chapter gave at page 151 a typical example of a substantial pessimism that has been overcome by a simple reformulation of the problem. The integration of analytical solvers in the resolution process would be of interest.

Part III

Robotics-related contributions

The ultimate application of this thesis is a new reliable SLAM method suited for long-term missions in poor environments. The resolution of this localization problem requires a set of elementary tools that have been developed in Part II. Chapter 7 aims at applying the new contractors on inter-temporal constraints of interest for the SLAM.

However, in order to soundly solve the problem, we still need to study a last constraint. This will be related to the development of another tool to prove loops along robot trajectories. Chapter 6 is our third contribution, in which we combine the topological theory together with interval analysis and tubes. The result is a new loop existence test that will be directly applied in the last chapter for underwater applications.

Looped trajectories: from detections to proofs

Contents

6.1	Introduction	164
6.1.1	The difference between detection and verification	164
6.1.2	Proprioceptive <i>vs.</i> exteroceptive measurements	165
6.1.3	The two-dimensional case	165
6.2	Proprioceptive loop detections	166
6.2.1	Formalization	166
6.2.2	Loop detections in a bounded-error context	167
6.2.3	Approximation of the solution set \mathbb{T}	168
6.3	Proving loops in detection sets	171
6.3.1	Formalism: zero verification	171
6.3.2	Topological degree for zero verification	173
6.3.3	Loop existence test	175
6.3.4	Reliable number of loops	179
6.4	Applications	181
6.4.1	The <i>Redermor</i> mission	182
6.4.2	The <i>Daurade</i> mission	187
6.4.3	Optimality of the approach	187
6.5	Conclusion	194

6.1 Introduction

6.1.1 The difference between detection and verification

In this chapter we present a reliable method to detect and verify the existence of loops along the uncertain trajectory of a robot, based on proprioceptive measurements only, using a bounded-error approach. In a reliable context, a distinction has to be made between the *detection* and the *verification* of a loop. Considering a set of feasible trajectories, some of them may cross themselves at some point; this will lead to a *detection*. In addition, when we verify that all the feasible trajectories are looped, then we can speak about a *loop proof* since a loop occurs whatever the considered uncertainties. Figure 6.1 provides an illustration of this distinction.

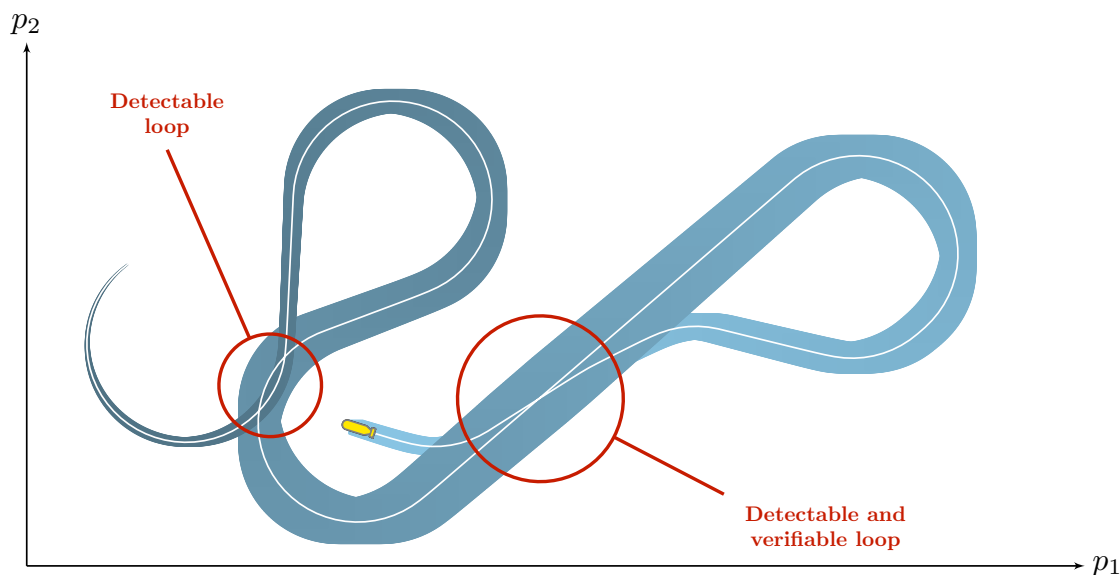


Figure 6.1: Only one loop can be *verified* in this set of trajectories, while at least two feasible loops are *detected*. Indeed, there exist trajectories that loop only once.

Section 6.2 focuses on loop detections, presenting the concepts introduced in [Aubry et al., 2013]. Our contribution is the object of Section 6.3 in which we propose a complementary tool for verification purposes. Such problem is not trivial, even in two-dimensional contexts. Our approach is to rely on the topological degree theory [Fonseca and Gangbo, 1995] to verify zeros in uncertain contexts. This study has been the object of a collaboration between Peter Franek, Clément Aubry, Luc Jaulin and the author, which led to the submission of [Rohou et al., 2018a].

6.1.2 Proprioceptive *vs.* exteroceptive measurements

A loop can be detected based on *exteroceptive* measurements, *i.e.* the perception of the outside, from scenes comparisons [Angeli et al., 2008, Cummins and Newman, 2008, Stachniss et al., 2004, Clemente et al., 2007]. However, it can be difficult to detect the closure due to poor estimations on both the robot's position and map-matchings. The problem appears even more challenging when dealing with homogeneous environments without any point of interest to rely on. This is typically the case one can encounter in underwater exploration with wide homogeneous seafloors. Such situation will unfortunately lead to a few detections of confident loop closures or, in the worst cases, to false detections that could lead to a wrong localization and mapping.

Besides exteroceptive measurements, it has been shown in [Aubry et al., 2013] that loops can be detected based on *proprioceptive* measurements only, namely: velocity vectors and inertial values knowing the kinematic of the robot. This approach has the advantage to be applicable regardless of the nature of the environment to explore. Of course, one should note that in this very case, the loop detections cannot improve by themselves the localization, as the approach will not bring new information or constraints to the problem.

However, this method is of high interest if combined with classical SLAM techniques that merge both proprioceptive and exteroceptive measurements, in order to decrease the computing burden of usual scenes recognitions. Indeed, the complexity of SLAM algorithms quickly increases with the exploration of wide environments, as it implies lots of loop closures to be identified among a dense set of data.

6.1.3 The two-dimensional case

Formally, a robot that performed a loop is a robot that came back to a previous position $\mathbf{p}(t)$. We will focus on the detection of loops along two-dimensional trajectories: $\mathbf{p}(t) \in \mathbb{R}^2$.

This choice is not a limitation made to keep things simple, but a practical requirement. Indeed, it is not possible to physically verify $\mathbf{p}(t_1) = \mathbf{p}(t_2)$ in higher dimensional spaces. A robot will never reach again the very same 3D atomic position, in contrast with two-dimensional cases. Furthermore, the amount of

uncertainties we have to deal with will always be too large to verify this. Therefore, it is not possible to prove three-dimensional loops, nor to verify that a robot came back to a previous *pose*, including both position and orientation, for the same reason.

Verify a two-dimensional loop is still interesting for many 3D applications. For instance, as pictured in Figure 1.14 (page 26), an underwater robot can apply a raw-data SLAM method using a sonar for exteroceptive measurements. In this configuration, the SLAM can be reduced to a 2D problem by merging vertical measurements, namely: depth from a pressure sensor and altitude from the sonar. Map-matching will then be achievable over each 2D crossing, as pictured in the figure with projections on the seafloor.

6.2 Proprioceptive loop detections

This section details how loops can be detected based on proprioceptive measurements only.

6.2.1 Formalization

In [Aubry et al., 2013], a loop is defined by a t -pair (t_1, t_2) such that $\mathbf{p}(t_1) = \mathbf{p}(t_2)$, $t_1 \neq t_2$, where $\mathbf{p}(t)$ is the two-dimensional position of the robot at t . The loop detection consists in computing the set \mathbb{T}^* of all loops:

$$\mathbb{T}^* = \left\{ (t_1, t_2) \in [t_0, t_f]^2 \mid \mathbf{p}(t_1) = \mathbf{p}(t_2), t_1 < t_2 \right\}, \quad (6.1)$$

Graphically, we represent the *loop set* \mathbb{T}^* in a so-called t -plane. An example of $\mathbb{T}^* = \{(t_a, t_b), (t_c, t_f), (t_d, t_e)\}$ is provided in Figure 6.2.

We consider a mobile robot moving on a horizontal plane. Its trajectory is made of several 2D positions defined by

$$\mathbf{p}(t) = \int_{t_0}^t \mathbf{v}(\tau) d\tau + \mathbf{p}_0, \quad (6.2)$$

where $\mathbf{v}(t) \in \mathbb{R}^2$ is the velocity vector expressed in the environment reference frame.

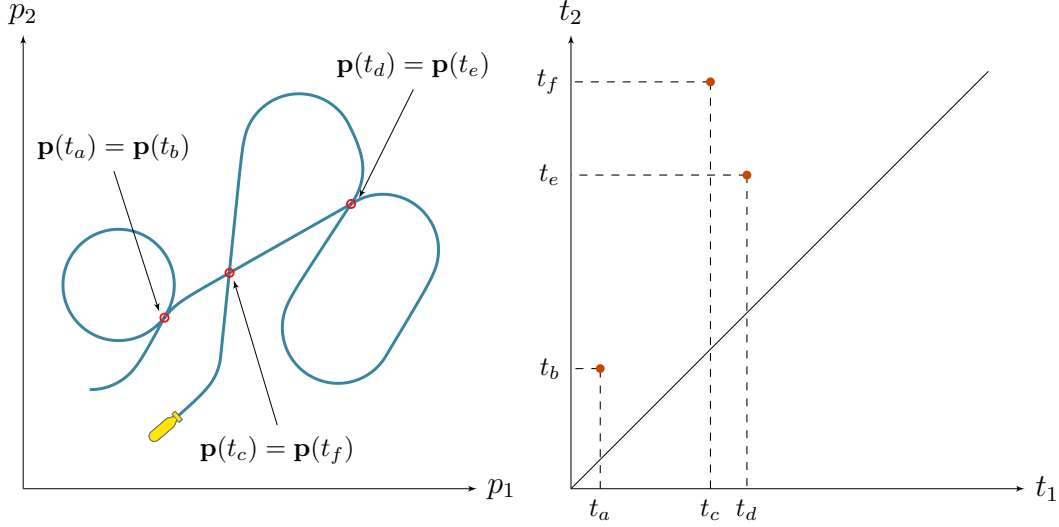


Figure 6.2: A robot performing three loops: its own trajectory has been crossed three times. A temporal representation provided by the t -plane (right-hand side) is used to picture the loops by t -pairs (t_a, t_b) , (t_c, t_f) , (t_d, t_e) . The diagonal line corresponds to the *no-delay* line for which $t_1 = t_2$.

Then, the loop set \mathbb{T}^* is

$$\mathbb{T}^* = \left\{ (t_1, t_2) \in [t_0, t_f]^2 \mid \int_{t_1}^{t_2} \mathbf{v}(\tau) d\tau = \mathbf{0}, t_1 < t_2 \right\}, \quad (6.3)$$

which means that for any $(t_1, t_2) \in \mathbb{T}^*$, robot's move from t_1 vanishes at t_2 . Therefore, any loop can be detected based on these velocity measurements.

6.2.2 Loop detections in a bounded-error context

In practice, trajectories are estimated by noisy measurements which leads to spatial uncertainties. Hence, from Equation (6.3), the set of t -pairs cannot be computed exactly. In what follows, we assume that the actual values of the velocity $\mathbf{v}^*(\cdot)$ are unknown, but guaranteed to lie in the known tube $[\mathbf{v}](\cdot)$. The loop detection problem then amounts to computing the set \mathbb{T} containing all feasible loops according to the given uncertainties:

$$\mathbb{T} = \left\{ (t_1, t_2) \mid \exists \mathbf{v}(\cdot) \in [\mathbf{v}](\cdot), \int_{t_1}^{t_2} \mathbf{v}(\tau) d\tau = \mathbf{0} \right\}, \quad (6.4)$$

or equivalently:

$$\mathbb{T} = \{(t_1, t_2) \mid \mathbf{0} \in [\mathbf{f}](t_1, t_2)\}, \quad (6.5)$$

with $[\mathbf{f}] : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ an inter-temporal inclusion function defined by

$$[\mathbf{f}](t_1, t_2) = \int_{t_1}^{t_2} [\mathbf{v}](\tau) d\tau. \quad (6.6)$$

The evaluation of $[\mathbf{f}]$ relies on Equation (3.9) given in page 79.

Hence, \mathbb{T} is a reliable enclosure of \mathbb{T}^* so that for each t -pair in \mathbb{T} , there exist values in the set of measurements that lead to the detection of a feasible loop. Therefore, the following relation is guaranteed:

$$\mathbb{T}^* \subseteq \mathbb{T} \subseteq [t_0, t_f]^2. \quad (6.7)$$

6.2.3 Approximation of the solution set \mathbb{T}

Estimating \mathbb{T} is a typical problem of set inversion. Section 2.4 (page 60) presented SIVIA: a set-membership algorithm able to approximate a solution set with sub-pavings. In our context, \mathbb{T} will be approximated by `proprioLoopSIVIA` provided in Algorithm 6 in a recursive form. Two inclusion tests are designed to decide whether a t -box $[\mathbf{t}]$ – enclosure of a t -pair – belongs completely or not to \mathbb{T} . In case of undecidability, the box is either bisected or kept in the outer approximation set.

Test: $[\mathbf{t}]$ outside the solution set

A t -box $[\mathbf{t}]$ is not a subset of \mathbb{T} if $\forall t \in [\mathbf{t}], \forall \mathbf{v}(\cdot) \in [\mathbf{v}](\cdot), \int_{t_1}^{t_2} \mathbf{v}(\tau) d\tau \neq \mathbf{0}$, and thus:

$$\left\{ \mathbf{0} \notin \int_{t_1}^{t_2} [\mathbf{v}](\tau) d\tau \right\} \implies [\mathbf{t}] \cap \mathbb{T} = \emptyset. \quad (6.8)$$

Furthermore, the $t_1 < t_2$ condition of Equation (6.1) is not verified when $[t_1] - [t_2] \subset \mathbb{R}^+$, which is another criterion for rejecting $[\mathbf{t}]$.

Finally, in the case of $[t_1] \cap [t_2] \neq \emptyset$, we will not be able to reject $[\mathbf{t}]$ since $\exists t_a \in [t_1], \exists t_a \in [t_2]$ in such a way that $\int_{t_a}^{t_a} [\mathbf{v}](\tau) d\tau = \mathbf{0}$. Now, if we prove that the function $\mathbf{p}(t)$ (Equation (6.2)) is injective inside $[t_1^-, t_2^+]$, then $\neg \exists (t_1, t_2) \in ([t_1], [t_2]) \mid \mathbf{p}(t_1) = \mathbf{p}(t_2)$. This can be verified by the following injectivity test:

$$\left\{ \mathbf{0} \notin [\mathbf{v}]\left([t_1^-, t_2^+]\right) \right\} \implies [\mathbf{t}] \cap \mathbb{T} = \emptyset. \quad (6.9)$$

Test: $[t]$ subset of the solution set

$[t]$ is a subset of \mathbb{T} if $\forall \mathbf{t} \in [t], \exists \mathbf{v}(\cdot) \in [\mathbf{v}](\cdot) \mid \int_{t_1}^{t_2} \mathbf{v}(\tau) d\tau = \mathbf{0}$, which can be reformulated from the intermediate value theorem as

$$\left\{ \int_{[t_1]}^{[t_2]} \mathbf{v}^-(\tau) d\tau \leq \mathbf{0} \leq \int_{[t_1]}^{[t_2]} \mathbf{v}^+(\tau) d\tau \right\} \implies [t] \subset \mathbb{T}. \quad (6.10)$$

where $\mathbf{v}^-(\cdot)$ and $\mathbf{v}^+(\cdot)$ are the bounds of the velocity tube (see for instance Figure 3.2, page 77) and

$$\int_{[t_1]}^{[t_2]} \mathbf{v}^-(\tau) d\tau = \left\{ \int_{t_1}^{t_2} \mathbf{v}^-(\tau) d\tau \mid t_1 \in [t_1], t_2 \in [t_2] \right\}. \quad (6.11)$$

In addition, $[t]$ will belong to \mathbb{T} only if the $t_1 < t_2$ condition is also met: $[t_1] - [t_2] \subset \mathbb{R}^-$.

The proof of these tests has been given in [Aubry et al., 2013, Sec. 4.1]. The following Algorithm 6 applies them in a SIVIA. Note that depending on the number of loops and the amount of uncertainties, the approximation of \mathbb{T} may consist of several connected components¹ denoted \mathbb{T}_i , see Figures 6.3 and 6.4.

This example has been computed in less than one second on a conventional computer. The evaluation of Equation (6.6) is optimized thanks to the new data structure of tubes proposed in this thesis and presented in Section 3.3.1, page 86. Local integrals are precomputed for each node of the tree, preventing from evaluations over each slice of the tube.

Furthermore, an optimized paving structure has been designed based on a binary tree in order to speed up the accesses to the $[t]_i$ boxes. This will be required to prove the existence of loops in these detection sets.

¹In topology, the term *connected set* refers to a component that cannot be made of two disjoint non-empty subsets.

Algorithm 6 proprioLoopSIVIA (in : $[\mathbf{v}](\cdot), [\mathbf{t}], \varepsilon, \text{inout} : \mathbb{T}^-, \mathbb{T}^+$)

- 1: **if** $[t_1] - [t_2] \subset \mathbb{R}^+$ **or** $\mathbf{0} \notin \int_{[t_1]}^{[t_2]} [\mathbf{v}](\tau) d\tau$ **or** $\mathbf{0} \notin [\mathbf{v}]([t_1^-, t_2^+])$ **then**
 - ▷ outside the solution set, $[\mathbf{t}] \cap \mathbb{T} = \emptyset$
 - ▷ the algorithm stops here
 - 2: **else if** $[t_1] - [t_2] \subset \mathbb{R}^-$ **and** $\int_{[t_1]}^{[t_2]} \mathbf{v}^-(\tau) d\tau \leq \mathbf{0} \leq \int_{[t_1]}^{[t_2]} \mathbf{v}^+(\tau) d\tau$ **then**
 - 3: $\mathbb{T}^+ \leftarrow \mathbb{T}^+ \cup [\mathbf{t}]$ ▷ outer approximation set
 - 4: $\mathbb{T}^- \leftarrow \mathbb{T}^- \cup [\mathbf{t}]$ ▷ inner approximation set
 - 5: **else if** $\text{width}([\mathbf{x}]) < \varepsilon$ **then**
 - 6: $\mathbb{T}^+ \leftarrow \mathbb{T}^+ \cup [\mathbf{t}]$ ▷ outer approximation set only
 - 7: **else** ▷ if we cannot conclude for the moment
 - 8: bisect($[\mathbf{t}]$) into $[\mathbf{t}]^{(1)}$ and $[\mathbf{t}]^{(2)}$
 - 9: proprioLoopSIVIA($[\mathbf{v}](\cdot), [\mathbf{t}]^{(1)}, \varepsilon, \mathbb{T}^-, \mathbb{T}^+$)
 - 10: proprioLoopSIVIA($[\mathbf{v}](\cdot), [\mathbf{t}]^{(2)}, \varepsilon, \mathbb{T}^-, \mathbb{T}^+$)
 - 11: **end if**
-

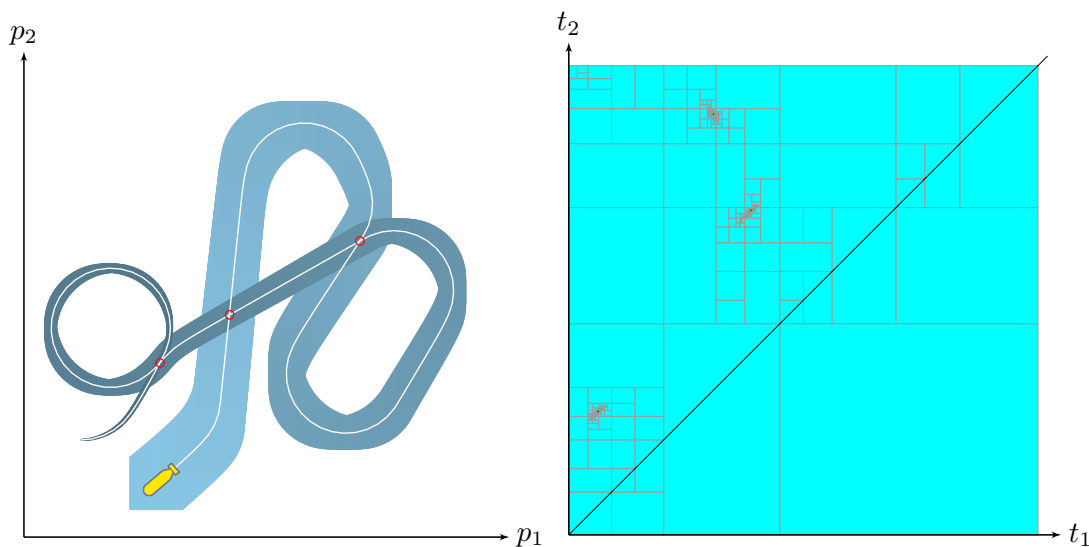


Figure 6.3: Loops detections in a bounded-error context. The approximation of \mathbb{T} with Algorithm 6 is presented on the right-hand side and let appear three connected subsets detailed in Figure 6.4.

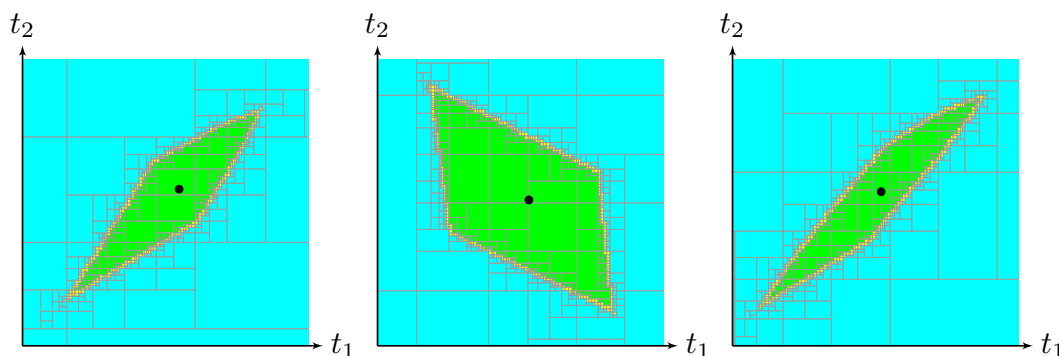


Figure 6.4: Zoom on the components of \mathbb{T} shown in Figure 6.3. Black dots represent actual loops as pictured in Figure 6.2.

6.3 Proving loops in detection sets

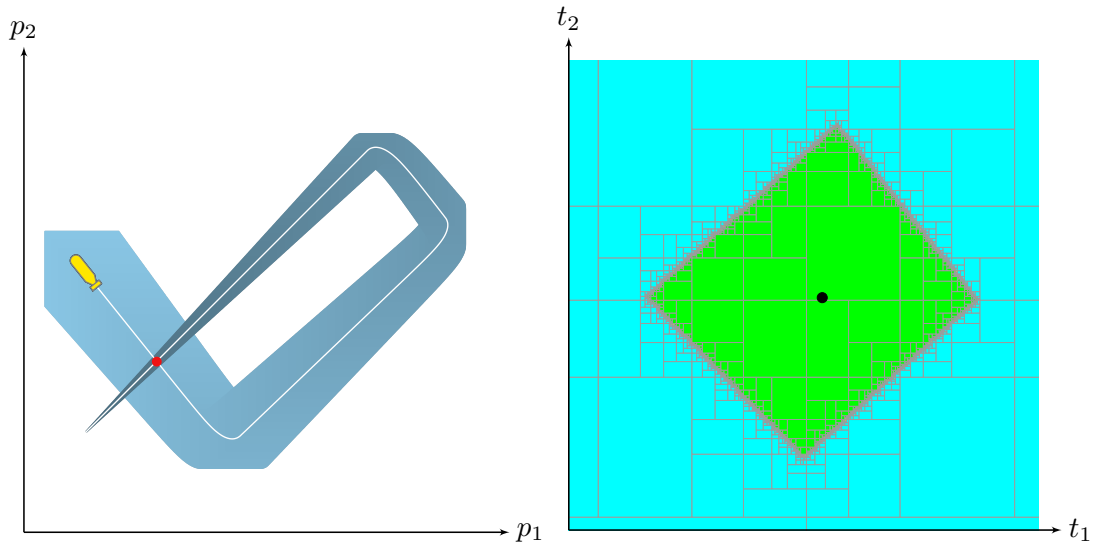
Figure 6.5 illustrates numerical approximations of \mathbb{T} on new examples of trajectories. As it can be seen, the detection of a potential loop is not a proof of its existence. For instance, Figures 6.5b–6.5c are two identical cases regarding the uncertainties: the detection \mathbb{T} pictured in the t -plane is the same while the actual trajectory may let appear one loop, two loops, or none.

6.3.1 Formalism: zero verification

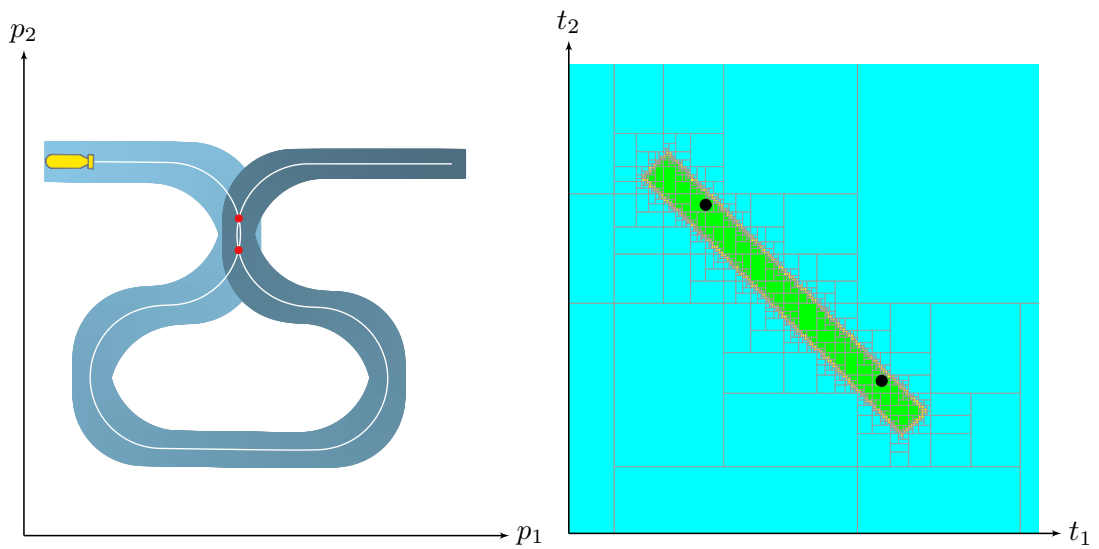
The only way to prove the existence of at least one loop in a given subset \mathbb{T}_i is to verify that $\forall \mathbf{f} \in [\mathbf{f}], \exists (t_1, t_2) \in \mathbb{T}_i$ such that $\mathbf{f}(t_1, t_2) = \mathbf{0}$, which is equivalent to verifying a zero of an unknown function² $\mathbf{f}^* \in [\mathbf{f}]$ on \mathbb{T}_i . This can be shown using the Newton test \mathcal{N} from [Moore, 1979]. Our contribution is to propose a new test \mathcal{T} based on topological degree that outperforms the previous method in most cases of ambiguous trajectories, *i.e.* non-robust zeros. This will be presented in this section.

We want to isolate and verify (prove the existence of) zeros of \mathbf{f}^* . It immediately follows from the definition that if $\mathbf{0} \notin [\mathbf{f}]([\mathbf{t}])$ for some box $[\mathbf{t}]$, then \mathbf{f}^* has no zero on $[\mathbf{t}]$. It is, however, harder to verify the *existence* of zero inside a region. If $\mathbf{0} \in [\mathbf{f}]([\mathbf{t}])$, we cannot disprove $\mathbf{f}^*(\mathbf{t}) = \mathbf{0}$ for some \mathbf{t} , but it is also not obvious how to prove the existence of such \mathbf{t} .

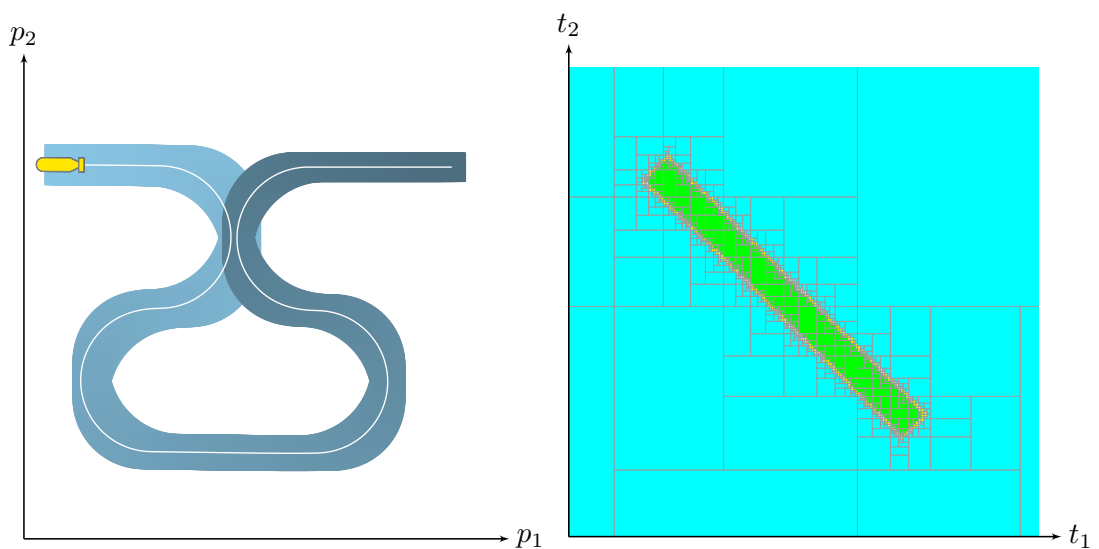
²The unknown function $\mathbf{f}^* : \mathbb{R}^2 \rightarrow \mathbb{R}^2$, defined as $\mathbf{f}^* = \int_{t_1}^{t_2} \mathbf{v}^*(\tau) d\tau$, cannot be evaluated as we do not know the actual velocity $\mathbf{v}^*(\cdot)$ of the robot.



(a) Loop detection over an undeniable looped trajectory.



(b) Loop detection over a doubtful looped trajectory. In this case the actual trajectory is made of two loops approximated within the same detection. Black dots represent actual t^* solutions.



(c) Loop detection over a doubtful looped trajectory. In this case the actual trajectory never crosses itself despite a loop detection.

Figure 6.5: Doubtful and undeniable loops in a set-membership context.

6.3.2 Topological degree for zero verification

A powerful tool for verifying zeros is the topological degree, denoted by $\deg(\mathbf{f}^*, \Omega)$. It is a unique integer assigned to \mathbf{f}^* and a compact set³ $\Omega \subset \mathbb{R}^n$ where $\mathbf{f}^*(\mathbf{t}) \neq \mathbf{0}$ for all $\mathbf{t} \in \partial\Omega$. In this definition, $\partial\Omega$ represents the boundary of the set Ω .

The topological degree satisfies certain properties, see [Fonseca and Gangbo, 1995, O'Regan et al., 2006, Furi et al., 2010] for detailed expositions. For our purposes, the most important property is that

$$\deg(\mathbf{f}^*, \Omega) \neq 0 \quad \implies \quad \exists \mathbf{t} \in \Omega \mid \mathbf{f}^*(\mathbf{t}) = \mathbf{0}. \quad (6.12)$$

Recent advances in computational topology generated many algorithms for computing the topological degree. Besides, it can be computed in case where only an inclusion function $[\mathbf{f}]$ of \mathbf{f}^* is given. It was argued in [Franek et al., 2016, Sec. 9] that the degree test is in many cases more powerful than more classical verification tools including interval Newton, Miranda's or Borsuk's tests. The reader interested in this line of research may refer to [Moore, 1977, Moore and Kioustelidis, 1980, Borsuk, 1933] for definitions and explanations.

Our application for detecting loops deals with the two-dimensional case: $\Omega \subset \mathbb{R}^2$ for the reason that loops are defined by couples of times. Then the degree has a particularly nice geometric interpretation: it is the *winding number* of the curve $\partial\Omega \xrightarrow{\mathbf{f}^*} \mathbb{R}^2 \setminus \{\mathbf{0}\}$ around $\mathbf{0}$, see Figure 6.6. If $[\mathbf{f}]$ is given, then the winding number can be computed by a number of elementary methods, the algorithm of [Franek and Ratschan, 2014] being one of them.

The following statement is a reformulation of [Franek and Ratschan, 2014, Theorem 2.9] adapted to our notation.

Theorem 6.1

Let Ω be a union of finitely many non-overlapping boxes in \mathbb{R}^n :

$$\Omega = \bigcup_{j=1}^l [\mathbf{t}]_j, \quad (6.13)$$

³In some references such as [Fonseca and Gangbo, 1995], Ω is assumed to be open and bounded, which corresponds to considering the *interior* of our Ω . The requirement $\mathbf{f}^*(\mathbf{t}) \neq \mathbf{0}, \forall \mathbf{t} \in \partial\Omega$ is unchanged.

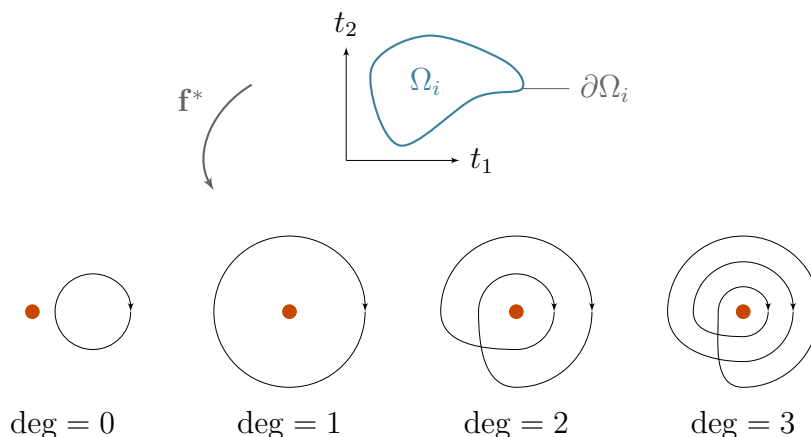


Figure 6.6: Computation of the degree of \mathbf{f}^* on Ω_i . The illustration shows several positive degree cases.

and assume that its boundary $\partial\Omega$ is a union of finitely many boxes⁴

$$\partial\Omega = \bigcup_{k=1}^p [\mathbf{b}]_k. \quad (6.14)$$

If $\mathbf{0} \notin [\mathbf{f}]([\mathbf{b}]_k)$ for all $k = 1, \dots, p$, then the degree $\deg(\mathbf{f}^*, \Omega)$ is uniquely determined and its computation can be done from the evaluations of $[\mathbf{f}]([\mathbf{b}]_k)$.

Under the assumptions of the theorem, it immediately follows that $\deg(\mathbf{g}, \Omega) = \deg(\mathbf{f}^*, \Omega)$ for any $\mathbf{g} \in [\mathbf{f}]$, because $[\mathbf{f}]$ is also an inclusion function for \mathbf{g} in such case.

Let $\Omega_1, \dots, \Omega_l$ be connected components of the union of such boxes $[\mathbf{t}]_j$ with potential zeros. On each Ω_i , if its boundary is covered by boxes $[\mathbf{b}]_k$ such that $\mathbf{0} \notin [\mathbf{f}]([\mathbf{b}]_k)$ for each k , we can compute $\deg(\mathbf{f}^*, \Omega_i)$. Whenever this degree is nonzero, we verified the existence of at least one $\mathbf{t} \in \Omega_i$ such that $\mathbf{f}^*(\mathbf{t}) = \mathbf{0}$. We emphasize that the function \mathbf{f}^* was unknown and we only worked with its inclusion function $[\mathbf{f}]$.

In the above paragraph, we never used derivatives of \mathbf{f}^* . Using additional information on derivatives, we can also count the number of solutions. Namely, if Ω is connected and $\deg(\mathbf{f}^*, \Omega) = \ell$ and we further know that the Jacobian matrix $\mathbf{J}_{\mathbf{f}^*}$ is non-singular everywhere on Ω , then \mathbf{f}^* has *exactly* $|\ell|$ zeros in Ω . This immediately

⁴We also consider degenerate boxes. In this case, $[\mathbf{b}]$'s are boxes in \mathbb{R}^n of topological dimension $n - 1$.

follows from the definition of the degree given, for example, in [Milnor, 1997, p. 27]. In particular, if the degree is ± 1 , then non-singularity immediately implies that there is a unique zero of \mathbf{f}^* in Ω .

6.3.3 Loop existence test

The topological degree theory will be used for proving the existence of loops. This section provides the proposed existence test with an explicit algorithm.

From topological degree to loops proofs

Let us consider a given domain \mathbb{T} in which we want to find zeros of \mathbf{f}^* corresponding to actual loops. The inclusion function $[\mathbf{f}]$ assumed in Section 6.3.2 is given by Equation (6.6), page 168.

With Algorithm 6, \mathbb{T} is firstly numerically approximated by two subpavings. The outer approximation \mathbb{T}^+ has the properties required for Ω . Indeed, an outer approximation set has no solution on its boundaries. Consequently, the set Ω will be \mathbb{T}^+ , that is: a finite union of boxes denoted by $[\mathbf{t}]_j$. The following relation is then guaranteed:

$$\mathbb{T}^* \subset \mathbb{T} \subset \left(\bigcup_i \Omega_i \right) \subset [t_0, t_f]^2. \quad (6.15)$$

Figure 6.7 gives an illustration of such reliable approximation with corresponding notations.

Each of these subpavings Ω_i constitutes a potential loop detection: there exists at least one trajectory with a $\mathbf{v}(\cdot) \in [\mathbf{v}](\cdot)$ that looped for one t -pair belonging to Ω_i . However, the trajectory related to the actual but unknown $\mathbf{v}^*(\cdot)$ may have never looped in reality despite the detection, as pictured by Figure 6.5. As a consequence, proving a loop amounts to verifying a zero of $\mathbf{f}^* : \mathbf{t} \mapsto \int_{t_1}^{t_2} \mathbf{v}^*(\tau) d\tau$ in Ω_i using the known inclusion function given by Equation (6.6).

By using the topological degree in this context, the consequent of the implication given in Equation (6.12) is a proof of a loop existence. The algorithm for numerical verification of $\deg(\mathbf{f}^*, \Omega_i) \neq 0$ is provided hereinafter.

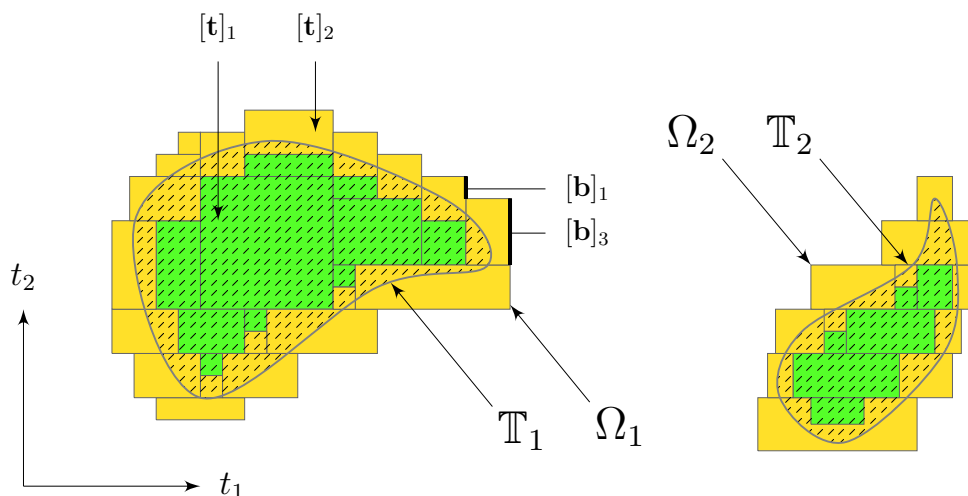


Figure 6.7: Approximation of a set $\mathbb{T} = \mathbb{T}_1 \cup \mathbb{T}_2$ with sets of non-overlapping boxes. In this chapter, only the outer approximations Ω_i will be assessed. The color code used in this figure is the one introduced at page 63.

Implementation

This section shows how to apply a simple version of the topological degree algorithm for the special case of a connected two-dimensional region Ω_i that consists of 2D boxes. The following algorithms are an adaptation of [Franek and Ratschan, 2014] for this special case.

Assume that $\Omega_i \subset \mathbb{R}^2$ is a union of finitely many boxes and the boundary $\partial\Omega_i$ is a topological circle⁵. Furthermore, let $\mathbf{a}_1, \dots, \mathbf{a}_p$ be points in $\partial\Omega_i$ and $[\mathbf{b}]_1, \dots, [\mathbf{b}]_p$ be edges covering the boundary $\partial\Omega_i$, such that $\partial[\mathbf{b}]_i = \{\mathbf{a}_{i+1}, \mathbf{a}_i\}$ for $i < p$ and $\partial[\mathbf{b}]_p = \{\mathbf{a}_1, \mathbf{a}_p\}$. We endow each $[\mathbf{b}]_i$ with an *orientation* such that \mathbf{a}_{i+1} is an end-point of $[\mathbf{b}]_i$ and \mathbf{a}_i is the starting-point of $[\mathbf{b}]_i$ for $i < p$ and, similarly, \mathbf{a}_1 is the end-point of $[\mathbf{b}]_p$ and \mathbf{a}_p the starting-point of $[\mathbf{b}]_p$. We define the *oriented boundary* of $[\mathbf{b}]_i$ to be $\mathbf{a}_{i+1} - \mathbf{a}_i$ for $i < p$ and the oriented boundary of $[\mathbf{b}]_p$ to be $\mathbf{a}_1 - \mathbf{a}_p$, where we introduce *oriented vertices* $\pm\mathbf{a}_j$ as formal symbols. This structure of oriented edges and oriented vertices can easily be represented in a computer.

Further, assume that an interval function $[\mathbf{f}]$ is given such that $\mathbf{0} \notin [\mathbf{f}]([\mathbf{b}]_i)$ for

⁵Hence, we shall assume that the set Ω_i is strictly included in $[t_0, t_f]^2$ so that a closed boundary $\partial\Omega_i$ can be assessed.

all i . This means that either the first or the second coordinate of the box $[\mathbf{f}]([\mathbf{b}]_i)$ has a constant sign, $+$ or $-$. We assign to the oriented box $[\mathbf{b}]_i$ the pair (c_i, s_i) where $c_i \in \{1, 2\}$ and $s_i \in \{+, -\}$ in such a way that the c_i -th coordinate of $[\mathbf{f}]([\mathbf{b}]_i)$ has a constant sign s_i . For example, $(2, -)$ indicates that the second coordinate of $[\mathbf{f}]([\mathbf{b}]_i)$ is negative: in particular f_2^* is negative on $[\mathbf{b}]_i$. Such choice (c_i, s_i) is not necessarily unique, but any choice will give us a correct result at the end.

The degree $\deg(\mathbf{f}^*, \Omega_i)$ can be computed using the following algorithms. The existence test \mathcal{T} is then a direct conclusion on the computed degree. One should note that, at this step, Algorithm 7 is not able to reject the feasibility of a loop. In case of a non-zero degree, it will prove a loop existence. Otherwise, the “ \emptyset ” output will reflect a non-conclusive test.

Algorithm 7 existenceTest \mathcal{T} (in : $\Omega_i, [\mathbf{f}]$ – out : true| \emptyset)

```

1:  $[\mathbf{b}]_1 \dots [\mathbf{b}]_p \leftarrow \text{getContour}(\Omega_i)$ 
2: if 2dTopoDegree( $[\mathbf{b}]_1 \dots [\mathbf{b}]_p, [\mathbf{f}]$ )  $\neq 0$  then  $\triangleright$  see Algorithm 8
3:   return true
4: else
5:   return  $\emptyset$   $\triangleright$  not able to conclude about existence
6: end if

```

An illustration of Algorithm 8 is given in Figure 6.8. Here the algorithm returns zero, because the if-conditions are satisfied only for the edge $[\mathbf{b}]_1$ where d will change from 0 to -1 , and then in edge $[\mathbf{b}]_4$ where d will be changed from -1 to 0.

If our representation of Ω_i comes from the previous SIVIA algorithm, we can assume that the `getContour` function (in Algorithm 7) is available and has linear time-complexity. A naive implementation of Algorithm 8 has quadratic complexity. Its input $[\mathbf{b}]_1, \dots, [\mathbf{b}]_p$ can be ordered and oriented in $\sim p^2$ steps so that the endpoint of $[\mathbf{b}]_j$ (resp. $[\mathbf{b}]_p$) coincides with the starting-point of $[\mathbf{b}]_{j+1}$ (resp. $[\mathbf{b}]_1$). The rest then amounts to finding the signs (c_j, s_j) in one pass over all j and adding 1 (resp. -1) to a global variable whenever $(c_j, s_j) = (1, +)$ and the next (resp. previous) sign is $(2, +)$. A better implementation in $O(p)$ is possible if we can access additional information, such as the boundary orientation of $[\mathbf{b}]_j$ induced from $\partial\Omega_i$.

Algorithm 8 2dTopoDegree (in : $[\mathbf{b}]_1 \dots [\mathbf{b}]_p, [\mathbf{f}]$ – out : d)

```

1:  $d \leftarrow 0$ 
2: for  $i = 1$  to  $p$  do
3:    $(c_i, s_i) \leftarrow \text{tagEdge}([\mathbf{b}]_i, [\mathbf{f}])$ 
4: end for
5:  $c_0 \leftarrow c_p, s_0 \leftarrow s_p, c_{p+1} \leftarrow c_1, s_{p+1} \leftarrow s_1$ 
6: for  $i = 1$  to  $p$  do
7:   if  $(c_i, s_i) = (1, +)$  then
8:     if  $(c_{i+1}, s_{i+1}) = (2, +)$  then
9:        $d \leftarrow d + 1$ 
10:    end if
11:   if  $(c_{i-1}, s_{i-1}) = (2, +)$  then
12:      $d \leftarrow d - 1$ 
13:   end if
14: end if
15: end for
16: return  $d$ 

```

Algorithm 9 tagEdge (in : $[\mathbf{b}], [\mathbf{f}]$ – out : (c, s))

```

1: if  $0 \notin [f_1]([\mathbf{b}])$  then
2:   if  $[f_1]([\mathbf{b}]) \subset \mathbb{R}^+$ , return  $(1, +)$ 
3:   else, return  $(1, -)$ 
4: else if  $0 \notin [f_2]([\mathbf{b}])$  then
5:   if  $[f_2]([\mathbf{b}]) \subset \mathbb{R}^+$ , return  $(2, +)$ 
6:   else, return  $(2, -)$ 
7: else
8:   return  $\emptyset$   $\triangleright$  note: this case should not happen
9: end if

```

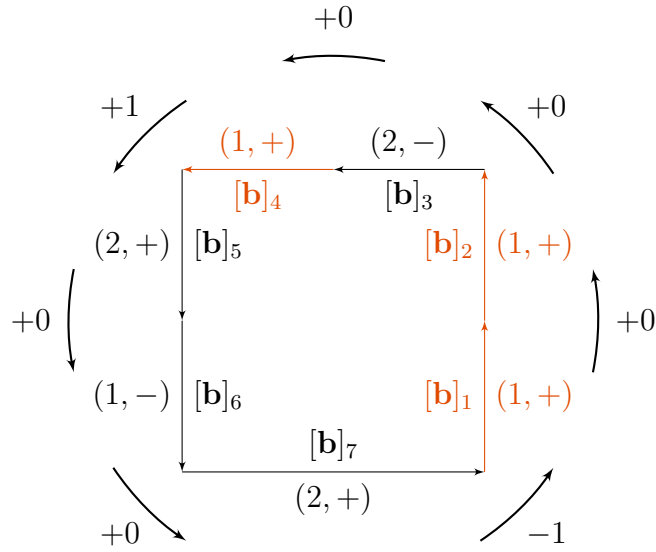


Figure 6.8: Illustration of the degree algorithm. The *selected* edges in this case are $[\mathbf{b}]_1, [\mathbf{b}]_2, [\mathbf{b}]_4$ but only $[\mathbf{b}]_1$ results in an addition by -1 and $[\mathbf{b}]_4$ in an addition of $+1$. The overall degree is $1 - 1 + 5 \times 0 = 0$ in this case.

6.3.4 Reliable number of loops

Aside from proving the existence of a loop, it may be interesting to count the number of solutions. This can be done using additional information on the derivatives. If Ω_i is a compact set as defined in Section 6.3.2 and if the Jacobian matrix $\mathbf{J}_{\mathbf{f}^*}$ is non-singular everywhere on Ω_i , then the absolute value of the degree is the exact number of solutions for $\mathbf{f}^* = \mathbf{0}$ in Ω_i .

The Jacobian matrix $\mathbf{J}_{\mathbf{f}^*}$ of the unknown \mathbf{f}^* can be approximated by $[\mathbf{J}_{\mathbf{f}}]$:

$$[\mathbf{J}_{\mathbf{f}}]([\mathbf{t}]) = \begin{pmatrix} \frac{\partial[f_1]}{\partial[t_1]} & \frac{\partial[f_1]}{\partial[t_2]} \\ \frac{\partial[f_2]}{\partial[t_1]} & \frac{\partial[f_2]}{\partial[t_2]} \end{pmatrix}. \quad (6.16)$$

From Leibniz's integral rule,

$$\frac{\partial}{\partial b} \left(\int_a^b f(x) dx \right) = f(b) \quad , \quad \frac{\partial}{\partial a} \left(\int_a^b f(x) dx \right) = -f(a). \quad (6.17)$$

Hence, we compute $[\mathbf{J}_f]([\mathbf{t}])$ as:

$$[\mathbf{J}_f]([\mathbf{t}]) = \begin{pmatrix} \frac{\partial[f_1]}{\partial[t_1]} & \frac{\partial[f_1]}{\partial[t_2]} \\ \frac{\partial[f_2]}{\partial[t_1]} & \frac{\partial[f_2]}{\partial[t_2]} \end{pmatrix} = \begin{pmatrix} -[v_1]([t_1]) & [v_1]([t_2]) \\ -[v_2]([t_1]) & [v_2]([t_2]) \end{pmatrix}, \quad (6.18)$$

where $[\mathbf{v}](\cdot)$ is the tube containing the unknown velocity $\mathbf{v}^*(\cdot)$ of the robot.

Proving the non-singularity of the Jacobian matrix amounts to verifying that its determinant is non-zero. Using the inclusion function from Equation (6.18), this is equivalent to verifying:

$$0 \notin \det([\mathbf{J}_f]) = -[v_1]([t_1]) \cdot [v_2]([t_2]) + [v_1]([t_2]) \cdot [v_2]([t_1]). \quad (6.19)$$

Algorithm 10 provided hereinafter returns the exact number of loops in a set Ω_i when the zeros are robust enough. Otherwise, nothing can be concluded regarding the uncertainties of the information.

Algorithm 10 loopsNumber (in : $\Omega_i, [\mathbf{f}], [\mathbf{J}_f]$ – out : ℓ)

```

1:  $[\mathbf{t}]_1 \dots [\mathbf{t}]_j \leftarrow \text{getBoxes}(\Omega_i)$ 
2: for  $k = 1$  to  $j$  do
3:   if  $0 \in \det([\mathbf{J}_f]([\mathbf{t}]_k))$  then
4:     return  $\emptyset$ 
5:   end if
6: end for
7:  $[\mathbf{b}]_1 \dots [\mathbf{b}]_p \leftarrow \text{getContour}(\Omega_i)$ 
8:  $\ell \leftarrow \text{2dTopoDegree}([\mathbf{b}]_1 \dots [\mathbf{b}]_p, [\mathbf{f}])$ 
9: return  $|\ell|$ 

```

Remark 6.1

The algorithm used to compute the set Ω_i may provide wide boxes $[\mathbf{t}]_k$ that will result in an over-approximation of the $[\mathbf{J}_f]([\mathbf{t}]_k)$. A bisection of the $[\mathbf{t}]_k$ may be applied when $0 \in \det([\mathbf{J}_f]([\mathbf{t}]_k))$ in order to deal with smaller boxes, thus reducing the pessimism of the Jacobian evaluation and increasing the chances to disprove $0 \in \det([\mathbf{J}_f]([\mathbf{t}]_k))$, see Figure 6.9. If the determinant approximation still contains 0 beyond a given precision ζ , then the algorithm should stop being not able to conclude. In the example of Figure 6.9, we used $\zeta = \varepsilon/10$ where ε is the precision of the SIVIA algorithm used to approximate \mathbb{T} .

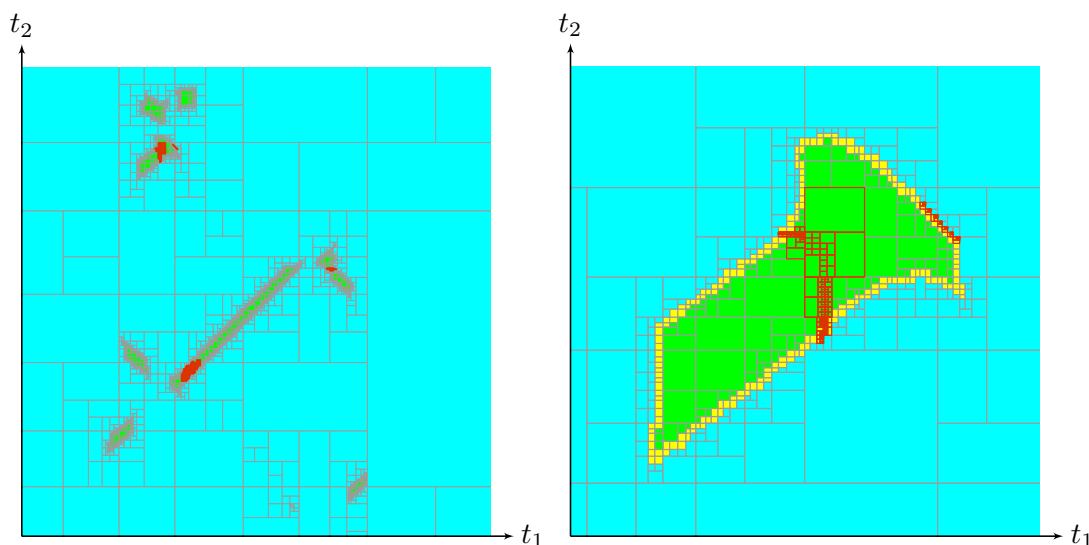


Figure 6.9: Auto-refinement of the approximation of loop sets for Jacobian evaluation purposes. Red boxes \bullet are bisections performed on the fly in order to increase the chances to disprove $0 \in \det([\mathbf{J}_{\mathbf{f}}]([\mathbf{t}]_k))$. Nevertheless, for one case detailed on the right-hand side, the algorithm stopped without being able to conclude about the non-singularity of the Jacobian matrix everywhere on this Ω_i .

6.4 Applications

The efficiency of the proposed test is demonstrated over two experiments involving actual underwater robots. The underwater case is challenging as robots do not benefit from GNSS fixes except at the very beginning of the mission. Hence, dead-reckoning methods usually apply for state estimation, leading to strong cumulative errors. Loops will be proven in this context.

Our method for loop detection is reliable under the assumption $\mathbf{f}^*([\mathbf{t}]) \subseteq [\mathbf{f}]([\mathbf{t}])$. This inclusion immediately follows from the assumption $\mathbf{v}^*(\cdot) \subseteq [\mathbf{v}](\cdot)$ but in fact, the former inclusion is much more robust with respect to random velocity errors than the latter.⁶ A quantitative analysis of error probabilities is a work in progress.

⁶The real displacement $\int_{t_a}^{t_b} \mathbf{v}^*(\tau) d\tau$ could lie outside $[\mathbf{f}](t_a, t_b)$ only if the velocity errors would *accumulate in one direction*. More precisely, the projection of $\mathbf{v}^{PL}(\cdot) - \mathbf{v}^*(\cdot)$ into one particular direction would have to be at least 2σ *in average*, over the whole time interval $[t_a, t_b]$. Under fairly general assumptions on the distribution of the velocity errors, such probability decreases exponentially with $(t_b - t_a)$.

6.4.1 The *Redermor* mission

This first application involves the *Redermor* AUV, see Figure 6.10. This test case has already been the subject of [Aubry et al., 2013, Sec. 6], in which the existence of 14 loops had been proved by using the test \mathcal{N} relying on the Newton operator. Our goal is to compare these results with the topological degree test \mathcal{T} we propose in this chapter.

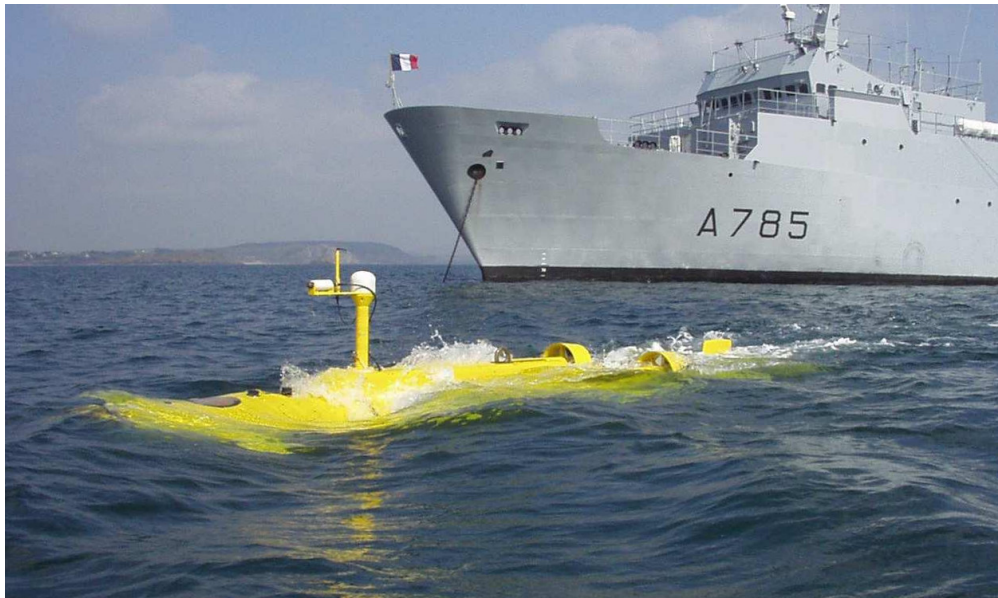
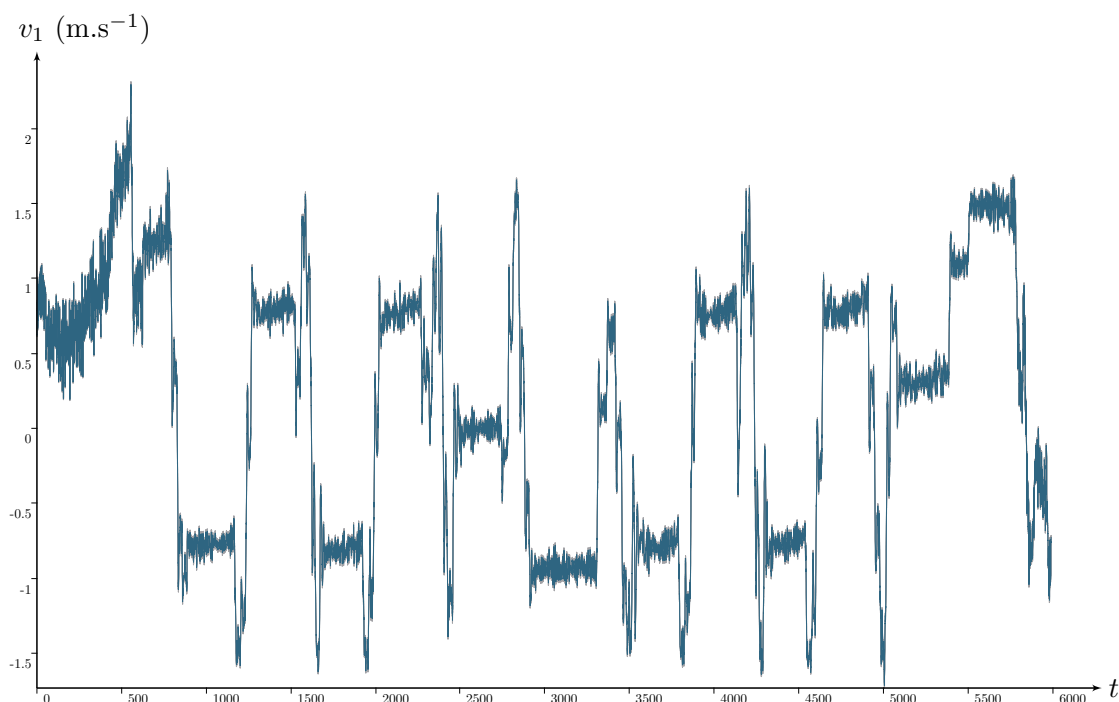


Figure 6.10: The *Redermor* AUV before a sea trial.

A two hours experimental mission has been done in the Douarnenez bay in Brittany (France). A top view of the area covered by the robot is pictured in Figure 6.12. *Redermor* performed 28 loops, 20m deep. The set-membership approach provides the enclosure of $\mathbf{v}^*(\cdot)$, see Figure 6.11, and then the approximation of \mathbb{T} pictured in the t -plane of Figure 6.13. A total of 25 complete loop-detection sets have been computed on this test-case, the other solutions being partial. By *complete detections* we mean loop detection sets Ω_i strictly included in the t -plane. Further comments on this application will only stand on these detections and the related actual loops.

In both Figures 6.12 and 6.13, the result of the degree test is displayed in orange when it proves the existence of a loop and in black when nothing can be concluded. This latter case means the robot's uncertainties are too large to demonstrate that a loop has been performed or not. In this example, there is only one situation for

Figure 6.11: Tube $[v_1](\cdot)$ enclosing the east velocity.

which nothing can be concluded. If we have a look at Figure 6.12, we can see this inconclusive case, black painted above the robot's trajectory. Figure 6.14 provides another view of it. Looking at the reliable envelope of feasible positions pictured in gray, it could have been a loop. We know it is not the case in reality: actual trajectories are not crossing. Here, the test does not reject the feasibility of a loop, it is simply not able to conclude.

We define the actual number of loops λ^* over a mission by:

$$\lambda^* = \#\{\mathbf{t} \mid \mathbf{f}^*(\mathbf{t}) = \mathbf{0}, t_1 < t_2\}, \quad (6.20)$$

where $\#$ denotes the cardinality of the following set. This application gives a comparison between the tests \mathcal{T} and \mathcal{N} . Corresponding computations provide the following results:

$$\lambda_{\mathcal{N}} = 14 \quad \lambda_{\mathcal{T}} = 24 \quad \lambda^* = 24$$

The white line in Figure 6.12 shows that the actual trajectory involves $\lambda^* = 24$ loops⁷. On this application, no other test than the topological degree would provide better results.

⁷Without considering the four loops in the components Ω_i that intersect the boundary of $[t_0, t_f]^2$.

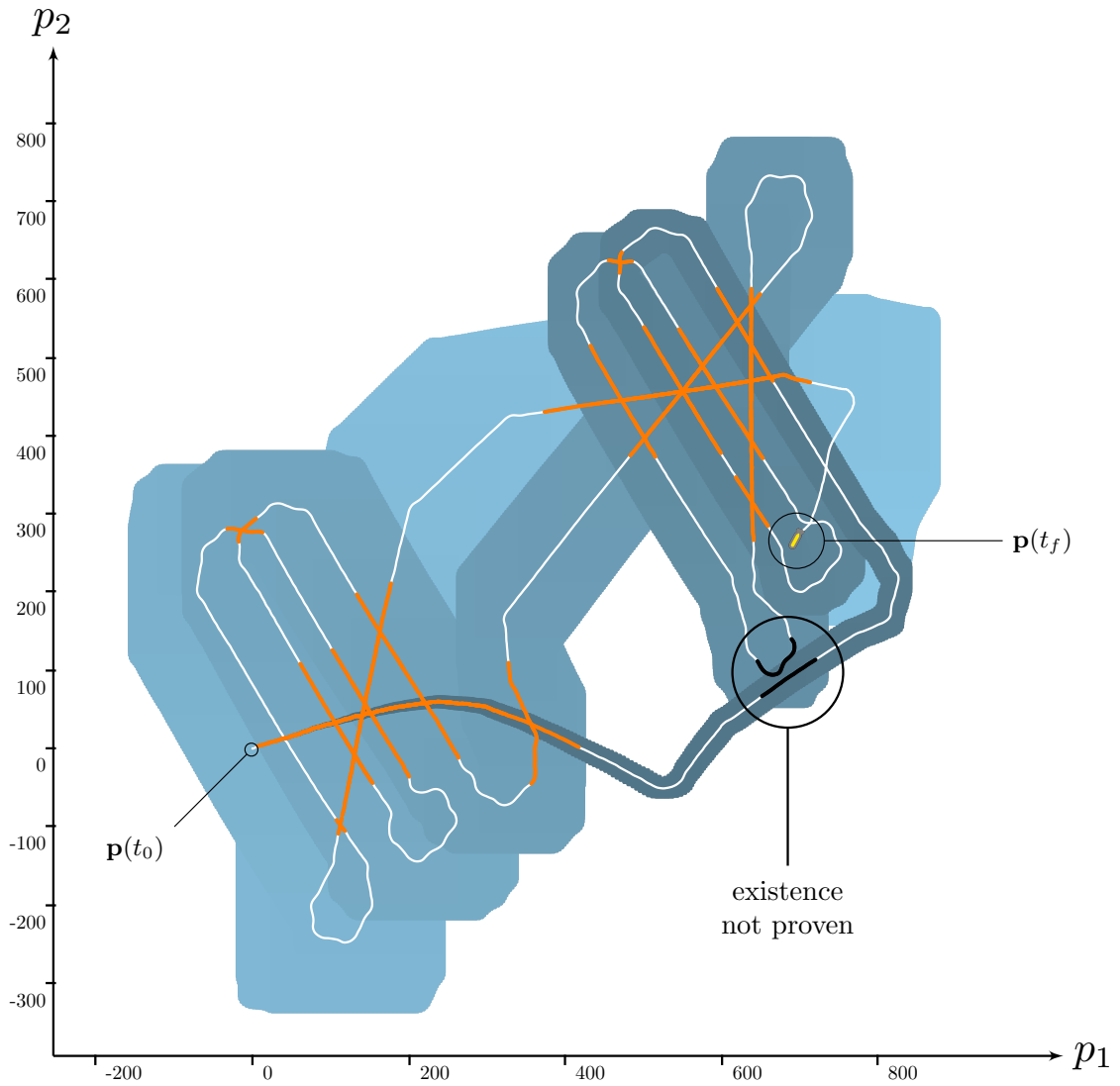


Figure 6.12: Map of the *Redermor* experiment. Orange \bullet and black \bullet lines are the projections of the results given by the topological degree test \mathcal{T} . This test case highlights one inconclusive result of \mathcal{T} .

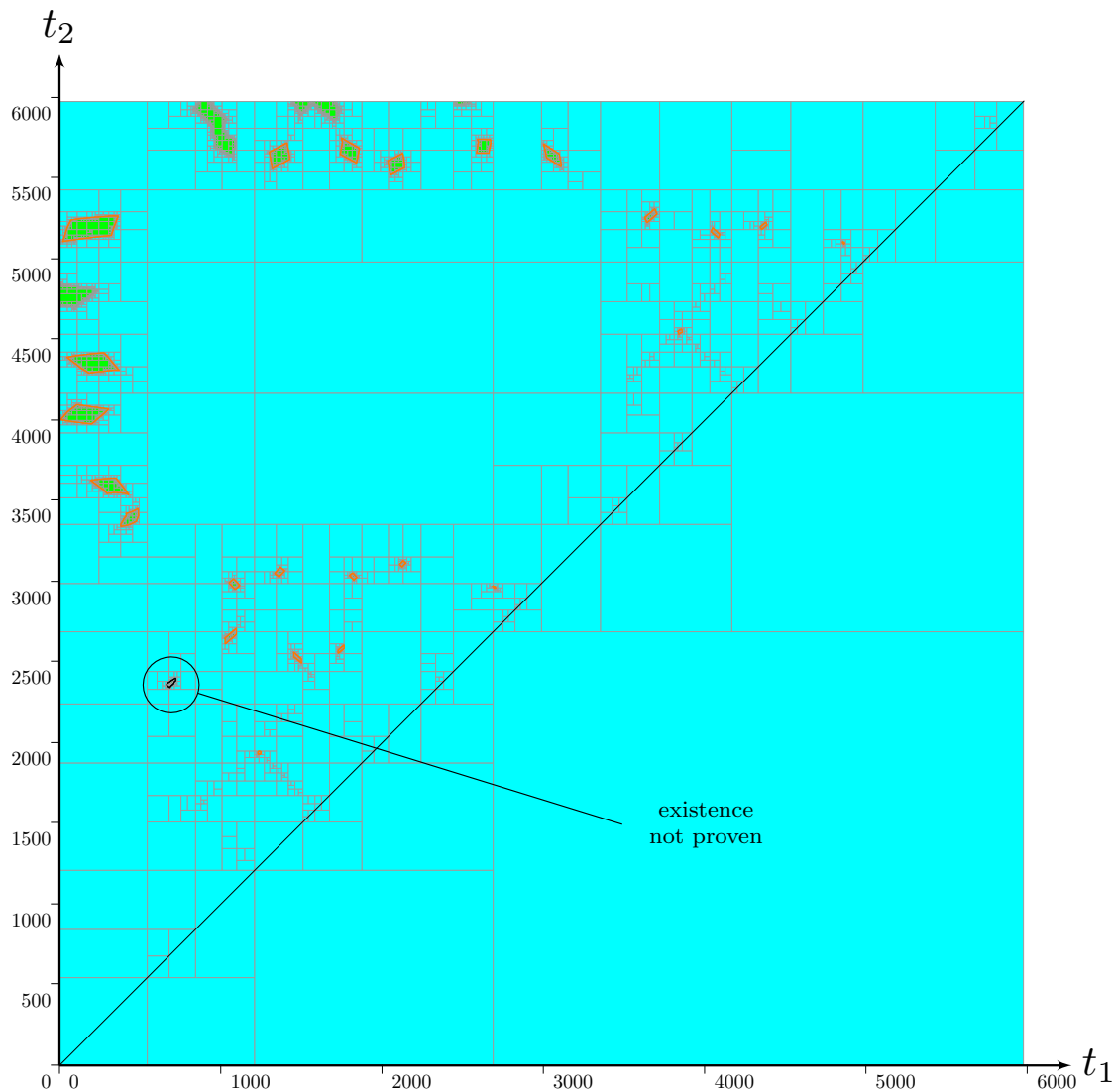


Figure 6.13: t -plane corresponding to *Redermor* experiment and computed with a SIVIA algorithm. There exist four partial detections Ω_i on t -plane's edges that will not be considered here since the $\partial\Omega_i$ are not totally defined. They enclose feasible loops (t_a, t_b) performed at the very beginning of the mission ($t_a \simeq t_0$) or at the end ($t_b \simeq t_f$). In this experiment, we used $\varepsilon = (t_f - t_0)/2000$.

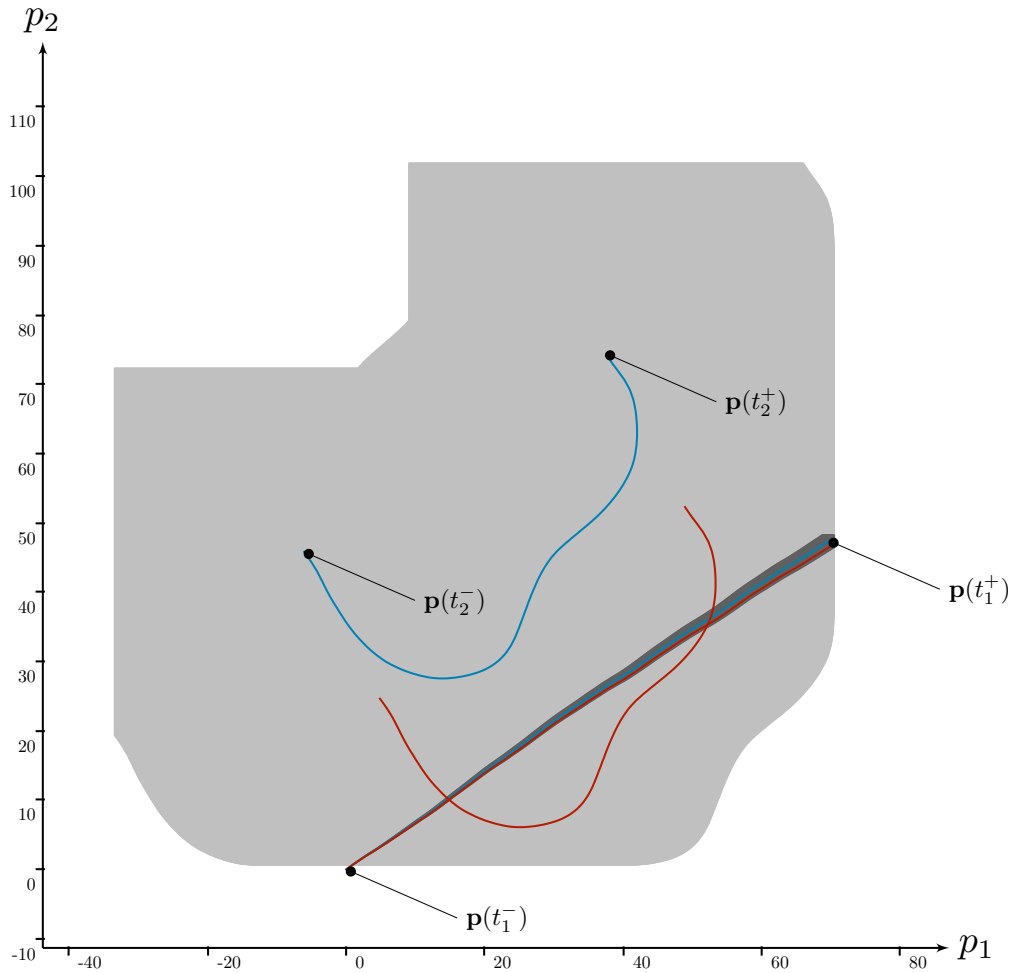


Figure 6.14: Independent projection of the non-conclusive case. Let us consider the loop-box $[t_1^-, t_1^+] \times [t_2^-, t_2^+]$ enclosing the corresponding Ω_i approximation. The actual trajectory over both $[t_1^-, t_1^+]$ and $[t_2^-, t_2^+]$ is plotted in blue \bullet . Its bounded approximation is pictured in dark gray for the first part and light gray then. Note that we do not represent the amount of uncertainties gathered before t_1^- : $\mathbf{p}(t_1^-)$ is centered in $(0, 0)$ in this independent view. However, the amount of uncertainties over $[t_2^-, t_2^+]$ is such that other crossing trajectories would have been possible given the assumed uncertainties, see *e.g.* the red one \bullet . This shows the impossibility to both disprove this loop detection and conclude about a loop existence.

6.4.2 The *Daurade* mission

We provide a complementary example involving the *Daurade* AUV. A similar mission has been performed without surfacing during 1h40. Figure 6.15 presents the corresponding trajectory together with its estimation and the test results. Figures 6.16 and 6.17 provide views of the t -plane.

For this test case, 116 subpavings Ω_i have been computed. The test \mathcal{T} proved the existence of loops in 114 of them. The uniqueness was also verified for each proof. The set of computations has been performed in less than one second on a conventional computer, which also demonstrates the relevancy of our approach for real applications.

The actual trajectory involved $\lambda^* = 118$ loops⁸ while we proved $\lambda_{\mathcal{T}} = 114$ of them. For two loop detection sets, the algorithm did not conclude due to strong uncertainties. One of these cases is highlighted in Figure 6.18.

The next section is a discussion about the optimality of our approach. The conclusion is that in this *Daurade* experiment, no more loops would have been proved by other means than the topological degree.

6.4.3 Optimality of the approach

In this section, we extend the aforementioned practical demonstration by a theoretical discussion of the degree test and its strength.

First of all, in a situation where the interval Newton test \mathcal{N} is strong enough to detect a (unique) solution of $\mathbf{f}^*(\mathbf{x}) = \mathbf{0}$ in a connected region Ω , then the Jacobian matrix $\mathbf{J}_{\mathbf{f}^*}$ is necessarily everywhere non-singular in Ω and the degree is either +1 or -1. However, the degree test does not use derivatives and can succeed even in cases where derivatives are either not at hand, or when the Jacobian matrix is potentially singular. For loop detection, this includes situations where the self-crossing is close to parallel. Figures 6.19 illustrate such situation with ambiguous crossings.

Similarly, the degree test can be shown to be more powerful than other interval-based verification tests, such as Miranda's or Borsuk's test, due to the following result [Franek et al., 2016, Thm 6]:

⁸See footnote⁷ at page 183.

Whenever a function \mathbf{f}^* has a robust zero (one that cannot be removed by arbitrary small perturbations), then it can be detected by the degree test, assuming that we have a sufficient subdivision and sufficiently precise interval-measurements.

One could still argue that such arbitrary precise interval approximations are practically not at hand. Here we state another variant of the optimality of the degree, which is adapted to the setting of our problem:

Proposition. Let Ω , $[\mathbf{f}]$, $[\mathbf{t}]_j$, $[\mathbf{b}]_k$ be as in Theorem 6.1 and assume further that the degree $\deg(\mathbf{f}^*, \Omega) = 0$ and that the interior of Ω is connected. Then there exists a function $\mathbf{g} \in [\mathbf{f}]$ such that

- $\mathbf{0} \notin \mathbf{g}(\Omega)$;
- $\mathbf{g}([\mathbf{t}]_j) \subseteq [\mathbf{f}]([\mathbf{t}]_j)$ for all j , and
- $\mathbf{g}([\mathbf{b}]_k) \subseteq [\mathbf{f}]([\mathbf{b}]_k)$ for all k .

In other words, whenever we detect a zero degree on some set Ω with connected interior, then it is still possible that \mathbf{f}^* has no zero: indeed, the unknown function \mathbf{f}^* may be the function \mathbf{g} from the theorem.

If we subdivided our domain more and obtained more data, our region Ω *could* split into more components — for example, Ω_1 with a degree 1, and Ω_2 with a degree -1 . Each Ω_i would then provably contain a zero. However, based only on the above interval evaluations, we cannot conclude the existence of a zero. In particular, for a given set of data, if we cannot conclude a zero based on the degree test then *no other test* (such as Newton) would conclude it either.

The proof of the last proposition is elementary⁹, but requires some necessary definitions from topology, so we omit it here in order to keep this chapter readable for a wide audience. Our main message is to underline the usefulness of the degree test for zero detection of functions with bounded uncertainty, and its relevancy for loop closure proofs.

⁹The main idea is to define the function \mathbf{g} to be equal to \mathbf{f}^* on $\partial\Omega$ and, in a small enough ε -neighborhood of the boundary, to extend it to a positive scalar multiple of \mathbf{f}^* such that its norm is small enough for any x that is ε -far from the boundary. This map takes $\{x : \text{dist}(x, \partial\Omega) = \varepsilon\}$ into a sphere of small diameter, and due to the fact that the degree is zero, can be extended to a function $\mathbf{g} : \Omega \rightarrow \mathbb{R}^n$ that it is still small farther from the boundary, and avoids zero.

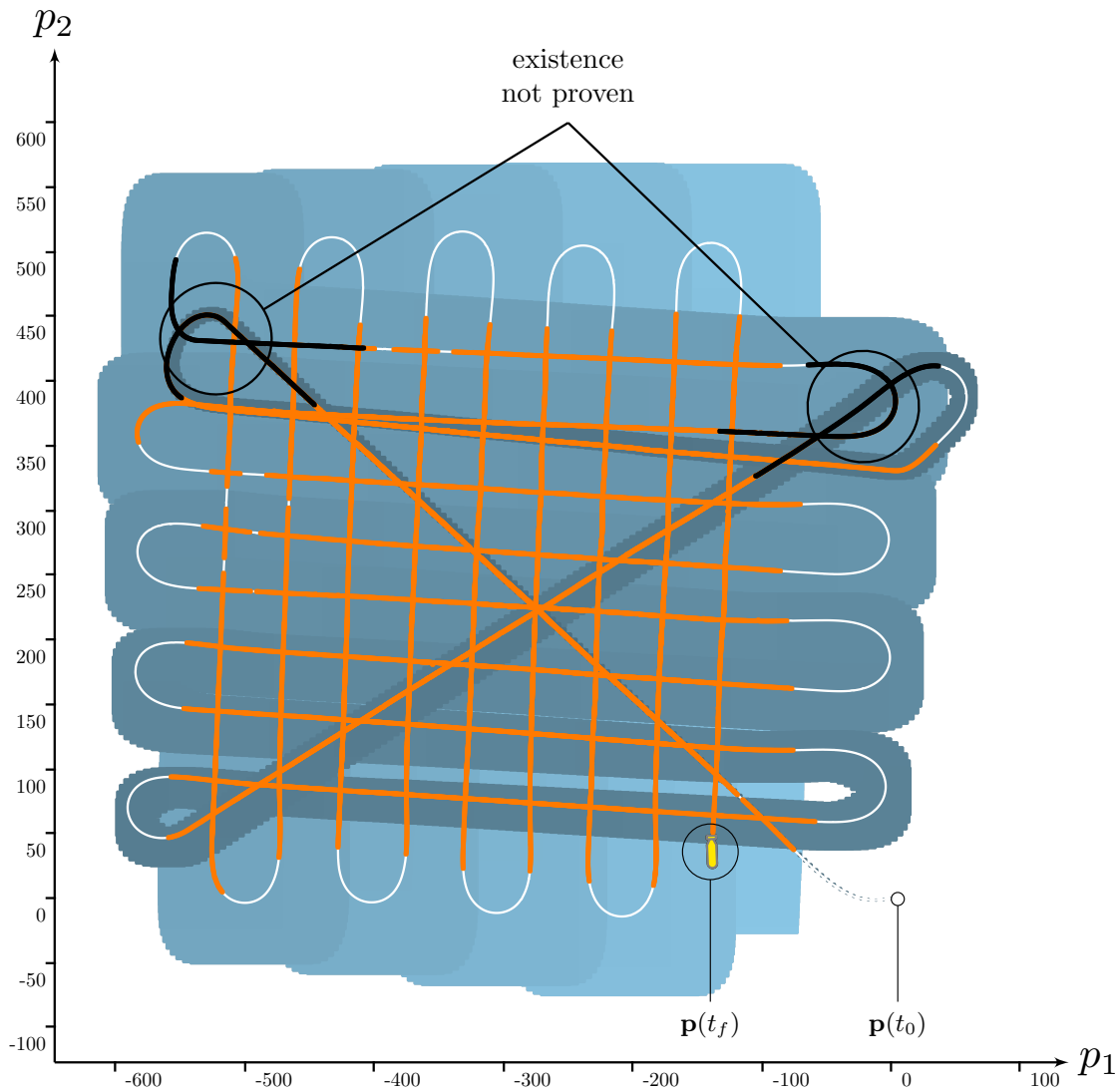


Figure 6.15: Map of the *Daurade* experiment. The topological test was not able to conclude for two loop detections involving a total of four actual loops. Figure 6.18 details one of these cases.

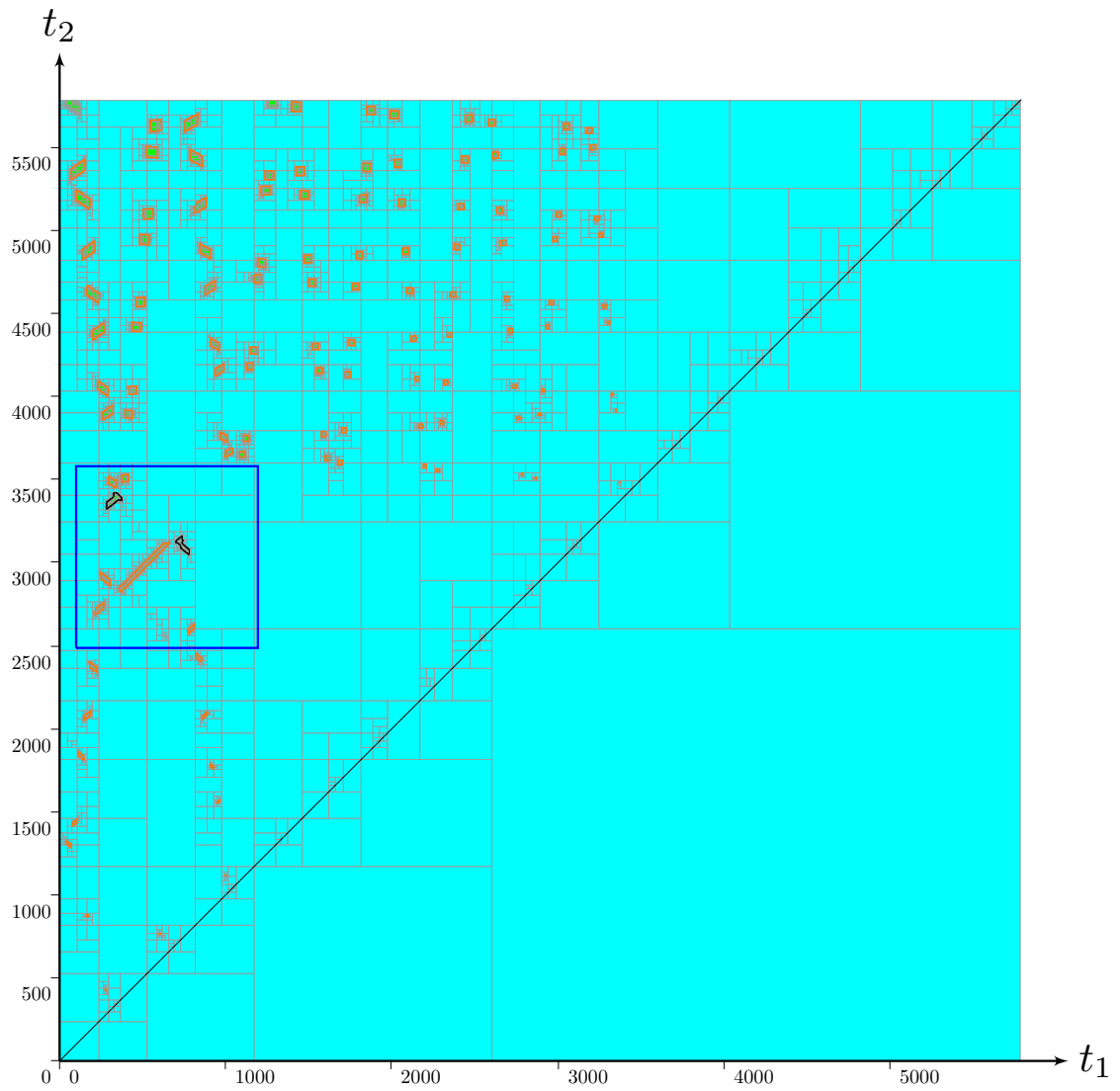


Figure 6.16: t -plane of the *Daurade* experiment. The blue box \bullet is detailed in Figure 6.17. In this experiment, we used $\varepsilon = (t_f - t_0)/2000$.

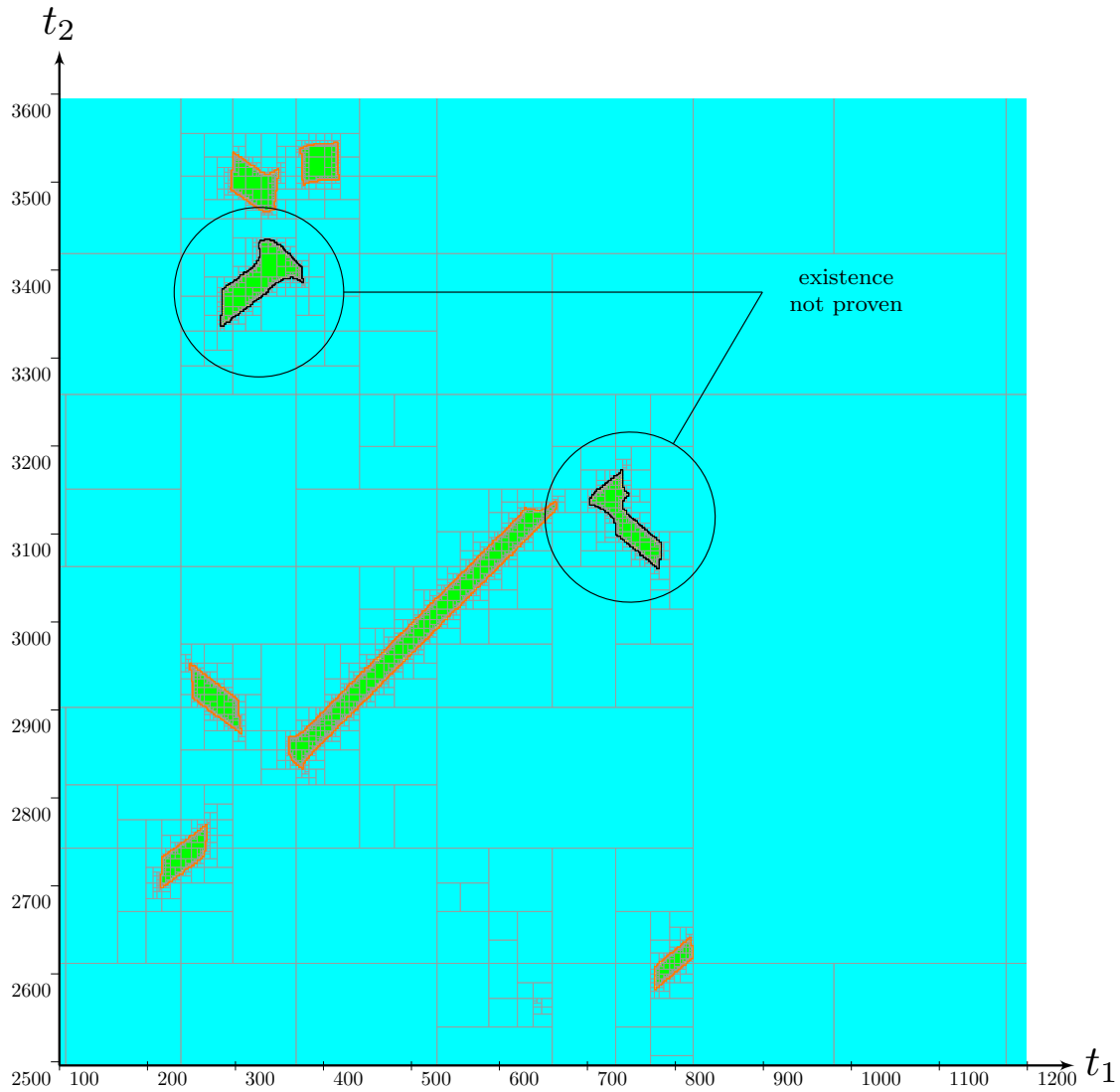


Figure 6.17: Zoom on t -plane of Figure 6.16, presenting eight clusters Ω_i corresponding to loop detection sets. Two of them, black painted, are non-conclusive cases with the topological degree test.

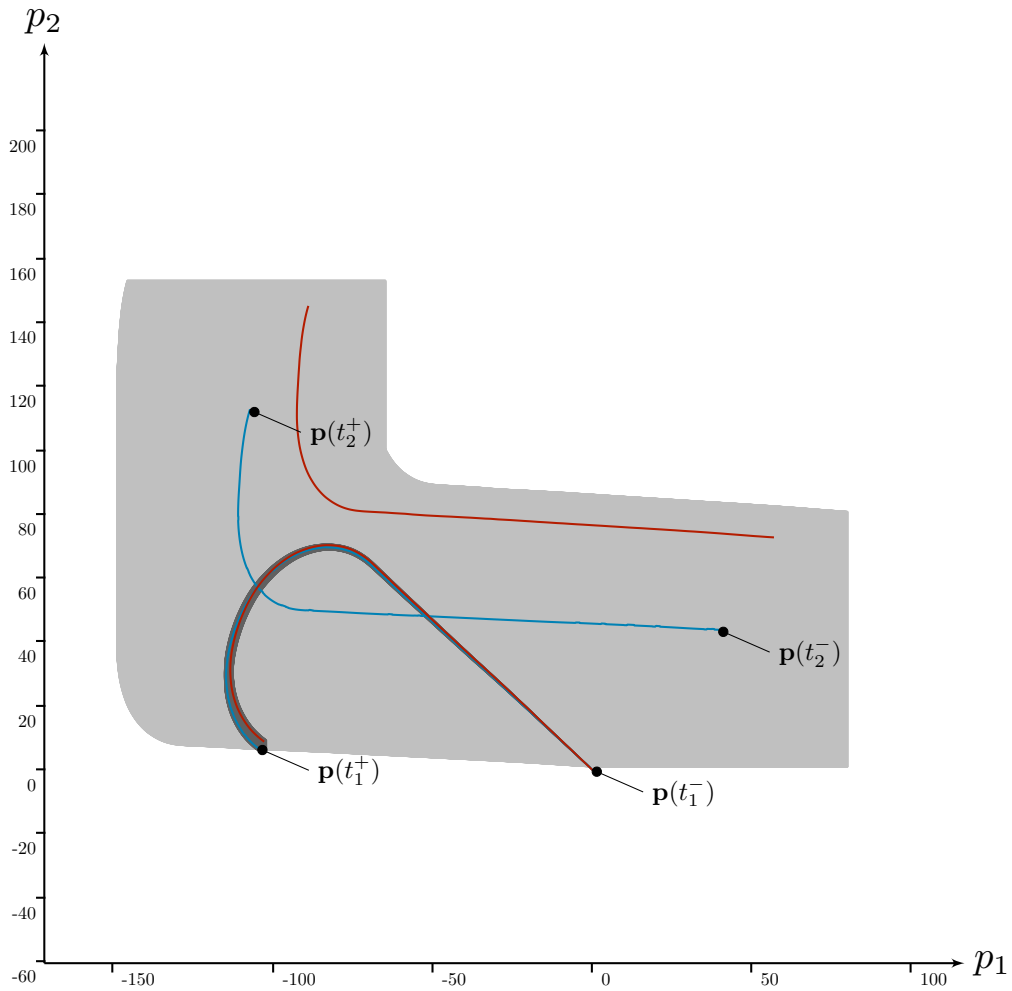
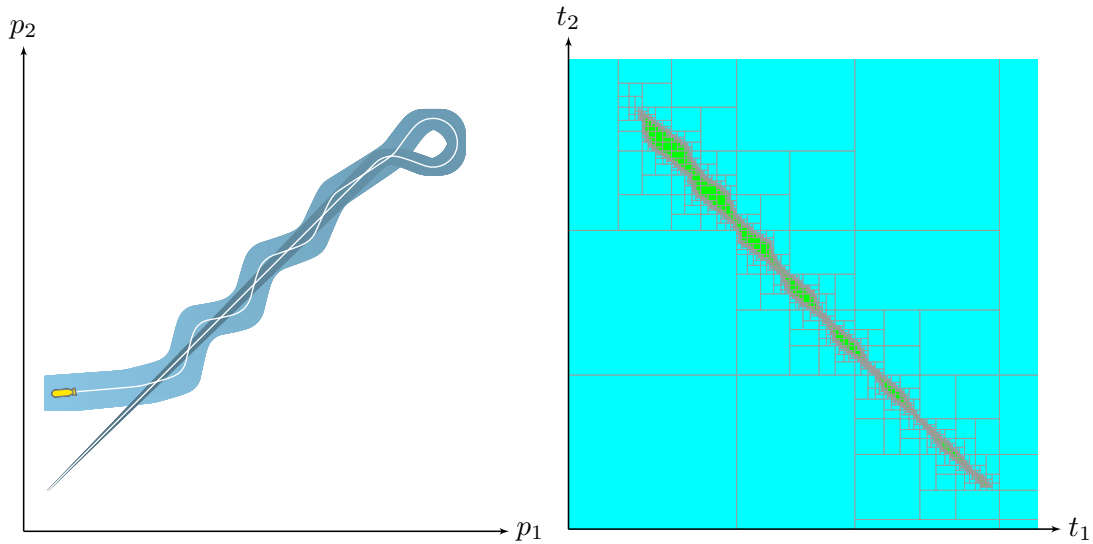
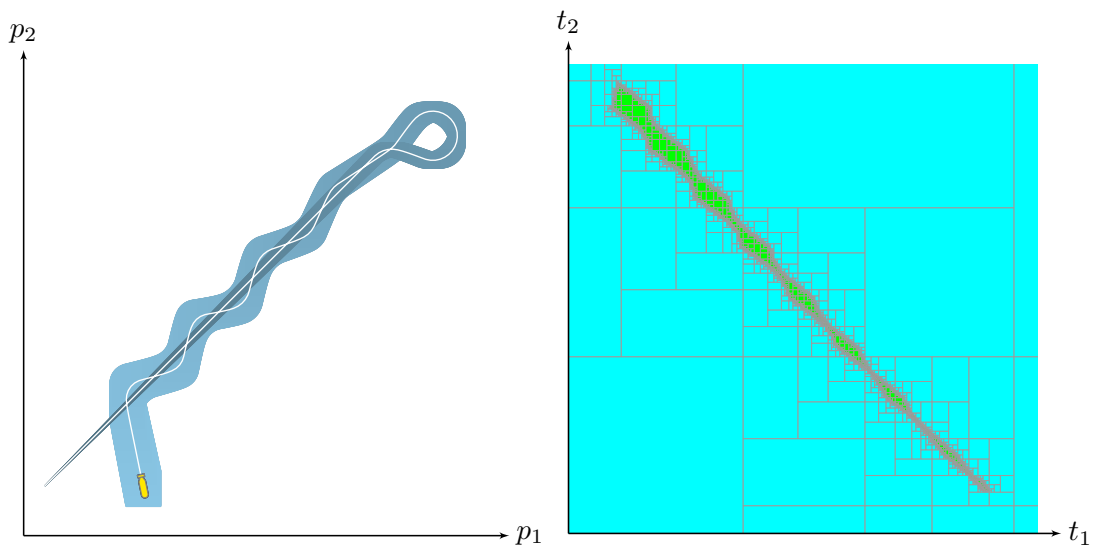


Figure 6.18: Independent projection of one of the two non-conclusive detection cases, as for the Redermor mission, see Figure 6.14. Contrary to the previous experiment, an actual loop plotted in blue \bullet has been performed, twice. However, the red \bullet trajectory reminds that a non-crossing case is still feasible.



(a) This case is non-provable since there exists one trajectory that never loops.



(b) This case is a typical situation where the Newton test would hardly succeed. However, the degree test \mathcal{T} is able to prove the existence of at least one loop in this situation, which is the expected result.

Figure 6.19: Ambiguous looped trajectories with maps and corresponding t -planes.

6.5 Conclusion

This chapter and [Rohou et al., 2018a] have presented a new method to prove the existence of loops in robot trajectories. The proposed algorithm allows one to verify that a robot crossed its own trajectory at some point. In this approach, conclusions can be taken considering proprioceptive measurements only and no scene observation. This is helpful to solve SLAM problems as it proves a previously-visited location to be recognized.

This topic has already been the subject of previous work but the offered *existence test*, relying on the Newton operator, did not give satisfactory results in some cases of undeniable looped trajectories. This was due to the use of Jacobian matrices not always invertible. Our contribution is to propose a new test relying on the topological degree theory. The algorithm behaves better as it does not use the information of the derivatives. Besides the loop existence proof, the same tool can provide the exact number of reliable loops performed by the robot, better than the Newton test did. The efficiency of the new method has been demonstrated on actual experiments involving the AUVs *Redermor* and *Daurade*.

Prospects

The discussion about the optimality of our approach suggests that the most efficient method has been studied for this problem of proprioceptive loop verification in a bounded-error context. Future work will consist in applying this new tool in classical SLAM algorithms and, in particular, those based on set-membership methods. This is the object of the next chapter.

Furthermore, complementary information could be coupled in the test such as second derivatives. This would allow the consideration of accelerations, not assessed here, or other kind of information such as the rotation of the axle of a wheeled robot.

Finally, this problem of loop detection/verification takes place in an Euclidean space while concrete problems may require to take into account non-planar surfaces. For instance, AUVs may explore wide areas up to several square kilometers, which necessarily leads to georeferencing problems due to the rounded shape of the Earth. Hence, new tools of loop detections on sphere or complex surfaces would be welcome.

A reliable temporal approach for the SLAM problem

Contents

7.1	Introduction	196
7.1.1	Motivations	196
7.1.2	SLAM formalism	198
7.1.3	Inter-temporalities	199
7.2	Temporal SLAM method	202
7.2.1	General assumptions	202
7.2.2	Temporal resolution	203
7.2.3	$\mathcal{L}_{\mathbf{p} \Rightarrow \mathbf{z}}$: inter-temporal implication constraint	204
7.2.4	The $\mathcal{C}_{\mathbf{p} \Rightarrow \mathbf{z}}$ contractor	208
7.2.5	Temporal SLAM algorithm	218
7.3	Underwater application: bathymetric SLAM	221
7.3.1	Context	221
7.3.2	<i>Daurade</i> 's underwater mission, 20 th October 2015	224
7.3.3	<i>Daurade</i> 's underwater mission, 19 th October 2015	227
7.3.4	Overview of the environment	232
7.4	Discussions	233
7.4.1	Relation to the state of the art	233
7.4.2	About a Bayesian resolution	234
7.4.3	Biased sensors	235
7.4.4	Fluctuating measurements	236
7.5	Conclusion	238

7.1 Introduction

7.1.1 Motivations

Research on SLAM topics is relatively recent [Smith et al., 1990] and gathers today a wide part of the robotic community. Several research issues have triggered the development of numerous SLAM solutions, mainly based on probabilistic approaches. The literature on this topic is wide and several books and papers have been written to capture the full picture of the various investigated methods, see for instance [Durrant-Whyte and Bailey, 2006, Bailey and Durrant-Whyte, 2006, Thrun and Leonard, 2008].

In SLAM, the noise coming from measurements is usually handled in a probabilistic way while only few attention has been paid to set-membership solutions [Yu et al., 2016, Di Marco et al., 2001, Jaulin, 2011]. The latter have the advantage to provide a reliable quality assessment of the localization and mapping, which might be expected for safety or military applications. In the underwater case, for instance, a public service in hydrography and maritime cartography will have to elaborate accurate maps for navigation purposes. One can easily understand the issues related to inaccurate maps and thus the need to comply with standards for hydrographic surveys, see for instance Table 7.1.

Table 7.1: Extract of standards for hydrographic surveys established by the International Hydrographic Organization [IHO, 2008]. Data must be qualified with a 95% confidence level. The vertical uncertainty is computed as $\pm\sqrt{a^2 + (b \times \text{depth})^2}$.

	Special order	Order 1a
Total horizontal uncertainty	2m	5m + 5% of depth
Total vertical uncertainty	$a = 0.25\text{m}$	$a = 0.5\text{m}$
	$b = 0.0075$	$b = 0.013$
Full seafloor search	required	required

AUVs involved in surveys will have to precisely estimate these uncertainties. In addition, for practical reasons mentioned in the introduction of this document, AUVs might have to survey without surfacing and thus a SLAM method could be considered.

Experiments involving *Daurade* have been undertaken by the SHOM and DGA-TN Brest for this purpose. It has been shown that the robot is able to maintain the *special order* (see Table 7.1) up to 45 minutes and then the *order 1a* during a few hours, using a dead-reckoning method. However, a SLAM approach would be welcomed to hold the special order for a longer period. Figure 7.1 depicts a *boustrophedon*, a typical pattern used for hydrographic surveys when covering a given area with parallel rails¹. In this example, the trajectory presents numerous loops that could be used in a SLAM method.

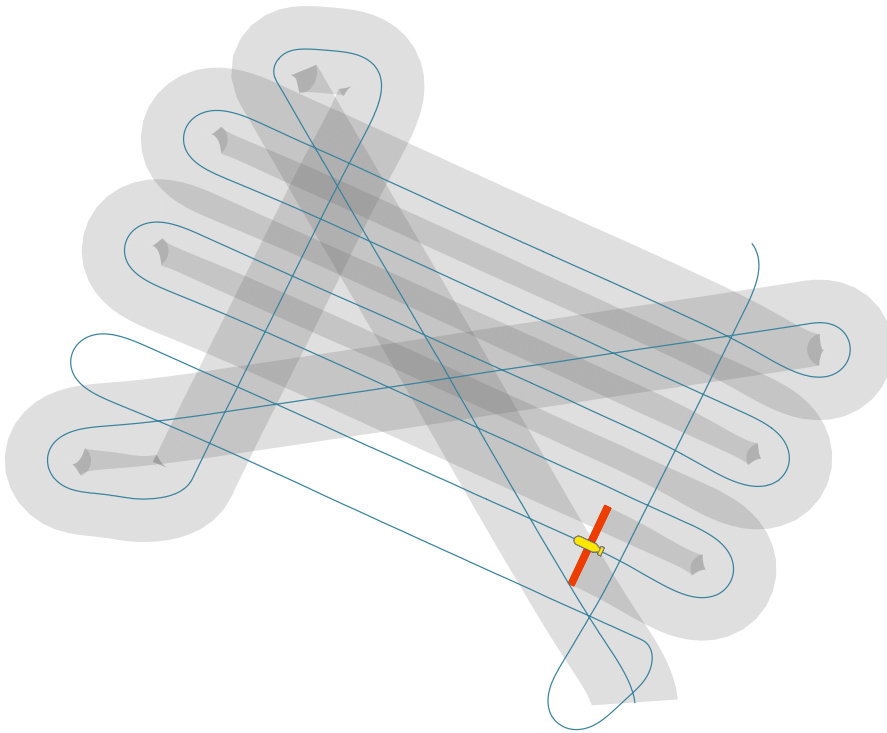


Figure 7.1: Illustration of a hydrographic survey with an AUV equipped with a multibeam echosounder. The red line • depicts the current sonar sensing while gray areas are parts of the seabed already explored. The vehicle follows a pattern called *boustrophedon* with a deliberate overlapping of rails. When planning this experiment, the operator opted for an overall crossing before performing the standard pattern. This allows easier recognitions of previously sensed areas, and thus localization refinements.

While this application highlights the need for methods providing guaranteed outputs, the same tools also present another way to deal with SLAM problems. We

¹The term *boustrophedon* is used as an analogy with bi-directional texts, mostly encountered in ancient manuscripts, for which every other line of writing is flipped, with reversed letters.

have seen in this thesis the relevancy of interval methods for constraint programming and the consideration of time uncertainties. Using a set-membership approach rather than usual probabilistic techniques, one could perfectly model a SLAM problem with a set of constraints involving robot's states and observations. Even in case of unstructured environments or poor datasets, the approach still ensures these constraints to be fulfilled at any time during the process, thus avoiding wrong localization and mappings.

7.1.2 SLAM formalism

In the literature, most SLAM problems are formalized by means of probabilistic models. To keep things independent from the resolution method, we will rather consider the following equations:

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), & (7.1a) \\ \mathbf{z}(t) = \mathbf{g}(\mathbf{m}, \mathbf{x}(t)), & (7.1b) \end{cases}$$

introducing the vector \mathbf{m} as the map of the environment. We recall that the input $\mathbf{u}(t)$ and the observation $\mathbf{z}(t)$ are measurements provided by sensors.

Once the SLAM process starts, the surroundings are poorly known and so the Equation (7.1b) provides a first estimation of the map \mathbf{m} from the quite accurate knowledge of the state $\mathbf{x}(t)$. Then the robot progressively gets lost and an approximation of $\mathbf{x}(t)$ becomes possible thanks to the accumulated knowledge on \mathbf{m} . This can be seen as a classical state estimation problem if we expand \mathbf{x} by $\mathbf{X} = (\mathbf{x}, \mathbf{m})^\top$, the map becoming a component of the state.

This formalism represents a wide part of SLAM topics. However, it does not allow one to deal with uncertain observation functions \mathbf{g} . As an example, let us consider the range-only SLAM problem [Newman and Leonard, 2003] that involves a distance observation function:

$$\begin{aligned} g : \mathbb{R}^n &\rightarrow \mathbb{R} \\ \mathbf{X} &\mapsto \sqrt{(x_1 - m_1)^2 + (x_2 - m_2)^2}. \end{aligned} \quad (7.2)$$

The position $(m_1, m_2)^\top$ of an emitting beacon can be estimated together with the state $(x_1, x_2)^\top$ of the robot. But when dealing with environments presenting unknown physical properties, the analytical expression of g is not at hand. In the

underwater case for instance, it is difficult to predict the path traveled by the sound between the beacon and the vehicle².

All is not lost and we may possibly state some mathematical properties of \mathbf{g} such as its monotonicity or particular symmetry features, still without being able to assess its formula. From this poor knowledge, we can deduce some relations among a set of observations. For instance, in the range-only example, g can be assumed as strictly increasing. This assumption will allow us to compare several measurements $z(t_1)$, $z(t_2)$ made at different times and then establish relations between the corresponding states $\mathbf{x}(t_1)$, $\mathbf{x}(t_2)$.

We propose to introduce a so-called *configuration* function, denoted $\mathbf{h} : \mathbb{R}^n \rightarrow \mathbb{R}^{n'}$, the expression of which being defined according to the assumptions on \mathbf{g} . Hence, \mathbf{h} becomes the function expressing the monotonicity, the symmetry, *etc.* In the range-only case, \mathbf{h} would depict a spherical symmetry centered on the beacon position.

In the following new formalism, \mathbf{g} disappears and an implication is introduced:

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), & (7.3a) \\ \mathbf{h}(\mathbf{x}(t_1)) = \mathbf{h}(\mathbf{x}(t_2)) \implies \mathbf{z}(t_1) = \mathbf{z}(t_2). & (7.3b) \end{cases}$$

The originality of our formalism lies in the implication (7.3b) which stands on inter-temporal measurements. In other words, when two states meet an equivalent configuration, then the related measurements should be identical. In the range-only example, if two positions are located on the same sphere centered on the beacon's location, then the range measurements should be the same.

Note that in practice, the reciprocal implication will be useful: if the measurements are not identical, then we can deduce that the states are not in the same configuration expressed by \mathbf{h} .

²Its analytical expression would depend in particular on the temperature, the depth and the salinity along the path

7.1.3 Inter-temporalities

Usual resolution techniques are not comfortable with inter-temporal relations. Instead, they rather break down the problem into the following equations:

$$\mathbf{h}(\mathbf{x}(t_1)) = \mathbf{h}(\mathbf{x}(t_2)) \iff \begin{cases} \mathbf{h}_1(\mathbf{x}(t_1), \mathbf{m}) = \mathbf{0} \\ \mathbf{h}_2(\mathbf{x}(t_2), \mathbf{m}) = \mathbf{0} \end{cases}, \quad (7.4)$$

employing the vector \mathbf{m} as the map of the environment. Hence, the temporal problem is transformed into a spatial one and the state estimation now amounts to approximating the new state vector $\mathbf{X} = (\mathbf{x}, \mathbf{m})^\top$.

Conversely, our approach consists in fully addressing Equation (7.3b) without performing a decomposition such as Equation (7.4). We will see that the resolution does not rely on the estimation of a vector \mathbf{m} and only focuses on the times of interest. However, we will still speak about SLAM since we apply the philosophy of exploring the surroundings while performing a localization refined by the observations. In our approach, the map is the time and any localization or mapping process will be related to the approximation of time references. This can be somehow related to Graph-SLAM methods [Thrun and Montemerlo, 2006] although they do not deal with time uncertainties. Indeed, when assessing both continuous measurements and noise on the observations, the uncertainties can be propagated to temporal variables. Suitable tools become necessary to handle the resolution. To our knowledge, such estimation of time values in a SLAM method is new.

To sum up, our approach applies when the observation function \mathbf{g} is completely unknown. In this context, a single measurement $\mathbf{z}(t_1)$ is not exploitable as it cannot be used to estimate $\mathbf{x}(t_1)$ and reciprocally, a prediction of the observation from the knowledge of $\mathbf{x}(t_1)$ is not achievable. The key point is the correlation that can be made between two identical measurements. The best illustration of this approach is the Borda's weighing technique.

Borda's double weighing method

This old technique is used to precisely estimate a weight whatever the precision of a balance scale. In this example, the balance represents an unknown *environment* because we admit an uncertainty on the identical length of its arms, see Figure 7.2.

Thus, when the equilibrium is reached, the mass to be estimated in the right pan may not be the same as the mass of reference m_0 in the left one. Borda's

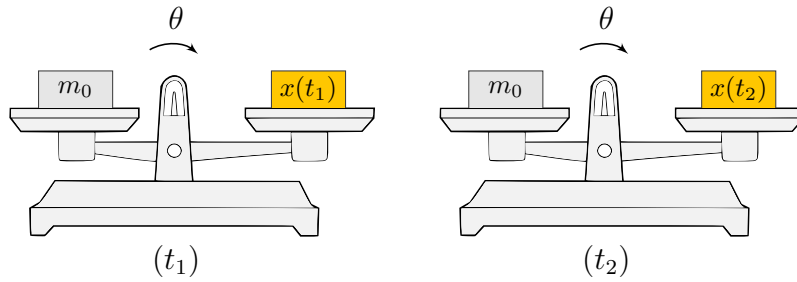


Figure 7.2: A flawed Roberval balance with arms of unknown length: an inter-temporal measurement is made at times t_1 and t_2 in order to precisely estimate $x(t_1)$ thanks to a well known mass $x(t_2)$.

method consists in removing the thing $x(t_1)$ to be weighted from the first pan and reaching again the equilibrium with a well-known weight $x(t_2)$. This last mass will give the true weight we want to estimate, even if we are using a false balance: $x(t_1) = x(t_2)$.

Using our formalism³,

$$\begin{aligned}
 h(x(t_1)) = h(x(t_2)) &\implies z(t_1) = z(t_2) \\
 h : (x(t)) &\mapsto x(t) \\
 g : (x) &\mapsto ?
 \end{aligned}
 \tag{7.5}$$

In this example, the definition of h is the identity function. The observation function g could depict the angle θ of the pointer as a function of the mass x . However, its expression is not at hand since we do not know the length of each arm. Nevertheless, a measurement z can be made looking at the pointer.

Here the environment is unknown but constant so that the observation function is time-independent. A single measurement does not give us any information about the weight we want to estimate. However, a couple of measurements made at different times cancels the errors of the balance thereby dispelling any uncertainty about the estimation. We have established an inter-temporal measurement for a state estimation. Our SLAM approach is similar.

³In this example, if we consider reasonable masses, the implication is actually an equivalence.

What to observe?

Any kind of observation is welcome, under the assumption that this information does not evolve during the mission in a non-predictable manner. Hence, many properties of the environment can be used such as radioactivity, magnetism, luminosity, temperature or, in our case, bathymetric measurements. It remains to define the function \mathbf{h} according to the considered observations.

For instance, the temperature only depends on the 3D position of the sensor while bathymetric values are related to 2D positions. In this latter case, the *configuration* function is simply defined as

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} \xrightarrow{\mathbf{h}} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \quad (7.6)$$

with $(x_1, x_2)^\top$ the horizontal position of the robot and x_3 its altitude. The other components of $\mathbf{x} \in \mathbb{R}^n$ may represent orientations or speeds, but this definition shows that only the horizontal position matters in this example.

Future research will focus on electrolocation [Boyer et al., 2015, Lebastard et al., 2013, Morel et al., 2016] based on the sensing of electric fields distortions. In this case, \mathbf{h} must be defined according to the pose of the robot, including its orientation. We believe the approach could allow a localization even without the knowledge of accurate electric models.

7.2 Temporal SLAM method

7.2.1 General assumptions

We consider the following requirements:

- for now, we do not deal with outliers: any bounded measurement is guaranteed to contain the actual value. The management of outliers will be the subject

of future work based on already existing tools [Jaulin, 2009, Drevelle and Bonnifait, 2009, Carbonnel et al., 2014];

- any change on the environment is assumed to be predictable: the surroundings may be static or may change according to some physical models with bounded uncertainties. For instance, in the case of underwater exploration, one could use tide models;
- we assume that there is sufficient spatial variations in the measurements to allow a localization. Indeed, an almost constant set of measurements will not be useful for the state estimation. In any case though, the reliability of the method will not be affected. We emphasize that the approach does not require the identification of points of interest: the environment may be poor without remarkable land/sea marks that could be used as reference.

7.2.2 Temporal resolution

We will solve this problem using a constraint propagation approach.

Constraint network

Our problem can be formulated with a CN:

$$\text{CN: } \left\{ \begin{array}{l}
 \mathbf{Variables: } \mathbf{x}(\cdot), \mathbf{z}(\cdot), \mathbf{u}(\cdot) \\
 \mathbf{Constraints:} \\
 \quad 1. \text{ Evolution constraint:} \\
 \quad \quad \text{— } \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \\
 \quad 2. \text{ Inter-temporal constraint:} \\
 \quad \quad \text{— } \mathbf{h}(\mathbf{x}(t_1)) = \mathbf{h}(\mathbf{x}(t_2)) \implies \mathbf{z}(t_1) = \mathbf{z}(t_2) \\
 \mathbf{Domains: } [\mathbf{x}](\cdot), [\mathbf{z}](\cdot), [\mathbf{u}](\cdot)
 \end{array} \right. \quad (7.7)$$

We recall that $\mathbf{x}(\cdot)$ is the set of states to be approximated and that $\mathbf{z}(\cdot)$ and $\mathbf{u}(\cdot)$ are observation and input measurements. The *configuration* function \mathbf{h} is defined according to the investigated problem. A complete example will be given in Section 7.3.

Decomposition of the inter-temporal constraint

It is important to break down the problem as much as possible. Here, the inter-temporal relation can be decomposed into:

$$\begin{cases} \mathbf{p}(t_1) = \mathbf{h}(\mathbf{x}(t_1)), & (7.8a) \\ \mathbf{p}(t_2) = \mathbf{h}(\mathbf{x}(t_2)), & (7.8b) \\ \mathbf{p}(t_1) = \mathbf{p}(t_2) \implies \mathbf{z}(t_1) = \mathbf{z}(t_2). & (7.8c) \end{cases}$$

These equations cannot be further simplified and a new elementary constraint, denoted $\mathcal{L}_{\mathbf{p} \Rightarrow \mathbf{z}}$, has to be addressed:

$$\mathcal{L}_{\mathbf{p} \Rightarrow \mathbf{z}}(\mathbf{p}(\cdot), \mathbf{w}(\cdot), \mathbf{z}(\cdot)) : \begin{cases} \mathbf{p}(t_1) = \mathbf{p}(t_2) \implies \mathbf{z}(t_1) = \mathbf{z}(t_2) \\ \dot{\mathbf{p}}(\cdot) = \mathbf{w}(\cdot) \end{cases} \quad (7.9)$$

The values t_1 and t_2 are internal variables that cover the whole domain $[t_0, t_f]$ of the trajectories. The presence of the derivative of $\mathbf{p}(\cdot)$, denoted $\mathbf{w}(\cdot)$, is explained by the time uncertainties this constraint will raise. Its values are given analytically or from measurements.

This leads us to the new CN:

$$\text{CN: } \left\{ \begin{array}{l} \mathbf{Variables: } \mathbf{x}(\cdot), \mathbf{v}(\cdot), \mathbf{p}(\cdot), \mathbf{w}(\cdot), \mathbf{z}(\cdot), \mathbf{u}(\cdot) \\ \mathbf{Constraints:} \\ \quad 1. \text{ Evolution constraint:} \\ \quad \quad - \mathbf{v}(\cdot) = \mathbf{f}(\mathbf{x}(\cdot), \mathbf{u}(\cdot)) \\ \quad \quad - \mathcal{L}_{\frac{d}{dt}}(\mathbf{x}(\cdot), \mathbf{v}(\cdot)) \\ \quad 2. \text{ Inter-temporal constraint:} \\ \quad \quad - \mathbf{p}(\cdot) = \mathbf{h}(\mathbf{x}(\cdot)) \\ \quad \quad - \mathbf{w}(\cdot) = \frac{d\mathbf{h}}{d\mathbf{x}(\cdot)} \cdot \mathbf{v}(\cdot) \quad (\text{expression of the derivative of } \mathbf{p}(\cdot)) \\ \quad \quad - \mathcal{L}_{\mathbf{p} \Rightarrow \mathbf{z}}(\mathbf{p}(\cdot), \mathbf{w}(\cdot), \mathbf{z}(\cdot)) \\ \mathbf{Domains: } [\mathbf{x}(\cdot), [\mathbf{v}(\cdot), [\mathbf{p}(\cdot), [\mathbf{w}(\cdot), [\mathbf{z}(\cdot), [\mathbf{u}(\cdot) \end{array} \right. \quad (7.10)$$

7.2.3 $\mathcal{L}_{\mathbf{p} \Rightarrow \mathbf{z}}$: inter-temporal implication constraint

Dealing with $\mathcal{L}_{\mathbf{p} \Rightarrow \mathbf{z}}$ is the main academic contribution of this chapter. To our knowledge, this is the first time an approach is proposed to solve such inter-temporal constraint. The aim of $\mathcal{L}_{\mathbf{p} \Rightarrow \mathbf{z}}$ is to remove trajectories not consistent with Equation (7.9).

Without using a set-membership method, this constraint is laborious to solve. A naive approach would be to test each feasible $\mathbf{p}(\cdot)$ and eliminate those for which no $\mathbf{t} \in [t_0, t_f]^2$ can lead to one possible $\mathbf{z}(\cdot)$ such that $\mathbf{p}(t_1) = \mathbf{p}(t_2) \implies \mathbf{z}(t_1) = \mathbf{z}(t_2)$. Figure 7.3 pictures this approach in a robotic context with three estimations of $\mathbf{p}(\cdot)$ and one well-known $\mathbf{z}(\cdot)$. One can easily understand that in case of uncertainties on both $\mathbf{p}(\cdot)$ and $\mathbf{z}(\cdot)$, this resolution becomes expensive in terms of calculations.

By using a set-membership approach, however, we can rely on the bounds of the related sets $[\mathbf{p}](\cdot)$ and $[\mathbf{z}](\cdot)$. We introduce in this section intermediate variables that will only appear inside the decomposition of $\mathcal{L}_{\mathbf{p} \Rightarrow \mathbf{z}}$. The next section will focus on the corresponding operator $\mathcal{C}_{\mathbf{p} \Rightarrow \mathbf{z}}$ built from the tools presented in Chapters 5 and 6.

Decomposition

First, we focus on each part of the implication:

$$\underbrace{\mathbf{p}(t_1) = \mathbf{p}(t_2)}_{\textcircled{1}} \implies \underbrace{\mathbf{z}(t_1) = \mathbf{z}(t_2)}_{\textcircled{2}}. \quad (7.11)$$

The key point consists in considering t_1, t_2 as classical variables to be estimated inside the constraint. These t -pairs belong to two sets defined either by the cause $\textcircled{1}$ or the effect $\textcircled{2}$:

$$\textcircled{1} \quad \mathbb{T}_{\mathbf{p}}^* = \left\{ (t_1, t_2) \in [t_0, t_f]^2 \mid \mathbf{p}(t_1) = \mathbf{p}(t_2), t_1 < t_2 \right\}, \quad (7.12)$$

$$\textcircled{2} \quad \mathbb{T}_{\mathbf{z}}^* = \left\{ (t_1, t_2) \in [t_0, t_f]^2 \mid \mathbf{z}(t_1) = \mathbf{z}(t_2), t_1 < t_2 \right\}. \quad (7.13)$$

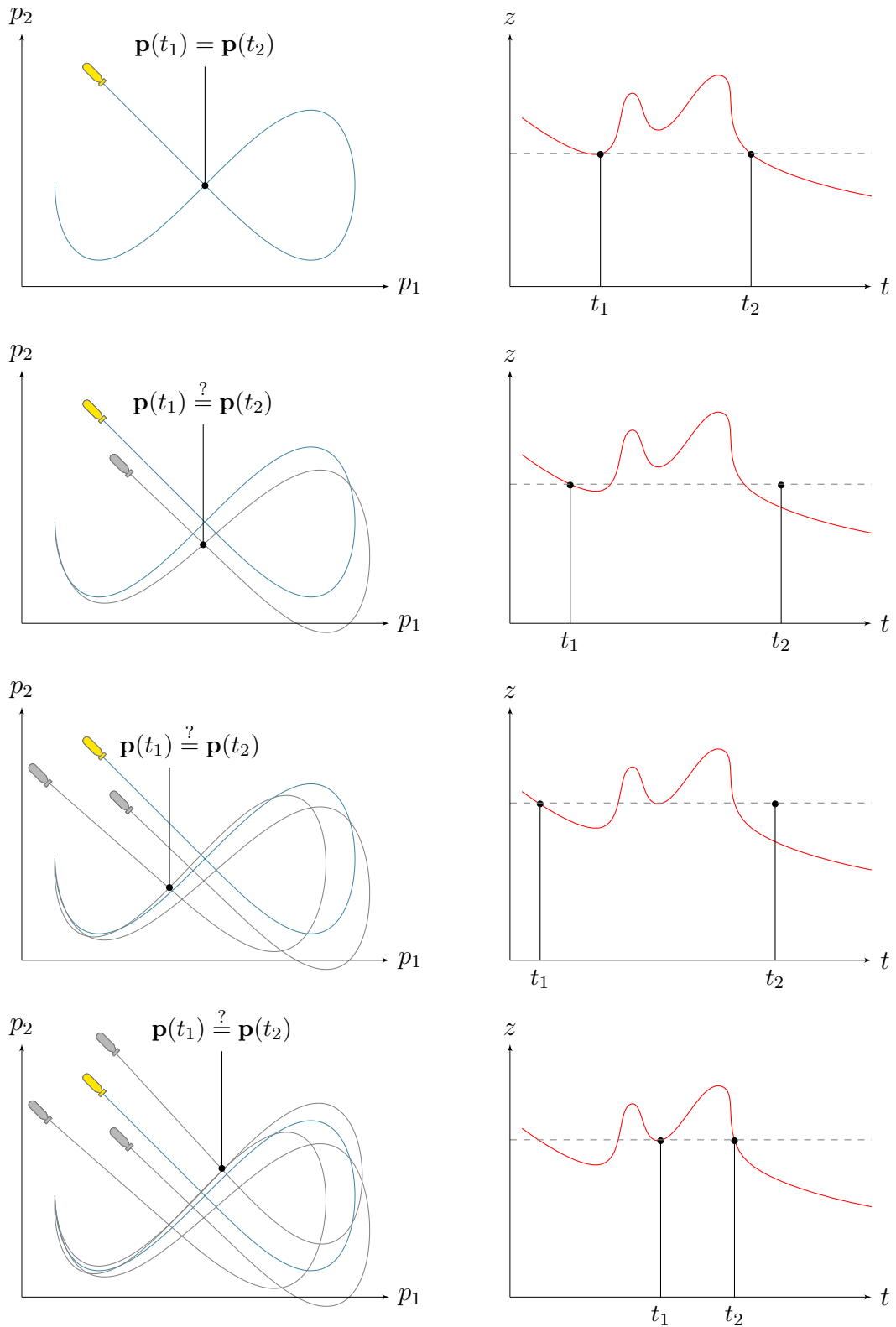


Figure 7.3: Physical interpretation of the $\mathcal{L}_{\mathbf{p} \rightarrow z}$ constraint: a robot coming back to a previous position $\mathbf{p} \in \mathbb{R}^2$ should sense the same observation $z \in \mathbb{R}$. A wrong estimation $\hat{\mathbf{p}}(\cdot)$ (plotted in gray \bullet) of the actual $\mathbf{p}^*(\cdot)$ (in blue \bullet) may be rejected if the constraint $\mathcal{L}_{\mathbf{p} \rightarrow z}(\hat{\mathbf{p}}(\cdot), \hat{\mathbf{p}}(\cdot), z(\cdot))$ is not fulfilled; it is the case for the second and third estimations as we obtain $z(t_1) \neq z(t_2)$. However, the last instance shows a case of *false alarm* with a wrong estimation $\hat{\mathbf{p}}(\cdot) \neq \mathbf{p}^*(\cdot)$ that meets the constraint.

In our robotic illustration, $\mathbb{T}_{\mathbf{p}}^*$ depicts the set of t -pairs leading to two identical configurations of the states $\mathbf{x}(t_1)$ and $\mathbf{x}(t_2)$. We recall that these configurations are described by the function \mathbf{h} . For instance, in the case of a bathymetric-based localization, only the horizontal position matters (see Equation (7.6), page 202) and so $\mathbb{T}_{\mathbf{p}}^*$ becomes what we called a *loop set* in Chapter 6. Subfigure 7.5a gives an example of $\mathbb{T}_{\mathbf{p}}^*$, obtained with proprioceptive measurements only (velocities).

In the same way, $\mathbb{T}_{\mathbf{z}}^*$ gathers all the inter-temporal keys corresponding to identical measurements. As an illustration, let us consider the set of measurements pictured in Figure 7.4. These values could correspond to bathymetric measurements. The corresponding set $\mathbb{T}_{\mathbf{z}}^*$ is pictured in blue \bullet in Subfigure 7.5b.

Furthermore, we can state from the implication ① \implies ② that

$$\mathbb{T}_{\mathbf{p}}^* \subset \mathbb{T}_{\mathbf{z}}^*. \quad (7.14)$$

$\mathbb{T}_{\mathbf{p}}^*$ is a subset of $\mathbb{T}_{\mathbf{z}}^*$ as it can be seen in Subfigure 7.5b.

Related constraint network

$$\mathcal{L}_{\mathbf{p} \Rightarrow \mathbf{z}} : \left\{ \begin{array}{l} \mathbf{Variables:} \mathbf{p}(\cdot), \mathbf{w}(\cdot), \mathbf{z}(\cdot) \\ \mathbf{Internal\ variables:} \mathbb{T}_{\mathbf{p}}^*, \mathbb{T}_{\mathbf{z}}^* \\ \mathbf{Constraints:} \\ \quad 1. \mathbb{T}_{\mathbf{p}}^* = \{(t_1, t_2) \mid \mathbf{p}(t_1) = \mathbf{p}(t_2)\} \\ \quad 2. \mathbb{T}_{\mathbf{z}}^* = \{(t_1, t_2) \mid \mathbf{z}(t_1) = \mathbf{z}(t_2)\} \\ \quad 3. \mathbb{T}_{\mathbf{p}}^* \subset \mathbb{T}_{\mathbf{z}}^* \\ \quad 4. \dot{\mathbf{p}}(\cdot) = \mathbf{w}(\cdot) \\ \mathbf{Domains:} [\mathbf{p}](\cdot), [\mathbf{w}](\cdot), [\mathbf{z}](\cdot), \mathbb{T}_{\mathbf{p}}, \mathbb{T}_{\mathbf{z}} \end{array} \right. \quad (7.15)$$

Note that this CN involves heterogeneous variables: trajectories and sets. We will employ the tools developed in this thesis to reduce their domains: trajectories are enclosed by tubes and sets will be approximated by subpavings.

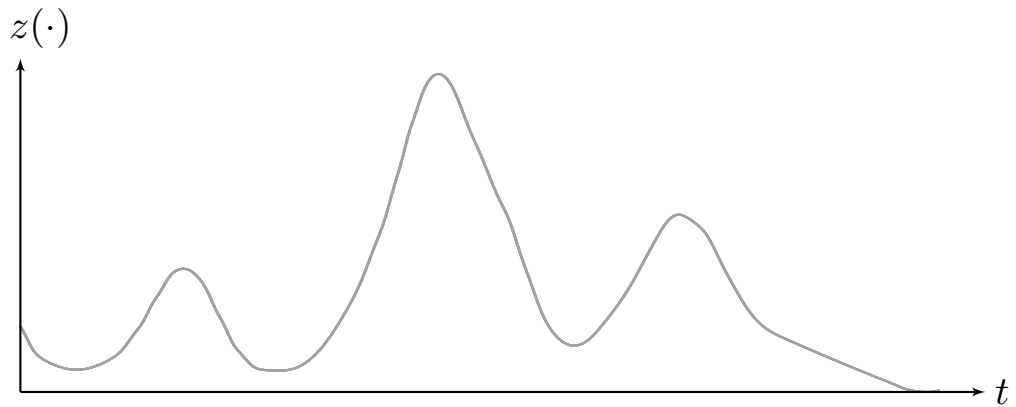


Figure 7.4: A theoretical example of scalar measurements $z(\cdot)$ sensed by a robot. The temporal set \mathbb{T}_z^* is defined according to $z(\cdot)$.

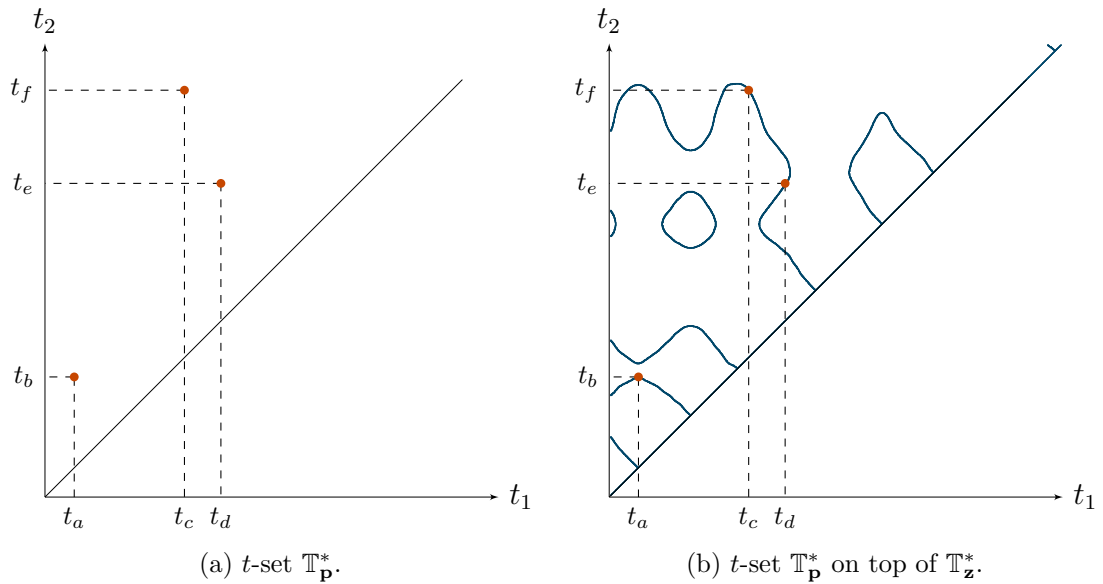


Figure 7.5: Illustration of t -sets \mathbb{T}_p^* and \mathbb{T}_z^* (see also the Figure 6.2 on page 167). Subfigure 7.5b highlights Equation (7.14).

7.2.4 The $\mathcal{C}_{p \Rightarrow z}$ contractor

The contractor related to $\mathcal{L}_{p \Rightarrow z}$ is not trivial. Besides, we cannot provide a mathematical definition of it as we did for $\mathcal{C}_{\frac{d}{dt}}$ or $\mathcal{C}_{\text{eval}}$. Here, several intermediate steps are necessary before the actual contraction.

Initial approximation of $\mathbb{T}_{\mathbf{p}}$ and $\mathbb{T}_{\mathbf{z}}$

The first step is to compute the domains $\mathbb{T}_{\mathbf{p}}$ and $\mathbb{T}_{\mathbf{z}}$ based on the Constraints (1) and (2) of CN (7.15).

Let us define the interval functions:

$$[\mathbf{f}_{\mathbf{p}}]([t_1], [t_2]) = [\mathbf{p}]([t_1]) - [\mathbf{p}]([t_2]), \quad (7.16)$$

$$[\mathbf{f}_{\mathbf{z}}]([t_1], [t_2]) = [\mathbf{z}]([t_1]) - [\mathbf{z}]([t_2]). \quad (7.17)$$

The estimation of $\mathbb{T}_{\mathbf{p}}$ and $\mathbb{T}_{\mathbf{z}}$ respectively amounts to computing the kernel of $[\mathbf{f}_{\mathbf{p}}]$ and $[\mathbf{f}_{\mathbf{z}}]$. The kernel characterization of an interval function has already been the subject of Section 2.4.4 at page 65. Hence, the results of the over-approximation of $\ker([\mathbf{f}_{\mathbf{p}}])$ will be a subpaving enclosing $\mathbb{T}_{\mathbf{p}}$. The same for $\ker([\mathbf{f}_{\mathbf{z}}])$. Figure 7.6 provides an illustration of these characterizations.

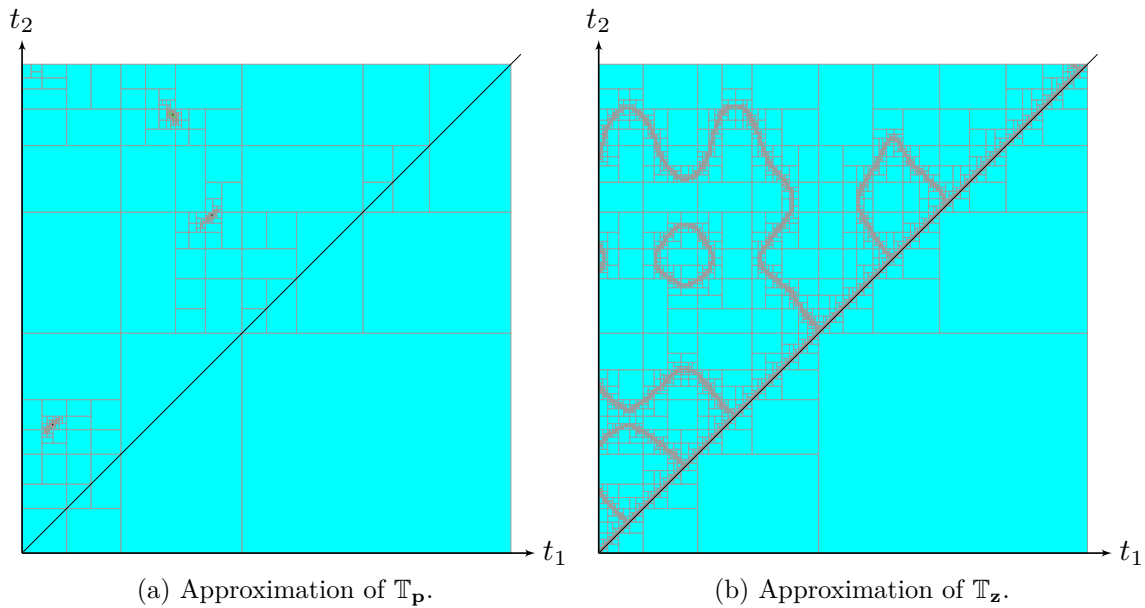


Figure 7.6: Approximation of the enclosure of t -sets presented in Figure 7.5 with SIVIA algorithms.

The sub-constraint $\mathbb{T}_{\mathbf{p}}^* \subset \mathbb{T}_{\mathbf{z}}^*$

We then focus on the relation between $\mathbf{p}(\cdot)$ and $\mathbf{z}(\cdot)$ which is at the core of $\mathcal{L}_{\mathbf{p} \Rightarrow \mathbf{z}}$. We remind that the sub-constraint $\mathbb{T}_{\mathbf{p}}^* \subset \mathbb{T}_{\mathbf{z}}^*$ is related to the implication $\mathbf{p}(t_1) = \mathbf{p}(t_2) \implies \mathbf{z}(t_1) = \mathbf{z}(t_2)$.

In a set-membership approach, only the reciprocal implication is assessed:

$$\mathbf{z}(t_1) \neq \mathbf{z}(t_2) \implies \mathbf{p}(t_1) \neq \mathbf{p}(t_2), \quad (7.18)$$

which is trivially applicable by intersecting the previously defined sets $\mathbb{T}_{\mathbf{p}}$ and $\mathbb{T}_{\mathbf{z}}$, see Figure 7.7. The information is kept in $\mathbb{T}_{\mathbf{p}}$:

$$\mathbb{T}_{\mathbf{p}} := \mathbb{T}_{\mathbf{p}} \cap \mathbb{T}_{\mathbf{z}}. \quad (7.19)$$

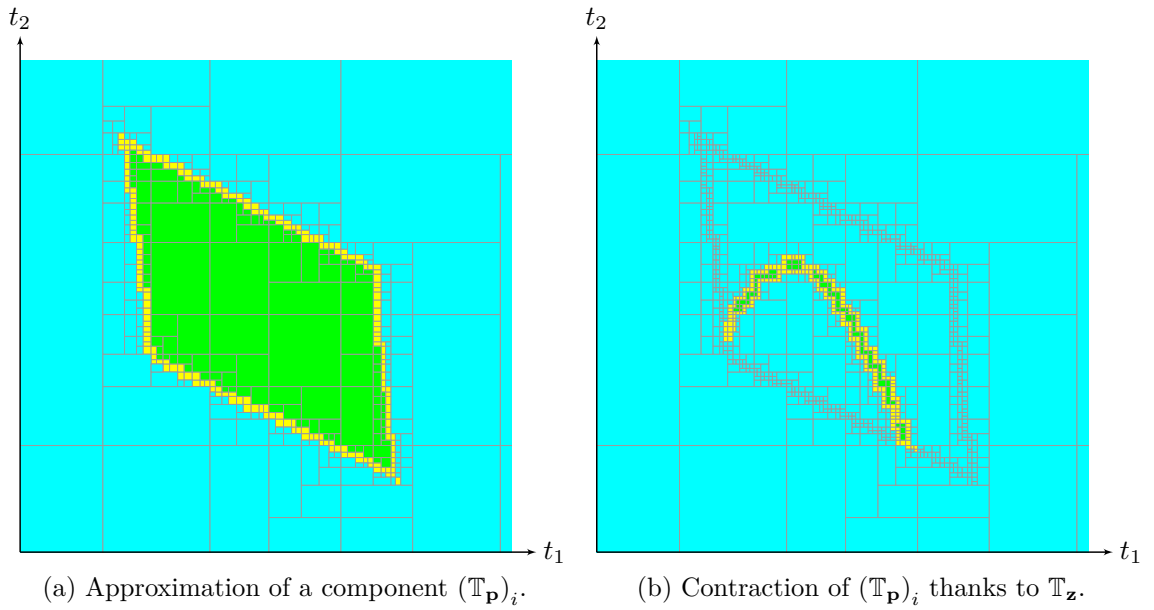


Figure 7.7: Contraction of a t -set from the $\mathcal{L}_{\mathbf{p} \Rightarrow \mathbf{z}}$ constraint.

From the temporal space to the trajectories' one

Now that the t -sets are estimated and contracted, it remains to propagate this temporal information to the tube $[\mathbf{p}](\cdot)$. The constraint of interest is $\mathbb{T}_{\mathbf{p}}^* = \{(t_1, t_2) \mid \mathbf{p}(t_1) = \mathbf{p}(t_2)\}$ that will now be considered in a *backward* way⁴, propagating information from the domain $\mathbb{T}_{\mathbf{p}}$ to $[\mathbf{p}](\cdot)$.

$\mathbb{T}_{\mathbf{p}}$ can be made of several connected subsets $(\mathbb{T}_{\mathbf{p}})_i$ such as in Figure 6.4, page 171. For each $(\mathbb{T}_{\mathbf{p}})_i$, a contraction of $[\mathbf{p}](\cdot)$ is affordable only if we can state that:

$$\exists \mathbf{t} \in (\mathbb{T}_{\mathbf{p}})_i \mid \mathbf{p}(t_1) = \mathbf{p}(t_2), \quad (7.20)$$

which is not necessarily the case due to the enclosure of $\mathbf{p}(\cdot)$ in $[\mathbf{p}](\cdot)$. This statement is equivalent to prove that:

$$\exists \mathbf{t} \in (\mathbb{T}_{\mathbf{p}})_i \mid \mathbf{f}_{\mathbf{p}}(t_1, t_2) = \mathbf{0}. \quad (7.21)$$

Verifying Equation (7.21) can be done using a zero verification test such as the one presented in Chapter 6. In this way, if we prove that $\forall \mathbf{f}_{\mathbf{p}} \in [\mathbf{f}_{\mathbf{p}}], \exists \mathbf{t} \in (\mathbb{T}_{\mathbf{p}})_i \mid \mathbf{f}_{\mathbf{p}}(t_1, t_2) = \mathbf{0}$, then the statement of Equation (7.20) is true and so we can proceed to a contraction of $[\mathbf{p}](\cdot)$.

We have seen that the connected subsets $(\mathbb{T}_{\mathbf{p}})_i$ are implemented by means of subpavings Ω_i made of t -boxes $[t]_j$. One should note that the verification of zeros of $\mathbf{f}_{\mathbf{p}}^*$ in Ω_i has to be done before the intersection of Ω_i by $\mathbb{T}_{\mathbf{z}}$ (Equation (7.19)). Indeed, we have seen at page 173 that one requirement for the topological degree test is that $\forall \mathbf{t} \in \partial\Omega_i, \mathbf{f}_{\mathbf{p}}^*(\mathbf{t}) \neq \mathbf{0}$. This is no longer the case if the boundary of Ω_i is reduced after its intersection by $\mathbb{T}_{\mathbf{z}}$.

We propose the contractor \mathcal{C}_{Ω} to apply Equation (7.20) on Ω_i and $[\mathbf{p}](\cdot)$:

$$\mathcal{C}_{\Omega}(\Omega_i, [\mathbf{p}](\cdot), [\mathbf{w}](\cdot)) = \bigcup_j \mathcal{C}_{t_1, t_2}([t_1]_j, [t_2]_j, [\mathbf{p}](\cdot), [\mathbf{w}](\cdot)), \quad (7.22)$$

where \mathcal{C}_{t_1, t_2} is a new generic inter-temporal contractor that handles tubes and boxes. The union is justified by the fact that we do not know which $[t]_j \subset \Omega_i$ surely contains an actual solution \mathbf{t}^* . We were only able to state that Ω_i is an enclosure of at least one \mathbf{t}^* .

⁴Here, the term *backward* does not refer to a temporal way but to a constraint propagation from one set back to another.

New inter-temporal contractor \mathcal{C}_{t_1, t_2}

Let us focus on the elementary constraint $\mathcal{L}_{t_1, t_2}(t_1, t_2, \mathbf{p}(\cdot), \mathbf{w}(\cdot))$ equivalent to:

$$\mathcal{L}_{t_1, t_2} : \left\{ \begin{array}{l} \text{Variables: } t_1, t_2, \mathbf{p}(\cdot), \mathbf{w}(\cdot) \\ \text{Constraints:} \\ \quad - \mathbf{p}(t_1) = \mathbf{p}(t_2) \\ \quad - \dot{\mathbf{p}}(\cdot) = \mathbf{w}(\cdot) \\ \text{Domains: } [t_1], [t_2], [\mathbf{p}(\cdot)], [\mathbf{w}(\cdot)] \end{array} \right. \quad (7.23)$$

\mathcal{L}_{t_1, t_2} is not a primitive constraint as it amounts to a composition of two $\mathcal{L}_{\text{eval}}$, which justifies the use of the derivative $\mathbf{w}(\cdot)$ in this CN:

$$\mathbf{p}(t_1) = \mathbf{p}(t_2) \iff \left\{ \begin{array}{l} \mathbf{a} = \mathbf{p}(t_1) \\ \mathbf{b} = \mathbf{p}(t_2) \\ \mathbf{a} = \mathbf{b} \end{array} \right. \iff \left\{ \begin{array}{l} \mathcal{L}_{\text{eval}}(t_1, \mathbf{b}, \mathbf{p}(\cdot), \mathbf{w}(\cdot)) \\ \mathcal{L}_{\text{eval}}(t_2, \mathbf{a}, \mathbf{p}(\cdot), \mathbf{w}(\cdot)) \\ \mathbf{a} = \mathbf{b} \end{array} \right. \quad (7.24)$$

We propose an implementation of the related contractor \mathcal{C}_{t_1, t_2} in Algorithm 11. Figure 7.8 depicts its application with inter-temporal contractions of an arbitrary tube. The contractor \mathcal{C}_{Ω} is a simple extension of \mathcal{C}_{t_1, t_2} that performs the contraction based on a subpaving Ω instead of a single box $[\mathbf{t}]$. The following Algorithm 12 is related to Equation (7.22).

Algorithm 11 \mathcal{C}_{t_1, t_2} (in : $[\mathbf{w}](\cdot)$, inout : $[t_1], [t_2], [\mathbf{p}](\cdot)$)

```

1: do
2:    $[t'_1](\cdot) \leftarrow [t_1](\cdot)$ 
3:    $[t'_2](\cdot) \leftarrow [t_2](\cdot)$ 
4:    $[\mathbf{p}'](\cdot) \leftarrow [\mathbf{p}](\cdot)$ 
5:    $[\mathbf{a}] \leftarrow [\mathbf{p}](t_1)$ 
6:    $\mathcal{C}_{\text{eval}}([t_2], [\mathbf{a}], [\mathbf{p}](\cdot), [\mathbf{w}](\cdot))$ 
7:    $[\mathbf{b}] \leftarrow [\mathbf{p}](t_2)$ 
8:    $\mathcal{C}_{\text{eval}}([t_1], [\mathbf{b}], [\mathbf{p}](\cdot), [\mathbf{w}](\cdot))$ 
9: while  $[t_1] \neq [t'_1]$  or  $[t_2] \neq [t'_2]$  or  $[\mathbf{p}](\cdot) \neq [\mathbf{p}'](\cdot)$ 

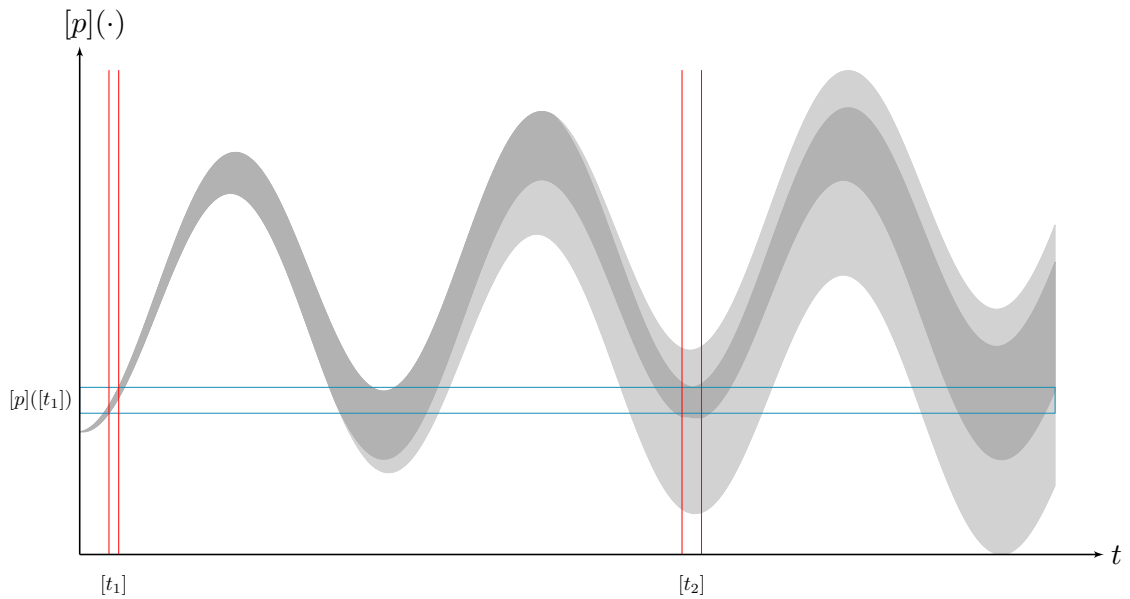
```

Algorithm 12 \mathcal{C}_Ω (in : $\Omega, [\mathbf{w}](\cdot)$, inout : $[\mathbf{p}](\cdot)$)

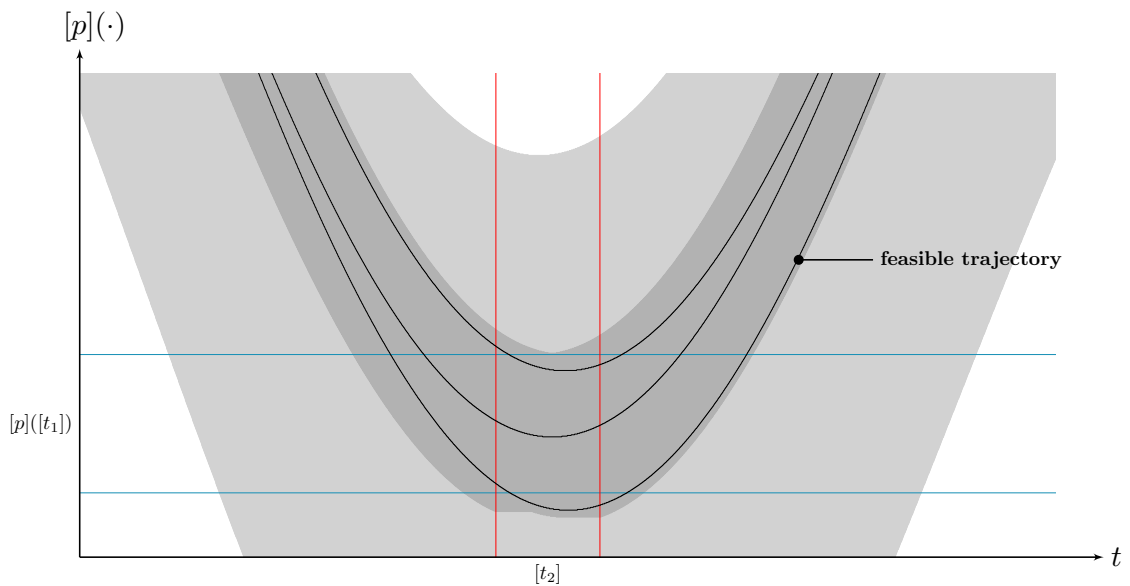
```

1:  $[\mathbf{p}'](\cdot) \leftarrow [\mathbf{p}](\cdot)$ 
2:  $[\mathbf{p}](\cdot) \leftarrow \emptyset(\cdot)$ 
3:  $[t]_1 \dots [t]_j \leftarrow \text{getBoxes}(\Omega)$ 
4: for  $k = 1$  to  $j$  do
5:    $[\mathbf{p}'' ](\cdot) \leftarrow [\mathbf{p}'](\cdot)$ 
6:    $\mathcal{C}_{t_1, t_2}([t_1]_k, [t_2]_k, [\mathbf{p}'' ](\cdot), [\mathbf{w}](\cdot))$ 
7:    $[\mathbf{p}](\cdot) \leftarrow [\mathbf{p}](\cdot) \cup [\mathbf{p}'' ](\cdot)$ 
8: end for

```



(a) Overview of the variables' domains.



(b) Zoom on the origin of the propagation with three hypothetical trajectories. Any $p(\cdot)$ going through the boxes $[t_2] \times [p](t_1)$ and $[t_1] \times [p](t_2)$ is kept in the contracted tube.

Figure 7.8: Inter-temporal contractions using \mathcal{C}_{t_1, t_2} . A tube $[p](\cdot)$ is contracted to the envelope of trajectories $p(\cdot)$ consistent with an inter-temporal constraint $p(t_1) = p(t_2)$ involving time uncertainties. Such contraction necessarily relies on the derivative of $p(\cdot)$, not represented.

The contractor $\mathcal{C}_{\mathbf{p} \Rightarrow \mathbf{z}}$

Now we have all the material to design the contractor applying the inter-temporal implicative constraint $\mathcal{L}_{\mathbf{p} \Rightarrow \mathbf{z}} : \mathbf{p}(t_1) = \mathbf{p}(t_2) \implies \mathbf{z}(t_1) = \mathbf{z}(t_2)$. The following Algorithm 13 summarizes the development of $\mathcal{C}_{\mathbf{p} \Rightarrow \mathbf{z}}([\mathbf{p}](), [\mathbf{w}](), [\mathbf{z}](), \varepsilon)$. The value $\varepsilon \in \mathbb{R}$ is a parameter defining the accuracy of internal approximations.

Algorithm 13 $\mathcal{C}_{\mathbf{p} \Rightarrow \mathbf{z}}$ (in : $[\mathbf{w}](), [\mathbf{z}](), \varepsilon$, inout : $[\mathbf{p}]()$)

```

1:  $j \leftarrow 0$   $\triangleright$  iteration identifier
2:  $[\mathbf{t}] \leftarrow [t_0, t_f]^2$ 
    $\triangleright$  defining inter-temporal inclusion functions:
3: define  $[\mathbf{f}_{\mathbf{p}}]$  ( $[a_1], [a_2]$ ) =  $[\mathbf{p}]([a_1]) - [\mathbf{p}]([a_2])$ 
4: define  $[\mathbf{f}_{\mathbf{z}}]$  ( $[a_1], [a_2]$ ) =  $[\mathbf{z}]([a_1]) - [\mathbf{z}]([a_2])$ 

5: do

6:    $j \leftarrow j + 1$ 
7:    $[\mathbf{p}'](\cdot) \leftarrow [\mathbf{p}]()$ 
8:    $\mathbb{T}_{\mathbf{p}}^- \leftarrow \emptyset, \mathbb{T}_{\mathbf{p}}^+ \leftarrow \emptyset$ 
9:    $\mathbb{T}_{\mathbf{z}}^- \leftarrow \emptyset, \mathbb{T}_{\mathbf{z}}^+ \leftarrow \emptyset$ 

10:   $(\mathbb{T}_{\mathbf{p}}^-, \mathbb{T}_{\mathbf{p}}^+) \leftarrow \text{kernelSIVIA}([\mathbf{f}_{\mathbf{p}}], [\mathbf{t}], \varepsilon)$   $\triangleright$  ①
11:   $(\mathbb{T}_{\mathbf{z}}^-, \mathbb{T}_{\mathbf{z}}^+) \leftarrow \text{kernelSIVIA}([\mathbf{f}_{\mathbf{z}}], [\mathbf{t}], \varepsilon)$   $\triangleright$  ②

12:   $\{\Omega_1, \dots, \Omega_k\} \leftarrow \text{extractSubpavings}(\mathbb{T}_{\mathbf{p}}^+)$ 

13:  for  $i = 1$  to  $k$  do
14:    if  $\text{existenceTest}\mathcal{T}(\Omega_i, [\mathbf{f}_{\mathbf{p}}])$  then  $\triangleright$  zero verification
15:       $\Omega_i \leftarrow \Omega_i \cap \mathbb{T}_{\mathbf{z}}^+$   $\triangleright$  fusion: implication ①  $\implies$  ②
16:       $\mathcal{C}_{\Omega}(\Omega_i, [\mathbf{p}](), [\mathbf{w}]())$   $\triangleright$  inter-temporal contraction
17:    end if
18:  end for

19: while  $[\mathbf{p}]() \neq [\mathbf{p}']()$ 

```

Some details are provided below:

Lines 3–4: we define inter-temporal inclusion functions in order to approximate the t -sets $\mathbb{T}_{\mathbf{p}}$ and $\mathbb{T}_{\mathbf{z}}$ based on kernel characterizations, see Equations (7.16)–(7.17);

Line 7: a copy of the tube $[\mathbf{p}](\cdot)$ is performed in order to detect a fixed point later on (line 19) in this iterative process. Note that this operation can be costly in terms of time and memory. The implementation of \mathcal{C}_{Ω} could return a boolean testifying whether a contraction on $[\mathbf{p}](\cdot)$ has been made or not;

Lines 10–11: the `kernelSIVIA` function is provided in Algorithm 2, page 68. Its output is an enclosure of $\mathbb{T}_{\mathbf{p}}$ with two subpavings $(\mathbb{T}_{\mathbf{p}}^-, \mathbb{T}_{\mathbf{p}}^+)$. The same for $\mathbb{T}_{\mathbf{z}}$. Note that for our application, only outer sets are being used. See also Figure 7.6;

Line 12: the `extractSubpavings` algorithm consists in detecting the list of closed and connected subsets in $\mathbb{T}_{\mathbf{p}}^+$. This algorithm is not detailed in this document;

Line 14: for a given subset Ω_i of the outer approximation of $\mathbb{T}_{\mathbf{p}}$, we verify the existence of a zero of $\mathbf{f}_{\mathbf{p}}^*$. See Algorithm 7 detailed in page 177;

Line 15: in case of a zero verification, we apply Equation (7.19) by performing the intersection of two subpavings. See Figure 7.7;

Line 16: once the t -set has been refined, a contraction of $[\mathbf{p}](\cdot)$ is achieved by \mathcal{C}_{Ω} , see Algorithm 12, page 213.

The overall process is repeated indefinitely until no more contractions are performed on $[\mathbf{p}](\cdot)$. Indeed, among the variables' domains $[\mathbf{p}](\cdot)$, $[\mathbf{w}](\cdot)$, $[\mathbf{z}](\cdot)$, only $[\mathbf{p}](\cdot)$ can be contracted. This is due firstly because of the implication constraint, since the propagation cannot reach $\mathbf{z}(\cdot)$. In addition, the derivative $\mathbf{w}(\cdot)$ cannot be constrained as discussed in Section 4.2.2, page 109.

The implementation of this algorithm can be multi-threaded from line 13 to 18: each subset Ω_i can be verified and contracted independently. The contraction of $[\mathbf{p}](\cdot)$ from Ω_i will however require some mutual exclusions. It should be noted that the only parameter to be set is ε , a scalar value depicting the precision of the approximation of t -sets. In practice⁵, ε is related to the tubes' timestep δ .

Remark 7.1

A faster implementation of $\mathcal{C}_{\mathbf{p} \Rightarrow \mathbf{z}}$ is possible with the following instructions provided in Algorithm 14.

By considering a contracted envelope $[\mathbf{b}] = [\Omega_i] \in \mathbb{IR}^2$ of the Ω_i instead of each t -box of the subpaving (see Figure 7.9), the computations related to \mathcal{C}_{t_1, t_2} are reduced to only one by Ω_i set. The counterpart is that we will obtain less precise results due to the wrapping effect of $[\mathbf{b}] \supseteq \Omega_i$.

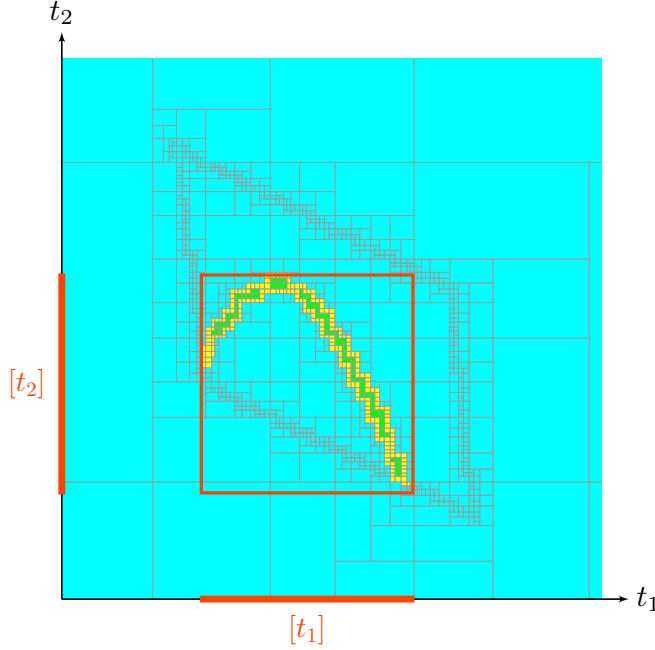


Figure 7.9: Minimal envelope $[\mathbf{b}]$ of Ω_i .

Algorithm 14 $\mathcal{C}_{\mathbf{p} \Rightarrow \mathbf{z}}^{\text{fast}}$ (in : $[\mathbf{w}](\cdot), [\mathbf{z}](\cdot), \varepsilon$, inout : $[\mathbf{p}](\cdot)$)

```

...
for  $i = 1$  to  $k$  do
    if existenceTest $\mathcal{T}$  ( $\Omega_i, [\mathbf{f}_p]$ ) then
         $\Omega_i \leftarrow \Omega_i \cap \mathbb{T}_z^+$ 
         $[\mathbf{b}] \leftarrow [\Omega_i]$ 
         $\mathcal{C}_{t_1, t_2} ([b_1], [b_2], [\mathbf{p}](\cdot), [\mathbf{w}](\cdot))$ 
    end if
end for
...

```

\triangleright zero verification
 \triangleright fusion: implication ① \implies ②
 \triangleright boxed envelope of Ω_i
 \triangleright inter-temporal contraction

⁵Note that it is not relevant to use $\varepsilon < \delta$ since bisections of subpavings (in a bi-temporal space) will not provide more accurate information.

7.2.5 Temporal SLAM algorithm

We are now able to address our SLAM problem defined in CN (7.10), page 204. Each constraint will be implemented by the corresponding contractor. We recall the variables involved in the problem:

- $\mathbf{x}(\cdot)$: states of the robot;
- $\mathbf{u}(\cdot)$: inputs of the system;
- $\mathbf{z}(\cdot)$: observation measurements.

Intermediate variables are defined for resolution purposes:

- $\mathbf{v}(\cdot)$: state evolutions (derivative of $\mathbf{x}(\cdot)$) that can be measured;
- $\mathbf{p}(\cdot)$: state configurations, $\mathbf{p}(\cdot) = \mathbf{h}(\mathbf{x}(\cdot))$;
- $\mathbf{w}(\cdot)$: derivative of $\mathbf{p}(\cdot)$.

In the following, we will focus on a SLAM involving measurements $\mathbf{z}(\cdot)$ that only depends on 2D positions, as it is the case for bathymetric sensing, see Figure 7.10. In this context, the *configuration* function \mathbf{h} is defined by

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} \xrightarrow{\mathbf{h}} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad (7.25)$$

and vanishes in case of a 2D loop.

\mathbf{h} will be used to compute the trajectory $\mathbf{p}(\cdot)$ from $\mathbf{x}(\cdot)$. The derivative $\mathbf{w}(\cdot)$ is also necessary for the constraint $\mathcal{L}_{\mathbf{p} \Rightarrow \mathbf{z}}(\mathbf{p}(\cdot), \mathbf{w}(\cdot), \mathbf{z}(\cdot))$. Note that in our context, $\mathbf{w}(\cdot)$ can be computed using the same function \mathbf{h} :

$$\mathbf{w}(\cdot) = \frac{d\mathbf{h}}{d\mathbf{x}(\cdot)} \cdot \mathbf{v}(\cdot) = \mathbf{h}(\mathbf{v}(\cdot)). \quad (7.26)$$

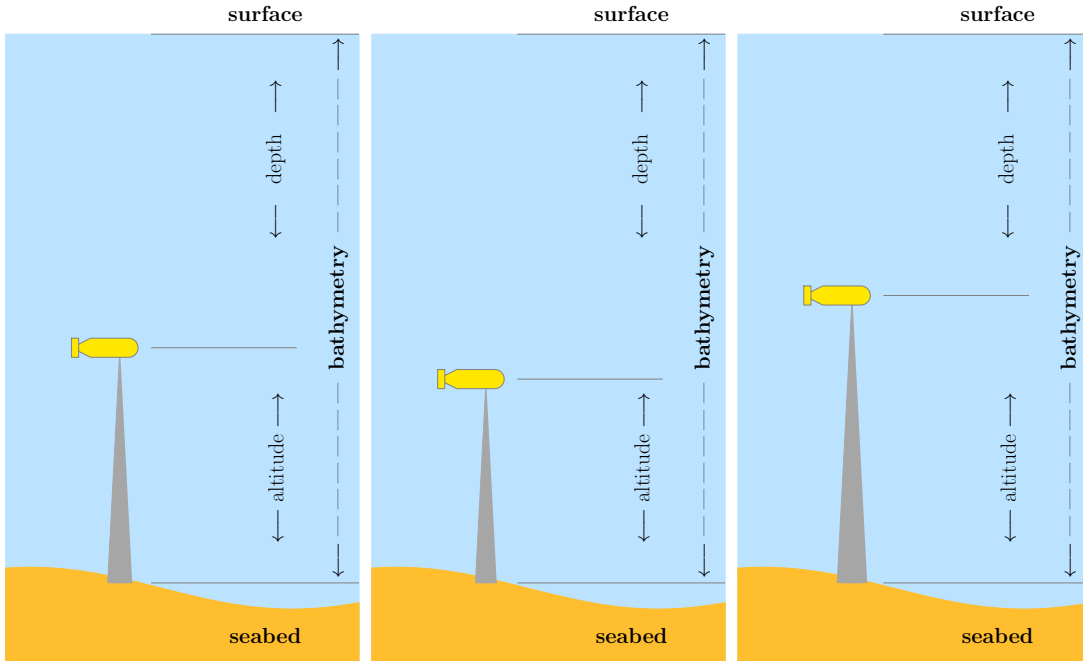


Figure 7.10: In the bathymetric SLAM application, the sensing will be the same for any depth and orientation of the robot. Since the bathymetry only depends on 2D positions, we can state that two different measurements necessarily mean two different positions. Such configuration is expressed by the function \mathbf{h} of Equation (7.25).

The constraint network related to this SLAM is:

$$\text{SLAM : } \left\{ \begin{array}{l}
 \mathbf{Variables: } \mathbf{x}(\cdot), \mathbf{v}(\cdot), \mathbf{p}(\cdot), \mathbf{w}(\cdot), \mathbf{z}(\cdot), \mathbf{u}(\cdot) \\
 \mathbf{Constraints:} \\
 \quad 1. \text{ Evolution constraint:} \\
 \quad \quad - \mathbf{v}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \\
 \quad \quad - \mathcal{L}_{\frac{d}{dt}}(\mathbf{x}(\cdot), \mathbf{v}(\cdot)) \\
 \quad 2. \text{ Inter-temporal constraint:} \\
 \quad \quad - \mathbf{p}(\cdot) = \mathbf{h}(\mathbf{x}(\cdot)) \\
 \quad \quad - \mathbf{w}(\cdot) = \mathbf{h}(\mathbf{v}(\cdot)) \\
 \quad \quad - \mathcal{L}_{\mathbf{p} \Rightarrow \mathbf{z}}(\mathbf{p}(\cdot), \mathbf{w}(\cdot), \mathbf{z}(\cdot)) \\
 \mathbf{Domains: } [\mathbf{x}(\cdot), [\mathbf{v}(\cdot), [\mathbf{p}(\cdot), [\mathbf{w}(\cdot), [\mathbf{z}(\cdot), [\mathbf{u}(\cdot)
 \end{array} \right. \quad (7.27)$$

In addition, we will overwrite the inter-temporal inclusion function $[\mathbf{f}_p]([t_1], [t_2]) = [\mathbf{p}]([t_2]) - [\mathbf{p}]([t_1])$ that will be used for the $\mathcal{C}_{p \Rightarrow z}$ contractor. This will allow a thinner approximation of the kernel of \mathbf{f}_p :

$$[\mathbf{f}_p]([t_1], [t_2]) = \left([\mathbf{p}]([t_2]) - [\mathbf{p}]([t_1]) \right) \cap \left(\int_{[t_1]}^{[t_2]} [\mathbf{w}](\tau) d\tau \right). \quad (7.28)$$

This formulation allows us to detect more loops in a non-dead-reckoning context, taking advantage of external constraints on both the position trajectories enclosed by $[\mathbf{p}](\cdot)$ and the velocities bounded by $[\mathbf{w}](\cdot)$. In practice, the characterization of $\ker([\mathbf{f}_p])$ is achieved using the `kernelSIVIA` algorithm (page 68) coupled with `proprioLoopSIVIA` (page 170) that will estimate loop sets based on bounded velocities $[\mathbf{v}](\cdot)$ only⁶. The results of these algorithms are subpavings that can be intersected to obtain the approximation of $\mathbb{T}_p = \ker([\mathbf{f}_p])$. See for instance Figure 7.6a, page 209.

We propose the following Algorithm 15 for the localization method, in which each constraint of CN (7.27) is applied by related contractors. We underline that the contractors \mathcal{C}_f and \mathcal{C}_h are trivially built from a composition of algebraic operators based on the expressions of \mathbf{f} and \mathbf{h} .

Algorithm 15 temporalSLAM (in : $[\mathbf{v}](\cdot), [\mathbf{u}](\cdot), [\mathbf{z}](\cdot), \varepsilon$, inout : $[\mathbf{x}](\cdot)$)

- 1: $\mathcal{C}_f([\mathbf{v}](\cdot), [\mathbf{x}](\cdot), [\mathbf{u}](\cdot))$ \triangleright evolution function
 - 2: $\mathcal{C}_{\frac{d}{dt}}([\mathbf{x}](\cdot), [\mathbf{v}](\cdot))$ \triangleright reaching a consistency state between $[\mathbf{x}](\cdot)$ and $[\mathbf{v}](\cdot)$
 - 3: $\mathcal{C}_h([\mathbf{p}](\cdot), [\mathbf{x}](\cdot))$ \triangleright new tube from the configuration function
 - 4: $\mathcal{C}_h([\mathbf{w}](\cdot), [\mathbf{v}](\cdot))$ \triangleright corresponding derivative
 - 5: $\mathcal{C}_{p \Rightarrow z}([\mathbf{p}](\cdot), [\mathbf{w}](\cdot), [\mathbf{z}](\cdot), \varepsilon)$ \triangleright inter-temporal resolution
 - 6: $\mathcal{C}_h([\mathbf{p}](\cdot), [\mathbf{x}](\cdot))$ \triangleright state contraction
-

Remark 7.2

A faster version of this algorithm is obtained using the $\mathcal{C}_{p \Rightarrow z}^{\text{fast}}$ contractor provided in Algorithm 14. It will be referenced as `fastTemporalSLAM` and illustrated in the following section.

⁶Note that we could also take into account second derivative information, if available.

7.3 Underwater application: bathymetric SLAM

The proposed method will be illustrated over two underwater experiments involving the *Daurade* AUV. We will call this problem *bathymetric SLAM* since the localization will be achieved based on altitude measurements obtained by acoustic means.

7.3.1 Context

Echosounders

Bathymetric SLAM has been the object of several studies [Barkby et al., 2009, Palomer et al., 2016], all of them presenting probabilistic approaches. They also mostly rely on multibeam echosounders that are expensive sonars providing a scanline of vertical points for each sensor pulse⁷. Figure 7.1, page 197, illustrates the footprint of this kind of sonar. A large amount of work has been undertaken with the aim of efficiently performing map-matchings over each data overlapping [Chailloux et al., 2011, Leblond et al., 2005].

However, multibeam sonars can be absent from vehicles due to their cost or the kind of mission to perform. Then the use of single beam echosounders has to be considered for SLAM, which has been the subject of very few studies [Barkby, 2011, Bichucher et al., 2015]. The problem is indeed challenging since overlaps are sparse. Therefore, because of the paucity of relevant information, we will deal with a one-dimensional observation vector: $z \in \mathbb{R}$.

Obtaining the altitude from a DVL sensor

Taking this line of thought further, we will also assume the altitude measurements will be provided by a DVL instead of a conventional single beam echosounder. This assumption has the advantage to be applicable on almost all AUVs equipped for long-range navigations. Indeed, the very same sensor is able to deliver both speed and altitude measurements, that are the necessary values for our localization method. In addition, using the same sensor for these measurements simplifies lever arms computations.

⁷For instance, a multibeam echosounder may provide data from 120 beams over 120 degrees.

However, the altitude information obtained from a DVL is extremely filtered. The sensing consists in the emission of four beams that may cover a wide area of the seafloor in case of high altitudes. Figure 7.11 gives an illustration of the range of the DVL lobes. Consequently, the higher the altitude of the robot, the more the data is filtered. Furthermore, it is difficult to estimate the measurements errors in this context: the datasheets rarely give reliable standard deviations about altitude measurements.

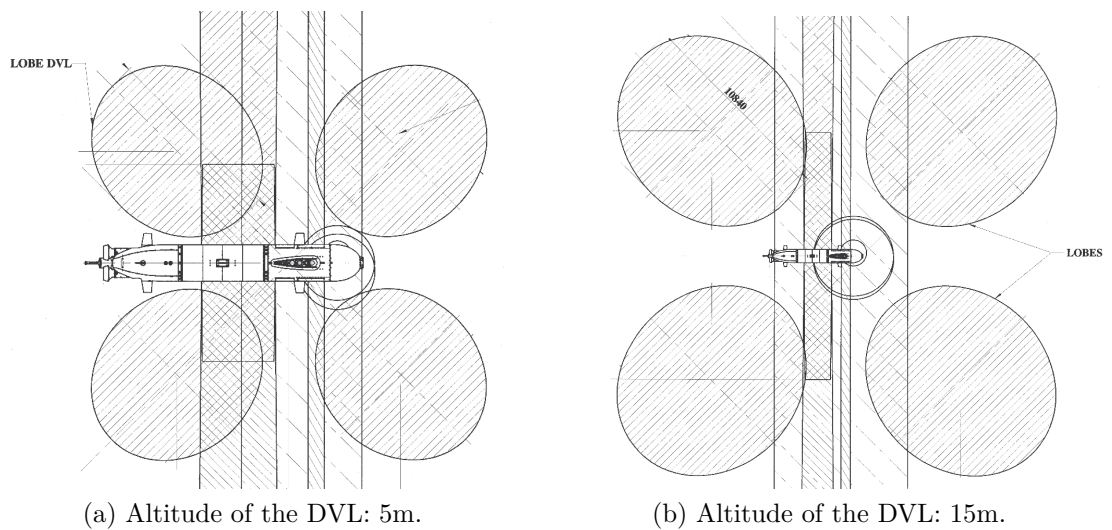


Figure 7.11: Footprint of the DVL beams according to the altitude (top view). Ellipses represent DVL lobes from which an altitude can be approximated. Hence, the higher the altitude of the robot, the more the data is filtered. *Source: Daurade technical datasheet.*

In the following sections, we will consider altitude measurements z_{alt} obtained from the DVL, keeping in mind that a single beam echosounder would provide more accurate results. Technical issues with *Daurade*'s embedded sonars prevented us from using precise measurements. We plan to perform new experiments with more accurate sounders for which standard deviations are provided by manufacturers.

For now, we have no choice but to use an arbitrary standard deviation on the measurements z_{alt} . Therefore, the experimental results presented in this chapter will not necessarily be relevant for assessing the reliability of our method for the guaranteed localization of *Daurade* with a DVL. However, this test case is still interesting to understand the basis of our SLAM, its iterative resolution and computation times.

Time-dependent measurements

The vertical position of the robot changes along the mission. The altitude observation has to be added to the depth z_{depth} obtained with an accurate pressure sensor. Additionally, it is important to take into account the rises and falls of sea levels along the mission. This is easily achievable with tide models that can be coupled with oceanographic observations for higher precisions. In our application, we will use a sinusoidal model:

$$z_{\text{tide}} = ma \cdot \sin^2\left(\frac{\pi/2 \times \Delta t}{Du}\right), \quad (7.29)$$

with:

- z_{tide} , the height variation from a given reference z_0 ;
- ma , the tidal range;
- Δt , the elapsed time since the chosen time reference;
- Du , the duration of the tide.

These values are available in classical tide charts. Note that the tide reference z_0 will have no impact in the resolution method, as it was the case for the m_0 mass in the Borda's example, page 201.

The final observation is expressed as

$$z = z_{\text{alt}} + z_{\text{depth}} + z_{\text{tide}}, \quad (7.30)$$

with respective standard deviations. This observation is vertically invariant as it only depends on the horizontal position of the robot: the heading of the AUV does not impact the measurement⁸ and we will further assume the pitch θ and roll ϕ angles are almost null⁹.

Proprioceptive measurements

The robot is described by a state vector \mathbf{x} where $(x_1, x_2)^\top$ is the horizontal position and x_3 the depth. To keep things simple, we will use the model introduced in

⁸Assuming the coverage of the DVL lobes is independent of the sensor's horizontal orientation.

⁹Which is a realistic assumption since *Daurade* is accurately regulated during the survey. Furthermore, $\sin(\theta) \simeq \sin(\phi) \simeq \sin(0) \simeq 0$ will have a little impact on the altitude measurement.

Section 4.4.4, page 126, without inertial hybridization:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \mathbf{R}(\psi, \theta, \varphi) \cdot \mathbf{v}_r, \quad (7.31)$$

where $\mathbf{R}(\psi, \theta, \varphi)$ is the Euler matrix given by Equation (4.32) at page 126 and $\mathbf{v}_r \in \mathbb{R}^3$ is the velocity vector provided by the DVL and expressed in the robot's coordinate system.

Hybridization models applied by INSs can also be considered and should be more accurate as they rely on inertial measurements too. However, the precise characterization of positioning errors is not always at hand, or defined by error models that may be questionable in a guaranteed context.

Assumption on the space

The space is assumed to be Euclidean, which is a rather realistic hypothesis when dealing with exploration areas up to a few hundred meters wide. In wider environments, the curvature of the Earth must be integrated in the equations in order to maintain the reliability of the outputs. This only impacts evolution and inter-temporal equations: the resolution method will remain the same.

7.3.2 *Daurade's* underwater mission, 20th October 2015

The experiments presented in this chapter took place in the *Rade de Brest* (France), see Figure 7.12. The first trial we present has been performed the 20th of October 2015 and was dedicated to this thesis.

Daurade surveyed a 25 hectares seabed area without surfacing during 1h40. The trajectory planned by the operators of the mission boat *Aventurière II* involved a classical boustrophedon after an overall crossing of the area, see Figure 7.16.

This application has already been presented in Section 6.4.2, page 187, where the existence of 114 loops had been proven. This time, we will use the model of Equation (7.31) and apply the temporal SLAM method on this dataset, coupled with bathymetric measurements pictured in Figure 7.14.

7.3. Underwater application: bathymetric SLAM

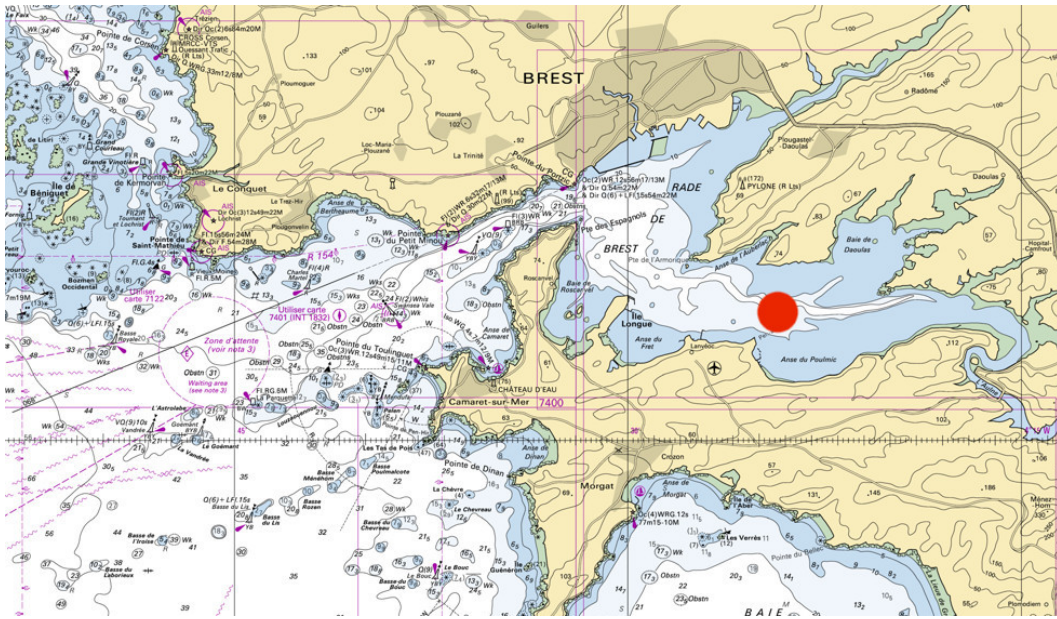


Figure 7.12: Location of the mission area (*Polygone de Rascas*) in the *Rade de Brest*. WGS84: 48° 18' 07.20" N, 4° 24' 19.68" W. Credits: SHOM.



Figure 7.13: *Daurade* before the experiment, on the working deck of the *Aventure II*, the 20th October 2015. Photo: S. Rohou.

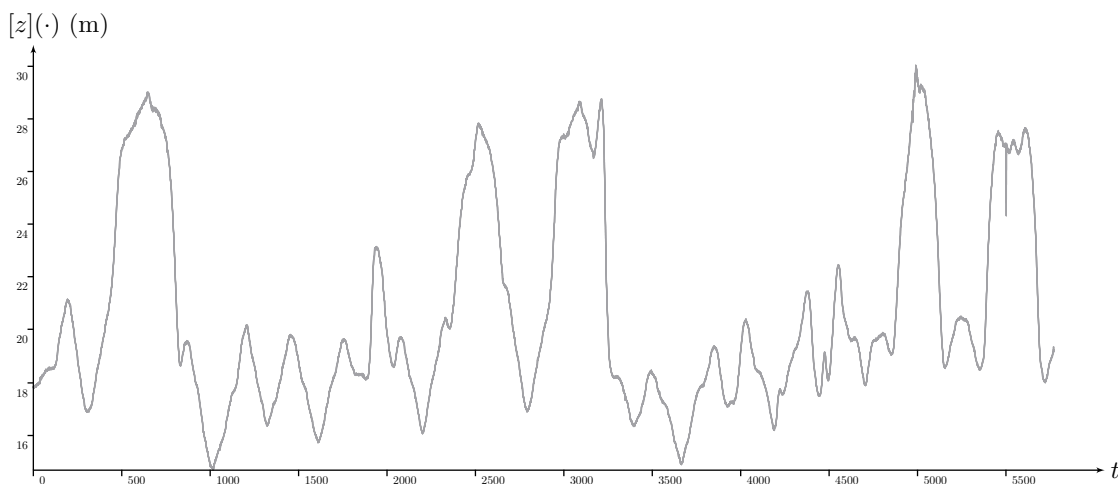


Figure 7.14: Bounded observations $[z](\cdot)$ representing bathymetric measurements obtained from the DVL, a pressure sensor and tide models. The tube is built without any filtering process on the data provided by the sensors.

As a first step, the `fastTemporalSLAM` algorithm is applied, calling the $\mathcal{C}_{\mathbf{p} \Rightarrow \mathbf{z}}^{\text{fast}}$ operator up to a fixed point in 5 iterations during 16 minutes. Table 7.2 and Figure 7.17 summarize each iteration. We can note that this iterative resolution lets appear successively new loop detections as well as new loop proofs, which raises new constraints for the following stages.

The main contractions on $[\mathbf{p}](\cdot)$ are performed on the first iteration with a drift reduced in 259 seconds by 63% for the last slice $[\mathbf{p}](t_f)$. Therefore, one could use the `fastTemporalSLAM` method without an iterative resolution process, if fast computations prevail on accurate results.

After reaching the fixed point, we then apply the full `temporalSLAM` algorithm in order to refine the results. The best outcomes are obtained after 143 minutes with a 77% drift reduction. We underline that this algorithm has been multithreaded and that computations have been performed on a eight-core processor. There are many opportunities to improve the calculation times, for instance by trying another parameter ε for the SIVIA algorithms, by improving the implementation of $\mathcal{C}_{\text{eval}}$ – an operator called thousands of times – or by using another computer representation for the tubes or the t -plane’s paving.

7.3.3 *Daurade's* underwater mission, 19th October 2015

The constraint based approach defended in this document allows us to consider any kind of observation at any time. Therefore, one could apply this SLAM method without any knowledge on the initial condition – see for instance the *kidnapped robot problem*, Section 4.4.3, page 125.

We will apply the temporal SLAM in this context with another experiment performed the day before, in the same area. Related figures are displayed at pages 230 and 231.

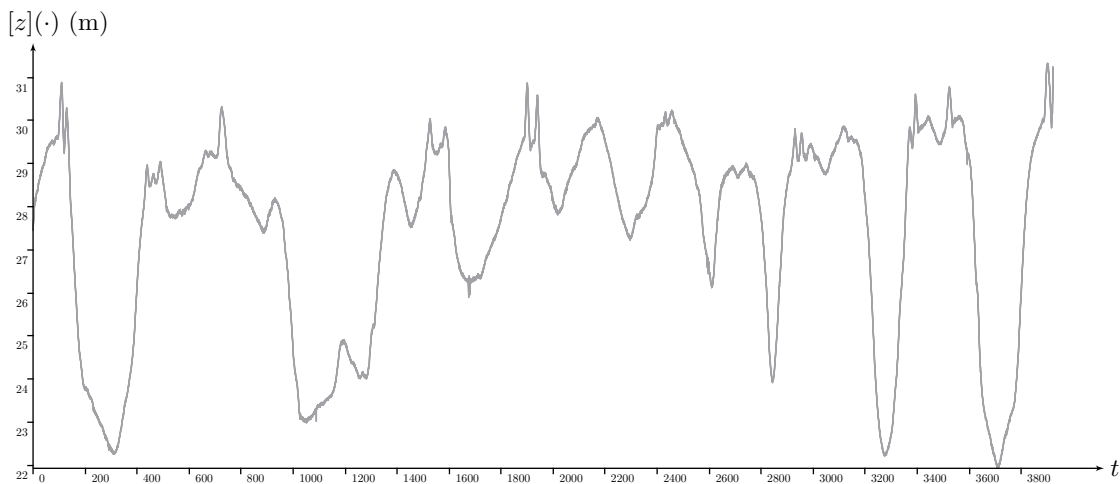


Figure 7.15: 19th October 2015: observation tube $[z](\cdot)$.

This time, an accurate positioning estimation is assumed in the middle of the mission. Figure 7.17 reveals backward and forward propagations from this reference.

In addition, this example demonstrates that the `fastTemporalSLAM` algorithm may provide poor results compared with the full `temporalSLAM` version. In this experiment, significant improvements are obtained with the accurate variant. It means that the wrapping effect on the loop sets Ω_i , enclosed by a box $[\mathbf{b}] = [\Omega_i]$, is too important on this application so that it becomes relevant to study each t -box of these subpavings. Such pessimistic effect could be evaluated in order to select the appropriate algorithm to apply, but the global impact of these enclosures on the contractions is not easily predictable.

Summary of the 20th October experiment

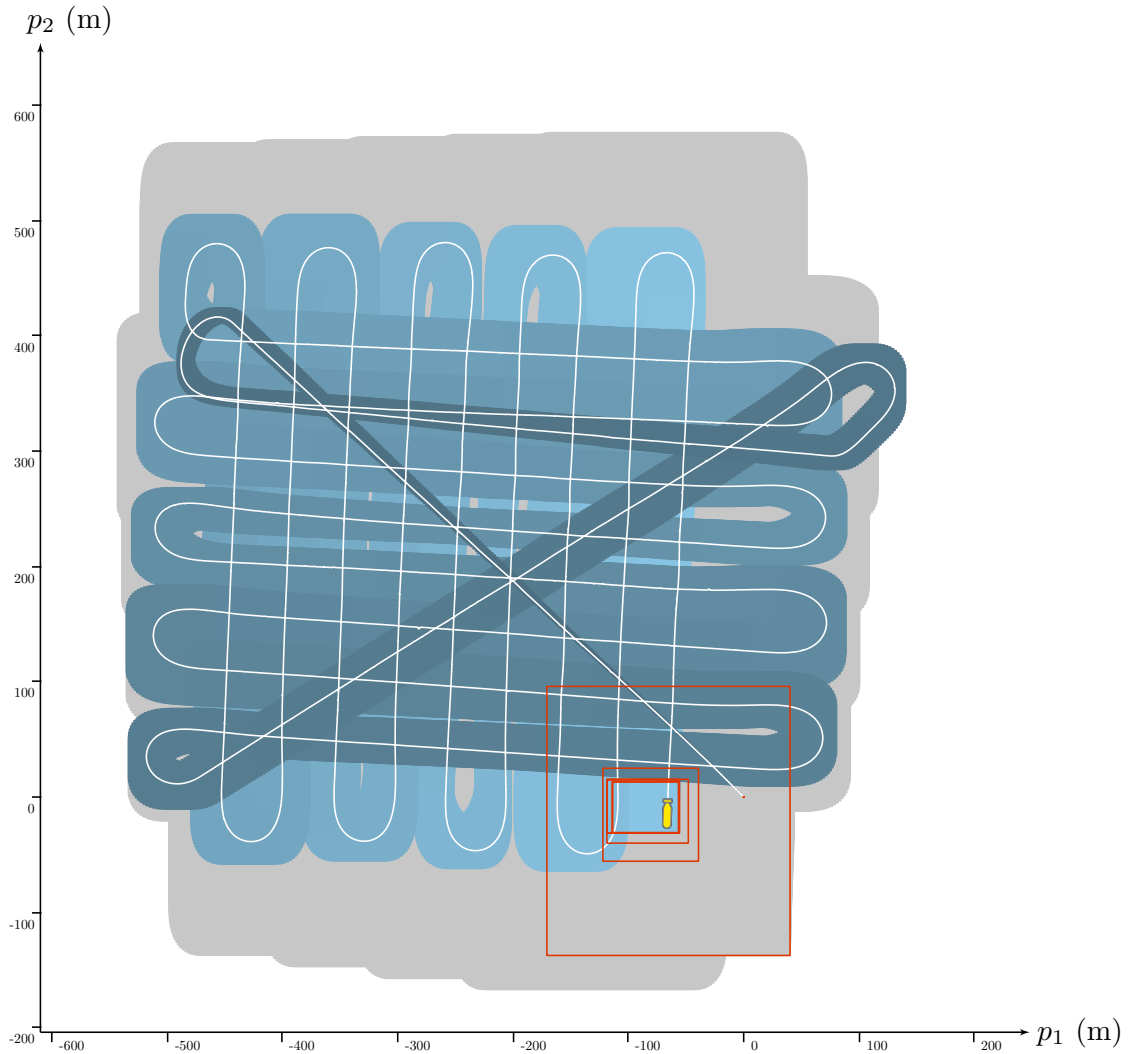


Figure 7.16: 20th October 2015: mission map. The robot evolved under the surface during 1h36. The gray area • depicts a bounded dead-reckoning method while the blue tube • is the result obtained with the temporal SLAM approach. The white line is *Daurade*'s trajectory estimated by means of its embedded INS filtered by a USBL. Red boxes • depict the last slice of the tube $[\mathbf{p}](\cdot)$, successively reduced by the iterative algorithm.

j	loop detections	loop proofs	computation time	cumulated comp. time	$[\mathbf{p}](t_f)$ contraction	SLAM algorithm
1	122	104	259s	259s	63.22%	fast
2	128	112	192s	451s	71.46%	fast
3	128	112	172s	623s	75.17%	fast
4	129	115	180s	803s	75.22%	fast
5	129	115	182s	985s (0h16)	75.22%	fast
fixed point						
6	129	115	2708s (0h45)	3693s (1h02)	76.91%	accurate
7	129	115	2506s (0h41)	6199s (1h43)	76.96%	accurate
8	129	115	2391s (0h40)	8590s (2h23)	76.96%	accurate
fixed point						

Table 7.2: 20th October 2015: SLAM iterations on *Daurade*'s experiment. Each line correspond to an iteration of the $\mathcal{C}_{\mathbf{p} \Rightarrow \mathbf{z}}$ or $\mathcal{C}_{\mathbf{p} \Rightarrow \mathbf{z}}^{\text{fast}}$ contractor. The percentage column expresses the contraction rate of the final position box $[\mathbf{p}](t_f)$, initially obtained with a dead-reckoning method.

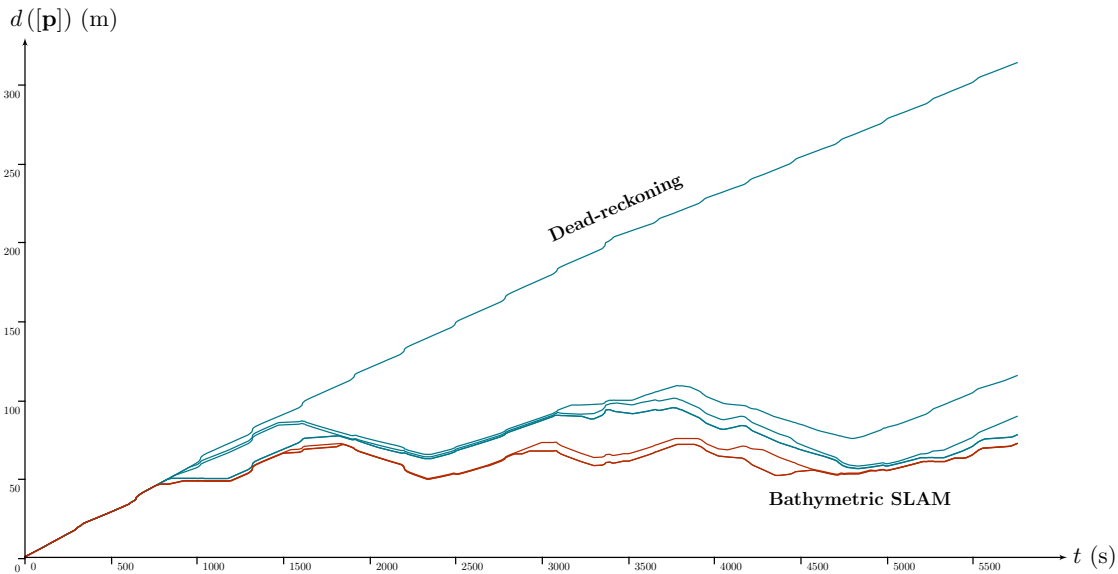


Figure 7.17: 20th October 2015: thickness graph depicting the drift of *Daurade* along the mission. The initial drift is linear due to the single integration of velocity measurements, Equation (7.31). The algorithm `fastTemporalSLAM` is then applied iteratively up to a fixed point. Its efficiency on $[\mathbf{p}](\cdot)$ is plotted in blue \bullet . Then, we apply the algorithm `temporalSLAM` in order to obtain thinner results drawn in red \bullet . The last values of this graph, at $t_f = 5760\text{s}$, depict the diagonal length of the red boxes drawn in Figure 7.16.

Summary of the 20th October experiment

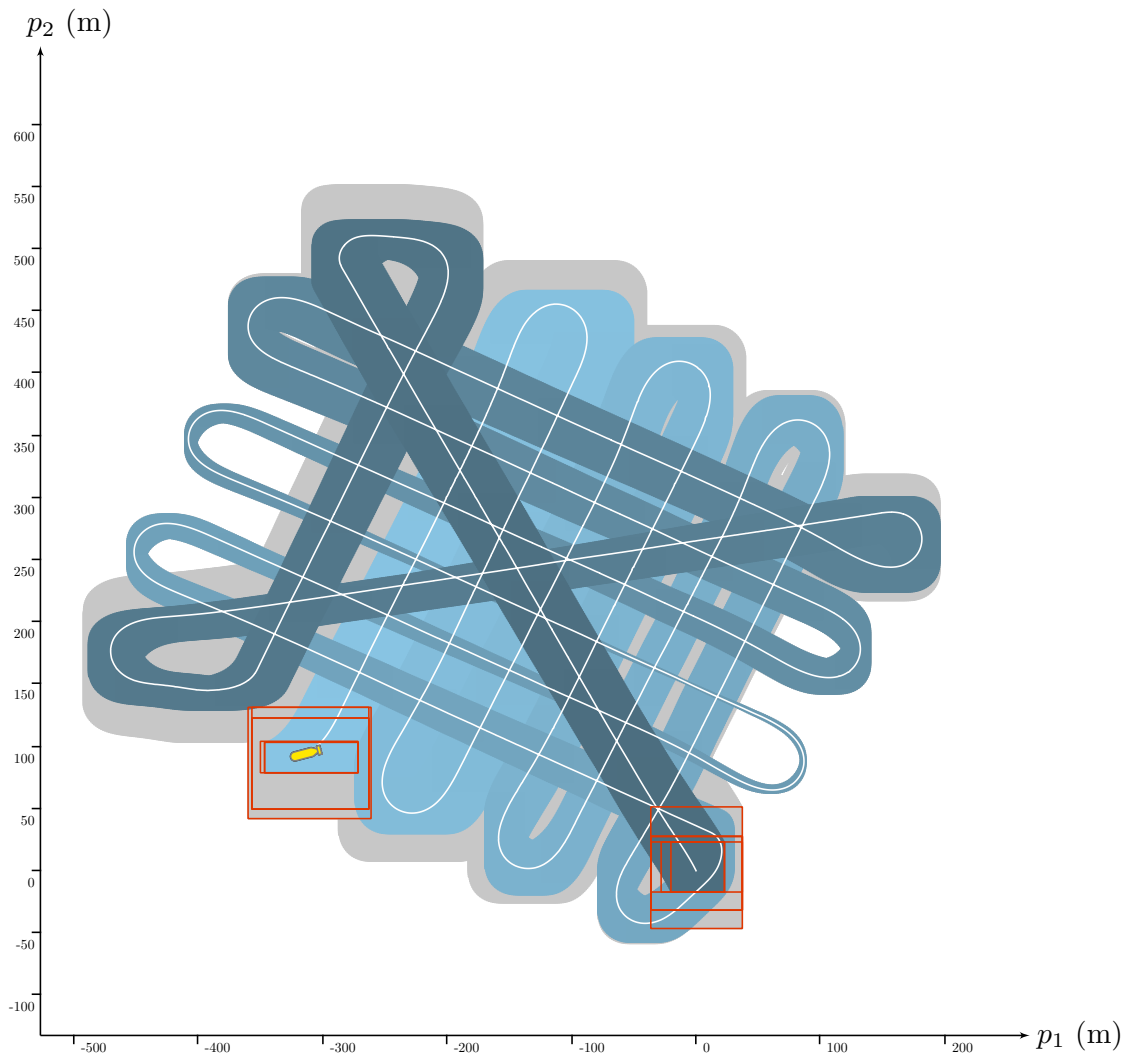


Figure 7.18: 19th October 2015: mission map. This time, the initial state is assumed unknown. The constraints are propagated from a known position in the middle of the mission in backward and forward ways. Red boxes • picture estimation improvements on initial and final states.

j	loop detections	loop proofs	computation time	cumulated comp. time	$[\mathbf{p}](t_0)$ contraction	SLAM algorithm
1	76	65	93s	93s	22.76%	fast
2	78	67	90s	183s	22.76%	fast
3	78	67	108s	291s (0h05)	22.76%	fast
fixed point						
4	78	67	1726s (0h29)	2017s (0h34)	31.47%	accurate
5	77	67	1392s (0h23)	3409s (0h57)	46.96%	accurate
6	77	67	1424s (0h24)	4833s (1h21)	51.85%	accurate
7	77	68	1470s (0h24)	6303s (1h45)	51.85%	accurate
fixed point						

Table 7.3: 19th October 2015: SLAM iterations on *Daurade*'s experiment. The robot evolved under the surface during 1h05. The percentage column expresses the contraction rate of the first position box $[\mathbf{p}](t_0)$, initially obtained with a backward dead-reckoning method.

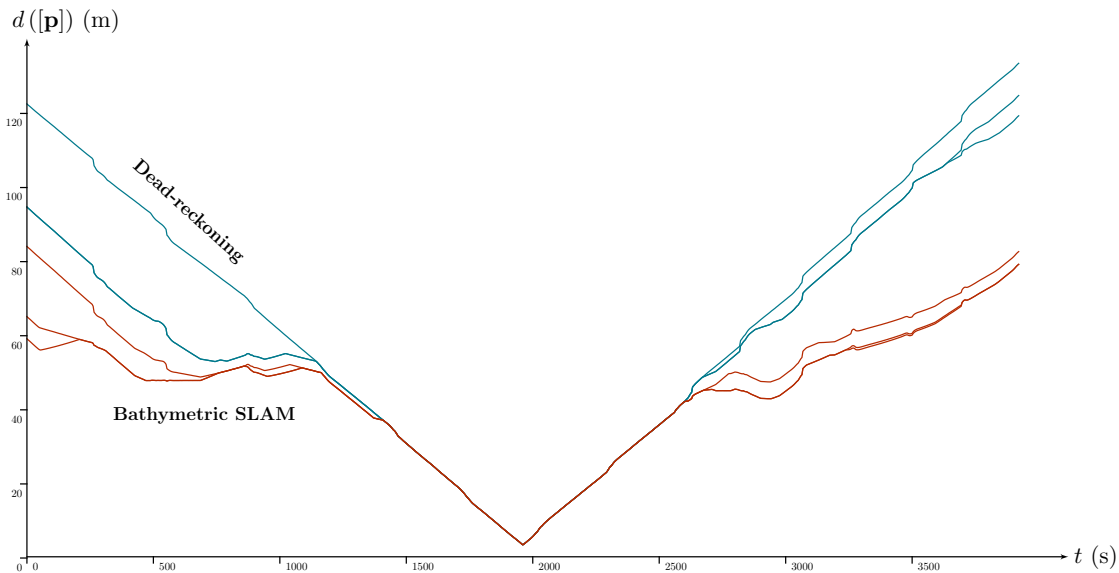


Figure 7.19: 19th October 2015: thickness graph. Constraints are propagated from a bounded state at $t = 1965s$.

7.3.4 Overview of the environment

Finally, we provide in Figure 7.20 a Digital Elevation Model (DEM) of the area covered by the robot during these two experiments. One should note that this map has not been used for the SLAM. It is however interesting to picture an overview of the feasible observations in order to understand the contractions over each loop. Unfortunately, this survey does not completely cover the 20th October's experiment.

Less convincing results have been obtained for the 19th October's experiment. This is probably due to the high number of loops in the 20th's trial, which weighs in favour of numerous constraints of interest for the localization. In addition, the DEM reveals a more homogeneous seafloor for the 19th's mission.

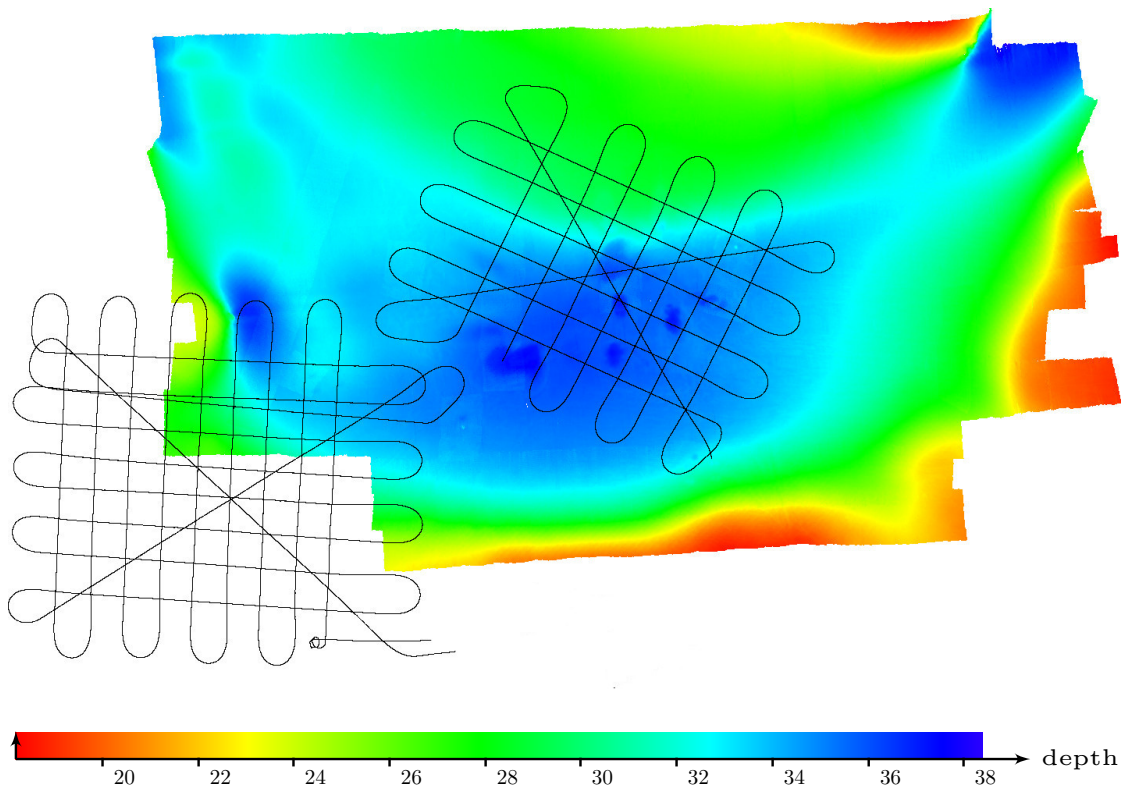


Figure 7.20: Digital Elevation Model (DEM) of the area obtained during another campaign by a vessel equipped with a multibeam echosounder and GNSS positioning. The trajectories of the presented experiments are drawn on the map in order to visualize the bathymetric measurements. *Credits: SHOM.*

7.4 Discussions

7.4.1 Relation to the state of the art

A considerable amount of work has been done to address the SLAM problem over the past decades. Numerous SLAM techniques have been studied and it would be impossible to provide an exhaustive list of methods in this chapter. We shall rather use a taxonomy given in [Thrun and Leonard, 2008] in order to distinguish the main approaches and place our contribution in this field.

- **full SLAM vs. online SLAM.** Online SLAM algorithms are filters that estimate the present robot location instead of the complete past trajectory. These methods are usually incremental. On the other hand, a full SLAM algorithm will identify antecedent states during a global resolution. Our approach has been studied in the context of full SLAM problems and its extension for online applications is a work in progress.
- **topological vs. metric.** Topological approaches build maps by defining relations between features; *e.g.* a point **a** is adjacent to **b**. This qualitative mapping is suited for navigation purposes. On the other hand, a metric SLAM will provide maps made of metric relations between the places, which is our case if we build a map from the state estimation that has been performed.
- **volumetric vs. feature-based.** Volumetric methods process a high resolution map, thus providing a realistic reconstruction of the environment. This usually implies heavy computations and high dimensional maps, contrary to feature-based methods that extract relevant information to be recognized along the exploration. In this latter case, the map amounts to a set of features which is an efficient way to summarize the information, at the risk of discarding useful data. From our point of view, the method proposed in this chapter is neither volumetric nor feature-based, but *temporal* in the sense that the relations between the items are entirely based on time references.
- **known vs. unknown correspondence.** When sensed items are identified, then correspondences can be searched. This is known as the data association problem. Other methods do not rely on the identification of landmarks and are classified as *unknown correspondence* approaches, which is the case of our SLAM method.

- **static vs. dynamic.** Dynamic methods assume that the environment may change over time, which is a complicated challenge. However in the literature, most of the methods are static, which is also the case of ours despite the use of evolution models such as Equation (7.29), page 223: any unpredictable change will disrupt the resolution.
- **small vs. large uncertainty.** Some approaches are efficient only in the case of small uncertainties in the location estimate. Conversely, other methods deal with large amounts of uncertainties which lead to the loop closure problem. Our method is comfortable in this situation, being able to detect and prove loop closings in the worst bounded-error contexts.
- **active vs. passive.** An active SLAM method will integrate the control of the robot into its own resolution process. In this way, the exploration can be more comprehensive and the localization easily refined by selecting the areas to visit again. Our approach is passive since we are purely observing datasets without acting upon them.
- **single-robot vs. multi-robot.** Multi-robot missions grew in popularity over the past years, providing new constraints of interest for the localization problem while multiplying the footprint of observation sensors. Our method involves a single robot and its extension to collaborative contexts will be the object of future work.
- **probabilistic vs. set-membership.** Lastly, our approach provides guaranteed results which is of high interest for safety purposes. This is a strong advantage of set-membership methods against probabilistic ones.

A synthesis of these categories is provided in Table 7.4.

7.4.2 About a Bayesian resolution

One could employ Bayesian methods to solve our temporal SLAM problem. However, we believe that set-membership tools are more suitable for this very case.

In particular, we may mention that tubes are infinite-dimensional spaces. Indeed, considering a trajectory defined over three times t_1, t_2, t_3 , then the corresponding space will be three-dimensional. In the continuous case, this space becomes of infinite dimension.

Table 7.4: Positioning of our approach among SLAM methods.

versus

full SLAM		online SLAM
topological		metric
volumetric	<i>temporal</i>	feature-based
known corresp.		unknown corresp.
static		dynamic
small uncertainty		large uncertainty
active		passive
single-robot		multi-robot
probabilistic		set-membership

Bayesian methods badly behave in case of large dimensional spaces. For instance, bisecting a box $[\mathbf{a}] \in \mathbb{IR}^{250}$ along each dimension will lead to 2^{250} possibilities¹⁰. These are all cases to test independently. Conversely, by working on bounds only, we believe that set-membership methods are more efficient in this situation.

In addition, our temporal approach relies on the implication constraint defined in Equation (7.11), page 205. It induces a strong dependency between proprioceptive and exteroceptive information, while Bayesian methods usually assume independent variables.

7.4.3 Biased sensors

We emphasize that this temporal approach is robust to biased observation sensors. An unknown – but constant – bias \mathbf{b} will have no impact on the relation $\mathbf{z}(t_1) = \mathbf{z}(t_2)$, which is equivalent to $\mathbf{z}(t_1) + \mathbf{b} = \mathbf{z}(t_2) + \mathbf{b}$. This is common when dealing with inter-temporal measurements, as the Borda’s double weighing does. In practice, the main advantage of this is the ability to apply the method with uncalibrated observation sensors. Naturally, this remark does not apply for evolution measurements.

¹⁰As an order of magnitude, 2^{250} is comparable to the number of atoms in the universe.

7.4.4 Fluctuating measurements

One can easily understand that the limits of our approach are reached in case of a constant observation over time, *i.e.* $\dot{\mathbf{z}}(\cdot) = \mathbf{0}$. Indeed, the kernel characterization of $[\mathbf{z}](\cdot)$ will be too uncertain to allow any loop set contraction. On the other hand, a signal presenting strong fluctuations will not efficiently contract a t -set $(\mathbb{T}_{\mathbf{p}})_i$. An effective contraction is presented in Figure 7.7, page 210, while this fluctuating effect is highlighted in Figure 7.21.

It is actually difficult to estimate the best properties expected for $\mathbf{z}(\cdot)$. A suitable fluctuation would be highly related to the temporal drift of the initial dead-reckoning method. In any case, though, any measurement is always welcome and the fusion of many p scalar properties is easily achievable with a single vector $\mathbf{z} \in \mathbb{R}^p$.

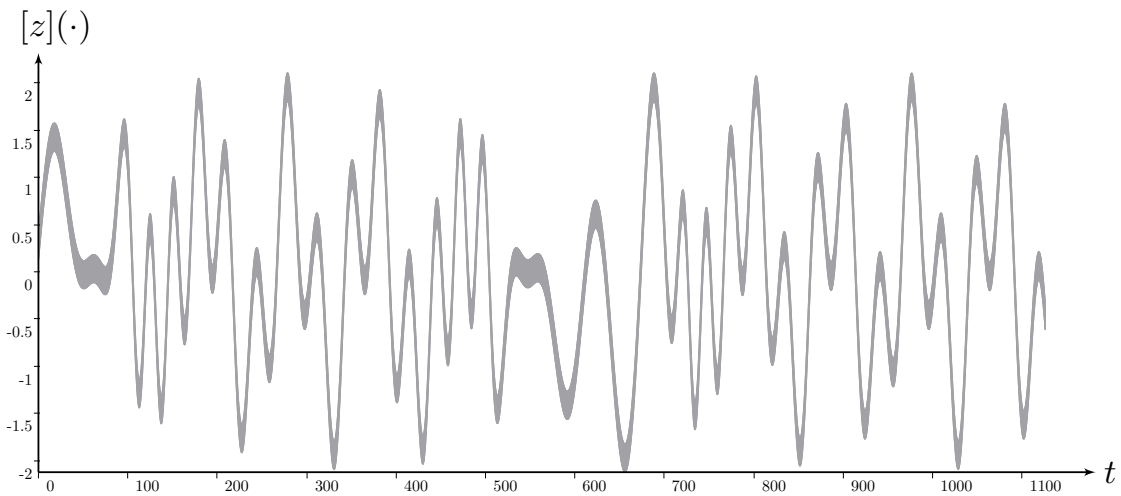
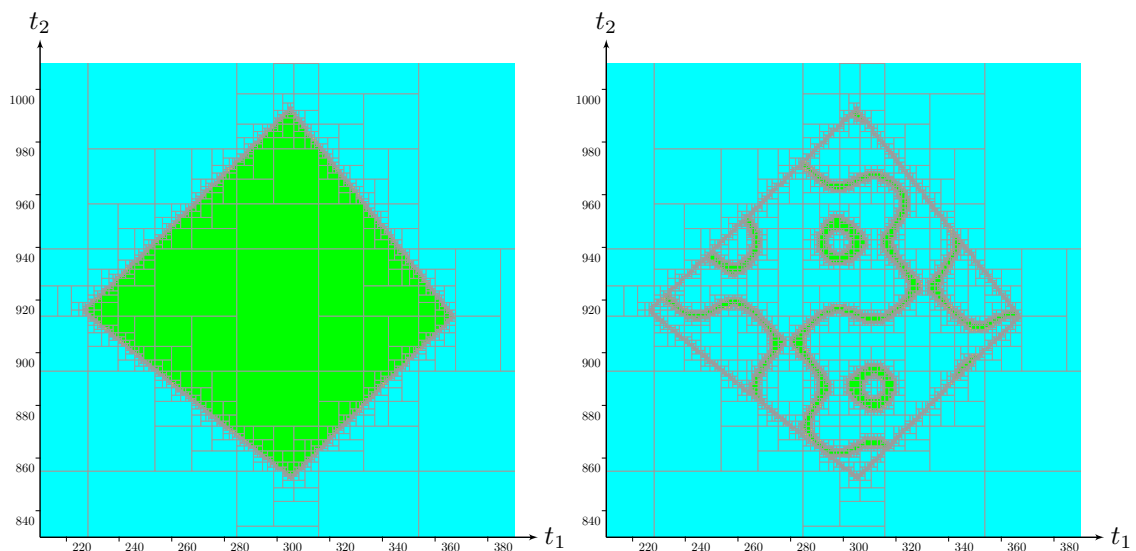
(a) Example of an observation tube $[z](\cdot)$ presenting lots of variations.(b) Approximation of a t -set $(\mathbb{T}_{\mathbf{p}})_i$ before and after the contraction based on Equation (7.19), page 210.

Figure 7.21: An example of inefficient temporal contractions in case of too fluctuating observations. The loop set is sparsely contracted which does not significantly reduce its envelope. Consequently, very few contractions on $[\mathbf{p}](\cdot)$ can then be expected.

7.5 Conclusion

Our fourth contribution was a new reliable SLAM method.

Robotic contribution

The main novelty lies in the temporal approach used for the resolution: time references become unknown variables to be estimated. This way, the localization process does not stand on a conventional map to be built, but on a set of temporal references to be approximated.

The method is illustrated over actual experiments involving the *Daurade* AUV. The proposed bathymetric SLAM, a topic that has been little studied in the case of single beam echosounders, reveals how the method behaves in case of strong positioning uncertainties and poor observation measurements. In addition, the approach stands on so-called inter-temporal measurements which allows the consideration of any kind of time-invariant observations, even when the analytical expression of the observation function is not known.

If we used bathymetric data in our illustrations, one could also employ other kinds of measurements. For instance, a terrestrial robot evolving in a dark room could successively sense the acoustic response of the place to a sound stimulus. Then the temporal SLAM method could refine the localization in case of a proven loop by comparing acoustic signals at different times.

Constraints contributions

The originality of this work was also to model a SLAM problem by a CN involving inter-temporal constraints and heterogeneous variables such as sets or trajectories. Using the tools provided in the previous chapters, we ensure the SLAM constraints to be fulfilled at any time during the process.

However, an operator for the specific *inter-temporal implication* constraint was still missing to achieve the resolution with a full contractor programming approach. This chapter introduced a new contractor $\mathcal{C}_{\mathbf{p} \Rightarrow \mathbf{z}}$ made as a composition of the tools provided in the previous parts of this document. In particular, the contractor $\mathcal{C}_{\text{eval}}$

presented in Chapter 5 revealed its full potential on this application, providing a way to couple trajectories with time variables in a bounded-error context. In addition, it is shown that $\mathcal{C}_{\mathbf{p} \Rightarrow \mathbf{z}}$ necessarily relies on a zero verification algorithm such as the topological degree test detailed in Chapter 6.

In sum, this SLAM problem was an opportunity to study the following elementary constraints:

1. Evolution constraint – Chapter 4

$$\mathcal{L}_{\frac{d}{dt}}(\mathbf{x}(\cdot), \mathbf{v}(\cdot)) : \dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot)$$

2. Evaluation constraint – Chapter 5

$$\mathcal{L}_{\text{eval}}(t, \mathbf{z}, \mathbf{p}(\cdot), \mathbf{w}(\cdot)) : \begin{cases} \mathbf{z} = \mathbf{p}(t) \\ \dot{\mathbf{p}}(\cdot) = \mathbf{w}(\cdot) \end{cases}$$

3. Inter-temporal evaluation constraint – Chapter 7

$$\mathcal{L}_{t_1, t_2}(t_1, t_2, \mathbf{p}(\cdot), \mathbf{w}(\cdot)) : \begin{cases} \mathbf{p}(t_1) = \mathbf{p}(t_2) \\ \dot{\mathbf{p}}(\cdot) = \mathbf{w}(\cdot) \end{cases}$$

4. Inter-temporal implication constraint – Chapters 6 and 7

$$\mathcal{L}_{\mathbf{p} \Rightarrow \mathbf{z}}(\mathbf{p}(\cdot), \mathbf{w}(\cdot), \mathbf{z}(\cdot)) : \begin{cases} \mathbf{p}(t_1) = \mathbf{p}(t_2) \implies \mathbf{z}(t_1) = \mathbf{z}(t_2) \\ \dot{\mathbf{p}}(\cdot) = \mathbf{w}(\cdot) \end{cases}$$

All of them have been ultimately involved in the algorithm `temporalSLAM` under the form of contractors.

Prospects

There is still a lot of work to be done on this SLAM topic. The first point to investigate is a meaningful comparison of our work with other SLAM approaches. If it may not be relevant to compare a set-membership method with a probabilistic one because of the heterogeneousness of the outcomes, we could however merge several methods and improve the state of the art in the probabilistic field.

About the bathymetric SLAM application, new experiments with more suited sensors such as a single beam echosounder would be welcomed. According to the availability of *Daurade* in the coming months, we could apply the method with a more complex evolution model based on an hybridization INS/DVL. This would allow thinner results in the trajectories estimations. Using an accurate bathymetric

sensor, we could then evaluate how the method is of interest to hold the special order of hydrographic surveys for a longer period. Further experiments involving smaller units such as the *Toutatis* AUVs, presented in the introduction of this document, or best known robots such as *Comet*¹¹, *Iver*¹² or *Remus 100*¹³ vehicles, would be interesting to state the usefulness of the approach for low-cost units and stronger measurement uncertainties.

We discussed in Section 7.4 that our method is passive: the algorithm does not output the control of the robot during its process. It would be interesting to study an active approach to decide where to move in order to improve the localization as efficiently as possible. Such strategy would induce relevant loops that could potentially raise valuable observation constraints. It is indeed interesting to provoke a loop over an heterogeneous part of the seabed instead of a flat seafloor area.

Another point is the mapping part of the method that has not been properly studied. Each measurement is referenced by a precise date t which spatially corresponds to a position box $[\mathbf{p}](t) \in \mathbb{I}\mathbb{R}^2$ that has been contracted during the process. In our application, several tools such as bathymetric interpolation methods could be integrated in order to build a complete map. A reliable mapping approximation can also be studied based on interval tools such as those presented in [Desrochers and Jaulin, 2017].

Finally, the constraint $\mathcal{L}_{\mathbf{p} \Rightarrow \mathbf{z}}(\mathbf{p}(\cdot), \mathbf{w}(\cdot), \mathbf{z}(\cdot))$ has been illustrated with an application involving 2D trajectories $\mathbf{p}(\cdot)$. The zero-verification algorithm requested for this constraint has to be scalable according to the dimension of $\mathbf{p}(\cdot)$. We provided a 2D implementation of the topological degree test in Chapter 6, but a higher dimensional test would be welcomed to apply $\mathcal{L}_{\mathbf{p} \Rightarrow \mathbf{z}}$ in the most generic way. This is currently a work in progress.

¹¹ *Comet* AUVs: <http://rtsys.eu/en/drones/comet>

¹² *Iver* AUVs: <http://www.iver-auv.com>

¹³ *Remus 100* AUVs: <https://www.km.kongsberg.com>

General conclusions and prospects

8.1 Conclusions

This thesis considers the localization of autonomous robots evolving in unstructured underwater environments. The common thread of this work is the use of temporal constraints and inter-temporal measurements raising new opportunities of localization, under-exploited so far. Besides, these robotic problems have motivated the study of academic research topics in the fields of interval analysis and constraint programming. Our goal was to develop new reliable tools to fit our needs. While their application on underwater experiments appears to be efficient, their use in other fields such as automatic and control, collision avoidance, path planning, terrestrial localization or spatial trajectory evaluations could also be considered.

Scope of the contributions

Our primary motivation is the localization of mobile robots in environments characterized by the paucity of relevant information. A SLAM approach would allow a concurrent localization of the vehicle and mapping of the area, without a prior knowledge on the environment nor the use of positioning systems. However, already existing approaches are not comfortable with poor measurements, unknown observation functions or strong positioning uncertainties. In addition, they do not provide guaranteed results that may be expected for safety reasons.

Our strategy consisted in taking a temporal approach to solve a spatial problem. Indeed, time references tie together state estimations and environment observations that are uncertain spatial values due to errors coming from the sensors. The matching of these values implies further uncertainties and we proposed to handle them in a bi-temporal space depicting inter-temporal configurations. This view differs from usual methods that only consider uncertainties in the spatial space without performing estimations of temporal references.

Chapter 7 detailed this temporal SLAM and provided illustrations on actual underwater experiments. Our choice was to address this problem with a constraint programming approach coupled with interval analysis. This decision was motivated by the simplicity and the genericity allowed by this paradigm. By depicting a problem with constraints and sets of feasible solutions, one can perfectly deal with complicated situations or poor datasets. Furthermore, the developed algorithms do not require sophisticated settings: in this document, only two parameters have been introduced: the time discretization of a tube δ and the precision ε of set-inversion algorithms (SIVIA).

To reach our goal, though, three primary constraints had to be studied. All of them apply to trajectories by means of new operators designed in purpose to reduce the domains of feasible dynamical solutions: so-called *tubes*. The first study of $\mathcal{L}_{\frac{d}{dt}} : \dot{x}(\cdot) = v(\cdot)$ has been the object of earlier work but a reliable *contractor* was still missing. Definitions and proofs of the new contractor $\mathcal{C}_{\frac{d}{dt}}$ were provided in Chapter 4 together with robotic applications. We also discussed the limits of the approach when it comes to overcome unwanted pessimism on the trajectories evaluations.

The second constraint of interest, $\mathcal{L}_{\text{eval}} : z = y(t)$, is fully investigated for the first time in this thesis. It allows the consideration of strong time uncertainties: a topic that has been poorly studied with set-membership methods and in the state estimation community. Chapter 5 presented $\mathcal{C}_{\text{eval}}$ which aims at considering any uncertainty about the evaluation of a trajectory at a given time. Its application on robotic problems, such as the correction of a drifting clock, is a first step towards new problem-solving techniques, allowing resolutions from a temporal aspect. Chapter 7 typically illustrated an original application made possible by this contractor.

Our SLAM problem has also triggered the development of a so-called inter-temporal implication constraint denoted $\mathcal{L}_{\mathbf{p} \Rightarrow \mathbf{z}} : \mathbf{p}(t_1) = \mathbf{p}(t_2) \implies \mathbf{z}(t_1) = \mathbf{z}(t_2)$. Its implementation required the study of a new zero verification test that has been successfully applied in robotics to prove the existence of loops along uncertain robot trajectories. Indeed, Chapter 6 demonstrated the efficiency of the topological degree theory when coupled with function evaluations in a bounded-error context. In our robotic applications, we discussed the optimality of the approach. These results strongly impact the effectiveness of the $\mathcal{C}_{\mathbf{p} \Rightarrow \mathbf{z}}$ contractor, that has then been detailed in Chapter 7 and illustrated in the context of a new bathymetric SLAM approach.

In a nutshell, this thesis is rooted in robotic problems which gradually leads to the development of new academic tools. It brought new advances in the field of constraint programming by proposing a declarative way to deal with dynamical systems. A reliable contractor programming framework is now at hand, allowing one to build solvers for dynamical systems. This set of tools has been illustrated along this document with realistic robotic applications.

8.2 Summary of the contributions

Papers

Chapters 4, 5, 6 and 7 are subject to publications in robotic journals. The last one is still a work in progress. We plan new experiments with accurate datasets in order to clearly assess the relevancy of our approach for underwater navigation.

The list of published or submitted papers of this thesis is summarized below:

- Rohou, S., Jaulin, L., Mihaylova, L., Le Bars, F., and Veres, S. M. (2017). Guaranteed computation of robot trajectories. *Robotics and Autonomous Systems*, 93:76–84
- Rohou, S., Jaulin, L., Mihaylova, L., Le Bars, F., and Veres, S. M. (2018b). Reliable non-linear state estimation involving time uncertainties. *Automatica*, 93:379–388
- Rohou, S., Franek, P., Aubry, C., and Jaulin, L. (2018a). Proving the existence of loops in robot trajectories. *Submitted to the International Journal of Robotics Research*

Open-source library

A significant contribution of this work is the development by the author of the new open-source library *Tubex*, freely available at:

- <http://www.simon-rohou.fr/research/tubex-lib>

This project gathers all the elementary tools presented in this document. The reader will be able to process the simulated examples and build its own solvers for the resolution of more dedicated dynamical problems.

8.3 Overall prospects

From now on, the reader should be able to address problems related to dynamical systems by breaking them down into a set of elementary constraints involving vectors, trajectories or sets. Then he could appropriately define initial domains of variables and use the presented contractors to approximate solution sets. However, build such solver requires technical skills to efficiently handle intervals, tubes, contractors and related settings. This knowledge is not at hand for many users who could benefit from this constraint programming approach. There is then a need to provide an extra abstraction level in the design of solvers.

The continuation of this thesis will be to design a dedicated language to define problems. This would imply syntax definitions and an appropriate semantic to maintain the ability to deal with a wide range of problems. A close link must be investigated between this language and its automatic implementation under the form of contractors. For instance, tubes should be automatically instantiated with an appropriate time discretization δ . Similarly, while we explained that the order of contractor calls has no impact on the final result, a smart scheduling could be processed in order to speed up the computations.

In addition, this would also be a good opportunity to push the limits of our approach by coupling it with further tools. One could merge several methods of guaranteed integration, such as those presented in the introduction of Chapter 4, which would be helpful to avoid some overestimation of trajectories sets. Furthermore, the complicated task of managing hybrid constraints with our approach could be dealt by merging it with some *Eulerian* approaches such as [Le Mézo et al., 2018].

All these optimizations can be hidden from user's view who would only focus on its problem definition. This opening is the subject of the new *Contredo* consortium¹ that will gather several academic and industrial partners on this topic over the next three years.

¹From the [French National Research Agency \(ANR\)](#). ANR Programme: (DS0702) 2016. Project ID: ANR-16-CE33-0024. Project coordinator: Pr. Gilles Trombettoni.

Bibliography

- [Abdallah et al., 2008] Abdallah, F., Gning, A., and Bonnifait, P. (2008). Box particle filtering for nonlinear state estimation using interval analysis. *Automatica*, 44(3):807–815.
- [Alexandre dit Sandretto and Chapoutot, 2016] Alexandre dit Sandretto, J. and Chapoutot, A. (2016). Validated Explicit and Implicit Runge-Kutta Methods. *Reliable Computing electronic edition*, 22.
- [Alexandre dit Sandretto et al., 2014] Alexandre dit Sandretto, J., Trombettoni, G., Daney, D., and Chabert, G. (2014). Certified Calibration of a Cable-Driven Robot Using Interval Contractor Programming. In Thomas, F. and Perez Gracia, A., editors, *Computational Kinematics: Proceedings of the 6th International Workshop on Computational Kinematics (CK2013)*, pages 209–217. Springer Netherlands, Dordrecht.
- [Angeli et al., 2008] Angeli, A., Filliat, D., Doncieux, S., and Meyer, J.-A. (2008). Fast and Incremental Method for Loop-Closure Detection Using Bags of Visual Words. *IEEE Transactions on Robotics*, 24(5):1027–1037.
- [Apt, 1999] Apt, K. R. (1999). The essence of constraint propagation. *Theoretical Computer Science*, 221(1):179–210.
- [Araya et al., 2008] Araya, I., Neveu, B., and Trombettoni, G. (2008). Exploiting Common Subexpressions in Numerical CSPs. In Stuckey, P. J., editor, *Principles and Practice of Constraint Programming: 14th International Conference, CP 2008, Sydney, Australia, September 14-18, 2008. Proceedings*, pages 342–357. Springer Berlin Heidelberg, Berlin, Heidelberg.
- [Araya et al., 2012] Araya, I., Trombettoni, G., and Neveu, B. (2012). A Contractor Based on Convex Interval Taylor. In Beldiceanu, N., Jussien, N., and Pinson, É., editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems: 9th International Conference, CPAIOR 2012, Nantes, France, May 28 - June 1, 2012. Proceedings*, pages 1–16. Springer Berlin Heidelberg, Berlin, Heidelberg.
- [Aubry et al., 2013] Aubry, C., Desmare, R., and Jaulin, L. (2013). Loop detection of mobile robots using interval analysis. *Automatica*, 49(2):463–470.
- [Aubry et al., 2014] Aubry, C., Desmare, R., and Jaulin, L. (2014). Kernel Characterization of an Interval Function. *Mathematics in Computer Science*, 8(3):379–390.

- [Bahr et al., 2009] Bahr, A., Leonard, J. J., and Fallon, M. F. (2009). Cooperative Localization for Autonomous Underwater Vehicles. *The International Journal of Robotics Research*, 28(6):714–728.
- [Bailey and Durrant-Whyte, 2006] Bailey, T. and Durrant-Whyte, H. (2006). Simultaneous localization and mapping (SLAM): part II. *IEEE Robotics & Automation Magazine*, 13(3):108–117.
- [Barkby, 2011] Barkby, S. (2011). *Efficient and Featureless Approaches to Bathymetric Simultaneous Localisation and Mapping*. PhD thesis, The University of Sydney.
- [Barkby et al., 2009] Barkby, S., Williams, S., Pizarro, O., and Jakuba, M. (2009). An efficient approach to bathymetric SLAM. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 219–224.
- [Benhamou and Older, 1997] Benhamou, F. and Older, W. J. (1997). Applying interval arithmetic to real, integer, and boolean constraints. *The Journal of Logic Programming*, 32(1):1–24.
- [Benhamou and Touraïvane, 1995] Benhamou, F. and Touraïvane, T. (1995). Prolog IV: langage et algorithmes. In *JFPLC*, pages 51–64.
- [Benjamin et al., 2010] Benjamin, M. R., Schmidt, H., Newman, P. M., and Leonard, J. J. (2010). Nested autonomy for unmanned marine vehicles with MOOS-IvP. *Journal of Field Robotics*, 27(6):834–875.
- [Berz, 1996] Berz, M., editor (1996). *Computational differentiation: techniques, applications, and tools*. Society for Industrial and Applied Mathematics, Philadelphia.
- [Bessiere, 2006] Bessiere, C. (2006). Constraint Propagation. In Rossi, F., van Beek, P., and Walsh, T., editors, *Foundations of Artificial Intelligence*, volume 2 of *Handbook of Constraint Programming*, pages 29–83. Elsevier.
- [Bethencourt and Jaulin, 2013] Bethencourt, A. and Jaulin, L. (2013). Cooperative localization of underwater robots with unsynchronized clocks. *Paladyn, Journal of Behavioral Robotics*, 4(4).
- [Bethencourt and Jaulin, 2014] Bethencourt, A. and Jaulin, L. (2014). Solving Non-Linear Constraint Satisfaction Problems Involving Time-Dependant Functions. *Mathematics in Computer Science*, 8(3):503–523.

-
- [Bichucher et al., 2015] Bichucher, V., Walls, J. M., Ozog, P., Skinner, K. A., and Eustice, R. M. (2015). Bathymetric factor graph SLAM with sparse point cloud alignment. In *IEEE OCEANS'15 conference*, pages 1–7. IEEE.
- [Borsuk, 1933] Borsuk, K. (1933). Drei Sätze über die n-dimensionale euklidische Sphäre. *Fundamenta Mathematicae*, 20(1):177–190.
- [Bouron, 2002] Bouron, P. (2002). *Méthodes ensemblistes pour le diagnostic, l'estimation d'état et la fusion de données temporelles*. PhD thesis, Compiègne.
- [Boyer et al., 2015] Boyer, F., Lebastard, V., Chevallereau, C., Mintchev, S., and Stefanini, C. (2015). Underwater navigation based on passive electric sense: New perspectives for underwater docking. *The International Journal of Robotics Research*, 34(9):1228–1250.
- [Caiti et al., 2005] Caiti, A., Garulli, A., Livide, F., and Prattichizzo, D. (2005). Localization of Autonomous Underwater Vehicles by Floating Acoustic Buoys: A Set-Membership Approach. *IEEE Journal of Oceanic Engineering*, 30(1):140–152.
- [Carbonnel et al., 2014] Carbonnel, C., Trombettoni, G., Vismara, P., and Chabert, G. (2014). Q-intersection Algorithms for Constraint-Based Robust Parameter Estimation. In *AAAI Conference on Artificial Intelligence, AAAI'14 - Twenty-Eighth Conference on Artificial Intelligence*, pages 2630–2636, Québec City, Canada.
- [Ceberio and Granvilliers, 2002] Ceberio, M. and Granvilliers, L. (2002). Horner's Rule for Interval Evaluation Revisited. *Computing*, 69(1):51–81.
- [Cerone, 1996] Cerone, V. (1996). Errors-in-variables models in parameter bounding. In *Bounding approaches to system identification*, pages 289–306. Springer.
- [Chabert, 2017] Chabert, G. (2017). IBEX, a C++ library for constraint processing over real numbers. <http://www.ibex-lib.org>.
- [Chabert and Jaulin, 2009] Chabert, G. and Jaulin, L. (2009). Contractor programming. *Artificial Intelligence*, 173(11):1079–1100.
- [Chablat et al., 2002] Chablat, D., Wenger, P., and Merlet, J. (2002). Workspace Analysis of the Orthoglide Using Interval Analysis. In *Advances in Robot Kinematics*, pages 397–406. Springer, Dordrecht.
- [Chailloux et al., 2011] Chailloux, C., Le Caillec, J.-M., Gueriot, D., and Zerr, B. (2011). Intensity-Based Block Matching Algorithm for Mosaicing Sonar Images. *IEEE Journal of Oceanic Engineering*, 36(4):627–645.

- [Choi et al., 2009] Choi, M., Choi, J., Park, J., and Chung, W. K. (2009). State estimation with delayed measurements considering uncertainty of time delay. pages 3987–3992.
- [Cleary, 1987] Cleary, J. G. (1987). Logical arithmetic. *Future Computing Systems*, 2(2):125–149.
- [Clemente et al., 2007] Clemente, L. A., Davison, A. J., Reid, I. D., Neira, J., and Tardós, J. D. (2007). Mapping Large Loops with a Single Hand-Held Camera. In *Robotics: Science and Systems*, volume 2.
- [Collins and Goldsztejn, 2008] Collins, P. and Goldsztejn, A. (2008). The Reach-and-Evolve Algorithm for Reachability Analysis of Nonlinear Dynamical Systems. *Electronic Notes in Theoretical Computer Science*, 223(Supplement C):87–102.
- [Combastel, 2005] Combastel, C. (2005). A State Bounding Observer for Uncertain Non-linear Continuous-time Systems based on Zonotopes. pages 7228–7234.
- [Creuze, 2014] Creuze, V. (2014). Robots marins et sous-marins. Perception, modélisation, commande. *Techniques de l’Ingenieur*, Collection Robotique, base documentaire : TIB398DUO:article : s7783.
- [Cruz and Barahona, 2003] Cruz, J. and Barahona, P. (2003). Constraint Satisfaction Differential Problems. In Rossi, F., editor, *Principles and Practice of Constraint Programming - CP 2003: 9th International Conference, CP 2003, Kinsale, Ireland, September 29 - October 3, 2003. Proceedings*, pages 259–273. Springer Berlin Heidelberg, Berlin, Heidelberg.
- [Cummins and Newman, 2008] Cummins, M. and Newman, P. (2008). FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance. *Int. J. Rob. Res.*, 27(6):647–665.
- [De Freitas et al., 2016] De Freitas, A., Mihaylova, L., Gning, A., Angelova, D., and Kadiramanathan, V. (2016). Autonomous crowds tracking with box particle filtering and convolution particle filtering. *Automatica*, 69:380–394.
- [Desrochers and Jaulin, 2016] Desrochers, B. and Jaulin, L. (2016). A minimal contractor for the polar equation: Application to robot localization. *Engineering Applications of Artificial Intelligence*, 55(Supplement C):83–92.
- [Desrochers and Jaulin, 2017] Desrochers, B. and Jaulin, L. (2017). Computing a Guaranteed Approximation of the Zone Explored by a Robot. *IEEE Transactions on Automatic Control*, 62(1):425–430.

-
- [Deville et al., 1998] Deville, Y., Janssen, M., and Van Hentenryck, P. (1998). Consistency Techniques in Ordinary Differential Equations. In Maher, M. and Puget, J.-F., editors, *Principles and Practice of Constraint Programming - CP98: 4th International Conference, CP98 Pisa, Italy, October 26-30, 1998 Proceedings*, pages 162–176. Springer Berlin Heidelberg, Berlin, Heidelberg.
- [Di Marco et al., 2001] Di Marco, M., Garulli, A., Lacroix, S., and Vicino, A. (2001). Set membership localization and mapping for autonomous navigation. *International Journal of Robust and Nonlinear Control*, 11(7):709–734.
- [Dillon, 2016] Dillon, J. (2016). Aided Inertial Navigation in GPS-denied Environments Using Synthetic Aperture Processing. Technical report, NRC-CNRC.
- [Drevelle, 2011] Drevelle, V. (2011). *Study of robust set estimation methods for a high integrity multi-sensor localization. Application to navigation in urban areas*. Theses, Université de Technologie de Compiègne.
- [Drevelle and Bonnifait, 2009] Drevelle, V. and Bonnifait, P. (2009). High integrity GNSS location zone characterization using interval analysis. In *ION GNSS 2009*, pages 2178–2187, Savannah, GA, United States.
- [Drevelle and Nicola, 2014] Drevelle, V. and Nicola, J. (2014). VIBes: A Visualizer for Intervals and Boxes. *Mathematics in Computer Science*, 8(3):563–572.
- [Dubins, 1957] Dubins, L. E. (1957). On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents. *American Journal of Mathematics*, 79(3):497.
- [Duracz, 2016] Duracz, A. (2016). *Rigorous Simulation : Its Theory and Applications*. PhD thesis, Halmstad University, Centre for Research on Embedded Systems (CERES).
- [Durrant-Whyte and Bailey, 2006] Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localization and mapping: part I. *IEEE Robotics & Automation Magazine*, 13(2):99–110.
- [Filippova et al., 1996] Filippova, T. F., Kurzhanski, A. B., Sugimoto, K., and Vályi, I. (1996). Ellipsoidal State Estimation for Uncertain Dynamical Systems. In Milanese, M., Norton, J., Piet-Lahanier, H., and Walter, É., editors, *Bounding Approaches to System Identification*, pages 213–238. Springer US, Boston, MA.
- [Fonseca and Gangbo, 1995] Fonseca, I. and Gangbo, W. (1995). *Degree theory in analysis and applications*. Number 2 in Oxford lecture series in mathematics and its applications. Clarendon Press ; Oxford University Press, Oxford : New York.

- [Fossen, 1994] Fossen, T. I. (1994). *Guidance and control of ocean vehicles*. Wiley, Chichester ; New York.
- [Franek and Ratschan, 2014] Franek, P. and Ratschan, S. (2014). Effective topological degree computation based on interval arithmetic. *Mathematics of Computation*, 84(293):1265–1290.
- [Franek et al., 2016] Franek, P., Ratschan, S., and Zgliczynski, P. (2016). Quasi-decidability of a Fragment of the First-Order Theory of Real Numbers. *Journal of Automated Reasoning*, 57(2):157–185.
- [Furi et al., 2010] Furi, M., Pera, M., and Spadini, M. (2010). A Set of Axioms for the Degree of a Tangent Vector Field on Differentiable Manifolds. *Fixed Point Theory and Applications*, 2010(1):845631.
- [Gning and Bonnifait, 2006] Gning, A. and Bonnifait, P. (2006). Constraints propagation techniques on intervals for a guaranteed localization using redundant data. *Automatica*, 42(7):1167–1175.
- [Gning et al., 2013] Gning, A., Ristic, B., Mihaylova, L., and Abdallah, F. (2013). An Introduction to Box Particle Filtering [Lecture Notes]. *IEEE Signal Processing Magazine*, 30(4):166–171.
- [Goldsztejn, 2006] Goldsztejn, A. (2006). A branch and prune algorithm for the approximation of non-linear AE-solution sets. page 1650. ACM Press.
- [Goldsztejn et al., 2011] Goldsztejn, A., Hayes, W., and Collins, P. (2011). Tinkerbell Is Chaotic. *SIAM Journal on Applied Dynamical Systems*, 10(4):1480–1501.
- [Goubault et al., 2014] Goubault, E., Mullier, O., Putot, S., and Kieffer, M. (2014). Inner Approximated Reachability Analysis. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control*, HSCC '14, pages 163–172, New York, NY, USA. ACM.
- [Hairer et al., 1993] Hairer, E., Nørsett, S. P., and Wanner, G. (1993). *Solving Ordinary Differential Equations I (2Nd Revised. Ed.): Nonstiff Problems*. Springer-Verlag New York, Inc., New York, NY, USA.
- [Hickey, 2000] Hickey, T. J. (2000). Analytic constraint solving and interval arithmetic. pages 338–351. ACM Press.
- [IHO, 2008] IHO (2008). IHO Standards for Hydrographic Surveys. Technical Report 44, International Hydrographic Organization.

- [Janssen et al., 2001] Janssen, M., Van Hentenryck, P., and Deville, Y. (2001). Optimal Pruning in Parametric Differential Equations. In Walsh, T., editor, *Principles and Practice of Constraint Programming - CP 2001: 7th International Conference, CP 2001 Paphos, Cyprus, November 26 - December 1, 2001 Proceedings*, pages 539–553. Springer Berlin Heidelberg, Berlin, Heidelberg.
- [Janssen et al., 2002] Janssen, M., Van Hentenryck, P., and Deville, Y. (2002). A Constraint Satisfaction Approach for Enclosing Solutions to Parametric Ordinary Differential Equations. *SIAM Journal on Numerical Analysis*, 40(5):1896–1939.
- [Jaulin, 2002] Jaulin, L. (2002). Nonlinear bounded-error state estimation of continuous-time systems. *Automatica*, 38(6):1079–1082.
- [Jaulin, 2009] Jaulin, L. (2009). Robust set-membership state estimation; application to underwater robotics. *Automatica*, 45(1):202–206.
- [Jaulin, 2011] Jaulin, L. (2011). Range-Only SLAM With Occupancy Maps: A Set-Membership Approach. *IEEE Transactions on Robotics*, 27(5):1004–1010.
- [Jaulin, 2015a] Jaulin, L. (2015a). *Mobile robotics*. OCLC: 986557752.
- [Jaulin, 2015b] Jaulin, L. (2015b). Pure range-only SLAM with indistinguishable landmarks; a constraint programming approach. *Constraints*, pages 1–20.
- [Jaulin et al., 2001] Jaulin, L., Kieffer, M., Didrit, O., and Walter, É. (2001). *Applied Interval Analysis*. Springer London, London.
- [Jaulin and Walter, 1993a] Jaulin, L. and Walter, É. (1993a). Guaranteed nonlinear parameter estimation via interval computations. *Interval Computation*, 3:61–75.
- [Jaulin and Walter, 1993b] Jaulin, L. and Walter, É. (1993b). Set inversion via interval analysis for nonlinear bounded-error estimation. *Automatica*, 29(4):1053–1064.
- [Jensen et al., 2011] Jensen, F. B., Kuperman, W. A., Porter, M. B., and Schmidt, H. (2011). *Computational Ocean Acoustics*. Springer New York, New York, NY.
- [Kalman, 1960] Kalman, R. E. (1960). Contributions to the theory of optimal control. *Boletín de la Sociedad Matemática Mexicana*, 5:102–119.
- [Konečný et al., 2016] Konečný, M., Taha, W., Bartha, F. A., Duracz, J., Duracz, A., and Ames, A. D. (2016). Enclosing the behavior of a hybrid automaton up to and beyond a Zeno point. *Nonlinear Analysis: Hybrid Systems*, 20(Supplement C):1 – 20.

- [Kurzanski and Filippova, 1993] Kurzanski, A. B. and Filippova, T. F. (1993). On the Theory of Trajectory Tubes - A Mathematical Formalism for Uncertain Dynamics, Viability and Control. In Kurzanski, A. B., editor, *Advances in Nonlinear Dynamics and Control: A Report from Russia*, pages 122–188. Birkhäuser Boston, Boston, MA.
- [Lasbouygues et al., 2014] Lasbouygues, A., Lapierre, L., Andreu, D., Hermoso, J. L., Jourde, H., and Ropars, B. (2014). Stable and reactive centering in conduits for karstic exploration. pages 2986–2991. IEEE.
- [Le Bars et al., 2012] Le Bars, F., Sliwka, J., Jaulin, L., and Reynet, O. (2012). Set-membership state estimation with fleeting data. *Automatica*, 48(2):381–387.
- [Le Gallo, 2016] Le Gallo, M. (2016). *Le grand livre des motifs bretons et celtiques: méthode de construction*. Coop Breizh, Spézet. OCLC: 981935364.
- [Le Mézo et al., 2018] Le Mézo, T., Jaulin, L., and Zerr, B. (2018). Bracketing the solutions of an ordinary differential equation with uncertain initial conditions. *Applied Mathematics and Computation*, 318:70–79.
- [Lebastard et al., 2013] Lebastard, V., Chevallereau, C., Girin, A., Servagent, N., Gossiaux, P.-B., and Boyer, F. (2013). Environment reconstruction and navigation with electric sense based on a Kalman filter. *The International Journal of Robotics Research*, 32(2):172–188.
- [Leblond et al., 2005] Leblond, I., Legris, M., and Solaiman, B. (2005). Use of classification and segmentation of sidescan sonar images for long term registration. In *IEEE OCEANS'05 conference*, pages 322–327 Vol. 1. IEEE.
- [Lemaire et al., 2007] Lemaire, T., Berger, C., Jung, I.-K., and Lacroix, S. (2007). Vision-Based SLAM: Stereo and Monocular Approaches. *International Journal of Computer Vision*, 74(3):343–364.
- [Leonard et al., 1998] Leonard, J. J., Bennett, A. A., Smith, C. M., Jacob, H., and Feder, S. (1998). Autonomous Underwater Vehicle Navigation. In *MIT Marine Robotics Laboratory Technical Memorandum*.
- [Leonard and Durrant-Whyte, 1991] Leonard, J. J. and Durrant-Whyte, H. F. (1991). Simultaneous map building and localization for an autonomous mobile robot. In *IEEE/RSJ International Workshop on Intelligent Robots and Systems '91. 'Intelligence for Mechanical Systems, Proceedings IROS '91*, pages 1442–1447 vol.3.

-
- [L'Hour and Creuze, 2016] L'Hour, M. and Creuze, V. (2016). French Archaeology's Long March to the Deep - The Lune Project: Building the Underwater Archaeology of the Future. In Hsieh, M. A., Khatib, O., and Kumar, V., editors, *Experimental Robotics*, volume 109, pages 911–927. Springer International Publishing, Cham.
- [Mackworth, 1977] Mackworth, A. K. (1977). Consistency in Networks of Relations. *Artif. Intell.*, 8(1):99–118.
- [Maksarov and Norton, 1996] Maksarov, D. G. and Norton, J. P. (1996). State bounding with ellipsoidal set description of the uncertainty. *International Journal of Control*, 65(5):847–866.
- [Matsuda et al., 2012] Matsuda, T., Maki, T., Sakamaki, T., and Ura, T. (2012). Performance Analysis on a Navigation Method of Multiple AUVs for Wide Area Survey. *Marine Technology Society Journal*, 46(2):45–55.
- [Matsuda et al., 2015] Matsuda, T., Maki, T., Sato, Y., and Sakamaki, T. (2015). Performance verification of the alternating landmark navigation by multiple AUVs through sea experiments. In *IEEE OCEANS'15 conference*, pages 1–9. IEEE.
- [Merlet, 2004] Merlet, J. P. (2004). Solving the Forward Kinematics of a Gough-Type Parallel Manipulator with Interval Analysis. *The International Journal of Robotics Research*, 23(3):221–235.
- [Milne, 1983] Milne, P. H. (1983). *Underwater acoustic positioning systems*. Gulf Publishing Company, Houston.
- [Milnor, 1997] Milnor, J. W. (1997). *Topology from the differentiable viewpoint*. Princeton landmarks in mathematics. Princeton University Press, Princeton, N.J, rev. ed edition.
- [Monnet et al., 2016] Monnet, D., Ninin, J., and Jaulin, L. (2016). Computing an Inner and an Outer Approximation of the Viability Kernel. *Reliable Computing*, 22.
- [Montemerlo et al., 2003] Montemerlo, M., Thrun, S., Roller, D., and Wegbreit, B. (2003). FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping That Provably Converges. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence, IJCAI'03*, pages 1151–1156, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

- [Moore, 1966] Moore, R. (1966). *Interval analysis*. Prentice-Hall series in automatic computation. Prentice-Hall.
- [Moore, 1979] Moore, R. (1979). *Methods and Applications of Interval Analysis*. Studies in Applied and Numerical Mathematics. Society for Industrial and Applied Mathematics.
- [Moore, 1977] Moore, R. E. (1977). A Test for Existence of Solutions to Nonlinear Systems. *SIAM Journal on Numerical Analysis*, 14(4):611–615.
- [Moore and Kioustelidis, 1980] Moore, R. E. and Kioustelidis, J. B. (1980). A Simple Test for Accuracy of Approximate Solutions to Nonlinear (or Linear) Systems. *SIAM Journal on Numerical Analysis*, 17(4):521–529.
- [Moore and Yang, 1959] Moore, R. E. and Yang, C. (1959). Interval analysis I. *Technical Document LMSD-285875, Lockheed Missiles and Space Division, Sunnyvale, CA, USA*.
- [Morel et al., 2016] Morel, Y., Lebastard, V., and Boyer, F. (2016). Neural-based underwater surface localization through electrolocation. pages 2596–2603. IEEE.
- [Munk et al., 1994] Munk, W. H., Spindel, R. C., Baggeroer, A., and Birdsall, T. G. (1994). The Heard Island Feasibility Test. *The Journal of the Acoustical Society of America*, 96(4):2330–2342.
- [Nedialkov and Jackson, 2000] Nedialkov, N. S. and Jackson, K. R. (2000). ODE Software that Computes Guaranteed Bounds on the Solution. In *Advances in Software Tools for Scientific Computing*, Lecture Notes in Computational Science and Engineering, pages 197–224. Springer, Berlin, Heidelberg.
- [Nedialkov et al., 1999] Nedialkov, N. S., Jackson, K. R., and Corliss, G. F. (1999). Validated Solutions of Initial Value Problems for Ordinary Differential Equations. *Applied Mathematics and Computation*, 105(1):21–68.
- [Nehmeier and von Gudenberg, 2011] Nehmeier, M. and von Gudenberg, J. W. (2011). `filib++`, Expression Templates and the Coming Interval Standard. *Reliable Computing*, 15(4):312–320.
- [Neuland et al., 2014] Neuland, R., Nicola, J., Maffei, R., Jaulin, L., Prestes, E., and Kolberg, M. (2014). Hybridization of Monte Carlo and set-membership methods for the global localization of underwater robots. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 199–204.

-
- [Newman and Leonard, 2003] Newman, P. and Leonard, J. (2003). Pure range-only sub-sea SLAM. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1921–1926.
- [Norton and Veres, 1993] Norton, J. P. and Veres, S. M. (1993). Outliers in bound-based state estimation and identification. In *1993 IEEE International Symposium on Circuits and Systems*, volume 1, pages 790–793.
- [O’Regan et al., 2006] O’Regan, D., Cho, Y. J., and Chen, Y. Q. (2006). *Topological degree theory and applications*. Number v. 10 in Series in mathematical analysis and applications. Chapman & Hall/CRC, Boca Raton, FL. OCLC: ocm64592216.
- [Palomer et al., 2016] Palomer, A., Ridao, P., and Ribas, D. (2016). Multibeam 3d Underwater SLAM with Probabilistic Registration. *Sensors*, 16(4):560.
- [Papoulis and Pillai, 2002] Papoulis, A. and Pillai, S. (2002). *Probability, random variables, and stochastic processes*. McGraw-Hill electrical and electronic engineering series. McGraw-Hill.
- [Paull et al., 2014] Paull, L., Seto, M., and Leonard, J. J. (2014). Decentralized cooperative trajectory estimation for autonomous underwater vehicles. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 184–191.
- [Pennec, 2010] Penneç, S. (2010). *Amélioration de la précision des systèmes de positionnement à base ultra-courte en acoustique sous-marine*. PhD dissertation, Université de Bretagne Occidentale, Brest, France.
- [Piazzi and Visioli, 1997] Piazzi, A. and Visioli, A. (1997). A global optimization approach to trajectory planning for industrial robots. In *IROS’97*, volume 3, pages 1553–1559 vol.3.
- [Picard et al., 2017] Picard, K., Brooke, B., and Coffin, M. (2017). Geological Insights from Malaysia Airlines Flight MH370 Search.
- [Pronzato and Walter, 1996] Pronzato, L. and Walter, É. (1996). Robustness to Outliers of Bounded-Error Estimators and Consequences on Experiment Design. In *Bounding Approaches to System Identification*, pages 199–212. Springer, Boston, MA.
- [Pruski and Rohmer, 1997] Pruski, A. and Rohmer, S. (1997). Robust Path Planning for Non-Holonomic Robots. *Journal of Intelligent and Robotic Systems*, 18(4):329–350.

- [Quidu et al., 2007] Quidu, I., Hétet, A., Dupas, Y., and Lefèvre, S. (2007). AUV (REDERMOR) obstacle detection and avoidance experimental evaluation. In *IEEE OCEANS'07 conference*, pages 1–6, Aberdeen (Scotland), United Kingdom.
- [Raïssi et al., 2004] Raïssi, T., Ramdani, N., and Candau, Y. (2004). Set membership state and parameter estimation for systems described by nonlinear differential equations. *Automatica*, 40(10):1771–1777.
- [Ramdani and Nedialkov, 2011] Ramdani, N. and Nedialkov, N. S. (2011). Computing reachable sets for uncertain nonlinear hybrid systems using interval constraint-propagation techniques. *Nonlinear Analysis: Hybrid Systems*, 5(2):149–162.
- [Revol et al., 2005] Revol, N., Makino, K., and Berz, M. (2005). Taylor models and floating-point arithmetic: proof that arithmetic operations are validated in COSY. *The Journal of Logic and Algebraic Programming*, 64(1):135–154.
- [Rohou et al., 2018a] Rohou, S., Franek, P., Aubry, C., and Jaulin, L. (2018a). Proving the existence of loops in robot trajectories. *Submitted to the International Journal of Robotics Research*.
- [Rohou et al., 2017] Rohou, S., Jaulin, L., Mihaylova, L., Le Bars, F., and Veres, S. M. (2017). Guaranteed computation of robot trajectories. *Robotics and Autonomous Systems*, 93:76–84.
- [Rohou et al., 2018b] Rohou, S., Jaulin, L., Mihaylova, L., Le Bars, F., and Veres, S. M. (2018b). Reliable non-linear state estimation involving time uncertainties. *Automatica*, 93:379–388.
- [Rokityanskiy and Veres, 2005] Rokityanskiy, D. Y. and Veres, S. M. (2005). Application of ellipsoidal estimation to satellite control design. *Mathematical and Computer Modelling of Dynamical Systems*, 11(2):239–249.
- [Rump, 1988] Rump, S. M. (1988). Reliability in Computing: The Role of Interval Methods in Scientific Computing. pages 109–126. Academic Press Professional, Inc., San Diego, CA, USA.
- [Sam-Haroud and Faltings, 1996] Sam-Haroud, D. and Faltings, B. (1996). Consistency techniques for continuous constraints. *Constraints*, 1(1-2):85–118.
- [Schichl and Neumaier, 2005] Schichl, H. and Neumaier, A. (2005). Interval Analysis on Directed Acyclic Graphs for Global Optimization. *Journal of Global Optimization*, 33(4):541–562.

-
- [Seddik, 2015] Seddik, S. I. (2015). *Localization of a Swarm of Underwater Robots Using Set-Membership Methods*. PhD dissertation, Université de Bretagne Occidentale, Brest, France.
- [Serra et al., 2015] Serra, R., Arzelier, D., Joldes, M., Lasserre, J.-B. B., Rondépierre, A., and Salvy, B. (2015). A Power Series Expansion based Method to compute the Probability of Collision for Short-term Space Encounters. Research Report, LAAS-CNRS.
- [Smith et al., 1990] Smith, R., Self, M., and Cheeseman, P. (1990). Estimating Uncertain Spatial Relationships in Robotics. In *Autonomous Robot Vehicles*, pages 167–193. Springer, New York, NY.
- [Stachniss et al., 2004] Stachniss, C., Hahnel, D., and Burgard, W. (2004). Exploration with active loop-closing for FastSLAM. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 2, pages 1505–1510 vol.2.
- [Taha et al., 2015] Taha, W., Duracz, A., Zeng, Y., Atkinson, K., Bartha, F. A., Brauner, P., Duracz, J., Xu, F., Cartwright, R., Konečný, M., Moggi, E., Masood, J., Andreasson, P., Inoue, J., Sant’Anna, A., Philippsen, R., Chapoutot, A., O’Malley, M., Ames, A., Gaspes, V., Hvatum, L., Mehta, S., Eriksson, H., and Grante, C. (2015). Acumen: An Open-Source Testbed for Cyber-Physical Systems Research. In *Internet of Things. IoT Infrastructures*, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pages 118–130. Springer, Cham.
- [Thrun et al., 2005] Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics*. The MIT Press.
- [Thrun and Leonard, 2008] Thrun, S. and Leonard, J. J. (2008). Simultaneous Localization and Mapping. In Siciliano, B. and Khatib, O., editors, *Springer Handbook of Robotics*, pages 871–889. Springer Berlin Heidelberg, Berlin, Heidelberg.
- [Thrun and Montemerlo, 2006] Thrun, S. and Montemerlo, M. (2006). The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures. *The International Journal of Robotics Research*, 25(5-6):403–429.
- [Toumelin and Lemaire, 2001] Toumelin, N. and Lemaire, J. (2001). New capabilities of the Redermor unmanned underwater vehicle. In *IEEE OCEANS’01 conference*, volume 2, pages 1032–1035. Marine Technol. Soc.

- [Tucker, 1999] Tucker, W. (1999). The Lorenz attractor exists. *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, 328(12):1197–1202.
- [Tuohy et al., 1996] Tuohy, S. T., Leonard, J. J., Bellingham, J. G., Patrikalakis, N. M., and Chryssostomidis, C. (1996). Map Based Navigation for Autonomous Underwater Vehicles. *International Journal of Offshore and Polar Engineering*, 6(1):9–18.
- [Tyrén, 1982] Tyrén, C. (1982). Magnetic Anomalies as a Reference for Ground-speed and Map-matching Navigation. *Journal of Navigation*, 35(02):242.
- [Vaganay et al., 1996] Vaganay, J., Leonard, J., and Bellingham, J. (1996). Outlier rejection for autonomous acoustic navigation. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 2174–2181. IEEE.
- [Van Hentenryck et al., 1998] Van Hentenryck, P., Michel, L., and Benhamou, F. (1998). Constraint programming over nonlinear constraints. *Science of Computer Programming*, 30(1):83–118.
- [Veres and Norton, 1996] Veres, S. M. and Norton, J. P. (1996). Parameter-Bounding Algorithms for Linear Errors-in-Variables Models. In *Bounding Approaches to System Identification*, pages 275–288. Springer, Boston, MA.
- [Walter and Piet-Lahanier, 1988] Walter, É. and Piet-Lahanier, H. (1988). Estimation of the parameter uncertainty resulting from bounded-error data. *Mathematical Biosciences*, 92(1):55–74.
- [Walter and Piet-Lahanier, 1989] Walter, É. and Piet-Lahanier, H. (1989). Exact recursive polyhedral description of the feasible parameter set for bounded-error models. *IEEE Transactions on Automatic Control*, 34(8):911–915.
- [Waltz, 1972] Waltz, D. L. (1972). Generating Semantic Descriptions From Drawings of Scenes With Shadows. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA.
- [Wilczak et al., 2017] Wilczak, D., Zgliczyński, P., Pilarczyk, P., Mrozek, M., Kapela, T., Galias, Z., Cyranka, J., and Capinski, M. (2017). Computer Assisted Proofs in Dynamics group, a C++ package for rigorous numerics. <http://capd.ii.uj.edu.pl>.
- [Yu et al., 2016] Yu, W., Zamora, E., and Soria, A. (2016). Ellipsoid SLAM: a novel set membership method for simultaneous localization and mapping. *Autonomous Robots*, 40(1):125–137.

List of Figures

1.1	<i>HMS Challenger</i> , mother vessel of the <i>Challenger Expedition</i>	2
1.2	The deep-diving research bathyscaphe <i>Trieste</i>	3
1.3	In the shallow waters of La Spezia (Italy)	4
1.4	Bathymetric survey of the search for MH370 aircraft	6
1.5	The <i>Redermor</i> AUV before a sea trial	8
1.6	Deployment of the <i>Daurade</i> AUV	9
1.7	An overview of the <i>Toutatis</i> AUVs project	10
1.8	Illustration of drifting state estimations with dead-reckoning	14
1.9	A LBL navigation system made of four hydrophones	16
1.10	A USBL system used to localize the <i>Daurade</i> AUV	18
1.11	An overview of positioning results obtained with a USBL system	19
1.12	A simple SLAM illustration	21
1.13	The case of loop closure in similar environments	22
1.14	Artist view of the loop-based localization method	26
1.15	First illustration of a tube of robot trajectories	31
1.16	Loop detections based on uncertain state estimations	33
2.1	Range-only robot localization among three beacons	39
2.2	Reliable set-membership approaches to enclose a set of solutions	40
2.3	An interval vector $[\mathbf{x}] \in \mathbb{IR}^2$	46
2.4	Inclusion functions $[\mathbf{f}]$, $[\mathbf{f}]^*$	47
2.5	Comparison of different interval evaluations	50
2.6	The wrapping effect revealed through several boxes rotations	51
2.7	A domain \mathbb{X} submitted to two constraints \mathcal{L}_1 and \mathcal{L}_2	53
2.8	A box $[\mathbf{x}]$ contracted by \mathcal{C}_1 and \mathcal{C}_2	54
2.9	Set-membership localization with range-only measurements	59
2.10	Inner and outer approximation of a set \mathbb{X}	61
2.11	Inclusion tests for set-inversion	63
2.12	Triskelions computed with a SIVIA algorithm	64
2.13	SIVIA results on a range-only problem with two beacons	66
2.14	The kernel of an interval function $[f]$	66
2.15	Bounds on an interval function $[f]$	67
2.16	Inclusion functions $[f^c]$ and $[f^\supset]$	68
2.17	An interval $[x]$ computed from a Gaussian distribution	70

2.18	Outward rounding of an interval	70
3.1	Motivating example of range-only localization	74
3.2	A one-dimensional tube $[x](\cdot)$	77
3.3	Tube inversion	79
3.4	Lower and upper bounds of the integral of a tube	80
3.5	Example of tube arithmetics	82
3.6	Illustration of tubes contractions	84
3.7	A tube implemented by a set of boxes	85
3.8	Outer approximation of the lower bound of $\int_a^b [x](\tau)d\tau$	86
3.9	A binary tree used for tube representation	88
3.10	A tube built from a linear interpolation	89
3.11	Simulation of a mobile robot from initial conditions	93
3.12	Accuracy of the estimations from the reliable dead-reckoning method	94
3.13	A mobile robot avoiding an obstacle	94
4.1	Reaching a consistency state with $\mathcal{C}_{\frac{d}{dt}}$, example 1	107
4.2	Reaching a consistency state with $\mathcal{C}_{\frac{d}{dt}}$, example 2	108
4.3	A step-by-step illustration of the $\mathcal{C}_{\frac{d}{dt}}$'s implementation	113
4.4	Propagation of an observation with $\mathcal{C}_{\frac{d}{dt}}$	116
4.5	Causal kinematic chains associated to state equations	117
4.6	A cyclic constraint network for $\dot{x} = -\sin(x)$	118
4.7	Inefficient resolution of the $\dot{x} = -\sin(x)$ problem	119
4.8	Forward simulation of a mobile robot	122
4.9	Forward/backward simulation of a mobile robot	123
4.10	A robot following a Lissajous curve	124
4.11	Mobile robot localization without prior knowledge on its initial position	125
4.12	Accuracy of the estimations for the kidnapped robot problem	126
4.13	<i>Daurade's</i> tube $[p_2](\cdot)$: north velocity	127
4.14	<i>Daurade's</i> tube $[p_2](\cdot)$: north position	128
4.15	State estimation of the <i>Daurade</i> AUV over a 45 minutes mission	129
5.1	A robot perceiving a plane wreck with a side scan sonar	134
5.2	The <i>Swansea</i> wreck perceived with a side scan sonar	135
5.3	Application of the $\mathcal{C}_{\text{eval}}$ contractor on a tube $[y](\cdot)$	137
5.4	Combined $\mathcal{C}_{\text{eval}}$ contractions on a theoretical example	142
5.5	A sliced tube wrongly contracted	143
5.6	Map of the range-only localization problem	145
5.7	Constraint network detailing the range-only problem	149

5.8	Accuracy of the range-only state estimation with time uncertainties	149
5.9	State estimation of a mobile robot amongst a set of low-cost beacons	150
5.10	The problem of a drifting clock corrected by ephemerides	154
5.11	Top view of \mathcal{B} 's trajectory	154
5.12	Reliable prevision of the distances between the boat and the beacon	158
5.13	Clock drift estimation	159
6.1	The difference between loop detection and verification	164
6.2	A robot performing three loops	167
6.3	Loops detections in a bounded-error context	170
6.4	Zoom on the components of \mathbb{T}	171
6.5	Doubtful and undeniable loops in a set-membership context	172
6.6	Illustrating the degree of \mathbf{f}^* on Ω_i	174
6.7	Outer approximation Ω of a set \mathbb{T}	176
6.8	Illustration of the degree algorithm	179
6.9	Refinements of loop sets for Jacobian evaluation purposes	181
6.10	The <i>Redermor</i> AUV before a sea trial	182
6.11	Tube $[v_1](\cdot)$ enclosing the east velocity during the <i>Redermor</i> mission	183
6.12	Map of the <i>Redermor</i> experiment	184
6.13	t -plane of the <i>Redermor</i> experiment	185
6.14	Independent projection of the non-conclusive loop case	186
6.15	Map of the <i>Daurade</i> experiment	189
6.16	t -plane of the <i>Daurade</i> experiment	190
6.17	Zoom on the t -plane of the experiment	191
6.18	Independent projection of a non-conclusive loop case	192
6.19	Ambiguous looped trajectories	193
7.1	Illustration of a hydrographic survey with an AUV	197
7.2	A flawed Roberval balance with arms of unknown length	201
7.3	Physical interpretation of the $\mathcal{L}_{\mathbf{p} \Rightarrow \mathbf{z}}$ constraint	206
7.4	An example of scalar measurements $z(\cdot)$	208
7.5	Illustration of t -sets $\mathbb{T}_{\mathbf{p}}^*$ and $\mathbb{T}_{\mathbf{z}}^*$	208
7.6	Approximation of the enclosure of t -sets with SIVIA algorithms	209
7.7	Contraction of a t -set from the $\mathcal{L}_{\mathbf{p} \Rightarrow \mathbf{z}}$ constraint	210
7.8	Inter-temporal contractions using \mathcal{C}_{t_1, t_2}	214
7.9	Minimal envelope of a contracted set Ω_i	217
7.10	A bathymetric sensing only depends on 2D positions	219
7.11	Footprint of the DVL beams according to the altitude	222
7.12	Location of the mission area	225

List of Figures

7.13	<i>Daurade</i> on the working deck of the <i>Aventurière II</i>	225
7.14	20 th October 2015: observation tube	226
7.15	19 th October 2015: observation tube	227
7.16	20 th October 2015: mission map	228
7.17	20 th October 2015: thickness graph	229
7.18	19 th October 2015: mission map	230
7.19	19 th October 2015: thickness graph	231
7.20	Digital Elevation Model of the area	232
7.21	Inefficient temporal contraction in case of high frequency observations	237

List of Tables

1.1	<i>Redermor's</i> main characteristics	8
1.2	<i>Daurade's</i> main characteristics	9
2.1	Beacons' positions and respective measurements	57
2.2	New beacons for the SIVIA illustration	65
5.1	List of beacons' locations for the low-cost beacon problem	147
5.2	List of measurements for the low-cost beacon problem	147
5.3	List of measurements for the drifting clock problem	155
7.1	IHO Standards for hydrographic surveys	196
7.2	SLAM iterations on 20 th 's experiment	229
7.3	SLAM iterations on 19 th 's experiment	231
7.4	Taxonomy of SLAM methods	235

List of Algorithms

1	SIVIA(in: $[\mathbf{f}], [\mathbf{x}], \mathbb{Y}, \varepsilon$ – inout: $\mathbb{X}^-, \mathbb{X}^+$)	62
2	kernelSIVIA (in : $[\mathbf{f}], [\mathbf{x}], \varepsilon$, inout : $\mathbb{X}^-, \mathbb{X}^+$)	68
3	$\mathcal{C}_{\frac{d}{dt}}^{\rightarrow}$ (in : $[x_0], [v](\cdot)$, inout : $[x](\cdot)$)	112
4	$\mathcal{C}_{\frac{d}{dt}}^{\leftarrow}$ (in : $[x_f], [v](\cdot)$, inout : $[x](\cdot)$)	112
5	$\mathcal{C}_{\frac{d}{dt}}$ (in : $[x_0], [x_f], [v](\cdot)$, inout : $[x](\cdot)$)	112
6	proprioLoopSIVIA (in : $[\mathbf{v}](\cdot), [\mathbf{t}], \varepsilon$, inout : $\mathbb{T}^-, \mathbb{T}^+$)	170
7	existenceTest \mathcal{T} (in : $\Omega_i, [\mathbf{f}]$ – out : true \emptyset)	177
8	2dTopoDegree (in : $[\mathbf{b}]_1 \dots [\mathbf{b}]_p, [\mathbf{f}]$ – out : d)	178
9	tagEdge (in : $[\mathbf{b}], [\mathbf{f}]$ – out : (c, s))	178
10	loopsNumber (in : $\Omega_i, [\mathbf{f}], [\mathbf{J}_f]$ – out : ℓ)	180
11	\mathcal{C}_{t_1, t_2} (in : $[\mathbf{w}](\cdot)$, inout : $[t_1], [t_2], [\mathbf{p}](\cdot)$)	213
12	\mathcal{C}_{Ω} (in : $\Omega, [\mathbf{w}](\cdot)$, inout : $[\mathbf{p}](\cdot)$)	213
13	$\mathcal{C}_{\mathbf{p} \Rightarrow \mathbf{z}}$ (in : $[\mathbf{w}](\cdot), [\mathbf{z}](\cdot), \varepsilon$, inout : $[\mathbf{p}](\cdot)$)	215
14	$\mathcal{C}_{\mathbf{p} \Rightarrow \mathbf{z}}^{\text{fast}}$ (in : $[\mathbf{w}](\cdot), [\mathbf{z}](\cdot), \varepsilon$, inout : $[\mathbf{p}](\cdot)$)	217
15	temporalSLAM (in : $[\mathbf{v}](\cdot), [\mathbf{u}](\cdot), [\mathbf{z}](\cdot), \varepsilon$, inout : $[\mathbf{x}](\cdot)$)	220

List of Abbreviations

- AUV** Autonomous Underwater Vehicle. 6–11, 15, 17–19, 22, 25, 26, 34, 121, 126, 133, 182, 187, 194, 196, 197, 221, 223, 238, 240
- BPF** Box Particle Filter. 75
- CN** Constraint Network. 52, 57, 83, 91, 115, 124, 137, 144, 148, 151, 155, 156, 203, 207, 209, 212, 218, 220, 238
- CSP** Constraint Satisfaction Problem. 52
- DEM** Digital Elevation Model. 6, 232
- DVL** Doppler Velocity Log. 8, 9, 11, 13, 127, 129, 221–224, 226, 239
- GNSS** Global Navigation Satellite System. 11, 15, 17, 181, 232
- IMU** Inertial Measurement Unit. 11, 13, 14, 127
- INS** Inertial Navigation System. 8, 9, 13–15, 17, 129, 224, 228, 239
- IVP** Initial Value Problem. 75, 95, 101, 102, 109, 118, 120, 130
- LBL** Long BaseLine. 16, 17
- ODE** Ordinary Differential Equation. 75, 100–103, 130
- PF** Particle Filter. 75
- SIVIA** Set-Inversion *via* Interval Analysis. 61, 62, 64–66, 151, 168, 169, 177, 185, 209, 226, 242
- SLAM** Simultaneous Localization And Mapping. 20–24, 28, 31, 32, 34, 134, 161, 165, 166, 194, 196–198, 200, 201, 218, 219, 221, 222, 224, 227–229, 231–235, 238, 239, 241, 242
- USBL** Ultra-Short BaseLine. 17–20, 127, 129, 228

Index

- acoustics, 5
- AUVs, 5, 7, 196
 - Daurade* AUV, 8, 18, 19, 126, 129, 133, 187, 221, 225
 - Redermor* AUV, 7, 182
 - Toutatis* AUV, 10
- bathymetry, 6, 219
- bathyscaphe, 2
- Borda's weighing method, 200
- boustrophedon, 196
- cameras, 4
- collaborative positioning, 17
- constraints, 29, 52
 - decompositions, 53, 91
 - in dynamical systems, 30
 - language, 29, 244
 - Constraint Networks, 52, 114, 149
 - constraint programming, 29, 102
 - propagations, 53, 117
- contractors, 54
 - Contractor Prog., 56, 103, 114
 - tube contractors, *see* tube
- dead-reckoning, 13, 90, 126, 129, 229
- DEM, 5, 232
- dependencies, 49
- dot notation (\cdot), 76
- drifting clock, 153
- DVL, 8, 9, 13, 126, 129, 221, 223
- Euler angles, 13, 126, 223
- fixed points, 58, 117, 141, 149
- floating points, 41, 70, 103
- temporal propagations, 104, 110
- Gaussian, 69
- GNSS, 11, 17
- guaranteed integration, *see* IVP
- hybridization, 9, 223
- hydrography, 6, 196
- IMU, 126
- INS, 8, 9, 13, 129, 223
- inter-temporal, 20, 25, 199
- intervals
 - interval arithmetic, 42
 - inclusion functions, 47
 - libraries, *see* numerical libraries
 - interval vectors, 46
- IVP, 75, 101, 102, 118
- kernel, 65, 209, 220
- kidnapped problem, 75, 100, 125, 231
- numerical libraries
 - Acumen*, 92
 - CAPD*, 101, 120, 121
 - Cosy*, 101
 - DynIBEX*, 101, 120
 - filib++*, 70
 - GAOL*, 70
 - IBEX*, 71
 - Tubex*, 71, 89, 142, 243
 - Vnode*, 101
- Lipschitz continuity, 109
- loops
 - loop closure, 21
 - loop detection, 33, 164
 - formalism, 166
 - loops number, 179, 183
 - loop proof/verification, 164, 171, 211
- Newton test \mathcal{N} , 171
- obstacle avoidance, 93
- ODEs, 75, 100, 102
- outliers, 19, 72
- outward rounding, 70
- path planning, 95
- probabilistics, 27

-
- range-only problem, 38, 57, 65, 66, 74, 145, 150
 - analytical reformulation, 151
 - sensors, 69, 87, 221
 - set-inversion, 60, 61, 63, 64, 66, 168, 209
 - SHOM, 8, 196, 232
 - SIVIA, *see* set-inversion
 - SLAM, 20, 196
 - Bathymetric SLAM, 221
 - complexity, 22, 165
 - formalism, 25, 198
 - taxonomy, 233
 - Temporal SLAM, 218
 - sonars, 5, 133, 221
 - state equations, 12, 74, 114, 132, 144
 - evolution function, 12
 - observation function, 12, 198
 - configuration function, 199, 201, 218
 - state vector, 12
 - strangle method, 120
 - subpaving, 40, 60, 61, 176
 - temporal sets, 166, 185, 190, 205
 - tide, 202, 223
 - time discretization, 85, 103, 142
 - time uncertainties, 25, 133
 - topological degree, 173, 211
 - trajectory, 76
 - tube, 31, 76
 - tube analysis, 78
 - contractors, 81
 - $\mathcal{C}_{\frac{d}{dt}}$, 104, 115
 - $\mathcal{C}_{\text{eval}}$, 115, 136
 - $\mathcal{C}_{\mathbf{p} \Rightarrow \mathbf{z}}$, 208
 - \mathcal{C}_{t_1, t_2} , 212
 - derivative, 87, 104, 109, 136
 - implementation, 83, 110
 - integral, 79, 103
 - inversion, 78, 156
 - underwater positioning, 15
 - LBL system, 16
 - USBL system, 17–19, 129, 228
 - unstructured env., 4, 23, 24
 - wrappers, 39
 - wrapping effect, 50, 120, 217
 - wrecks, 5, 133

Reliable robot localization: a constraint programming approach over dynamical systems

The localization of underwater robots remains a challenging issue. Usual sensors, such as Global Navigation Satellite System (GNSS) receivers, cannot be used under the surface and other inertial systems suffer from a strong integration drift. On top of that, the seabed is generally uniform and unstructured, making it difficult to apply usual Simultaneous Localization and Mapping (SLAM) methods to perform a localization.

Hence, innovative approaches have to be explored. The presented method can be characterized as a raw-data SLAM approach, but we propose a temporal resolution – which differs from usual methods – by considering time as a standard variable to be estimated. This concept raises new opportunities for state estimation, under-exploited so far. However, such temporal resolution is not straightforward and requires a set of theoretical tools in order to achieve the main purpose of localization.

This thesis is thus not only a contribution in the field of mobile robotics, it also offers new perspectives in the areas of constraint programming and set-membership approaches. We provide a reliable contractor programming framework in order to build solvers for dynamical systems. This set of tools is illustrated throughout this document with realistic robotic applications.

Keywords: mobile robotics, dynamical systems, constraint programming, interval analysis, localization, SLAM, AUVs

Localisation fiable de robots : une approche de programmation par contraintes sur des systèmes dynamiques

Aujourd'hui, la localisation de robots sous-marins demeure une tâche complexe. L'utilisation de capteurs habituels est impossible sous la surface, tels que ceux reposant sur les systèmes de géolocalisation par satellites. Les approches inertielles sont quant à elles limitées par leur forte dérive dans le temps. De plus, les fonds marins sont généralement homogènes et non structurés, rendant difficile l'utilisation de méthodes SLAM connues, qui couplent la localisation et la cartographie de manière simultanée.

Il devient donc nécessaire d'explorer de nouvelles alternatives. Notre approche consiste à traiter un problème de SLAM de manière purement temporelle. L'originalité de ce travail est de représenter le temps comme une variable classique qu'il faut estimer. Cette stratégie soulève de nouvelles opportunités dans le domaine de l'estimation d'état, permettant de traiter de nombreux problèmes sous un autre angle. Toutefois, une telle résolution temporelle demande un ensemble d'outils théoriques qu'il convient de développer.

Cette thèse n'est donc pas seulement une contribution dans le monde de la robotique mobile, elle propose également une nouvelle démarche dans les domaines de la propagation de contraintes et des méthodes ensemblistes. Cette étude apporte de nouveaux outils de programmation par contracteurs qui permettent le développement de solveurs pour des systèmes dynamiques. Les composants étudiés sont mis en application tout au long de ce document autour de problèmes robotiques concrets.

Mots-clefs : robotique mobile, systèmes dynamiques, programmation par contraintes, analyse par intervalles, localisation, SLAM, AUVs