Set-membership Terrain Based Navigation

Simon Rohou

ENSTA Bretagne, Lab-STICC, UMR CNRS 6285, Brest, France

LS2N Seminar / ARMEN team
26$^{\text{th}}$ April 2019

Section 1

Motivations

Motivations

Autonomous wrecks search

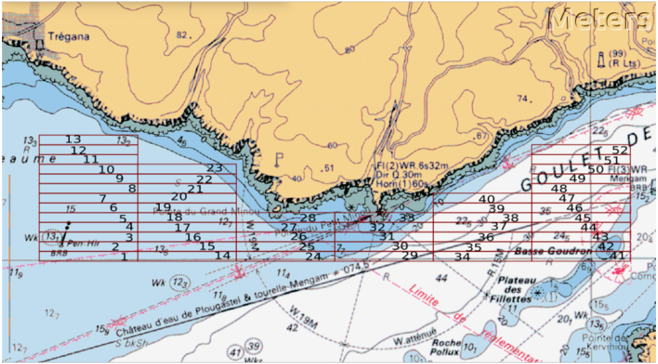
1512: the Breton *Cordelière* and the English *Regent* flagships are lost during the Battle of Saint-Mathieu (near Brest)

The simultaneous destruction of the *Cordelière* and the *Regent*
(depicted by Pierre-Julien Gilbert)

Motivations

Autonomous wrecks search

2019: autonomous research of the wrecks.
Problem: exploration of a wide underwater area without surfacing.



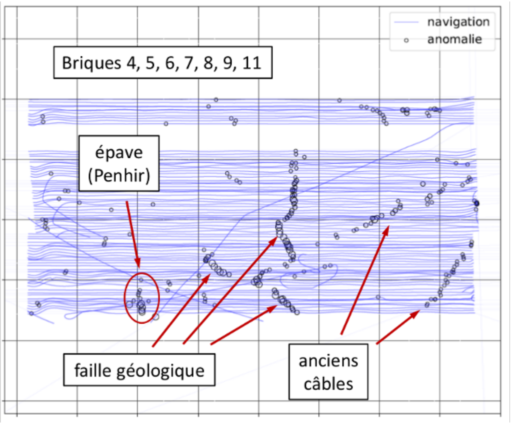
Underwater areas to explore near Brest

Image: Luc Jaulin

Motivations

Autonomous wrecks search

Magnetic sensors $\Longrightarrow$ navigation near the seabed
$\Longrightarrow$ good position estimation



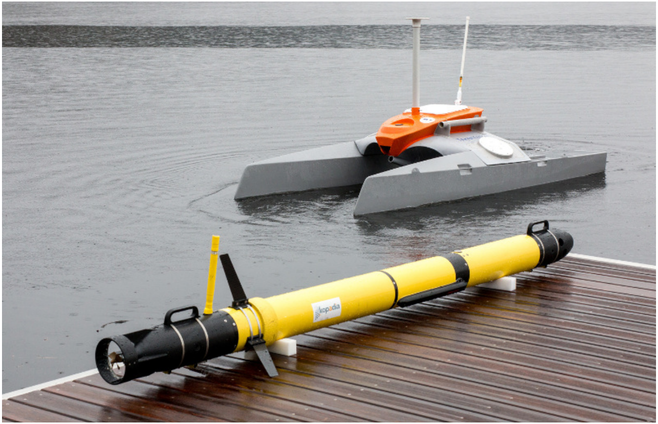
First results of a previous magnetic survey (2018)

Image: Romain Schwab

Motivations

Autonomous wrecks search

Magnetic sensors $\Longrightarrow$ navigation near the seabed
$\Longrightarrow$ good position estimation



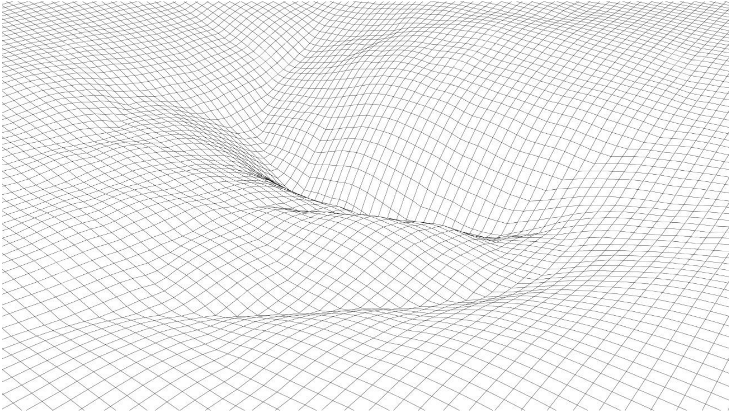
Autonomous vehicles that will be used for the next wreck search (2019)
(Sub-Meeting workshop: https://www.ensta-bretagne.fr/jaulin/submeeting2019.html)

Motivations

Problem: wide environment and poor observations

Under the surface:

- **no satellite** navigation systems available
- **no seamarks** or points of interest

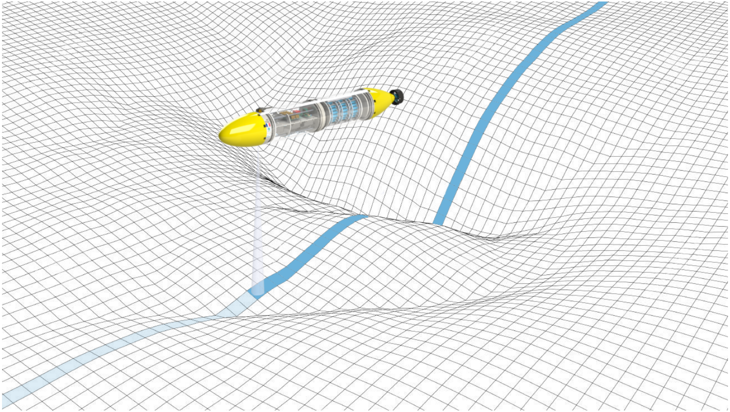


Motivations

Problem: wide environment and poor observations

Available data:

- **bathymetric** measurements (scalar values)
- prior knowledge of the environment (embedded map)

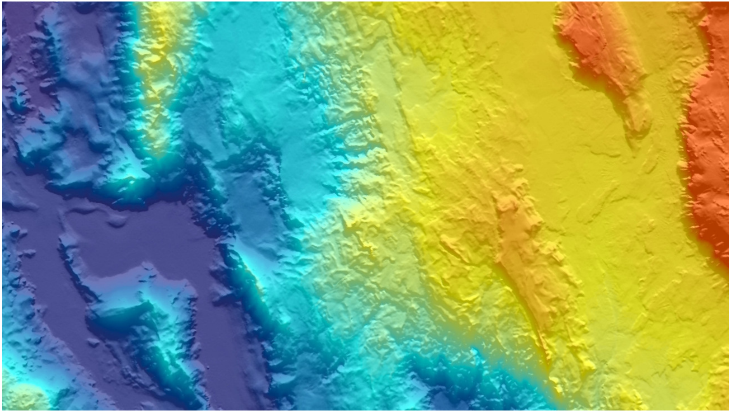


Motivations

Problem: wide environment and poor observations

Available data:

- **bathymetric** measurements (scalar values)
- prior knowledge of the environment (embedded map)

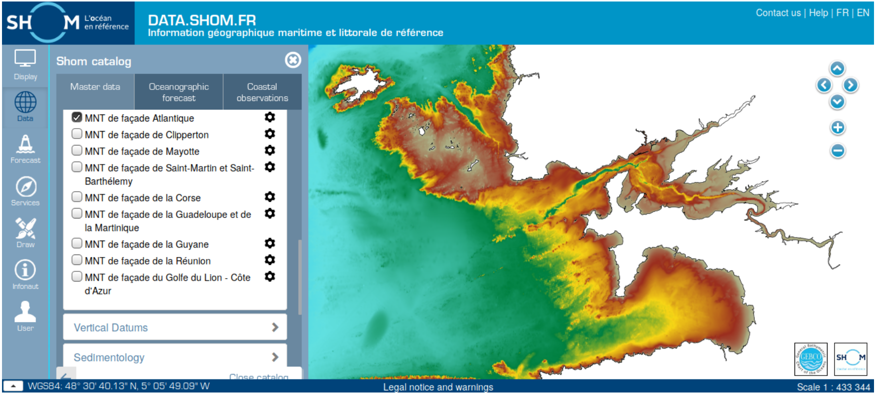


Motivations

Available Digital Elevation Models (DEM)

Portail data.shom.fr

- Shom: Service Hydrographique et Océanographique de la Marine
- many marine DEM available



Motivations

Available Digital Elevation Models (DEM)

ENSTA Bretagne

- hydrographic department and *Panopée* boat
- several surveys made in Rade de Brest

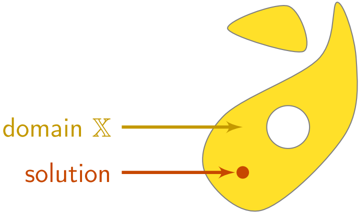
Section 2

Constraint programming

Constraint programming

Constraint programming: overall concept

- system described by a *constraint network*
- **variables** belonging to **domains** $\mathbb{X}$



Constraint network:

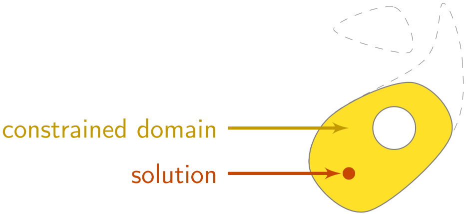
$$\begin{cases} \textbf{Variables: } \mathbf{x} \\ \textbf{Constraints:} \\ \\ \\ \\ \textbf{Domains: } \mathbb{X} \end{cases}$$

Contractor Programming
Chabert, Jaulin *Artifical Intelligence*, 2009

Constraint programming

Constraint programming: overall concept

- system described by a *constraint network*
- **variables** belonging to **domains** $\mathbb{X}$
- continuous **constraints** $\mathcal{L}$: non-linear equations, inequalities, ...



Constraint network:

Variables: $\mathbf{x}$

Constraints:

1. $\mathcal{L}_1(\mathbf{x})$

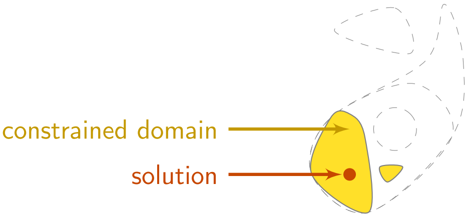
Domains: $\mathbb{X}$

Contractor Programming
Chabert, Jaulin *Artifical Intelligence*, 2009

Constraint programming

Constraint programming: overall concept

- system described by a *constraint network*
- **variables** belonging to **domains** $\mathbb{X}$
- continuous **constraints** $\mathcal{L}$: non-linear equations, inequalities, ...



Constraint network:

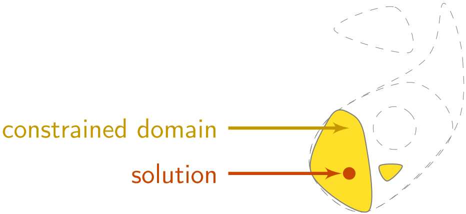
$$\left\{\begin{array}{l}\textbf{Variables: } \mathbf{x}\\ \textbf{Constraints:}\\ \quad 1.\ \mathcal{L}_1(\mathbf{x})\\ \quad 2.\ \mathcal{L}_2(\mathbf{x})\\ \textbf{Domains: } \mathbb{X}\end{array}\right.$$

Contractor Programming
Chabert, Jaulin *Artifical Intelligence*, 2009

Constraint programming

Constraint programming: overall concept

- system described by a *constraint network*
- **variables** belonging to **domains** $\mathbb{X}$
- continuous **constraints** $\mathcal{L}$: non-linear equations, inequalities, ...



Constraint network:

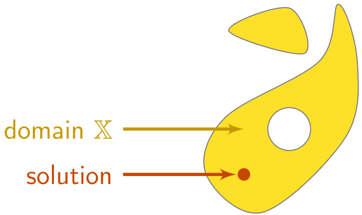
$$\left\{\begin{array}{l}\textbf{Variables: } \mathbf{x} \\ \textbf{Constraints:} \\ \quad 1.\ \mathcal{L}_1(\mathbf{x}) \\ \quad 2.\ \mathcal{L}_2(\mathbf{x}) \\ \quad 3.\ \ldots \\ \textbf{Domains: } \mathbb{X}\end{array}\right.$$

Contractor Programming
Chabert, Jaulin *Artifical Intelligence*, 2009

Constraint programming

Constraint programming: overall concept

- system described by a *constraint network*
- **variables** belonging to **domains** $\mathbb{X}$
- continuous **constraints** $\mathcal{L}$: non-linear equations, inequalities, ...



Constraint network:

Variables: $\mathbf{x}$

Constraints:

1. $\mathcal{L}_1(\mathbf{x})$
2. $\mathcal{L}_2(\mathbf{x})$
3. ...

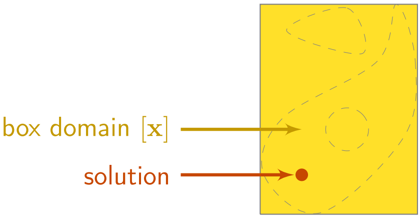
Domains: $\mathbb{X}$

Contractor Programming
Chabert, Jaulin *Artifical Intelligence*, 2009

Constraint programming

Constraint programming: overall concept

- system described by a *constraint network*
- **variables** belonging to **domains** $\mathbb{X}$
- continuous **constraints** $\mathcal{L}$: non-linear equations, inequalities, ...
- representable domains: *e.g.* boxes $[\mathbf{x}]$



Constraint network:

Variables: $\mathbf{x}$

Constraints:

1. $\mathcal{L}_1(\mathbf{x})$
2. $\mathcal{L}_2(\mathbf{x})$
3. ...

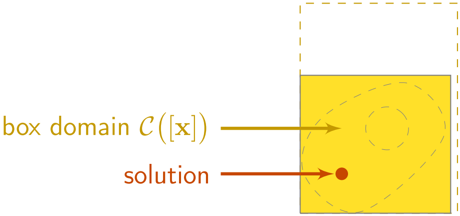
Domains: $[\mathbf{x}]$

Contractor Programming
Chabert, Jaulin *Artifical Intelligence*, 2009

Constraint programming

Constraint programming: overall concept

- system described by a *constraint network*
- **variables** belonging to **domains** $\mathbb{X}$
- continuous **constraints** $\mathcal{L}$: non-linear equations, inequalities, ...
- representable domains: *e.g.* boxes $[\mathbf{x}]$
- resolution by **contractors**, $\mathcal{C}_{\mathcal{L}}([\mathbf{x}])$



Constraint network:

Variables: $\mathbf{x}$

Constraints:

1. $\mathcal{L}_1(\mathbf{x})$
2. $\mathcal{L}_2(\mathbf{x})$
3. ...

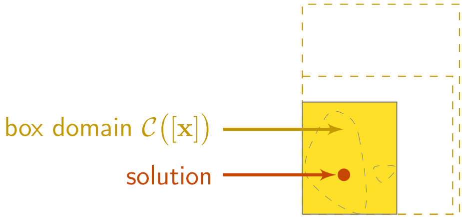
Domains: $[\mathbf{x}]$

Contractor Programming
Chabert, Jaulin *Artifical Intelligence*, 2009

Constraint programming

Constraint programming: overall concept

- system described by a *constraint network*
- **variables** belonging to **domains** $\mathbb{X}$
- continuous **constraints** $\mathcal{L}$: non-linear equations, inequalities, ...
- representable domains: *e.g.* boxes $[\mathbf{x}]$
- resolution by **contractors**, $\mathcal{C}_{\mathcal{L}}([\mathbf{x}])$



Constraint network:

$$\left\{\begin{array}{l}\textbf{Variables: } \mathbf{x} \\ \textbf{Constraints:} \\ \quad 1.\ \mathcal{L}_1(\mathbf{x}) \\ \quad 2.\ \mathcal{L}_2(\mathbf{x}) \\ \quad 3.\ \ldots \\ \textbf{Domains: } [\mathbf{x}]\end{array}\right.$$

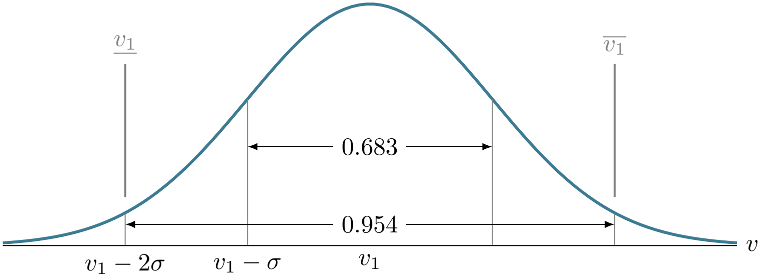
Contractor Programming
Chabert, Jaulin *Artifical Intelligence*, 2009

Constraint programming

Domains from sensor data

Uncertainties:

- datasheets $\Longrightarrow$ standard deviation σ for each sensor
- 95% confidence rate: $v_1^* \in [v_1] = [v_1 - 2\sigma, v_1 + 2\sigma]$

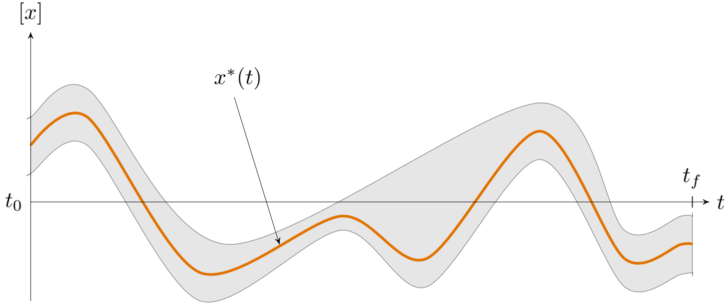


- uncertainties then reliably propagated in the system

Constraint programming

Tubes: domains for trajectories

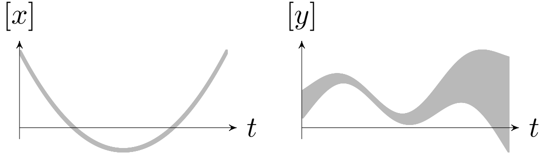
Tube $[x](\cdot)$: interval of trajectories $[x^-(\cdot), x^+(\cdot)]$
such that $\forall t \in \mathbb{R},\ x^-(t) \leqslant x^+(t)$



Tube $[x](\cdot)$ enclosing an uncertain trajectory $x^*(\cdot)$

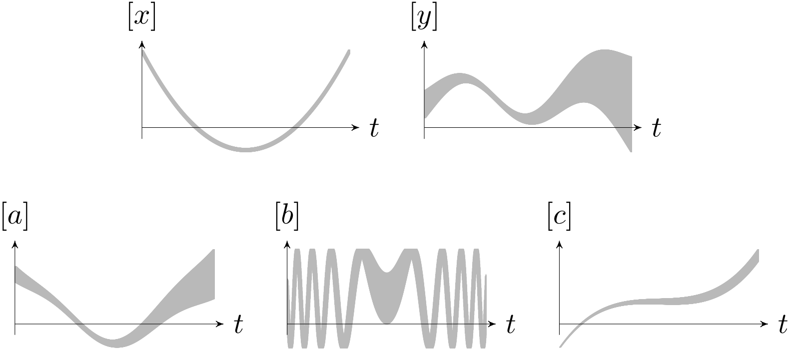
Constraint programming

Tubes arithmetic



Constraint programming

Tubes arithmetic



$[a](\cdot) = [x](\cdot) + [y](\cdot)$

$[b](\cdot) = \sin\big([x](\cdot)\big)$

$[c](\cdot) = \int_0 [x](\tau)d\tau$

Constraint programming

Terrain Based Navigation: formalization

Robot localization = state estimation problem.
Classically, we have:

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}\big(\mathbf{x}(t), \mathbf{u}(t)\big) & \text{(navigation)} \\ \end{cases}$$

With:

- $\mathbf{x}$: state vector (position, bearing, ...)
- $\mathbf{u}$: input vector (command)
- $\mathbf{f}$: *evolution* function

Constraint programming

Terrain Based Navigation: formalization

Robot localization = state estimation problem.
Classically, we have:

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}\big(\mathbf{x}(t), \mathbf{u}(t)\big) & \text{(navigation)} \\ z_i = g\big(\mathbf{x}(t_i), \mathbb{M}\big) & \text{(measurements)} \end{cases}$$

With:

- $\mathbf{x}$: state vector (position, bearing, …)
- $\mathbf{u}$: input vector (command)
- $\mathbf{f}$: *evolution* function
- g: *observation* function
- z_i: scalar measurements (at t_i)
- $\mathbb{M}$: terrain knowledge

Constraint programming

Terrain Based Navigation: formalization

Robot localization = state estimation problem.
Classically, we have:

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}\big(\mathbf{x}(t), \mathbf{u}(t)\big) & \text{(navigation)} \\ z_i = g\big(\mathbf{x}(t_i), \mathbb{M}\big) & \text{(measurements)} \end{cases}$$

With:

- $\mathbf{x}$: state vector (position, bearing, ...)
- $\mathbf{u}$: input vector (command)
- $\mathbf{f}$: *evolution* function
- g: *observation* function
- z_i: scalar measurements (at t_i) (ex: scalar bathymetric values)
- $\mathbb{M}$: terrain knowledge (ex: digital elevation map)

Constraint programming

Terrain Based Navigation: formalization

Robot localization = state estimation problem.
Classically, we have:

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}\big(\mathbf{x}(t), \mathbf{u}(t)\big) & \text{(navigation)} \\ z_i = \mathbb{M}\big(\mathbf{x}(t_i)\big) & \text{(measurements)} \end{cases}$$

With:

- $\mathbf{x}$: state vector (position, bearing, . . .)
- $\mathbf{u}$: input vector (command)
- $\mathbf{f}$: *evolution* function
- g: *observation* function
- z_i: scalar measurements (at t_i) (ex: scalar bathymetric values)
- $\mathbb{M}$: terrain knowledge (ex: digital elevation map)

Constraint programming

Constraint programming for TBN

Using intervals and tubes to enclose our solutions:

Variables:

- reals: z_i, t_i
- trajectories: $\mathbf{x}(\cdot)$

Constraint programming

Constraint programming for TBN

Using intervals and tubes to enclose our solutions:

Variables:

- reals: z_i, t_i
- trajectories: $\mathbf{x}(\cdot)$

Numerical domains of the variables:

- intervals: $[z_i]$, $[t_i]$
- tubes: $[\mathbf{x}](\cdot)$

Constraint programming

Constraint programming for TBN

Using contractors to reduce the domains of the variables:

System:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \\ z_i = \mathbb{M}\big(\mathbf{x}(t_i)\big) \end{cases}$$

Constraint programming

Constraint programming for TBN

Using contractors to reduce the domains of the variables:

System:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \\ z_i = \mathbb{M}\big(\mathbf{x}(t_i)\big) \end{cases}$$

Elementary constraints decomposition:

- $\mathbf{v}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big)$

Constraint programming

Constraint programming for TBN

Using contractors to reduce the domains of the variables:

System:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \\ z_i = \mathbb{M}\big(\mathbf{x}(t_i)\big) \end{cases}$$

Elementary constraints decomposition:

- $\mathbf{v}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big)$
- $\dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot)$

Constraint programming

Constraint programming for TBN

Using contractors to reduce the domains of the variables:

System:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \\ z_i = \mathbb{M}\big(\mathbf{x}(t_i)\big) \end{cases}$$

Elementary constraints decomposition:

- $\mathbf{v}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big)$
- $\dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot)$
- $\mathbf{p}_i = \mathbf{x}(t_i)$

Constraint programming

Constraint programming for TBN

Using contractors to reduce the domains of the variables:

System:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \\ z_i = \mathbb{M}\big(\mathbf{x}(t_i)\big) \end{cases}$$

Elementary constraints decomposition:

- $\mathbf{v}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big)$
- $\dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot)$
- $\mathbf{p}_i = \mathbf{x}(t_i)$
- $z_i = \mathbb{M}\big(\mathbf{p}_i\big)$

Constraint programming

Constraint programming for TBN

Using contractors to reduce the domains of the variables:

System:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \\ z_i = \mathbb{M}\big(\mathbf{x}(t_i)\big) \end{cases}$$

Elementary constraints decomposition:

- $\mathbf{v}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \rightarrow$ algebraic constraint
- $\dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot)$
- $\mathbf{p}_i = \mathbf{x}(t_i)$
- $z_i = \mathbb{M}\big(\mathbf{p}_i\big)$

Constraint programming for TBN

Using contractors to reduce the domains of the variables:

System:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \\ z_i = \mathbb{M}\big(\mathbf{x}(t_i)\big) \end{cases}$$

Elementary constraints decomposition:

- $\mathbf{v}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big)$ $\rightarrow$ algebraic constraint
- $\dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot)$ $\rightarrow$ derivative constraint
- $\mathbf{p}_i = \mathbf{x}(t_i)$
- $z_i = \mathbb{M}\big(\mathbf{p}_i\big)$

Constraint programming

Constraint programming for TBN

Using contractors to reduce the domains of the variables:

System:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \\ z_i = \mathbb{M}\big(\mathbf{x}(t_i)\big) \end{cases}$$

Elementary constraints decomposition:

- $\mathbf{v}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \rightarrow$ algebraic constraint
- $\dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot) \rightarrow$ derivative constraint $\rightarrow \mathcal{L}_{\frac{d}{dt}}\big(\mathbf{x}(\cdot), \mathbf{v}(\cdot)\big)$
- $\mathbf{p}_i = \mathbf{x}(t_i)$
- $z_i = \mathbb{M}\big(\mathbf{p}_i\big)$

Constraint programming

Constraint programming for TBN

Using contractors to reduce the domains of the variables:

System:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \\ z_i = \mathbb{M}\big(\mathbf{x}(t_i)\big) \end{cases}$$

Elementary constraints decomposition:

- $\mathbf{v}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big)$ $\rightarrow$ algebraic constraint
- $\dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot)$ $\rightarrow$ derivative constraint $\rightarrow \mathcal{L}_{\frac{d}{dt}}\big(\mathbf{x}(\cdot), \mathbf{v}(\cdot)\big)$
- $\mathbf{p}_i = \mathbf{x}(t_i)$ $\rightarrow$ evaluation constraint
- $z_i = \mathbb{M}\big(\mathbf{p}_i\big)$

Constraint programming

Constraint programming for TBN

Using contractors to reduce the domains of the variables:

System:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \\ z_i = \mathbb{M}\big(\mathbf{x}(t_i)\big) \end{cases}$$

Elementary constraints decomposition:

- $\mathbf{v}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big)$ $\rightarrow$ algebraic constraint
- $\dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot)$ $\rightarrow$ derivative constraint $\rightarrow \mathcal{L}_{\frac{d}{dt}}\big(\mathbf{x}(\cdot), \mathbf{v}(\cdot)\big)$
- $\mathbf{p}_i = \mathbf{x}(t_i)$ $\rightarrow$ evaluation constraint $\rightarrow \mathcal{L}_{\text{eval}}\big(t_i, \mathbf{p}_i, \mathbf{x}(\cdot)\big)$
- $z_i = \mathbb{M}\big(\mathbf{p}_i\big)$

Constraint programming

Constraint programming for TBN

Using contractors to reduce the domains of the variables:

System:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \\ z_i = \mathbb{M}\big(\mathbf{x}(t_i)\big) \end{cases}$$

Elementary constraints decomposition:

- $\mathbf{v}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \rightarrow$ algebraic constraint
- $\dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot) \rightarrow$ derivative constraint $\rightarrow \mathcal{L}_{\frac{d}{dt}}\big(\mathbf{x}(\cdot), \mathbf{v}(\cdot)\big)$
- $\mathbf{p}_i = \mathbf{x}(t_i) \rightarrow$ evaluation constraint $\rightarrow \mathcal{L}_{\text{eval}}\big(t_i, \mathbf{p}_i, \mathbf{x}(\cdot)\big)$
- $z_i = \mathbb{M}\big(\mathbf{p}_i\big) \rightarrow$ map constraint $\mathcal{L}_{\mathbb{M}}$

Constraint programming

Constraint programming for TBN

Using contractors to reduce the domains of the variables:

System:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \\ z_i = \mathbb{M}\big(\mathbf{x}(t_i)\big) \end{cases}$$

Elementary constraints decomposition:

- $\mathbf{v}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \rightarrow$ algebraic constraint
- $\dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot) \rightarrow$ derivative constraint $\rightarrow \mathcal{L}_{\frac{d}{dt}}\big(\mathbf{x}(\cdot), \mathbf{v}(\cdot)\big)$
- $\mathbf{p}_i = \mathbf{x}(t_i) \rightarrow$ evaluation constraint $\rightarrow \mathcal{L}_{\mathrm{eval}}\big(t_i, \mathbf{p}_i, \mathbf{x}(\cdot)\big)$
- $z_i = \mathbb{M}\big(\mathbf{p}_i\big) \rightarrow$ map constraint $\mathcal{L}_{\mathbb{M}}$

Note: a solver could break down such problem automatically

Constraint programming

Algebraic constraints on trajectories

Example 1: consider the constraint $a(\cdot) = x(\cdot) + y(\cdot)$
A minimal **contractor** to apply this constraint is:

$$\left(\begin{array}{c}[a](\cdot)\\ [x](\cdot)\\ [y](\cdot)\end{array}\right) \overset{\mathcal{C}_+}{\longmapsto} \left(\begin{array}{c}[a](\cdot)\cap([x](\cdot)+[y](\cdot))\\ [x](\cdot)\cap([a](\cdot)-[y](\cdot))\\ [y](\cdot)\cap([a](\cdot)-[x](\cdot))\end{array}\right)$$

Contractor programming: $\mathcal{C}_+\big([a](\cdot),[x](\cdot),[y](\cdot)\big)$

Constraint programming

Algebraic constraints on trajectories

Example 1: consider the constraint $a(\cdot) = x(\cdot) + y(\cdot)$
A minimal **contractor** to apply this constraint is:

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

Contractor programming: $\mathcal{C}_+\big([a](\cdot), [x](\cdot), [y](\cdot)\big)$

Example 2: consider the constraint $c(\cdot) = \int_0^{\cdot} x(\tau)d\tau$
A non-minimal **contractor** to apply this constraint is:

$$\begin{pmatrix} [x](\cdot) \\ [c](\cdot) \end{pmatrix} \xrightarrow{\mathcal{C}_{\int}} \begin{pmatrix} [x](\cdot) \\ [c](\cdot) \cap \int_0^{\cdot} [x](\tau)\, d\tau \end{pmatrix}$$

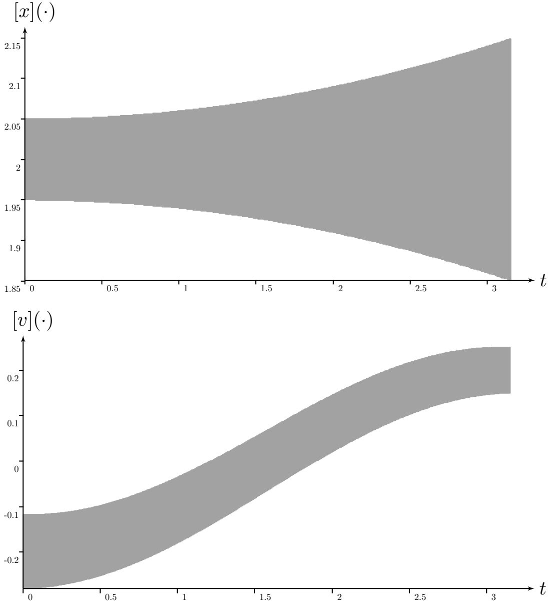
Contractor programming: $\mathcal{C}_{\int}\big([x](\cdot), [c](\cdot)\big)$

Constraint programming

Derivative constraint

Differential constraint:

- $\dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot)$
- one trajectory and its derivative



Constraint programming

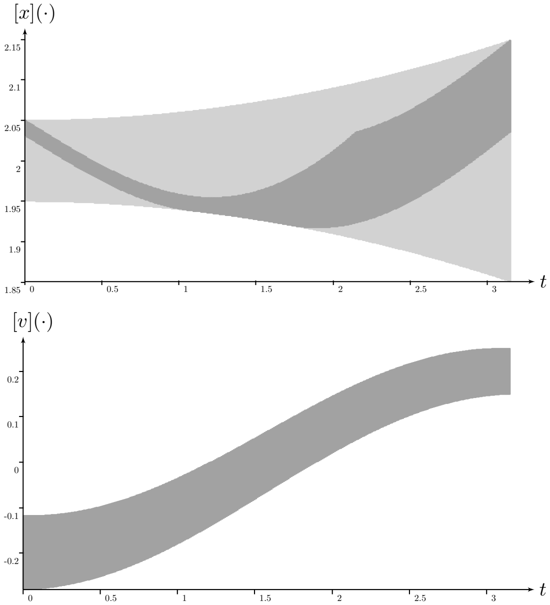
Derivative constraint

Differential constraint:

- $\dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot)$
- one trajectory and its derivative

Contractor programming:

- $\mathcal{C}_{\frac{d}{dt}}\big([\mathbf{x}](\cdot),[\mathbf{v}](\cdot)\big)$



Guaranteed computation of robot trajectories

Rohou, Jaulin, Mihaylova, Le Bars, Veres

Robotics and Autonomous Systems, 2017

Constraint programming

Evaluation constraint

Trajectory evaluation $\left\{ \mathbf{z} = \mathbf{y}(t) \right.$

Reliable non-linear state estimation involving time uncertainties
Rohou, Jaulin, Mihaylova, Le Bars, Veres *Automatica*, 2018

Constraint programming

Evaluation constraint

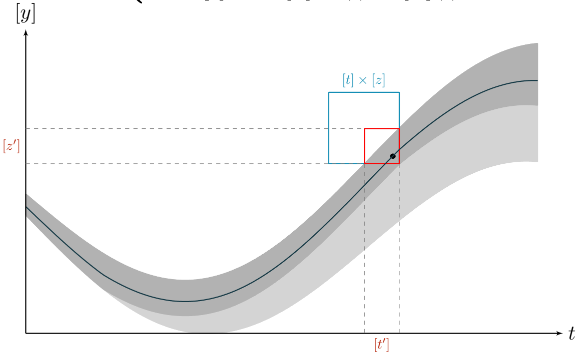
$$\text{Trajectory evaluation} \begin{cases} \mathbf{z} = \mathbf{y}(t) \\ \\ t \in [t], \mathbf{z} \in [\mathbf{z}], \mathbf{y}(\cdot) \in [\mathbf{y}](\cdot) \end{cases}$$

Reliable non-linear state estimation involving time uncertainties
Rohou, Jaulin, Mihaylova, Le Bars, Veres *Automatica*, 2018

Constraint programming

Evaluation constraint

Trajectory evaluation $\begin{cases} \mathbf{z} = \mathbf{y}(t) \\ t \in [t], \mathbf{z} \in [\mathbf{z}], \mathbf{y}(\cdot) \in [\mathbf{y}](\cdot) \end{cases}$

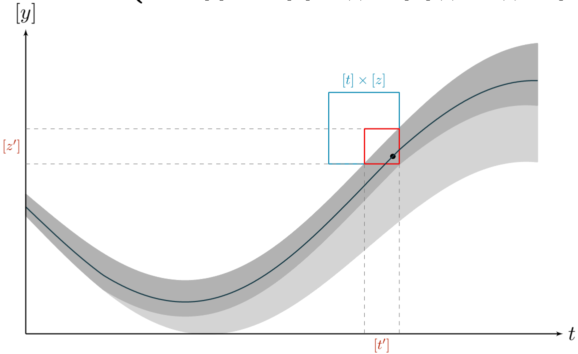


Reliable non-linear state estimation involving time uncertainties
Rohou, Jaulin, Mihaylova, Le Bars, Veres *Automatica*, 2018

Constraint programming

Evaluation constraint

Trajectory evaluation $\begin{cases} \mathbf{z} = \mathbf{y}(t) \\ \dot{\mathbf{y}}(\cdot) = \mathbf{w}(\cdot) \\ t \in [t], \mathbf{z} \in [\mathbf{z}], \mathbf{y}(\cdot) \in [\mathbf{y}](\cdot), \mathbf{w}(\cdot) \in [\mathbf{w}](\cdot) \end{cases}$



Contractor programming: $\mathcal{C}_{\text{eval}}\big([t], [\mathbf{z}], [\mathbf{y}](\cdot), [\mathbf{w}](\cdot)\big)$

Reliable non-linear state estimation involving time uncertainties
Rohou, Jaulin, Mihaylova, Le Bars, Veres *Automatica*, 2018

Constraint programming

Constraint programming for TBN

Using constraints to reduce the domains of the variables:

System:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \\ z_i = \mathbb{M}\big(\mathbf{x}(t_i)\big) \end{cases}$$

Elementary constraints decomposition:

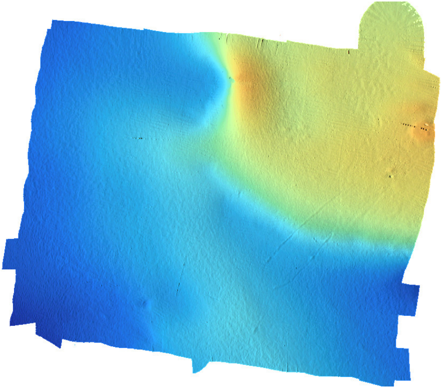
- $\mathbf{v}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big)$ → algebraic constraint ✓
- $\dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot)$ → derivative constraint → $\mathcal{L}_{\frac{d}{dt}}\big(\mathbf{x}(\cdot), \mathbf{v}(\cdot)\big)$ ✓
- $\mathbf{p}_i = \mathbf{x}(t_i)$ → evaluation constraint → $\mathcal{L}_{\text{eval}}\big(t_i, \mathbf{p}_i, \mathbf{x}(\cdot)\big)$ ✓
- $z_i = \mathbb{M}\big(\mathbf{p}_i\big)$ → $\mathcal{L}_{\mathbb{M}}$, related contractor $\mathcal{C}_{\mathbb{M}}$ to be defined

Constraint programming

The map constraint $\mathcal{L}_{\mathbb{M}}\big(z_i, \mathbf{p}_i\big) : z_i = \mathbb{M}(\mathbf{p}_i)$

Definition of the related operator $\mathcal{C}_{\mathbb{M}}([z_i], [\mathbf{p}_i])$:

$$\begin{pmatrix} [z_i] \\ [\mathbf{p}_i] \end{pmatrix} \mapsto \begin{pmatrix} [z_i] \cap \mathbb{M}([\mathbf{p}_i]) \\ \bigsqcup \big(\mathbf{p}_i \in [\mathbf{p}_i] \mid \mathbb{M}(\mathbf{p}_i) \in [z_i]\big) \end{pmatrix} \tag{1}$$

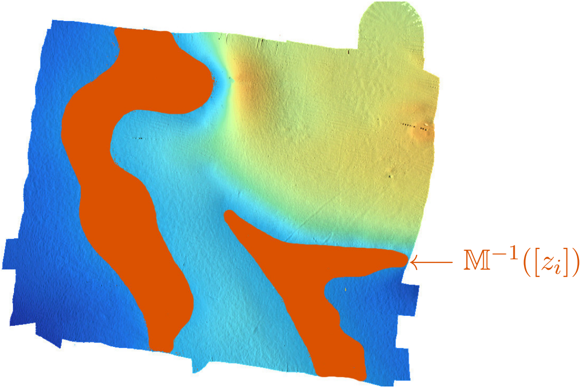


Constraint programming

The map constraint $\mathcal{L}_{\mathbb{M}}\big(z_i,\mathbf{p}_i\big)\,:\,z_i=\mathbb{M}(\mathbf{p}_i)$

Definition of the related operator $\mathcal{C}_{\mathbb{M}}([z_i],[\mathbf{p}_i])$:

$$\begin{pmatrix}[z_i]\\ [\mathbf{p}_i]\end{pmatrix}\mapsto\begin{pmatrix}[z_i]\cap\mathbb{M}([\mathbf{p}_i])\\ \bigsqcup\left(\mathbf{p}_i\in[\mathbf{p}_i]\mid\mathbb{M}(\mathbf{p}_i)\in[z_i]\right)\end{pmatrix}\qquad(1)$$

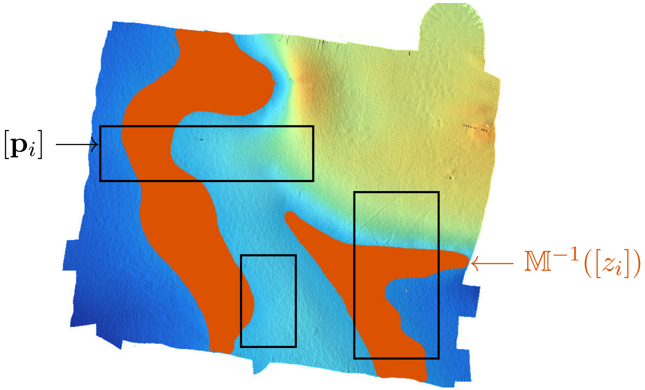


Constraint programming

The map constraint $\mathcal{L}_{\mathbb{M}}\big(z_i, \mathbf{p}_i\big) : z_i = \mathbb{M}(\mathbf{p}_i)$

Definition of the related operator $\mathcal{C}_{\mathbb{M}}([z_i], [\mathbf{p}_i])$:

$$\begin{pmatrix} [z_i] \\ [\mathbf{p}_i] \end{pmatrix} \mapsto \begin{pmatrix} [z_i] \cap \mathbb{M}([\mathbf{p}_i]) \\ \bigsqcup \big(\mathbf{p}_i \in [\mathbf{p}_i] \mid \mathbb{M}(\mathbf{p}_i) \in [z_i]\big) \end{pmatrix} \qquad (1)$$

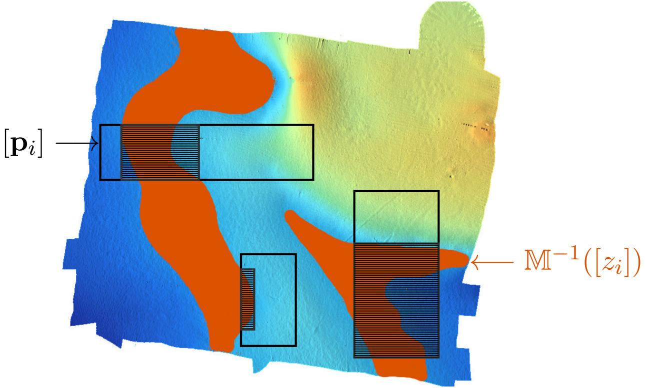


Constraint programming

The map constraint $\mathcal{L}_{\mathbb{M}}\big(z_i, \mathbf{p}_i\big) : z_i = \mathbb{M}(\mathbf{p}_i)$

Definition of the related operator $\mathcal{C}_{\mathbb{M}}([z_i], [\mathbf{p}_i])$:

$$\begin{pmatrix} [z_i] \\ [\mathbf{p}_i] \end{pmatrix} \mapsto \begin{pmatrix} [z_i] \cap \mathbb{M}([\mathbf{p}_i]) \\ \bigsqcup \left(\mathbf{p}_i \in [\mathbf{p}_i] \mid \mathbb{M}(\mathbf{p}_i) \in [z_i]\right) \end{pmatrix} \tag{1}$$



Constraint programming

TBN: resolution method

System:

$$\left\{\begin{array}{l}\dot{\mathbf{x}}(\cdot)=\mathbf{f}\big(\mathbf{x}(\cdot)\big)\\ z_i=\mathbb{M}\big(\mathbf{x}(t_i)\big)\end{array}\right.$$

Constraint programming

TBN: resolution method

System:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \\ z_i = \mathbb{M}\big(\mathbf{x}(t_i)\big) \end{cases}$$

Contractor programming:

- $\mathcal{C}_{\mathbf{f}}\big([\mathbf{x}](\cdot), [\mathbf{v}](\cdot)\big)$
- $\mathcal{C}_{\frac{d}{dt}}\big([\mathbf{x}](\cdot), [\mathbf{v}](\cdot)\big)$
- $\mathcal{C}_{\text{eval}}\big([t_i], [\mathbf{p}_i], [\mathbf{x}](\cdot)\big)$
- $\mathcal{C}_{\mathbb{M}}\big([z_i], [\mathbf{p}_i]\big)$

Section 3

Application

Application

Experimental mission with the Daurade AUV

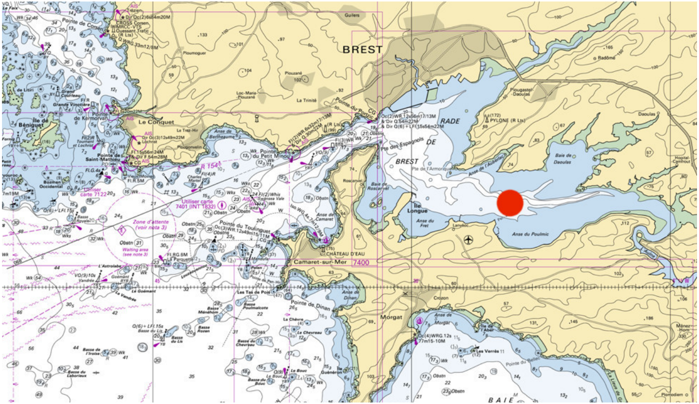
- 2 hours experimental mission
- *Rade de Brest*, Brittany

Location: *Polygone de Rascas* – Credits: Shom

Application

Experimental mission with the Daurade AUV

- 2 hours experimental mission
- *Rade de Brest*, Brittany



Location: *Polygone de Rascas* – Credits: Shom

Application

Experimental mission with the Daurade AUV

- Daurade: Autonomous Underwater Vehicle (AUV)
- weight: 1010kg – length: 5m – max depth: 300m

Special thanks to DGA-TN Brest (formerly GESMA)

Application

Experimental mission with the Daurade AUV

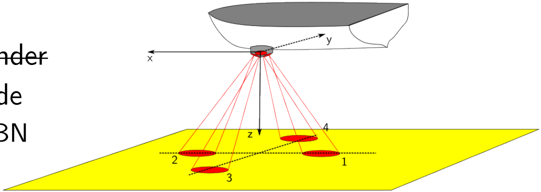
Mission data

Proprioceptive sensors:

- Inertial Navigation System (INS): Euler angles, accelerations
- Doppler Velocity Log (DVL): absolute velocities

Exteroceptive sensors:

- ~~single beam echo-sounder~~
- DVL $\rightarrow$ vehicle altitude
  - not suitable for TBN

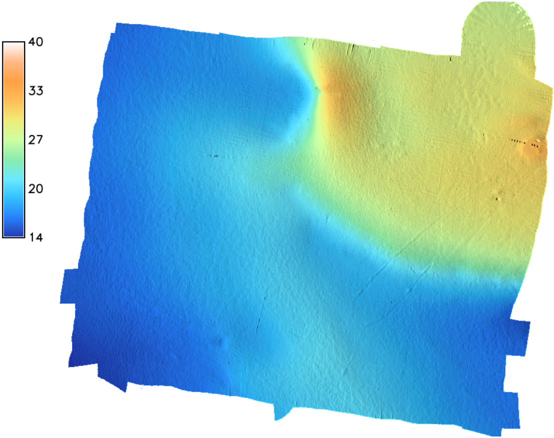


Double Janus DVL with 4 beams (image: Michel Legris)

Application

Map $\mathbb{M}$ of the environment

Prior knowledge: bathymetric map of Rascas, denoted $\mathbb{M}$

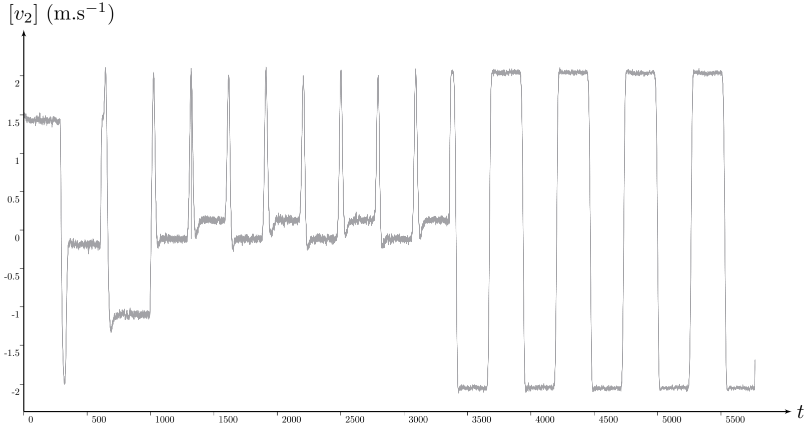


Thanks to Irène Mopin, Pierre Simon,
Romain Schwab, Michel Legris, Rodéric Moitié

Application

Evolution measurements

- velocity measurements obtained with a DVL
- considering uncertainties, building a tube $[\mathbf{v}](\cdot)$

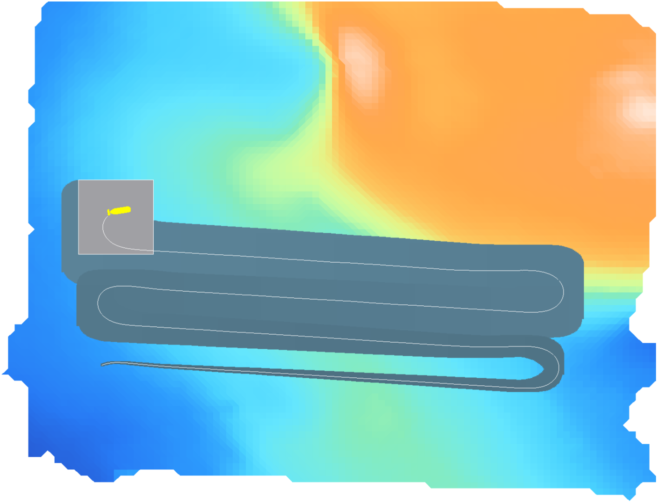


North speed velocity tube $[v_2](\cdot)$

Application

Dead reckoning from evolution measurements only

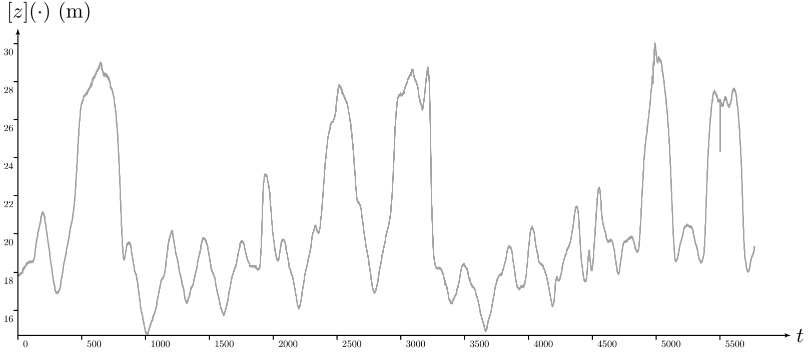
Video



Application

Observations measurements: bathymetric values

- DVL, same sensor, can provide **altitude measurements** z_{alt}
- pressure sensor: depth values z_{depth}
- time-dependent measurements, use of **tide models**
- $z = z_{\text{alt}} + z_{\text{depth}} + z_{\text{tide}}$

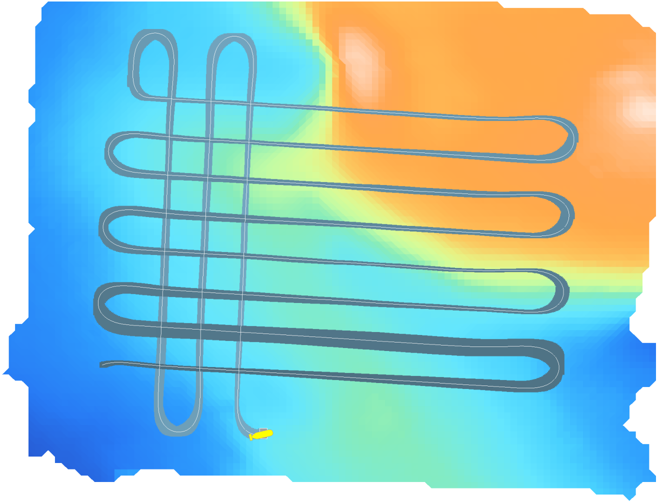


Observation tube $[z](\cdot)$

Application

Full example of TBN

Video



Section 4

Conclusions

Conclusions

Assets of constraint programming

- **simplicity** of the approach
  transparent application of contractors on elementary constraints

Conclusions

Assets of constraint programming

- **simplicity** of the approach
  transparent application of contractors on elementary constraints
- **reliability** of the results: no solution can be lost
  useful for proof purposes and the safety of systems

Conclusions

Assets of constraint programming

- **simplicity** of the approach
  transparent application of contractors on elementary constraints
- **reliability** of the results: no solution can be lost
  useful for proof purposes and the safety of systems
- focus on **the *what* instead of the *how***
  no expertise required on how to solve a problem

Conclusions

Assets of constraint programming

- **simplicity** of the approach
  transparent application of contractors on elementary constraints
- **reliability** of the results: no solution can be lost
  useful for proof purposes and the safety of systems
- focus on **the *what* instead of the *how***
  no expertise required on how to solve a problem
- **complex systems** easily handled
  non-linearities, differential equations, values from datasets

Conclusions

Assets of constraint programming

- **simplicity** of the approach
  transparent application of contractors on elementary constraints
- **reliability** of the results: no solution can be lost
  useful for proof purposes and the safety of systems
- focus on **the *what* instead of the *how***
  no expertise required on how to solve a problem
- **complex systems** easily handled
  non-linearities, differential equations, values from datasets

Tubex library: open-source library providing tools for constraint programming over dynamical systems

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

Conclusions

About this TBN method

- **fast implementation** with constraint programming
  use of already existing contractors, one more to be defined

Conclusions

About this TBN method

- **fast implementation** with constraint programming
  use of already existing contractors, one more to be defined
- **reliability** of the results
  guaranteed enclosure of the actual trajectory

Conclusions

About this TBN method

- **fast implementation** with constraint programming
  use of already existing contractors, one more to be defined
- **reliability** of the results
  guaranteed enclosure of the actual trajectory
- based on **poor observations**
  scalar bathymetric values, *e.g.* from a single beam echo sounder

Conclusions

About this TBN method

- **fast implementation** with constraint programming
  use of already existing contractors, one more to be defined
- **reliability** of the results
  guaranteed enclosure of the actual trajectory
- based on **poor observations**
  scalar bathymetric values, *e.g.* from a single beam echo sounder
- **no initial condition** required
  the constraints apply in any order up to a fixed point

Conclusions

About this TBN method

- **fast implementation** with constraint programming
  use of already existing contractors, one more to be defined
- **reliability** of the results
  guaranteed enclosure of the actual trajectory
- based on **poor observations**
  scalar bathymetric values, *e.g.* from a single beam echo sounder
- **no initial condition** required
  the constraints apply in any order up to a fixed point

Todo:

- use this algorithm in real time
  AUV equipped with a DVL and a sonar

Conclusions

About this TBN method

- **fast implementation** with constraint programming
  use of already existing contractors, one more to be defined
- **reliability** of the results
  guaranteed enclosure of the actual trajectory
- based on **poor observations**
  scalar bathymetric values, *e.g.* from a single beam echo sounder
- **no initial condition** required
  the constraints apply in any order up to a fixed point

Todo:

- use this algorithm in real time
  AUV equipped with a DVL and a sonar
- find the *Cordelière*