Dealing with evaluation constraints in uncertain dynamical systems

Simon Rohou[1], Luc Jaulin[2], Lyudmila Mihaylova[3], Fabrice Le Bars[2], Sandor M. Veres[3]

[1] IMT Atlantique, LS2N (OGRE), Nantes, France
[2] ENSTA Bretagne, Lab-STICC, Brest, France
[3] University of Sheffield, Sheffield, UK
simon.rohou@imt-atlantique.fr

SCAN
12th September 2018, Tokyo

Section 1

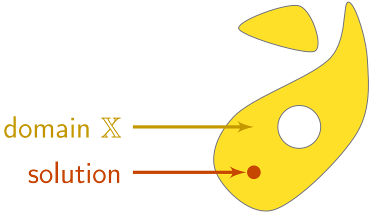
Constraint programming over dynamical systems

Constraint programming over dynamical systems

Constraint programming in a nutshell

Example in $\mathbb{R}^2$:

- system solving described by a *constraint network*
- **variables** (vectors $\mathbf{x} \in \mathbb{R}^n$) belonging to **domains** $\mathbb{X}$



Constraint network:

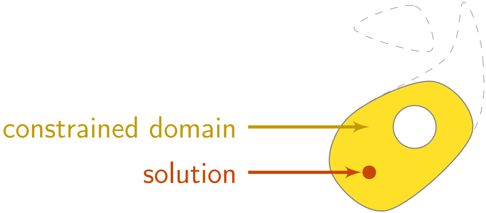
Variables: $\mathbf{x}$
Domains: $\mathbb{X}$
Constraints:

Constraint programming over dynamical systems

Constraint programming in a nutshell

Example in $\mathbb{R}^2$:

- system solving described by a *constraint network*
- **variables** (vectors $\mathbf{x} \in \mathbb{R}^n$) belonging to **domains** $\mathbb{X}$
- continuous **constraints** $\mathcal{L}$: non-linear equations, inequalities, ...



Constraint network:

Variables: $\mathbf{x}$

Domains: $\mathbb{X}$

Constraints:

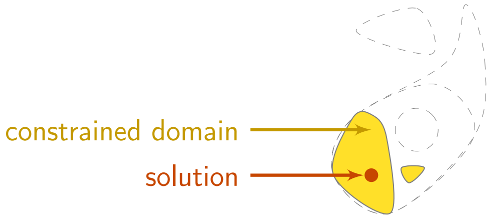
1. $\mathcal{L}_1(\mathbf{x})$

Constraint programming over dynamical systems

Constraint programming in a nutshell

Example in $\mathbb{R}^2$:

- system solving described by a *constraint network*
- **variables** (vectors $\mathbf{x} \in \mathbb{R}^n$) belonging to **domains** $\mathbb{X}$
- continuous **constraints** $\mathcal{L}$: non-linear equations, inequalities, ...



Constraint network:

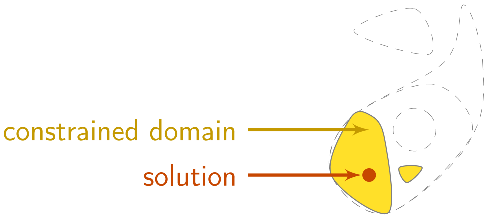
$$\begin{cases} \textbf{Variables: } \mathbf{x} \\ \textbf{Domains: } \mathbb{X} \\ \textbf{Constraints:} \\ \quad 1.\ \mathcal{L}_1(\mathbf{x}) \\ \quad 2.\ \mathcal{L}_2(\mathbf{x}) \end{cases}$$

Constraint programming over dynamical systems

Constraint programming in a nutshell

Example in $\mathbb{R}^2$:

- system solving described by a *constraint network*
- **variables** (vectors $\mathbf{x} \in \mathbb{R}^n$) belonging to **domains** $\mathbb{X}$
- continuous **constraints** $\mathcal{L}$: non-linear equations, inequalities, ...



Constraint network:

Variables: $\mathbf{x}$

Domains: $\mathbb{X}$

Constraints:

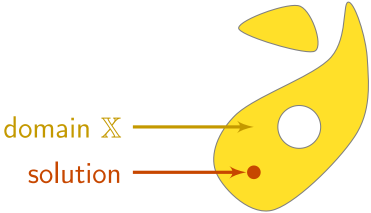
1. $\mathcal{L}_1(\mathbf{x})$
2. $\mathcal{L}_2(\mathbf{x})$
3. ...

Constraint programming over dynamical systems

Constraint programming in a nutshell

Example in $\mathbb{R}^2$:

- system solving described by a *constraint network*
- **variables** (vectors $\mathbf{x} \in \mathbb{R}^n$) belonging to **domains** $\mathbb{X}$
- continuous **constraints** $\mathcal{L}$: non-linear equations, inequalities, ...



Constraint network:

Variables: $\mathbf{x}$
Domains: $\mathbb{X}$
Constraints:

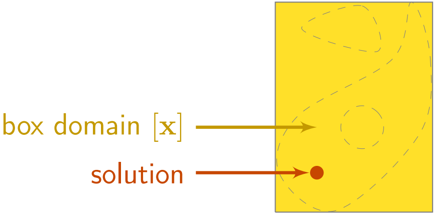
1. $\mathcal{L}_1(\mathbf{x})$
2. $\mathcal{L}_2(\mathbf{x})$
3. ...

Constraint programming over dynamical systems

Constraint programming in a nutshell

Example in $\mathbb{R}^2$:

- system solving described by a *constraint network*
- **variables** (vectors $\mathbf{x} \in \mathbb{R}^n$) belonging to **domains** $\mathbb{X}$
- continuous **constraints** $\mathcal{L}$: non-linear equations, inequalities, ...
- representable domains: interval-vectors $[\mathbf{x}] \in \mathbb{IR}^n$



Constraint network:

Variables: $\mathbf{x}$

Domains: $\mathbb{X} \subseteq [\mathbf{x}]$

Constraints:

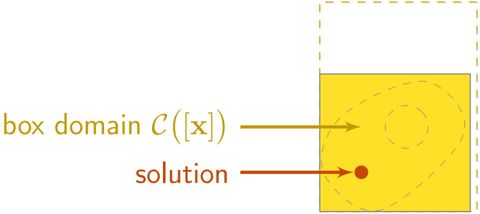
1. $\mathcal{L}_1(\mathbf{x})$
2. $\mathcal{L}_2(\mathbf{x})$
3. ...

Constraint programming over dynamical systems

Constraint programming in a nutshell

Example in $\mathbb{R}^2$:

- system solving described by a *constraint network*
- **variables** (vectors $\mathbf{x} \in \mathbb{R}^n$) belonging to **domains** $\mathbb{X}$
- continuous **constraints** $\mathcal{L}$: non-linear equations, inequalities, ...
- representable domains: interval-vectors $[\mathbf{x}] \in \mathbb{IR}^n$
- resolution by **contractors**, $\mathcal{C}_{\mathcal{L}}([\mathbf{x}])$



Constraint network:

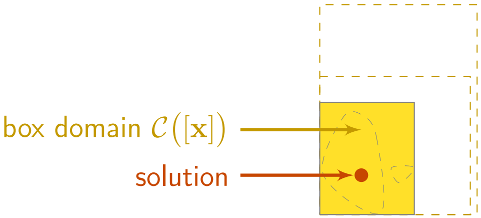
$$\left\{ \begin{array}{l} \textbf{Variables: } \mathbf{x} \\ \textbf{Domains: } \mathbb{X} \subseteq [\mathbf{x}] \\ \textbf{Constraints:} \\ \quad 1.\ \mathcal{L}_1(\mathbf{x}) \rightarrow \mathcal{C}_1([\mathbf{x}]) \\ \quad 2.\ \mathcal{L}_2(\mathbf{x}) \\ \quad 3.\ \ldots \end{array} \right.$$

Constraint programming over dynamical systems

Constraint programming in a nutshell

Example in $\mathbb{R}^2$:

- system solving described by a *constraint network*
- **variables** (vectors $\mathbf{x} \in \mathbb{R}^n$) belonging to **domains** $\mathbb{X}$
- continuous **constraints** $\mathcal{L}$: non-linear equations, inequalities, ...
- representable domains: interval-vectors $[\mathbf{x}] \in \mathbb{IR}^n$
- resolution by **contractors**, $\mathcal{C}_{\mathcal{L}}([\mathbf{x}])$



Constraint network:

Variables: $\mathbf{x}$

Domains: $\mathbb{X} \subseteq [\mathbf{x}]$

Constraints:

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

Constraint programming over dynamical systems

Extension to dynamical systems

Only few work on **constraints for dynamical systems**:

- Hickey 2000
- Janssen, Van Hentenryck, and Deville 2002
- Cruz and Barahona 2003

Constraint programming over dynamical systems

Extension to dynamical systems

Only few work on **constraints for dynamical systems**:

- Hickey 2000
- Janssen, Van Hentenryck, and Deville 2002
- Cruz and Barahona 2003

New approach:

- variables: **trajectories**, $\mathbf{x}(\cdot) : \mathbb{R} \to \mathbb{R}^n$
- domains: **tubes**, $[\mathbf{x}](\cdot) : \mathbb{R} \to \mathbb{IR}^n$

Set-membership state estimation with fleeting data
F. Le Bars, J. Sliwka, L. Jaulin, O. Reynet *Automatica*, 2012

Solving Non-Linear Constraint Satisfaction Problems Involving Time-Dependant Functions
A. Bethencourt, L. Jaulin. *Mathematics in Computer Science*, 2014

Constraint programming over dynamical systems

Extension to dynamical systems

Only few work on **constraints for dynamical systems**:

- Hickey 2000
- Janssen, Van Hentenryck, and Deville 2002
- Cruz and Barahona 2003

New approach:

- variables: **trajectories**, $\mathbf{x}(\cdot):\mathbb{R}\to\mathbb{R}^n$
- domains: **tubes**, $[\mathbf{x}](\cdot):\mathbb{R}\to\mathbb{IR}^n$

Set-membership state estimation with fleeting data
F. Le Bars, J. Sliwka, L. Jaulin, O. Reynet *Automatica*, 2012

Solving Non-Linear Constraint Satisfaction Problems Involving Time-Dependant Functions
A. Bethencourt, L. Jaulin. *Mathematics in Computer Science*, 2014

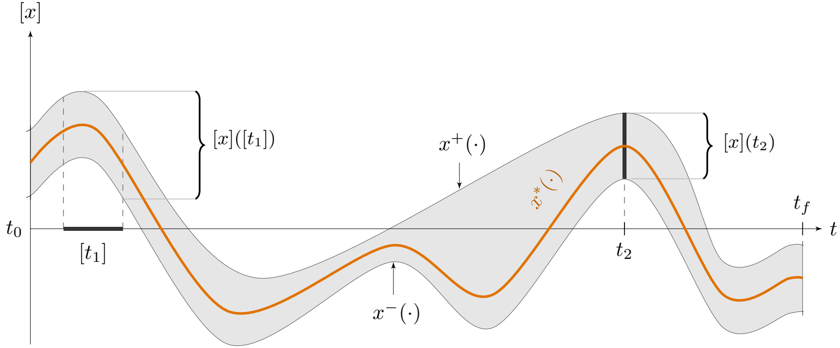
Our contribution:

- develop **primitive dynamical contractors**

Constraint programming over dynamical systems

Tubes

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

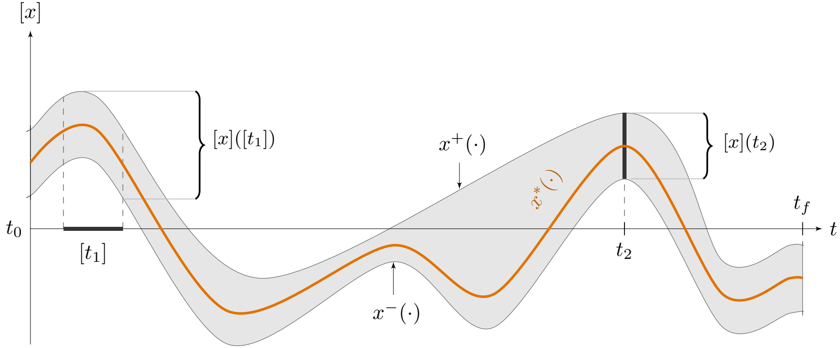


Tube $[x](\cdot)$ enclosing an uncertain trajectory $x^*(\cdot)$

Constraint programming over dynamical systems

Tubes

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

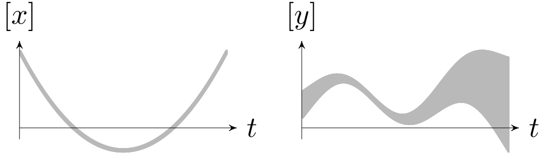


Tube $[x](\cdot)$ enclosing an uncertain trajectory $x^*(\cdot)$

- dot notation $(\cdot)$

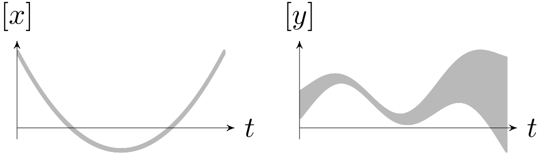
Constraint programming over dynamical systems

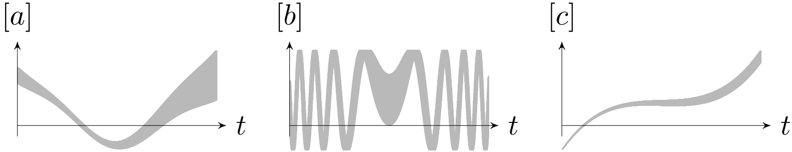
Tubes arithmetic



Constraint programming over dynamical systems

Tubes arithmetic





$[a](\cdot) = [x](\cdot) + [y](\cdot)$ $\quad$ $[b](\cdot) = \sin\big([x](\cdot)\big)$ $\quad$ $[c](\cdot) = \int_0^{\cdot} [x](\tau)d\tau$

Constraint programming over dynamical systems

Tube contractor

Contractor on boxes can be extended to sets of trajectories (tubes).

Definition

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:

Constraint programming over dynamical systems

Tube contractor

Contractor on boxes can be extended to sets of trajectories (tubes).

Definition

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:

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

Constraint programming over dynamical systems

Tube contractor

Contractor on boxes can be extended to sets of trajectories (tubes).

Definition

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:

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

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

Constraint programming over dynamical systems

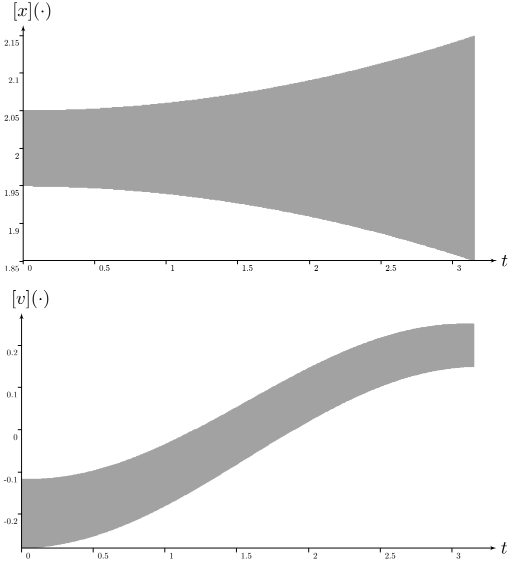
Constraint $\dot{x}(\cdot) = v(\cdot)$

Differential constraint:

$$\mathcal{L}_{\frac{d}{dt}}\big(x(\cdot), v(\cdot)\big) : \dot{x}(\cdot) = v(\cdot)$$

Constraint programming over dynamical systems

Constraint $\dot{x}(\cdot) = v(\cdot)$



Differential constraint:

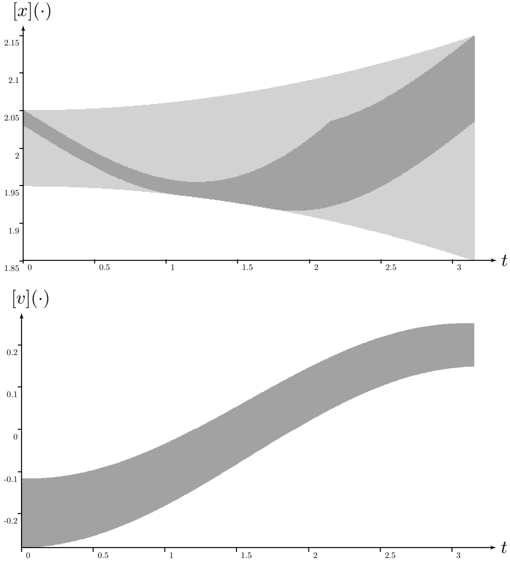
$$\mathcal{L}_{\frac{d}{dt}}\big(x(\cdot), v(\cdot)\big) : \dot{x}(\cdot) = v(\cdot)$$

Related contractor $\mathcal{C}_{\frac{d}{dt}}$:

- $x(\cdot) \in [x](\cdot)$
- $v(\cdot) \in [v](\cdot)$

Constraint programming over dynamical systems

Constraint $\dot{x}(\cdot) = v(\cdot)$



Differential constraint:

$$\mathcal{L}_{\frac{d}{dt}}\big(x(\cdot), v(\cdot)\big) : \dot{x}(\cdot) = v(\cdot)$$

Related contractor $\mathcal{C}_{\frac{d}{dt}}$:

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

Constraint programming over dynamical systems

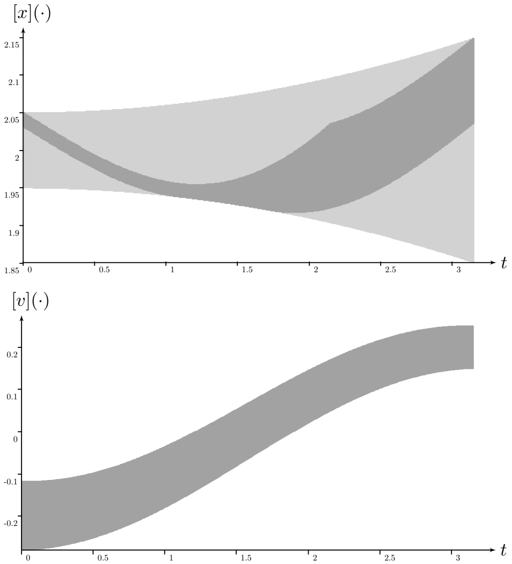
Constraint $\dot{x}(\cdot) = v(\cdot)$

Differential constraint:

$$\mathcal{L}_{\frac{d}{dt}}\big(x(\cdot), v(\cdot)\big) : \dot{x}(\cdot) = v(\cdot)$$

Related contractor $\mathcal{C}_{\frac{d}{dt}}$:

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



Guaranteed computation of robot trajectories

Rohou, Jaulin, Mihaylova, Le Bars, Veres

Robotics and Autonomous Systems, 2017

Constraint programming over dynamical systems

State estimation

Classical formalization:

$$\left\{ \begin{array}{ll} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot), \mathbf{u}(\cdot)\big) & \qquad\qquad \text{(evolution)} \\ z = g\big(\mathbf{x}(t)\big) & \qquad\qquad \text{(observations)} \end{array} \right.$$

Constraint programming over dynamical systems

State estimation

Classical formalization:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot), \mathbf{u}(\cdot)\big) & \text{(evolution)} \\ z = g\big(\mathbf{x}(t)\big) & \text{(observations)} \end{cases}$$

Decomposition:

1. $\mathbf{v}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot), \mathbf{u}(\cdot)\big)$
2. $\dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot)$
3. $y(\cdot) = g(\mathbf{x}(\cdot))$
4. $z = y(t)$

Constraint programming over dynamical systems

State estimation

Classical formalization:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot), \mathbf{u}(\cdot)\big) & \text{(evolution)} \\ z = g\big(\mathbf{x}(t)\big) & \text{(observations)} \end{cases}$$

Decomposition:

1. $\mathbf{v}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot), \mathbf{u}(\cdot)\big)$
2. $\dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot)$
3. $y(\cdot) = g(\mathbf{x}(\cdot))$
4. $z = y(t)$

Constraints:

1. $\mathcal{L}_{\mathbf{f}}\big(\mathbf{v}(\cdot), \mathbf{x}(\cdot), \mathbf{u}(\cdot)\big)$ (arithmetic composition)
2. $\mathcal{L}_{\frac{d}{dt}}\big(\mathbf{x}(\cdot), \mathbf{v}(\cdot)\big)$
3. $\mathcal{L}_{g}\big(y(\cdot), \mathbf{x}(\cdot)\big)$ (arithmetic composition)

Constraint programming over dynamical systems

State estimation

Classical formalization:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot), \mathbf{u}(\cdot)\big) & \text{(evolution)} \\ z = g\big(\mathbf{x}(t)\big) & \text{(observations)} \end{cases}$$

Decomposition:

1. $\mathbf{v}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot), \mathbf{u}(\cdot)\big)$
2. $\dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot)$
3. $y(\cdot) = g(\mathbf{x}(\cdot))$
4. $z = y(t)$

Constraints:

1. $\mathcal{L}_{\mathbf{f}}\big(\mathbf{v}(\cdot), \mathbf{x}(\cdot), \mathbf{u}(\cdot)\big)$ (arithmetic composition)
2. $\mathcal{L}_{\frac{d}{dt}}\big(\mathbf{x}(\cdot), \mathbf{v}(\cdot)\big)$
3. $\mathcal{L}_{g}\big(y(\cdot), \mathbf{x}(\cdot)\big)$ (arithmetic composition)
4. $\mathcal{L}_{\text{eval}}\big(t, z, y(\cdot)\big)$

Constraint programming over dynamical systems

State estimation

Classical formalization:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot), \mathbf{u}(\cdot)\big) & \text{(evolution)} \\ z = g\big(\mathbf{x}(t)\big) & \text{(observations)} \end{cases}$$

Decomposition:

1. $\mathbf{v}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot), \mathbf{u}(\cdot)\big)$
2. $\dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot)$
3. $y(\cdot) = g(\mathbf{x}(\cdot))$
4. $z = y(t)$

Constraints $\longrightarrow$ Contractors:

1. $\mathcal{C}_{\mathbf{f}}\big(\mathbf{v}(\cdot), \mathbf{x}(\cdot), \mathbf{u}(\cdot)\big)$ (arithmetic composition)
2. $\mathcal{C}_{\frac{d}{dt}}\big(\mathbf{x}(\cdot), \mathbf{v}(\cdot)\big)$
3. $\mathcal{C}_g\big(y(\cdot), \mathbf{x}(\cdot)\big)$ (arithmetic composition)
4. $\mathcal{C}_{\text{eval}}\big(t, z, y(\cdot)\big)$

Section 2

Constraint $\mathcal{L}_{\text{eval}}$: $z = y(t)$

Constraint $\mathcal{L}_{\text{eval}}$: $z = y(t)$

Definition

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

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

Reliable non-linear state estimation involving time uncertainties
S. Rohou, L. Jaulin, L. Mihaylova, F. Le Bars, S. M. Veres. *Automatica*, 2018

Constraint $\mathcal{L}_{\text{eval}}$: $z = y(t)$

Definition

$$\mathcal{L}_{\text{eval}} : \begin{cases} \textbf{Variables: } t,\ z,\ y(\cdot),\ w(\cdot) \\ \textbf{Domains: } [t],\ [z],\ [y](\cdot),\ [w](\cdot) \\ \textbf{Constraints:} \\ \quad 1.\ z = y(t) \\ \quad 2.\ \dot{y}(\cdot) = w(\cdot) \end{cases}$$

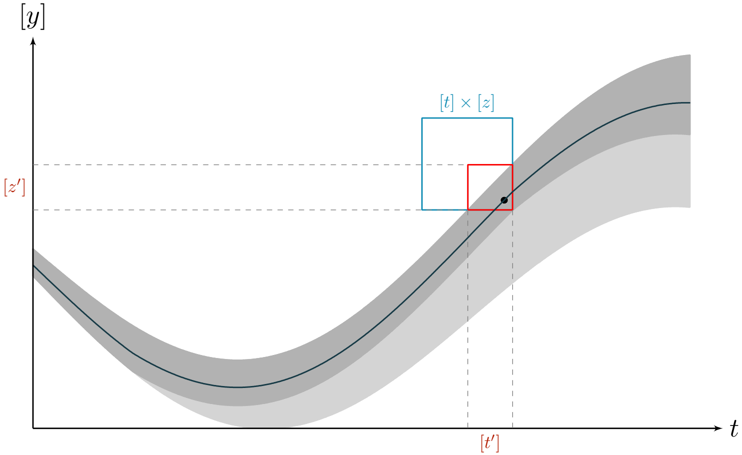
$\mathcal{L}_{\text{eval}}$ equivalent to:

$\exists t \in [t],\ \exists z \in [z],\ \exists y(\cdot) \in [y](\cdot) \ \mid\ z = y(t)$

Reliable non-linear state estimation involving time uncertainties
S. Rohou, L. Jaulin, L. Mihaylova, F. Le Bars, S. M. Veres. *Automatica*, 2018

Constraint $\mathcal{L}_{\text{eval}}$: $z = y(t)$

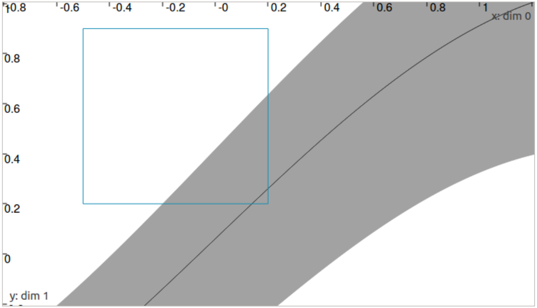
Contractor $\mathcal{C}_{\text{eval}}$: illustration



Bounded evaluation with contractions of $[y](\cdot)$ and both $[t]$ and $[z]$ by means of $\mathcal{C}_{\text{eval}}$. The tube's contracted part is depicted in light gray.

Constraint $\mathcal{L}_{\text{eval}}$: $z = y(t)$

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

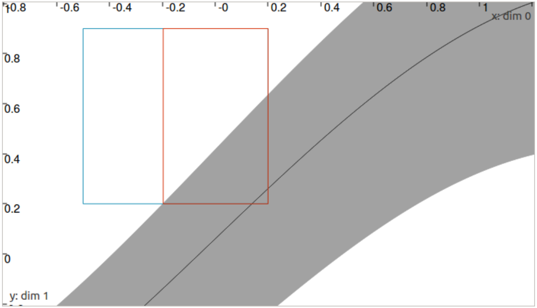


Definition:

$$\begin{pmatrix} [t] \\ [z] \\ [y](\cdot) \\ [w](\cdot) \end{pmatrix} \xmapsto{\mathcal{C}_{\text{eval}}} \begin{pmatrix} \\ \\ \\ \end{pmatrix}$$

Constraint $\mathcal{L}_{\text{eval}}$: $z = y(t)$

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

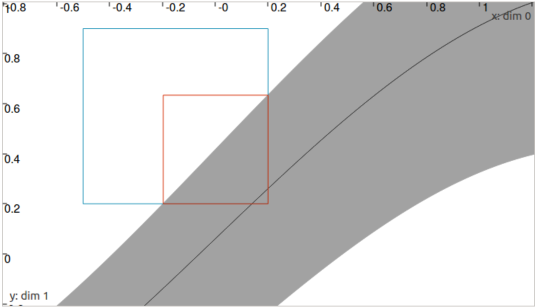


Definition:

$$\begin{pmatrix} [t] \\ [z] \\ [y](\cdot) \\ [w](\cdot) \end{pmatrix} \xmapsto{\mathcal{C}_{\text{eval}}} \begin{pmatrix} [t] \cap [y]^{-1}([z]) \\ \\ \\ \\ \end{pmatrix}$$

Constraint $\mathcal{L}_{\text{eval}}$: $z = y(t)$

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

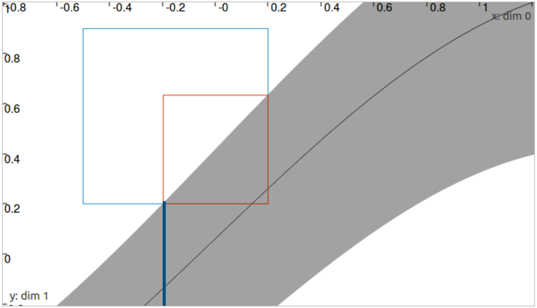


Definition:

$$\begin{pmatrix} [t] \\ [z] \\ [y](\cdot) \\ [w](\cdot) \end{pmatrix} \xmapsto{\mathcal{C}_{\text{eval}}} \begin{pmatrix} [t] \cap [y]^{-1}([z]) \\ [z] \cap [y]([t]) \\ \\ \\ \end{pmatrix}$$

Constraint $\mathcal{L}_{\text{eval}}$: $z = y(t)$

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

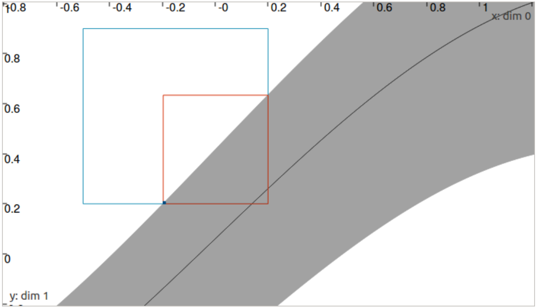


Definition:

$$\begin{pmatrix}[t]\\ [z]\\ [y](\cdot)\\ [w](\cdot)\end{pmatrix} \xmapsto{\mathcal{C}_{\text{eval}}} \begin{pmatrix}[t]\cap[y]^{-1}([z])\\ [z]\cap[y]([t])\\ \big([y](t_1)\qquad\big)\\ \end{pmatrix}$$

Constraint $\mathcal{L}_{\text{eval}}$: $z = y(t)$

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

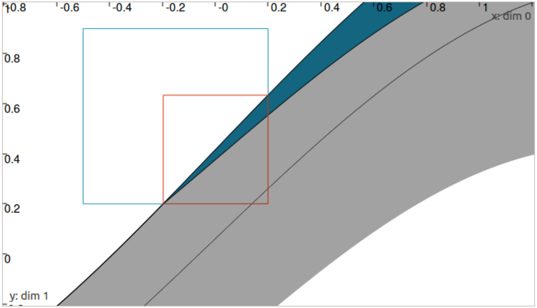


Definition:

$$\begin{pmatrix} [t] \\ [z] \\ [y](\cdot) \\ [w](\cdot) \end{pmatrix} \xmapsto{\mathcal{C}_{\text{eval}}} \begin{pmatrix} [t] \cap [y]^{-1}([z]) \\ [z] \cap [y]([t]) \\ \Big(([y](t_1) \cap [z]) \qquad\qquad \Big) \\ \\ \end{pmatrix}$$

Constraint $\mathcal{L}_{\text{eval}}$: $z = y(t)$

$$\mathcal{C}_{\text{eval}}\Big([t], [z], [y](\cdot), [w](\cdot)\Big)$$

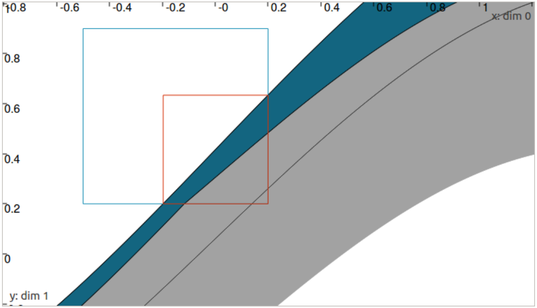


Definition:

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

Constraint $\mathcal{L}_{\text{eval}}$: $z = y(t)$

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

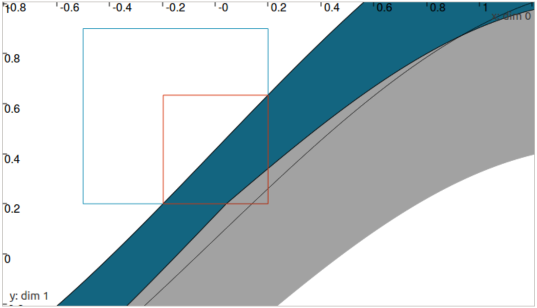


Definition:

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

Constraint $\mathcal{L}_{\text{eval}}$: $z = y(t)$

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

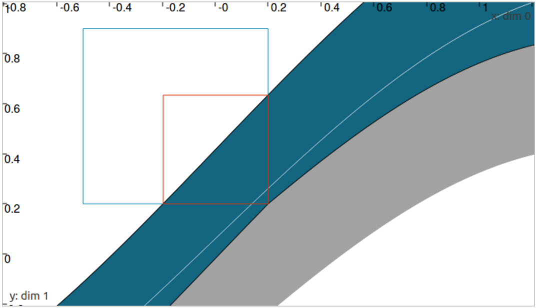


Definition:

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

Constraint $\mathcal{L}_{\text{eval}}$: $z = y(t)$

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

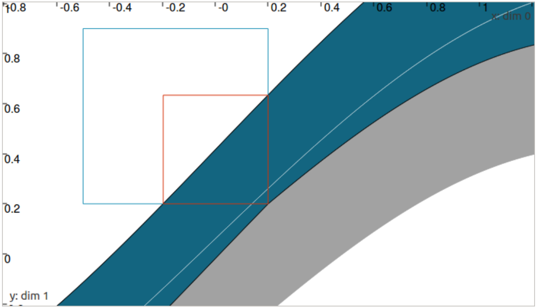


Definition:

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

Constraint $\mathcal{L}_{\text{eval}}$: $z = y(t)$

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

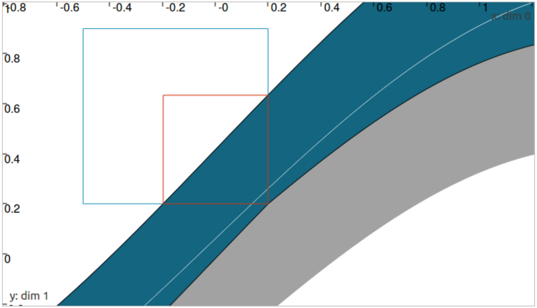


Definition:

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

Constraint $\mathcal{L}_{\text{eval}}$: $z = y(t)$

$$\mathcal{C}_{\text{eval}}\Big([t],[z],[y](\cdot),[w](\cdot)\Big)$$



Definition:

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

Section 3

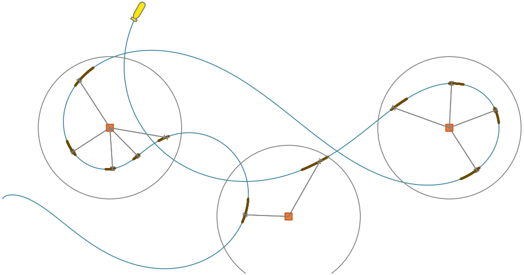
Application: robot localization

Application: robot localization

State estimation based on low-cost beacons

Robot $\mathcal{R}$ evolving amongst low-cost beacons $\mathcal{B}_k$:

- initial value unknown
- discrete set of range-only measurements



- beacons $\mathcal{B}_k$
- $\mathcal{R}$'s unknown trajectory $\mathbf{x}^*$

Bounded measurements:

- command vector $\mathbf{u}(\cdot) \in [\mathbf{u}](\cdot)$
- range values $z_i \in [z_i]$ (distance $\mathcal{R} \leftrightarrow \mathcal{B}_k$, discrete observations)
- related time measurements $t_i \in [t_i]$

Application: robot localization

State equations — $\mathbf{x} = \{x_1, x_2, \psi, \vartheta\}^\intercal$

1. **Evolution state equation**, $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$
   System modeled by the following evolution function:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 = \dot{\psi} \\ \dot{x}_4 = \dot{\vartheta} \end{pmatrix} \xmapsto{\mathbf{f}} \begin{pmatrix} \vartheta \cos(\psi) \\ \vartheta \sin(\psi) \\ u_1 \\ u_2 \end{pmatrix} \tag{1}$$

Application: robot localization

State equations — $\mathbf{x} = \{x_1, x_2, \psi, \vartheta\}^\intercal$

1. **Evolution state equation**, $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$
   System modeled by the following evolution function:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 = \dot{\psi} \\ \dot{x}_4 = \dot{\vartheta} \end{pmatrix} \xmapsto{\mathbf{f}} \begin{pmatrix} \vartheta\cos(\psi) \\ \vartheta\sin(\psi) \\ u_1 \\ u_2 \end{pmatrix} \tag{1}$$

   Input $\mathbf{u}(t) \in \mathbb{R}^2$, bounded as:

$$\mathbf{u}(t) \in [\mathbf{u}](t) = \begin{pmatrix} -9/20\cos(t/5) \\ 1/10 + \sin(t/4) \end{pmatrix} + \frac{1}{1000}\begin{pmatrix} [-1,1] \\ [-1,1] \end{pmatrix} \tag{2}$$

Application: robot localization

State equations — $\mathbf{x} = \{x_1, x_2, \psi, \vartheta\}^\intercal$

1. **Evolution state equation**, $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$
   System modeled by the following evolution function:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 = \dot{\psi} \\ \dot{x}_4 = \dot{\vartheta} \end{pmatrix} \xmapsto{\mathbf{f}} \begin{pmatrix} \vartheta \cos(\psi) \\ \vartheta \sin(\psi) \\ u_1 \\ u_2 \end{pmatrix} \tag{1}$$

   Input $\mathbf{u}(t) \in \mathbb{R}^2$, bounded as:

$$\mathbf{u}(t) \in [\mathbf{u}](t) = \begin{pmatrix} -{}^{9}\!/_{20} \cos({}^{t}\!/_{5}) \\ {}^{1}\!/_{10} + \sin({}^{t}\!/_{4}) \end{pmatrix} + \frac{1}{1000} \begin{pmatrix} [-1, 1] \\ [-1, 1] \end{pmatrix} \tag{2}$$

2. **Observation state equation**, $z_i = g_k(\mathbf{x}(t_i))$
   Distance function:

$$g_k(\mathbf{x}) = \sqrt{(x_1 - b_{k,1})^2 + (x_2 - b_{k,2})^2} \tag{3}$$

Application: robot localization

Beacons' location and list of measurements $([t_i],[z_i])$

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

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	$[56.75,57.55]$	$[17.45,18.45]$

Application: robot localization

Variables:

Domains:

Constraints:

Note: initial position $(x_1(0), x_2(0))$ not priorly known.

Application: robot localization

Variables:
$\mathbf{u}(\cdot), \mathbf{v}(\cdot), \mathbf{x}(\cdot)$

Domains:
$[\mathbf{u}](\cdot), [\mathbf{v}](\cdot), [\mathbf{x}](\cdot)$

Constraints:

1. State evolution:
   - $\mathbf{v}(\cdot) = \mathbf{f}(\mathbf{x}(\cdot), \mathbf{u}(\cdot))$
   - $\dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot)$

Note: initial position $(x_1(0), x_2(0))$ not priorly known.

Application: robot localization

Variables:
$\mathbf{u}(\cdot), \mathbf{v}(\cdot), \mathbf{x}(\cdot), \{y_k(\cdot)\}$

Domains:
$[\mathbf{u}](\cdot), [\mathbf{v}](\cdot), [\mathbf{x}](\cdot), \{[y_k](\cdot)\}$

Constraints:

1. State evolution:
   - $\mathbf{v}(\cdot) = \mathbf{f}(\mathbf{x}(\cdot), \mathbf{u}(\cdot))$
   - $\dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot)$
2. Beacon-robot distance function:
   - $y_k(\cdot) = \sqrt{(x_1(\cdot) - b_{k,1})^2 + (x_2(\cdot) - b_{k,2})^2}$

Note: initial position $(x_1(0), x_2(0))$ not priorly known.

Application: robot localization

Variables:
$\mathbf{u}(\cdot), \mathbf{v}(\cdot), \mathbf{x}(\cdot), \{y_k(\cdot)\}, \{(t_i, z_i)\}$

Domains:
$[\mathbf{u}](\cdot), [\mathbf{v}](\cdot), [\mathbf{x}](\cdot), \{[y_k](\cdot)\}, \{([t_i], [z_i])\}$

Constraints:

1. State evolution:
   - $\mathbf{v}(\cdot) = \mathbf{f}(\mathbf{x}(\cdot), \mathbf{u}(\cdot))$
   - $\dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot)$
2. Beacon-robot distance function:
   - $y_k(\cdot) = \sqrt{(x_1(\cdot) - b_{k,1})^2 + (x_2(\cdot) - b_{k,2})^2}$
3. Measurements:
   - $z_i = y_k(t_i)$

Note: initial position $(x_1(0), x_2(0))$ not priorly known.

Application: robot localization

Variables:
$\mathbf{u}(\cdot), \mathbf{v}(\cdot), \mathbf{x}(\cdot), \{y_k(\cdot)\}, \{(t_i, z_i)\}, \{w_k(\cdot)\}$

Domains:
$[\mathbf{u}](\cdot), [\mathbf{v}](\cdot), [\mathbf{x}](\cdot), \{[y_k](\cdot)\}, \{([t_i], [z_i])\}, \{[w_k](\cdot)\}$

Constraints:

1. State evolution:
   - $\mathbf{v}(\cdot) = \mathbf{f}(\mathbf{x}(\cdot), \mathbf{u}(\cdot))$
   - $\dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot)$
2. Beacon-robot distance function:
   - $y_k(\cdot) = \sqrt{(x_1(\cdot) - b_{k,1})^2 + (x_2(\cdot) - b_{k,2})^2}$
3. Measurements:
   - $z_i = y_k(t_i)$
   - $\dot{y}_k(\cdot) = w_k(\cdot)$

Note: initial position $(x_1(0), x_2(0))$ not priorly known.

Application: robot localization

Variables:
$\mathbf{u}(\cdot), \mathbf{v}(\cdot), \mathbf{x}(\cdot), \{y_k(\cdot)\}, \{(t_i, z_i)\}, \{w_k(\cdot)\}$

Domains:
$[\mathbf{u}](\cdot), [\mathbf{v}](\cdot), [\mathbf{x}](\cdot), \{[y_k](\cdot)\}, \{([t_i], [z_i])\}, \{[w_k](\cdot)\}$

Constraints:

1. State evolution:
   - $\mathbf{v}(\cdot) = \mathbf{f}(\mathbf{x}(\cdot), \mathbf{u}(\cdot))$
   - $\dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot)$
2. Beacon-robot distance function:
   - $y_k(\cdot) = \sqrt{(x_1(\cdot) - b_{k,1})^2 + (x_2(\cdot) - b_{k,2})^2}$
   - $w_k(\cdot) = \dfrac{dy_k(\cdot)}{d\cdot}$
3. Measurements:
   - $z_i = y_k(t_i)$
   - $\dot{y}_k(\cdot) = w_k(\cdot)$

Note: initial position $(x_1(0), x_2(0))$ not priorly known.

Application: robot localization

Variables:
$\mathbf{u}(\cdot), \mathbf{v}(\cdot), \mathbf{x}(\cdot), \{y_k(\cdot)\}, \{(t_i, z_i)\}, \{w_k(\cdot)\}$

Domains:
$[\mathbf{u}](\cdot), [\mathbf{v}](\cdot), [\mathbf{x}](\cdot), \{[y_k](\cdot)\}, \{([t_i], [z_i])\}, \{[w_k](\cdot)\}$

Constraints:

1. State evolution:
   - $\mathbf{v}(\cdot) = \mathbf{f}(\mathbf{x}(\cdot), \mathbf{u}(\cdot))$ → $\mathcal{C}_{\mathbf{f}}\big([\mathbf{v}](\cdot), [\mathbf{x}](\cdot), [\mathbf{u}](\cdot)\big)$
   - $\dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot)$ → $\mathcal{C}_{\frac{d}{dt}}\big([\mathbf{x}](\cdot), [\mathbf{v}](\cdot)\big)$
2. Beacon-robot distance function:
   - $y_k(\cdot) = \sqrt{(x_1(\cdot) - b_{k,1})^2 + (x_2(\cdot) - b_{k,2})^2}$ → $\mathcal{C}^k_{\text{dist}}\big([y_k](\cdot), [\mathbf{x}](\cdot)\big)$
   - $w_k(\cdot) = \dfrac{dy_k(\cdot)}{d\cdot}$ → $\mathcal{C}^k_{\text{ddist}}\big([w_k](\cdot), [\mathbf{x}](\cdot)\big)$
3. Measurements:
   - $z_i = y_k(t_i)$
   - $\dot{y}_k(\cdot) = w_k(\cdot)$

   → $\mathcal{C}_{\text{eval}}\big([t_i], [z_i], [y_k](\cdot), [w_k](\cdot)\big)$

Contractor programming algorithm:

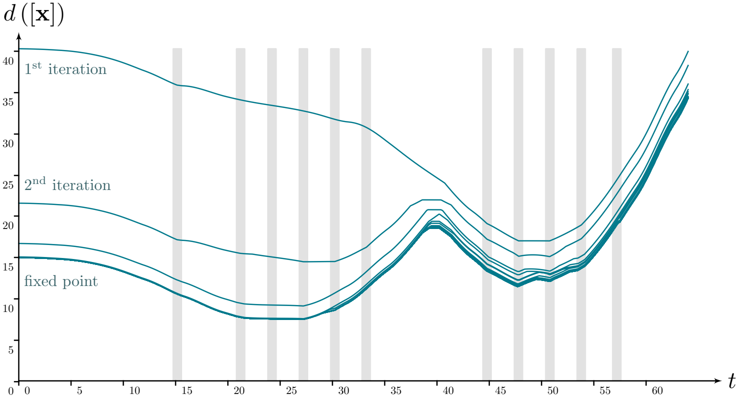
Note: initial position $(x_1(0), x_2(0))$ not priorly known.

Application: robot localization

An iterative resolution process

Let us define $d : \mathbb{IR}^2 \to \mathbb{R}$ the diagonal of a position box $[x_1] \times [x_2]$:

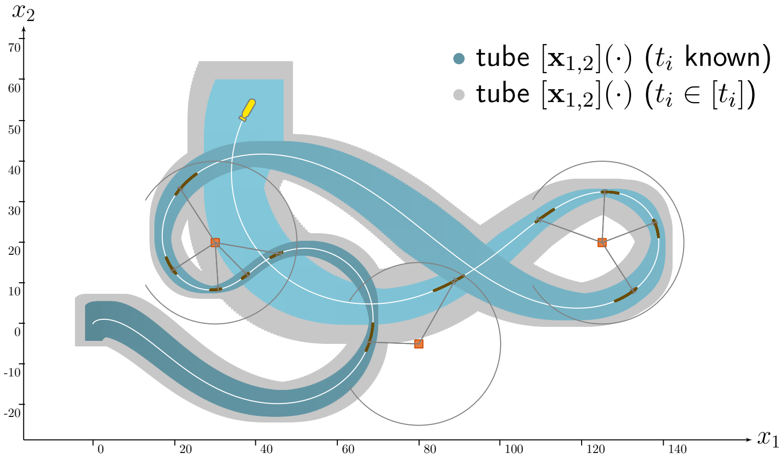
$$d([\mathbf{x}]) = \sqrt{\left(x_1^+ - x_1^-\right)^2 + \left(x_2^+ - x_2^-\right)^2}$$



Thicknesses of robot's positions estimation $[x_1](\cdot) \times [x_2](\cdot)$ for each iteration step.

Application: robot localization

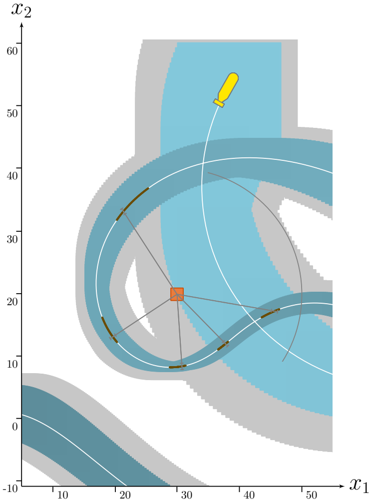
Overview of state estimations

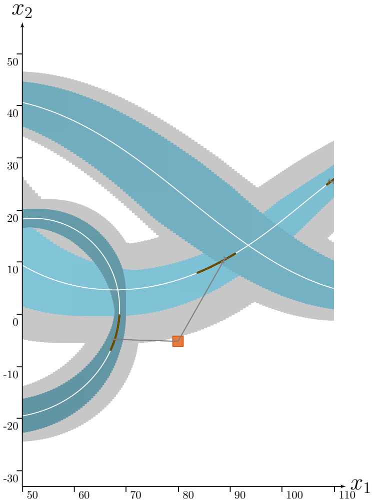


Projections of the resulting tube $[\mathbf{x}](\cdot)$ in blue and gray.
In gray: computed envelope assuming time uncertainties.
In blue: estimations that would have been obtained without time uncertainties.

Application: robot localization

Overview of state estimations





Section 4

Conclusions

Conclusions

Contractor programming on tubes:

- **non-linear** and **differential** systems
- **generic** and **simple** estimation approaches

Contractor $\mathcal{C}_{\text{eval}}$:

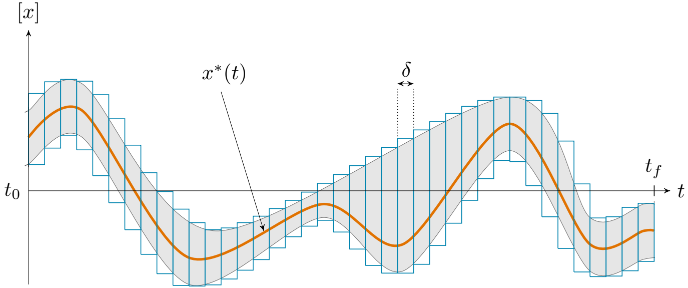
- elementary **operator** in the contractor prog. framework
- original method to deal with (strong) **time uncertainties**
- allows one to consider state estimation problems from a **temporal point of view** where the time t becomes an unknown variable to be estimated

Conclusions

Tubex library

An open-source C++ library based on IBEX and providing tools for constraint programming over dynamical systems.

- `Tube`, `TubeVector`, ...
- contractors $\mathcal{C}_{\frac{d}{dt}}$, $\mathcal{C}_{\text{eval}}$, $\mathcal{C}_{\text{delay}}$, ...
- robotic tools and applications



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

References

- **Contractor Programming**
G. Chabert, L. Jaulin. *Artificial Intelligence*, 2009

- **A Constraint Satisfaction Approach for Enclosing Solutions to Parametric ODEs**
M. Janssen, P. Van Hentenryck, Y. Deville. *SIAM Journal on Numerical Analysis*, 2002

- **Analytic constraint solving and interval arithmetic**
T. J. Hickey. *ACM Press*, 2000

- **Constraint Satisfaction Differential Problems**
J. Cruz, P. Barahona. *Springer Berlin Heidelberg*, 2003

- **Set-membership state estimation with fleeting data**
F. Le Bars, J. Sliwka, L. Jaulin, O. Reynet *Automatica*, 2012

- **Solving Non-Linear Constraint Satisfaction Problems Involving Time-Dependant Functions**
A. Bethencourt, L. Jaulin. *Mathematics in Computer Science*, 2014

- **Guaranteed computation of robot trajectories**
S. Rohou, L. Jaulin, L. Mihaylova, F. Le Bars, S. M. Veres. *Robotics and Autonomous Systems*, 2017

- **Reliable non-linear state estimation involving time uncertainties**
S. Rohou, L. Jaulin, L. Mihaylova, F. Le Bars, S. M. Veres. *Automatica*, 2018

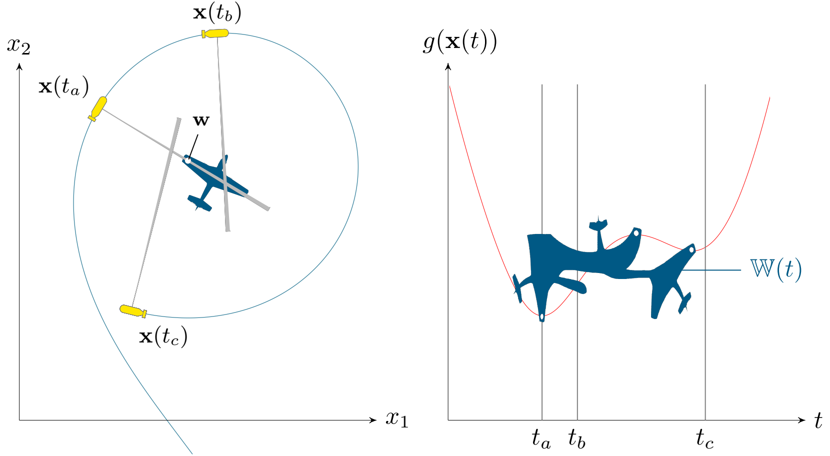
- **Reliable robot localization: a constraint programming approach over dynamical systems**
S. Rohou. *PhD thesis*, 2017

Section 5

Appendices

Appendices

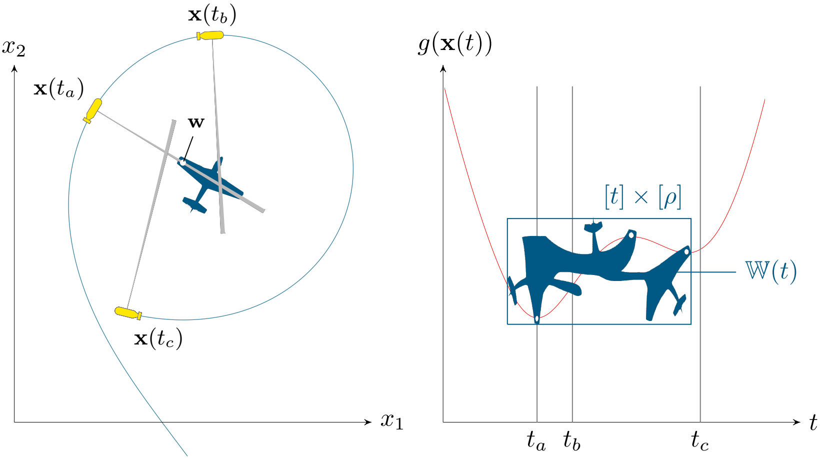
Time uncertainties in state estimation



A robot $\mathcal{R}$ perceiving a plane wreck with a side scan sonar.

Appendices

Time uncertainties in state estimation

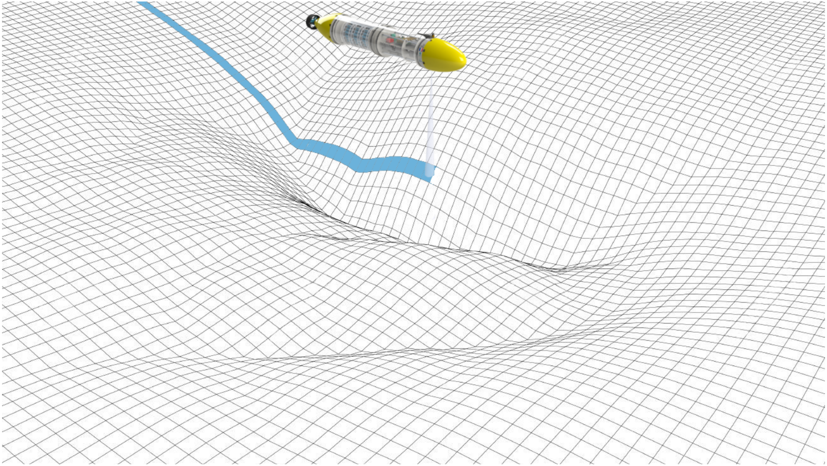


A robot $\mathcal{R}$ perceiving a plane wreck with a side scan sonar.

Appendices

Robot localization $\rightarrow$ temporal resolution

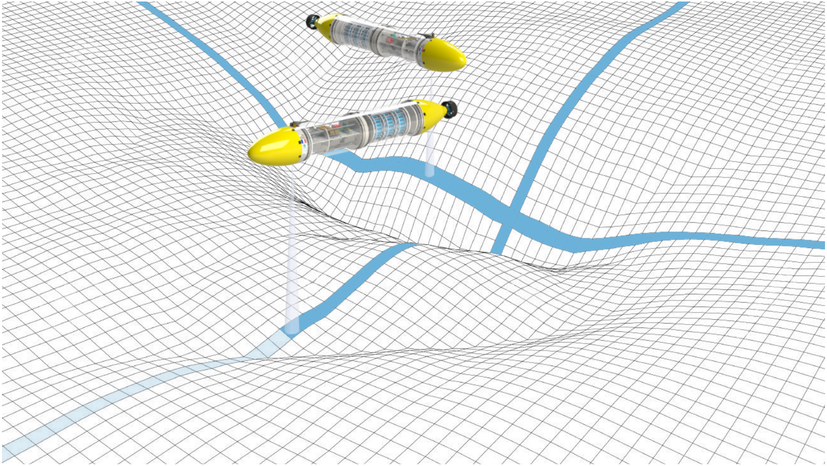
Trajectory $\mathbf{p}(\cdot) : \mathbb{R} \rightarrow \mathbb{R}^2$ crossed at times t_1, t_2: $\mathbf{p}(t_1) = \mathbf{p}(t_2)$.



Appendices

Robot localization $\rightarrow$ temporal resolution

Trajectory $\mathbf{p}(\cdot) : \mathbb{R} \rightarrow \mathbb{R}^2$ crossed at times t_1, t_2: $\mathbf{p}(t_1) = \mathbf{p}(t_2)$.



Appendices

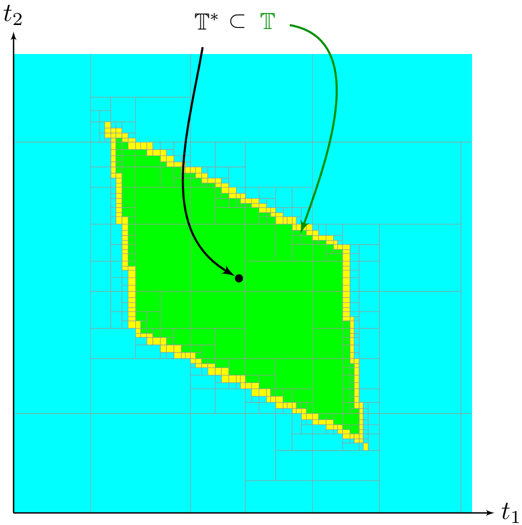
Robot localization $\rightarrow$ temporal resolution

Constraint:

- $\mathbf{p}(t_1) = \mathbf{p}(t_2)$
- $t_1 \in [t_1]$, $t_2 \in [t_2]$

1. approximation of a temporal set $\mathbb{T}$ with evolution constraints

2. contraction of $\mathbb{T}$ thanks to exteroceptive measurements (ex: bathymetry)



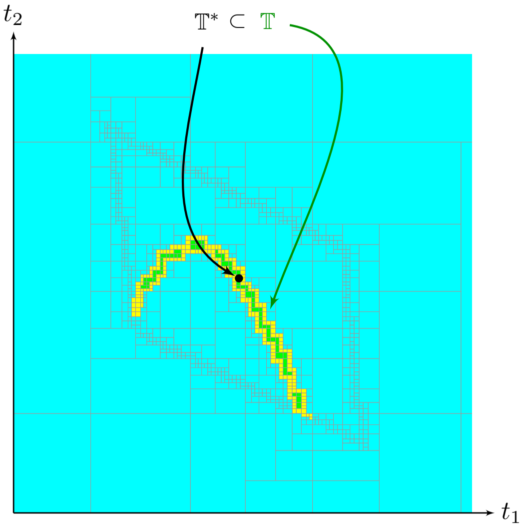
Appendices

Robot localization $\rightarrow$ temporal resolution

Constraint:

- $\mathbf{p}(t_1) = \mathbf{p}(t_2)$
- $t_1 \in [t_1]$, $t_2 \in [t_2]$

1. approximation of a temporal set $\mathbb{T}$ with evolution constraints
2. contraction of $\mathbb{T}$ thanks to exteroceptive measurements (ex: bathymetry)



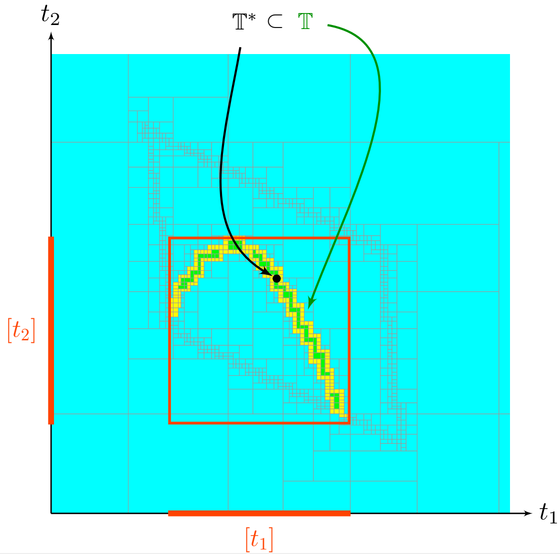
Appendices

Robot localization → temporal resolution

Constraint:

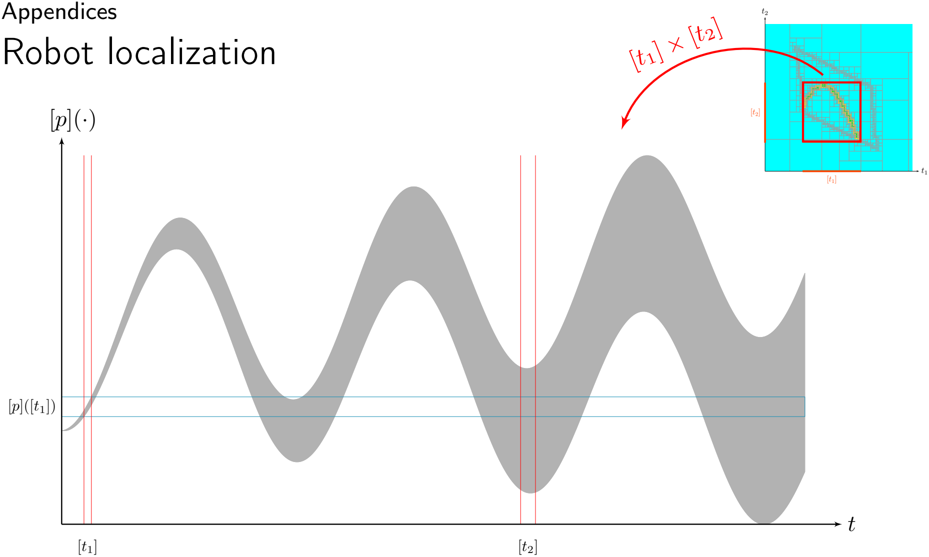
- $\mathbf{p}(t_1) = \mathbf{p}(t_2)$
- $t_1 \in [t_1]$, $t_2 \in [t_2]$

1. approximation of a temporal set $\mathbb{T}$ with evolution constraints
2. contraction of $\mathbb{T}$ thanks to exteroceptive measurements (ex: bathymetry)



Appendices

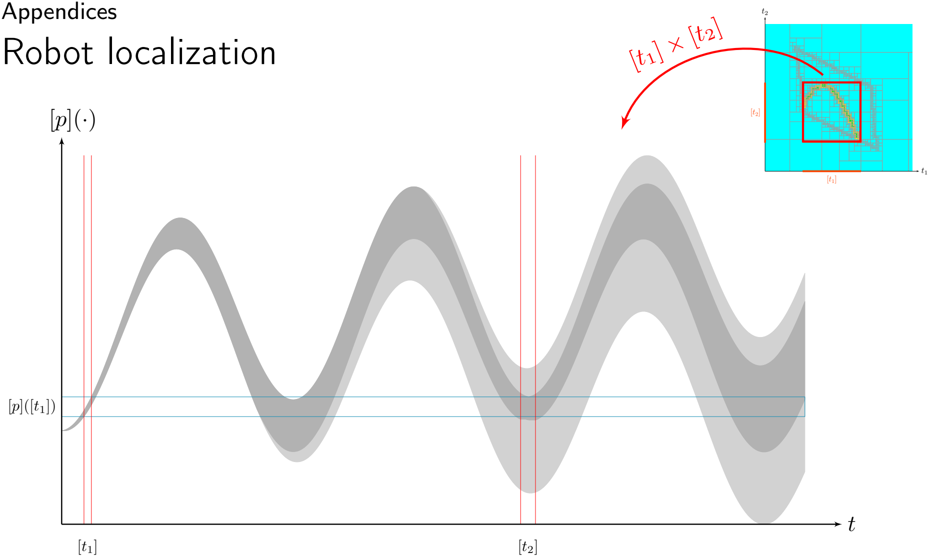
Robot localization



Constraint $\mathcal{L}_{t_1,t_2}(t_1, t_2, \mathbf{p}(\cdot), \mathbf{w}(\cdot)) : \begin{cases} \mathbf{p}(t_1) = \mathbf{p}(t_2) \\ \dot{\mathbf{p}}(\cdot) = \mathbf{w}(\cdot) \end{cases}$

Appendices

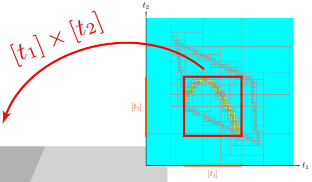
Robot localization

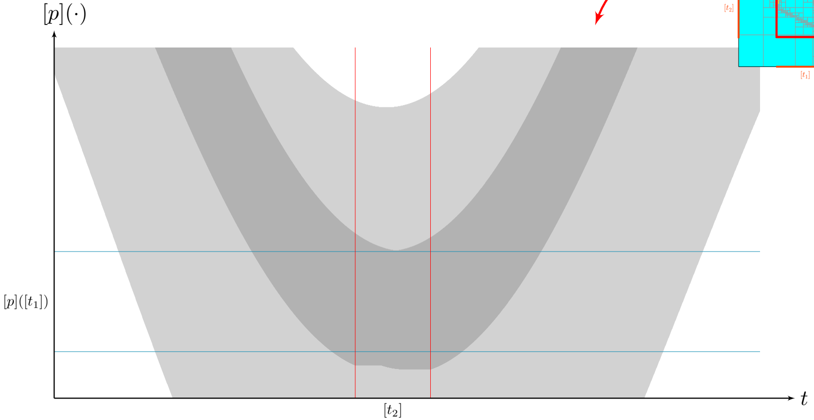


Constraint $\mathcal{L}_{t_1,t_2}\big(t_1, t_2, \mathbf{p}(\cdot), \mathbf{w}(\cdot)\big) : \begin{cases} \mathbf{p}(t_1) = \mathbf{p}(t_2) \\ \dot{\mathbf{p}}(\cdot) = \mathbf{w}(\cdot) \end{cases}$

Appendices

Robot localization

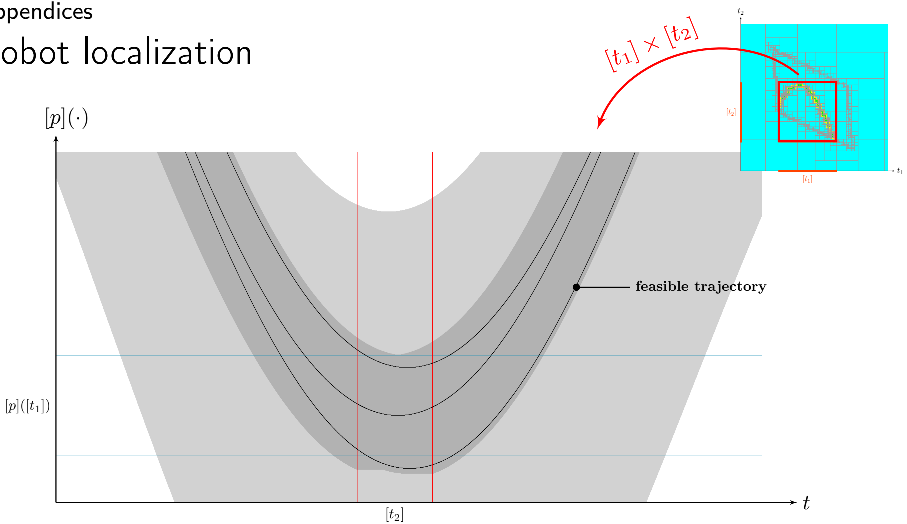




Constraint $\mathcal{L}_{t_1,t_2}\big(t_1, t_2, \mathbf{p}(\cdot), \mathbf{w}(\cdot)\big)$: $\begin{cases} \mathbf{p}(t_1) = \mathbf{p}(t_2) \\ \dot{\mathbf{p}}(\cdot) = \mathbf{w}(\cdot) \end{cases}$

Appendices

Robot localization

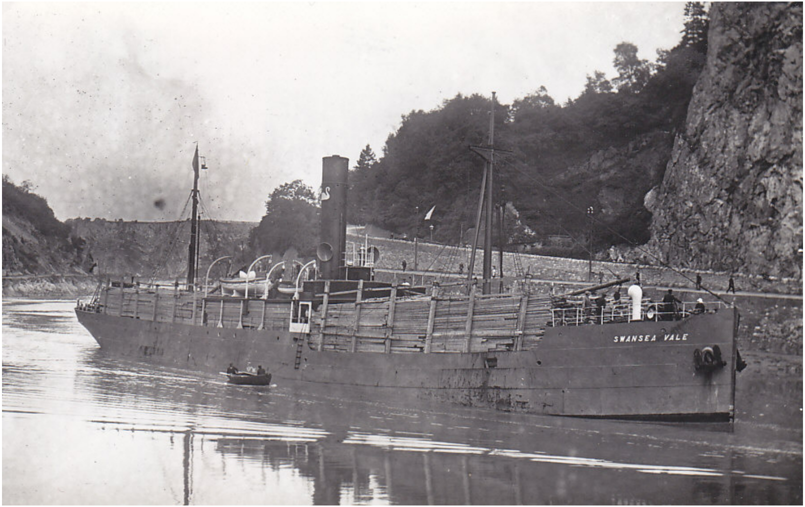


Constraint $\mathcal{L}_{t_1,t_2}\big(t_1, t_2, \mathbf{p}(\cdot), \mathbf{w}(\cdot)\big)$: $\begin{cases} \mathbf{p}(t_1) = \mathbf{p}(t_2) \\ \dot{\mathbf{p}}(\cdot) = \mathbf{w}(\cdot) \end{cases}$

Appendices

Time uncertainties in state estimation

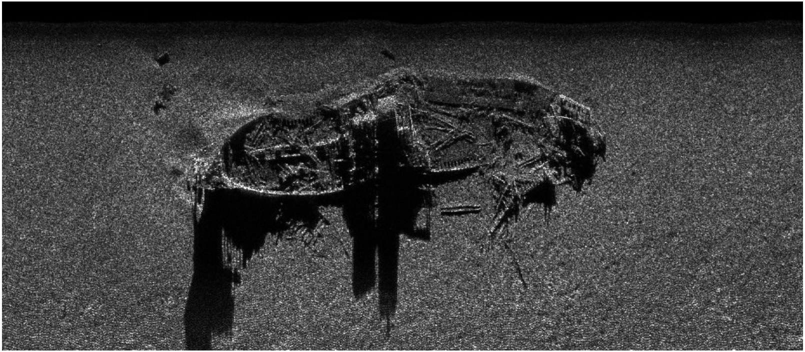
Application example: wreck based localization



Appendices

Time uncertainties in state estimation

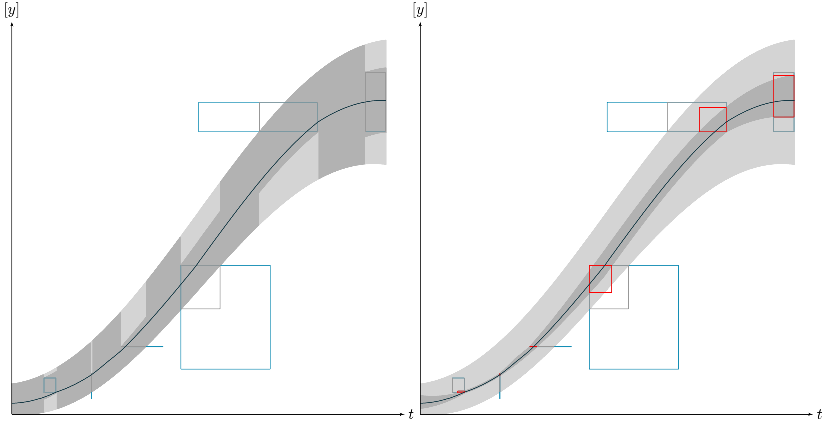
Application example: wreck based localization



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

Appendices

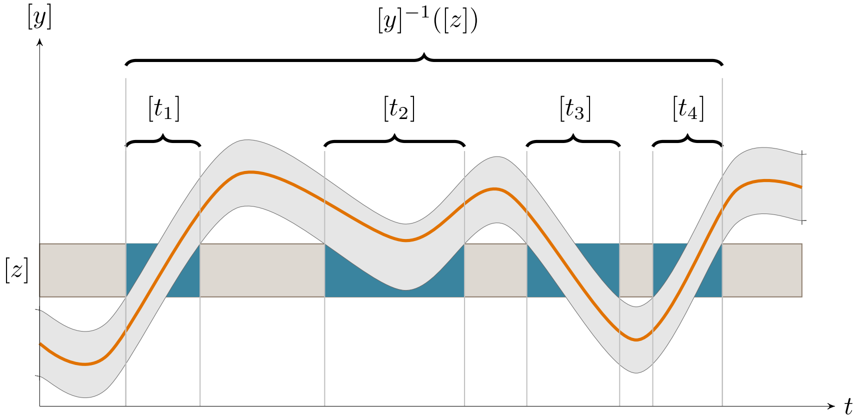
Several evaluations: fixed point iteration



Left: one iteration. Right: fixed point result.

Appendices

Tube inversion



Tube set-inversion $[y]^{-1}([z]) = \bigsqcup_{z\in[z]} \{t \mid y \in [y](t)\}$.