Robust set-membership state estimation

Luc Jaulin
ENSIETA,
2 rue François Verny, 29806 Brest Cédex 09

Abstract: This paper proposes a new observer for estimating the state vector of a nonlinear system. This observer, which is robust with respect to outliers, assumes that the measurement errors as well as the number of outliers that could occur within a given time window are bounded. The principle of the approach is to use interval analysis to deal properly with the nonlinearities involved in the system (without any linearization nor approximation) and to propagate through the time, in a forward and backward manner, the assumptions made about outliers. A testcase related to the localization and control of an underwater robot is also proposed to illustrate the efficiency of the approach.

Keywords: bounded-error, constraint propagation, control, interval analysis, localization, nonlinear estimation, observer, outliers, underwater robotics.

1. Introduction

Consider the discrete-time dynamic system described by the following nonlinear state equations

$$\left\{ \begin{array}{lcl} \mathbf{x}(k+1) & = & \mathbf{f}_k(\mathbf{x}(k)) \\ \mathbf{y}(k) & = & \mathbf{g}_k(\mathbf{x}(k)). \end{array} \right. \tag{1.1}$$

This formulation encloses situations where the state equations depend of some inputs that are known exactly. In a bounded-error context, we generally assume that for all k, the output vectors $\mathbf{y}(k)$ belong to some known sets $\mathbb{Y}(k)$. These sets are obtained from measurements $\tilde{\mathbf{y}}(k)$ of the output vector $\mathbf{y}(k)$ and take into account some bounded error noises that could corrupt the measurements. We can thus define the feasible set $\mathbb{X}(k)$ of all state vectors $\mathbf{x}(k)$ that are consistent

with the past as follows

$$\mathbb{X}(k) = \left\{ \mathbf{x}(k) \in \mathbb{R}^n, \left| \begin{array}{l} \exists \mathbf{x}(0), \ldots, \exists \mathbf{x}(k-1), \\ \forall k_1 \in \{0, \ldots, k-1\} \\ \mathbf{x}(k_1+1) = \mathbf{f}_{k_1}(\mathbf{x}(k_1)) \\ \bigwedge\limits_{i \in \{0,\ldots,\ell\}} (\mathbf{g}_{k_1}(\mathbf{x}(k_1)) \in \mathbb{Y}(k_1)) \end{array} \right. \right\}. \tag{1.2}$$

In this formula the operator $\wedge$ is the logical *and* operator. The set $\mathbb{X}(k)$ can be computed recursively [BR71] as follows

$$\mathbb{X}(k+1) = \mathbf{f}_k\left(\mathbb{X}(k)\right) \cap \left(\mathbf{f}_k \circ \mathbf{g}_k^{-1}\right)\left(\mathbb{Y}(k)\right). \tag{1.3}$$

In practice, it may happen that some of the $\mathbf{y}(k)$, the actual value the output vector at time k, do not belong to their corresponding sets $\mathbb{Y}(k)$. In such a case, we say that $\mathbb{Y}(k)$ is an outlier. Dealing with outliers has already been consider by several authors, in a set membership context (see, *e.g.*, [NV93], [LWG87], [PW96], [KLP+03]). To robustify bounded error methods against these outliers, we shall make the following assumption:

MNO (Minimal Number of Outliers) assumption: Outliers may exist for the outputs but within any time window of length ℓ we never have more than q outliers.

We can thus define the feasible set $\mathbb{X}(k)$ as the set of all feasible values for $\mathbf{x}(k)$, *i.e.*, the set of all $\mathbf{x}(k)$ that could be reached by assuming less than q outliers within any time window of length ℓ. More rigorously, $\mathbb{X}(k)$ is defined as follows:

$$\mathbb{X}(k) = \left\{ \mathbf{x}(k) \in \mathbb{R}^n, \left| \begin{array}{l} \exists \mathbf{x}(-\ell), \ldots, \exists \mathbf{x}(0), \ldots, \exists \mathbf{x}(k-1), \\ \forall k_1 \in \{-\ell, \ldots, k-1\}, \mathbf{x}(k_1+1) = \mathbf{f}_{k_1}(\mathbf{x}(k_1)) \\ \forall k_2 \in \{0, \ldots, k-1\}, \bigwedge\limits^{\{q\}}_{i \in \{0,\ldots,\ell\}} (\mathbf{g}_{k_2-i}(\mathbf{x}(k_2-i)) \in \mathbb{Y}(k_2-i)) \end{array} \right. \right\}. \tag{1.4}$$

In this formula the operator $\bigwedge^{\{q\}}$ is the *relaxed 'and' operator.* It means that q of the $\ell+1$ set-membership constraints are allowed to be violated. For instance

$$\bigwedge^{\{1\}} (a \in \mathbb{A}, b \in \mathbb{B}, c \in \mathbb{C}) \Leftrightarrow (a \in \mathbb{A} \wedge b \in \mathbb{B}) \vee (a \in \mathbb{A} \wedge c \in \mathbb{B}) \vee (b \in \mathbb{A} \wedge c \in \mathbb{B}),$$

where $\vee$ and $\wedge$ stand for *or* and *and* respectively.

If our MNO assumption is true, then for each k, the true value for the state vector belongs to $\mathbb{X}(k)$. This paper proposes an interval constraint propagation approach to recursively compute a box which encloses $\mathbb{X}(k)$, for all k. Or, equivalently and in a control point of view, we shall build an interval observer robust with respect to outliers. The two integers q and ℓ will be the parameters

of this observer. Note that interval constraint propagation methods have been shown successful for several state estimation problems (see, *e.g.*, [JKBW01], [RRC04] or [GB06]). However, these techniques have never been used for state estimation in the context where outliers could occur.

Section 2 provides a recursive definition of the set of feasible set for the state vectors. The methodology that is used for characterizing the feasible set is detailed in Section 3. As an illustration, Section 4 deals with the control and the localization problem of an underwater robot moving inside a swimming pool. Section 5 will then conclude the paper.

2. Recursive formulation for the feasible set

2.1. Relaxed intersection

Consider n sets $\mathbb{X}_1, \ldots, \mathbb{X}_m$ of $\mathbb{R}^n$. The q-relaxed intersection denoted by $\bigcap^{\{q\}} \mathbb{X}_i$ is the set of all $\mathbf{x} \in \mathbb{R}^n$ which belong to all $\mathbb{X}_i$'s, except q at most. Figure 2.1 illustrates this notion for $m = 6$ and $q = 2, 3, 4$. For this example, we have

$$\bigcap^{\{0\}} \mathbb{X}_i = \bigcap^{\{1\}} \mathbb{X}_i = \emptyset, \ \bigcap^{\{5\}} \mathbb{X}_i = \bigcup \mathbb{X}_i \text{ and } \bigcap^{\{6\}} \mathbb{X}_i = \mathbb{R}^2. \tag{2.1}$$

Since the q-relaxed intersection can be written as a combination of unions and intersections, it is inclusion monotonic, *i.e.*,

$$(\mathbb{X}_1 \subset \mathbb{Y}_1, \ldots, \mathbb{X}_6 \subset \mathbb{Y}_6) \Rightarrow \bigcap_{i\in\{1,\ldots,m\}}^{\{q\}} \mathbb{X}_i \subset \bigcap_{i\in\{1,\ldots,m\}}^{\{q\}} \mathbb{Y}_i.$$

This inclusion monotonicity is illustrated by Figure 2.2 in the case where the $\mathbb{Y}_i$'s are boxes. Note that the q-relaxed intersection of m boxes is not necessarily a box.

2.2. Feasible sets for the state

The following new theorem provides a recursive definition for the feasible set $\mathbb{X}(k+1)$ for the state vector.

Theorem 2.1. *If all $\mathbf{f}_k$ are bijective functions, then feasible set for the state vector assuming less than q outliers inside a time window of length ℓ (it is our MNO assumption made on the introduction) is*

$$\mathbb{X}(k+1) = \mathbf{f}_k\left(\mathbb{X}(k)\right) \ \cap \bigcap_{i\in\{0,\ldots,\ell\}}^{\{q\}} \mathbf{f}_k^i \circ \mathbf{g}_{k-i}^{-1}\left(\mathbb{Y}(k-i)\right), \tag{2.2}$$

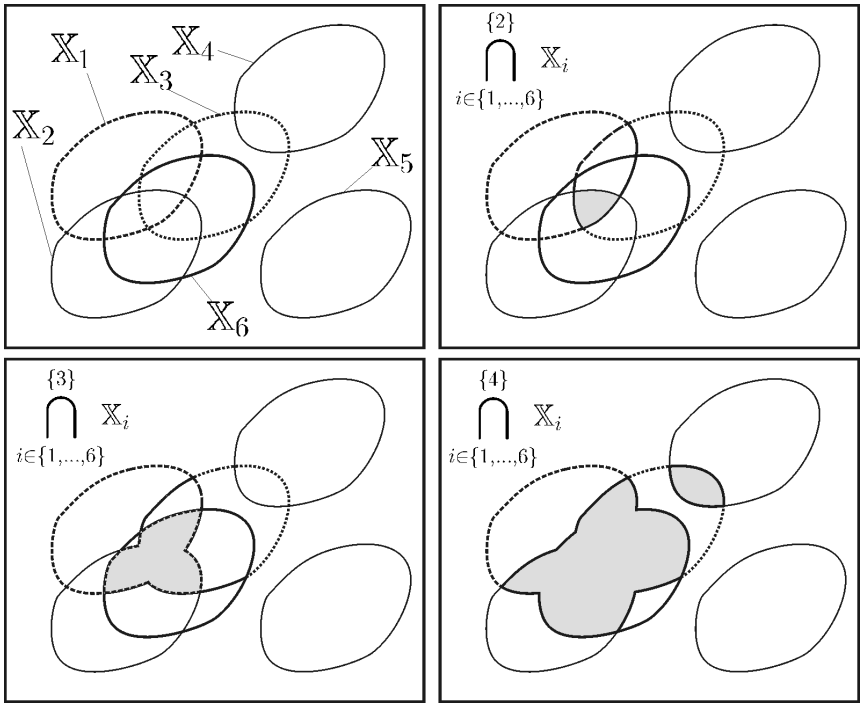

Figure 2.1: Illustration (in gray) of the q-relaxed intersection the 6 sets $\mathbb{X}_1, \dots, \mathbb{X}_6$ where $q \in \{2, 3, 4\}$

where

$$\mathbf{f}_k^i(\mathbf{x}(k-i)) = \mathbf{f}_k \circ \mathbf{f}_{k-1} \circ \dots \circ \mathbf{f}_{k-i}\left(\mathbf{x}(k-i)\right). \tag{2.3}$$

Moreover, if

$$\mathbb{X}(k+1) \cap \mathbf{f}_k^i \circ \mathbf{g}_{k-i}^{-1}\left(\mathbb{Y}(k-i)\right) = \emptyset, \tag{2.4}$$

*then the $k-i$ data set $\mathbb{Y}(k-i)$ an outlier (*i.e.*, $\mathbf{y}(k-i) \notin \mathbb{Y}(k-i)$).*

This theorem is illustrated by Figure 2.3 for $\ell = 2$ and $q = 1$. The actual state vectors and outputs are represented by the small black disks. In the situation represented on the figure, an outlier occurs at time $k-1$. Moreover, since $\mathbb{X}(k+1) \cap \mathbf{f}_k^1 \circ \mathbf{g}_{k-1}^{-1}\left(\mathbb{Y}(k-1)\right) = \emptyset$, then we detect that $\mathbf{y}(k-1)$ is an outlier.

Proof of the theorem: From Equation (1.4), we have

$$\mathbb{X}(k+1) = \left\{ \mathbf{x}(k+1) \in \mathbb{R}^n, \left| \begin{array}{l} \exists \mathbf{x}(-\ell), \dots, \exists \mathbf{x}(0), \dots, \exists \mathbf{x}(k), \\ \forall k_1 \in \{-\ell, \dots, k\}, \mathbf{x}(k_1+1) = \mathbf{f}_{k_1}(\mathbf{x}(k_1)) \\ \forall k_2 \in \{0, \dots, k\}, \bigwedge_{i\in\{0,\dots,\ell\}}^{\{q\}} \left(\mathbf{g}_{k_2-i}(\mathbf{x}(k_2-i)) \in \mathbb{Y}(k_2-i)\right). \end{array} \right. \right\} \tag{2.5}$$

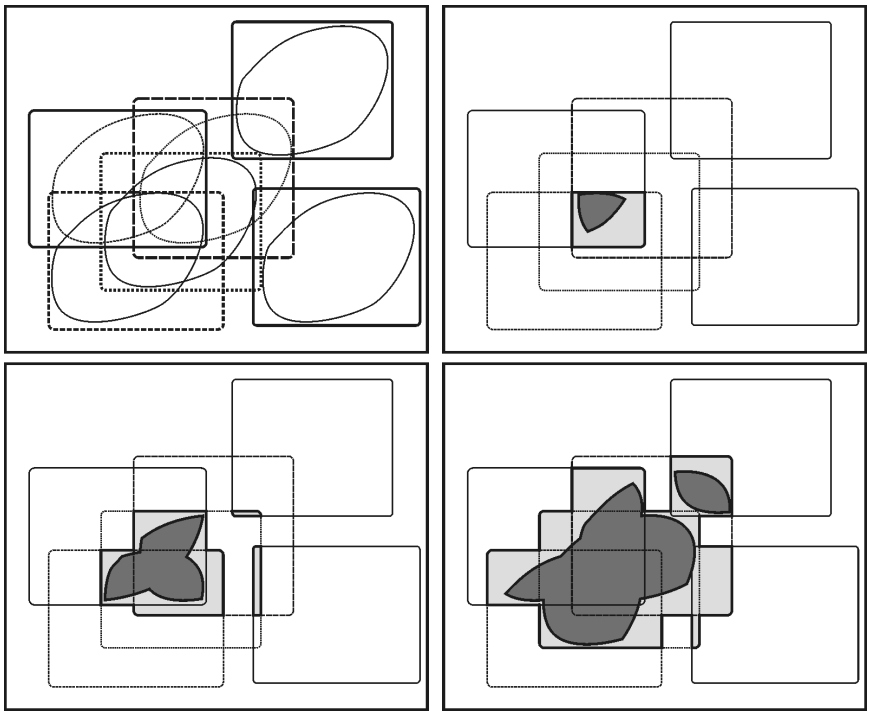

Figure 2.2: Illustration of the inclusion monotonicity of the q-relaxed intersection (in dark grey for the $\mathbb{X}_i$'s and in light grey for the $\mathbb{Y}_i$'s)

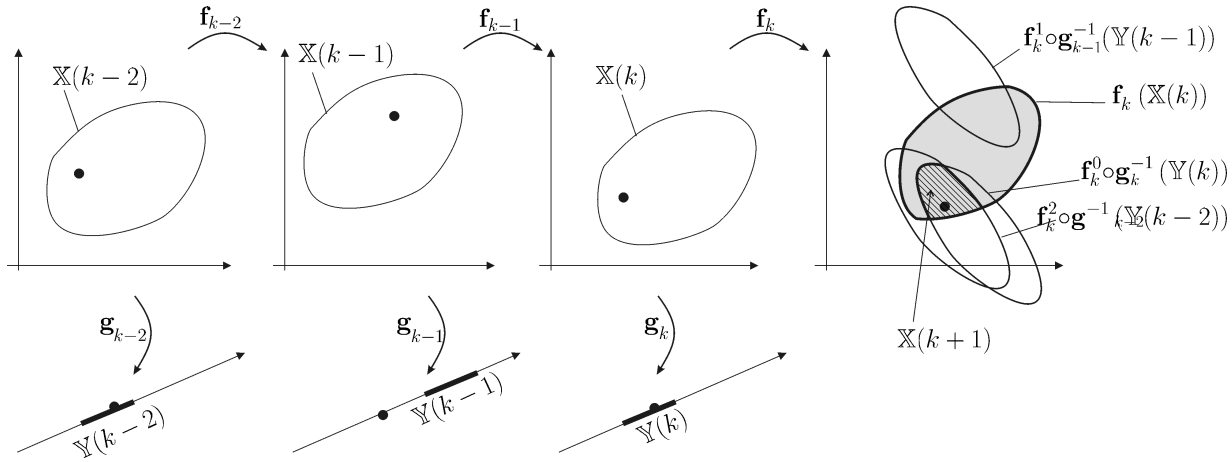

Figure 2.3: The feasible set for the state vector $\mathbb{X}(k+1)$, assuming at most $q=1$ outlier, can be defined recursively from $\mathbb{X}(k)$ and from the data sets $\mathbb{Y}(k), \mathbb{Y}(k-1), \mathbb{Y}(k-2)$.

Since

$$(\forall k_1 \in \{-\ell, \dots, k\}, \mathbf{x}(k_1+1) = \mathbf{f}_{k_1}(\mathbf{x}(k_1)))$$

$$\Leftrightarrow \quad \begin{pmatrix} \mathbf{x}(k+1) = \mathbf{f}_k(\mathbf{x}(k)) \\ \mathbf{x}(k) = \mathbf{f}_{k-1}(\mathbf{x}(k-1))) \\ \vdots \\ \mathbf{x}(-\ell+1) = \mathbf{f}_{-\ell}(\mathbf{x}(-\ell)) \end{pmatrix}$$

$$\Leftrightarrow \quad \begin{pmatrix} \mathbf{x}(k+1) = \mathbf{f}_k(\mathbf{x}(k)) \\ \mathbf{x}(k+1) = \mathbf{f}_k \circ \mathbf{f}_{k-1}(\mathbf{x}(k-1)) \\ \vdots \\ \mathbf{x}(k+1) = \mathbf{f}_k \circ \mathbf{f}_{k-1} \circ \cdots \circ \mathbf{f}_{-\ell}(\mathbf{x}(-\ell)) \end{pmatrix}$$

$$\overset{(2.3)}{\Leftrightarrow} \quad \begin{pmatrix} \mathbf{x}(k+1) = \mathbf{f}_k^0(\mathbf{x}(k)) \\ \mathbf{x}(k+1) = \mathbf{f}_k^1(\mathbf{x}(k-1)) \\ \vdots \\ \mathbf{x}(k+1) = \mathbf{f}_k^{k+\ell}(\mathbf{x}(-\ell)) \end{pmatrix}$$

$$\Leftrightarrow \quad \begin{pmatrix} \mathbf{x}(k) = \left(\mathbf{f}_k^0\right)^{-1}(\mathbf{x}(k+1)) \\ \mathbf{x}(k-1) = \left(\mathbf{f}_k^1\right)^{-1}(\mathbf{x}(k+1)) \\ \vdots \\ \mathbf{x}(-\ell) = \left(\mathbf{f}_k^{k+\ell}\right)^{-1}(\mathbf{x}(k+1)) \end{pmatrix},$$

the proposition (2.5) becomes

$$\mathbb{X}(k+1) = \left\{ \mathbf{x}(k+1) \in \mathbb{R}^n, \forall k_2 \in \{0, \dots, k\}, \bigwedge_{i \in \{0,\dots,\ell\}}^{\{q\}} \mathbf{g}_{k_2-i} \circ \left(\mathbf{f}_k^{k-k_2+i}\right)^{-1}(\mathbf{x}(k+1)) \in \mathbb{Y}(k_2-i) \right\}$$

By applying the equivalence

$$\mathbf{g}(\mathbf{b}) \in \mathbb{A} \Leftrightarrow \mathbf{b} \in \mathbf{g}^{-1}(\mathbb{A}), \tag{2.6}$$

we get

$$\mathbb{X}(k+1) = \left\{ \mathbf{x}(k+1) \in \mathbb{R}^n, \forall k_2 \in \{0, \dots, k\}, \bigwedge_{i \in \{0,\dots,\ell\}}^{\{q\}} \mathbf{x}(k+1) \in \mathbf{f}_k^{k-k_2+i} \circ \mathbf{g}_{k_2-i}^{-1}(\mathbb{Y}(k_2-i)) \right\}$$

or equivalently

$$\mathbb{X}(k+1) = \bigcap_{k_2 \in \{0,\dots,k\}} \bigcap_{i \in \{0,\dots,\ell\}}^{\{q\}} \mathbf{f}_k^{k-k_2+i} \circ \mathbf{g}_{k_2-i}^{-1}(\mathbb{Y}(k_2-i)). \tag{2.7}$$

By decomposing the intersection indexed by $k_2 \in \{0,\dots,k\}$ into two parts ($\{0,\dots,k-1\}$ and $\{k\}$), we get

$$\mathbb{X}(k+1) = \left(\bigcap_{k_2\in\{0,\dots,k-1\}} \bigcap_{i\in\{0,\dots,\ell\}}^{\{q\}} \mathbf{f}_k^{k-k_2+i} \circ \mathbf{g}_{k_2-i}^{-1}\left(\mathbb{Y}(k_2-i)\right) \right) \cap \left(\bigcap_{i\in\{0,\dots,\ell\}}^{\{q\}} \mathbf{f}_k^{i} \circ \mathbf{g}_{k-i}^{-1}\left(\mathbb{Y}(k-i)\right) \right). \tag{2.8}$$

We shall now prove that the left part of this intersection corresponds to $\mathbf{f}_k\left(\mathbb{X}(k)\right)$. We have

$$\mathbf{f}_k\left(\mathbb{X}(k)\right) \overset{(2.7)}{=} \mathbf{f}_k \left(\bigcap_{k_2\in\{0,\dots,k-1\}} \bigcap_{i\in\{0,\dots,\ell\}}^{\{q\}} \mathbf{f}_{k-1}^{k-1-k_2+i} \circ \mathbf{g}_{k_2-i}^{-1}\left(\mathbb{Y}(k_2-i)\right) \right).$$

Now, since $\mathbf{f}_k$ is bijective, it is distributive with respect the intersection and union. Moreover, the relaxed intersection can be defined as a composition of intersections and unions. As a consequence,

$$\mathbf{f}_k\left(\mathbb{X}(k)\right) = \bigcap_{k_2\in\{0,\dots,k-1\}} \bigcap_{i\in\{0,\dots,\ell\}}^{\{q\}} \mathbf{f}_k \circ \mathbf{f}_{k-1}^{k-1-k_2+i} \circ \mathbf{g}_{k_2-i}^{-1}\left(\mathbb{Y}(k_2-i)\right). \tag{2.9}$$

Now, by using the definition of $\mathbf{f}_k^i$ (see (2.3)), we have

$$\mathbf{f}_k \circ \mathbf{f}_{k-1}^{k-1-k_2+i} = \mathbf{f}_k \circ \mathbf{f}_{k-1} \circ \dots \circ \mathbf{f}_{(k-1)-(k-1-k_2+i)} = \mathbf{f}_k \circ \dots \circ \mathbf{f}_{k_2-i} = \mathbf{f}_k \circ \dots \circ \mathbf{f}_{k-(k-k_2+i)} = \mathbf{f}_k^{k-k_2+i}.$$

Thus

$$\mathbf{f}_k\left(\mathbb{X}(k)\right) = \bigcap_{k_2\in\{0,\dots,k-1\}} \bigcap_{i\in\{0,\dots,\ell\}}^{\{q\}} \mathbf{f}_k^{k-k_2+i} \circ \mathbf{g}_{k_2-i}^{-1}\left(\mathbb{Y}(k_2-i)\right) \tag{2.10}$$

which corresponds to the left part of the intersection in (2.8). This terminates the proof of Equation (2.2). We shall now prove the last part of the theorem given by (2.4). If the assumption of the theorem are satisfied, the true value for the state at time $k+1$ satisfies $\mathbf{x}(k+1) \in \mathbb{X}(k+1)$. If $\mathbb{X}(k+1) \cap \mathbf{f}_k^i \circ \mathbf{g}_{k-i}^{-1}\left(\mathbb{Y}(k-i)\right) = \emptyset$, then

$$\mathbf{x}(k+1) \notin \mathbf{f}_k^i \circ \mathbf{g}_{k-i}^{-1}\left(\mathbb{Y}(k-i)\right) \tag{2.11}$$

which implies that

$$\mathbf{y}(k-i) = \mathbf{g}_{k-i} \circ \left(\mathbf{f}_k^i\right)^{-1}\left(\mathbf{x}(k+1)\right) \notin \mathbb{Y}(k-i). \tag{2.12}$$

The $k-i$ data is thus an outlier. ■

Figure 2.4 assumes the conditions of Figure 2.3, with $q=1$. Whereas in the situation of Figure 2.3, we were able to detect that $\mathbf{y}(k-1)$ were the outlier, in the situation represented on the left part of Figure 2.4, we are only able to detect that either $\mathbf{y}(k-1)$ or $\mathbf{y}(k-2)$ is an outlier and that $\mathbf{y}(k)$ cannot be an outlier. On the right of Figure 2.4, since $\bigcap_i \mathbf{f}_k^i \circ \mathbf{g}_{k-i}^{-1}\left(\mathbb{Y}(k-i)\right)$ is not empty, we are not able to detect if there exists an outlier. Note that in both cases represented on Figure 2.4, $\mathbf{y}(k-1)$ is an outlier and $\mathbb{X}(k+1)$ encloses the actual state vector. This property will be true as long as a maximum of q outliers could occur on a window of length ℓ (which is our MNO assumption).

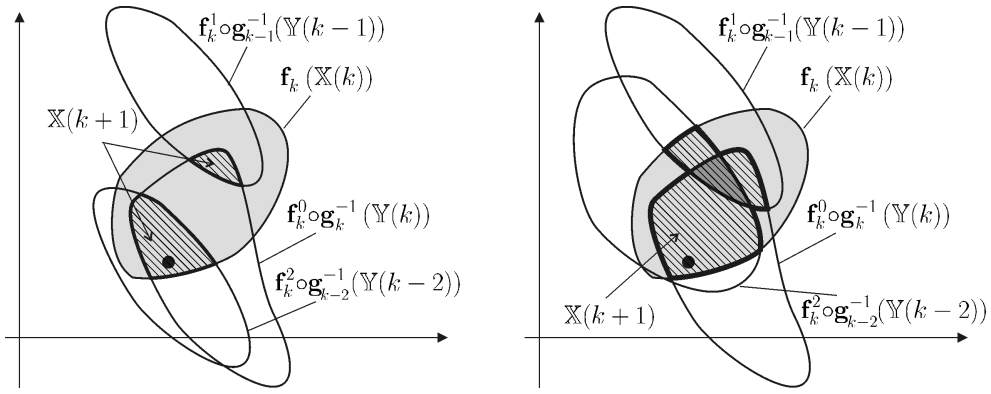

Figure 2.4: Left: we are able two detect that either $\mathbf{y}(k-1)$ or $\mathbf{y}(k-2)$ is an oulier; Right: we are not able to detect that there exists an outlier.

2.3. Enclosing the feasible sets

The sequence of sets given by Equation (2.2) is very difficult to compute exactly, except for some unrealistic situations. Here, we shall assume that the sets $\mathbb{Y}(k)$ are boxes or can be enclosed by boxes denoted by $[\mathbf{y}](k)$. We shall also assume also that the set $\mathbb{X}(k)$ can be included into a box $[\mathbf{x}](k)$. Because of the inclusion monotonicity of the q-relaxed intersection, we have the enclosure

$$\mathbb{X}(k+1) \subset \mathbf{f}_k\left([\mathbf{x}](k)\right) \cap \bigcap_{i\in\{0,\ldots,\ell\}}^{\{q\}} \mathbf{f}_k^i \circ \mathbf{g}_{k-i}^{-1}\left([\mathbf{y}](k-i)\right). \tag{2.13}$$

If we are able to compute a box $[\mathbf{x}](k+1)$ such that

$$[\mathbf{x}](k+1) \supset \mathbf{f}_k\left([\mathbf{x}](k)\right) \cap \bigcap_{i\in\{0,\ldots,\ell\}}^{\{q\}} \mathbf{f}_k^i \circ \mathbf{g}_{k-i}^{-1}\left([\mathbf{y}](k-i)\right) \tag{2.14}$$

then, we will be able to generate a sequence of boxes which encloses the sequence of sets $\mathbb{X}(k)$. The following section will explain how such an enclosing box $[\mathbf{x}](k+1)$ can be computed.

3. Relaxed set inversion

3.1. Problem

We shall consider in this section the *relaxed set inversion problem* which consists in finding a box enclosing the set defined by

$$\mathbb{X} \stackrel{\text{def}}{=} [\mathbf{x}] \cap \bigcap_{i\in\{1,\ldots,\ell\}}^{\{q\}} \mathbf{f}_i^{-1}\left([\mathbf{y}](i)\right). \tag{3.1}$$

Note that this problem is very similar to finding a box enclosing the set $\mathbb{X}(k+1)$ defined by (2.2), except that the notations have been simplified for the sake of clarity. The correspondence between this problem and that of finding $[\mathbf{x}](k+1)$ satisfying (2.14) is as follows: $\mathbf{f}_i^{-1} \leftrightarrow \mathbf{f}_k^i \circ \mathbf{g}_{k-i}^{-1}$, $[\mathbf{y}](i) \leftrightarrow [\mathbf{y}](k-i)$, $\mathbb{X} \leftrightarrow \mathbb{X}(k+1)$ and $[\mathbf{x}]$ corresponds to a box enclosing $\mathbf{f}_k([\mathbf{x}](k))$.

3.2. Algorithm

We shall now present a new algorithm which solves the relaxed set inversion problem The principle of the method for solving the relaxed set inversion problem is illustrated by figure 3.1. In this figure, we used the notation

$$\mathbb{X}_i = \mathbf{f}_i^{-1}([\mathbf{y}](i)). \tag{3.2}$$

Subfigure (a) represents the sets $\mathbb{X}_i$ with the solution set $\mathbb{X}$ (hatched), representing the q-relaxed intersection we would like to enclose (here, $q = 1$). For each i, we first enclose the sets $[\mathbf{x}] \cap \mathbb{X}_i$ by boxes $[\mathbf{x}](i)$ as represented with dash line boxes on subfigure (b). On subfigure (c), the two grey boxes represents the q-relaxed intersection of the boxes $[\mathbf{x}](i)$. We compute a box enclosure (hatched box) of this q-relaxed intersection. On subfigure (d), we are in the same situation as we were on subfigure (a). The current box still encloses $\mathbb{X}$ but is now smaller. The process can be iterated once more as illustrated by subfigures (e) and (f). We will then converge to a steady box quickly, even if the dimension n of $\mathbf{x}$ is high. The accuracy of the enclosure can be controlled by allowing several bisections of the current box into subboxes and by iterating the contraction procedure on each subbox. The algorithm RSIVIA (for *Relaxed Set Inverter Via Interval Analysis*) for solving the relaxed set inversion problem is given by Table 3.1.

Table 3.1: Algorithm for solving the relaxed set inversion problem

Algorithm RSIVIA(in: $[\mathbf{x}], \mathbf{f}_1, \ldots, \mathbf{f}_\ell, q$, out: $[\bar{\mathbf{x}}]$)

$\mathcal{L} := \{[\mathbf{x}]\}$;
while we still have time and $\mathcal{L} \neq \emptyset$
  pull $([\mathbf{x}], \mathcal{L})$;
  Repeat several times
    for $i = 1$ to ℓ, compute $[\mathbf{x}](i)$ which encloses $[\mathbf{x}] \cap \mathbf{f}_i^{-1}([\mathbf{y}](i))$
    $[\mathbf{x}] := \left[\bigcap_{i \in \{1,\ldots,\ell\}}^{\{q\}} [\mathbf{x}](i) \right]$
  end repeat
  if $[\mathbf{x}] \neq \emptyset$, bisect $[\mathbf{x}]$ and push the resulting boxes into $\mathcal{L}$
end while
$[\bar{\mathbf{x}}] = \sqcup(\mathcal{L})$

Step 1: The list $\mathcal{L}$ contains boxes, the union of which encloses $\mathbb{X}$. It is a queue (*i.e.*, it has a first

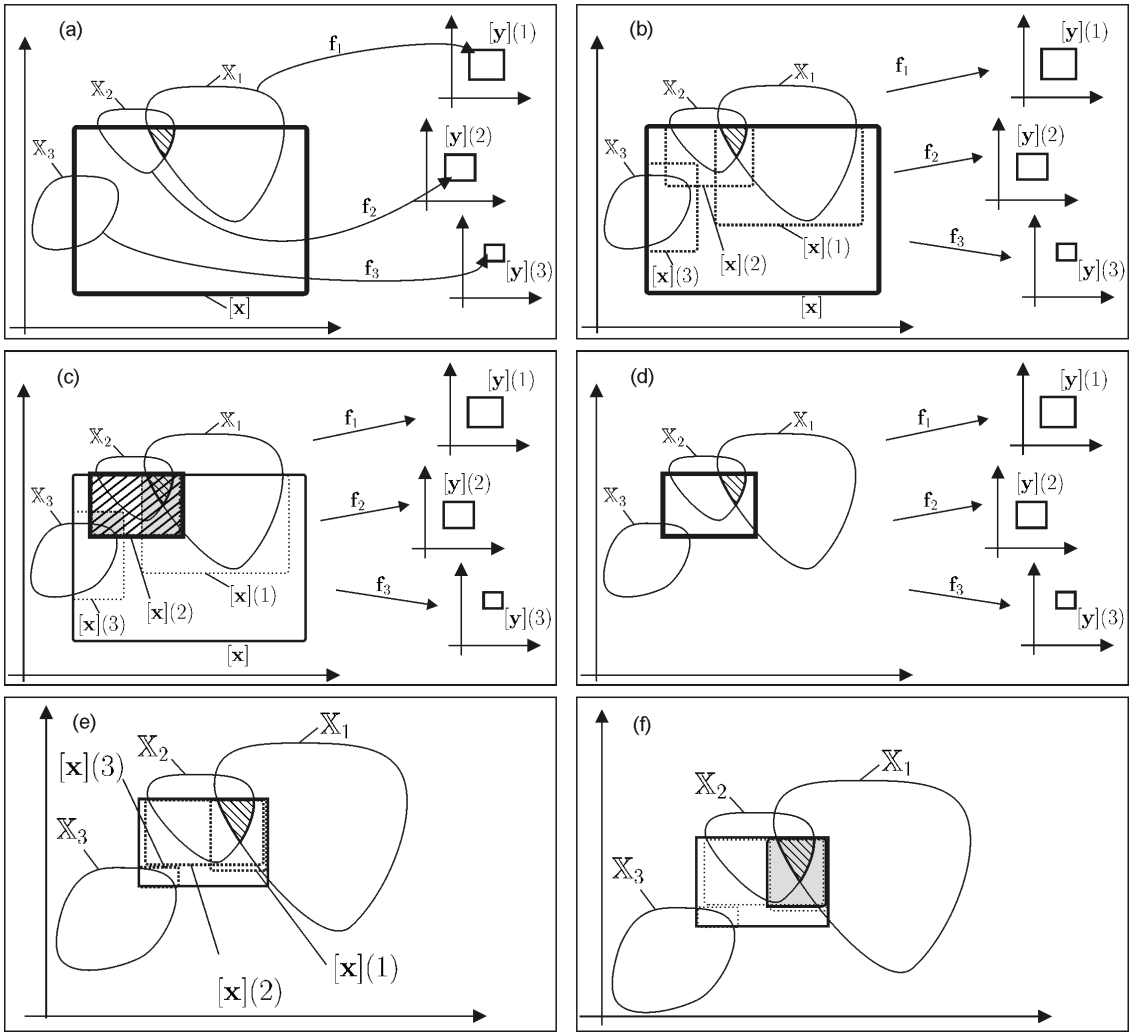

Figure 3.1: Principle of the contraction procedure for the relaxed set inversion problem

in first out structure) and is initialized with the single box $[\mathbf{x}]$.

Step 2: The algorithm RSIVIA should take less than the sampling time (between k and $k+1$) for real time applications. This is why we should contract as much as possible the list $\mathcal{L}$ within the allocated time.

Step 3: The first box (*i.e.*, the one which is waiting to be processed since the longest time) is pulled out from the list.

Step 4: The contraction procedure is iterated a number of times fixed in advance (for instance 10 times) for real time applications. If no real time implementation is required, it is better to contract until no more significant contractions of $[\mathbf{x}]$ can be observed.

Step 5: For all i, a box $[\mathbf{x}](i)$ enclosing $[\mathbf{x}] \cap \mathbf{f}_i^{-1}([\mathbf{y}](i))$ is computed. This can be done efficiently using interval analysis [Moo66] combined with constraint propagation methods. For the application presented in this paper, a single forward-backward contraction procedure (see *e.g.* [L. 01]) is implemented to compute the $[\mathbf{x}](i)$'s.

Step 6: A box enclosing the q-relaxed intersection of the $[\mathbf{x}](i)$'s is computed. Here $[\mathbb{A}]$ represents a box (as small as possible) enclosing $\mathbb{A}$. Note that computing the q relaxed intersection of ℓ boxes has a polynomial complexity, if the dimension n of the boxes is fixed (see, *e.g.*, [BR07]), but the complexity of this problem is exponential with respect to n. Figure 3.2 illustrates the principle of such an algorithm. First, generate $(2\ell-1)^n$ boxes as on the subfigure (b). Select all boxes whose centers belong to at least $\ell - q$ of the ℓ initial boxes Finally, take the enveloping box of all selected boxes. For the test case of Section 4 a much efficient algorithm has of course been implemented but the theoretical complexity remains the same (*i.e.*, $O(\ell^n)$).

Step 7: The current box is bisected into two smaller boxes. These two boxes are pushed at the end of the queue $\mathcal{L}$.

Step 8: The algorithm returns the smallest box $[\bar{\mathbf{x}}]$ enclosing all boxes stored in $\mathcal{L}$ (represented here by the box union operator $\sqcup$).

3.3. Application to the enclosure of the feasible set

Recall that (see (2.14)), at each step, we have to compute a box enclosure $[\mathbf{x}](k+1)$ for the set

$$\mathbf{f}_k([\mathbf{x}](k)) \cap \bigcap_{i\in\{0,\ldots,\ell\}}^{\{q\}} \mathbf{f}_k^i \circ \mathbf{g}^{-1}([\mathbf{y}](k-i)) \tag{3.3}$$

which has been proven to enclose $\mathbb{X}(k+1)$, in Section 2.3. Using interval arithmetic [Moo66], a box $[\mathbf{z}](k)$ enclosing the set $\mathbf{f}_k([\mathbf{x}](k))$ can easily be computed. Since the functions $\mathbf{f}_k$ have been assumed to be bijective, this is also the case for the functions $\mathbf{f}_k^i$. Thus, we can call the algorithm

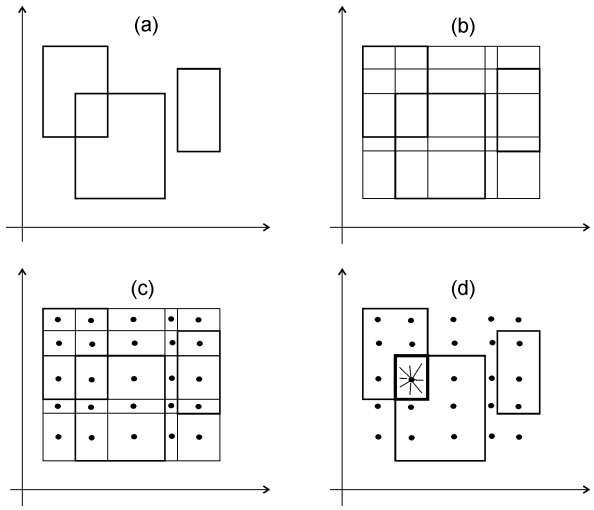

Figure 3.2: Illustration of the principle of polynomial method which computes the smallest box which contans the q-relaxed intersection of ℓ boxes

RSIVIA$([\mathbf{z}], \mathbf{g}_k \circ (\mathbf{f}_k^0)^{-1}, \ldots, \mathbf{g}_{k-\ell} \circ \left(\mathbf{f}_k^\ell\right)^{-1}, q)$ which will return a box $[\mathbf{x}](k+1)$ enclosing $\mathbb{X}(k+1)$. An illustration of the procedure is given on Figure 3.3.

4. Application to the localization and control of an underwater robot

To illustrate the efficiency of the approach, we shall consider the problem of the localization of an underwater robot. Note that set-membership methods have often been considered for the localization of robots (see, *e.g.*, [MPRH96], [HM96], in the case where the problem is linear and also [CGLP02] when the robot is underwater). In situations where strong nonlinearities are involved, interval analysis has been shown to be particularly useful (see, *e.g.*, [MLJW02], where the first localization of an actual robot has been solved with interval methods). Here, the approach is made more efficient by the addition of constraint propagation techniques.

Assume the robot is described by the following state equation

$$\left\{\begin{array}{rcl} \dot{x} &=& v\cos\theta \\ \dot{y} &=& v\sin\theta \\ \dot{\theta} &=& u_2 - u_1 \\ \dot{v} &=& u_1 + u_2 - v. \end{array}\right. \tag{4.1}$$

This model corresponds to an underwater robot with a constant depth (the depth regulation of the robot is assumed to be already solved and will not be considered here) and with no roll and pitch. Thus, our robot can be seen as a two-dimensional robot. The localization problem for this

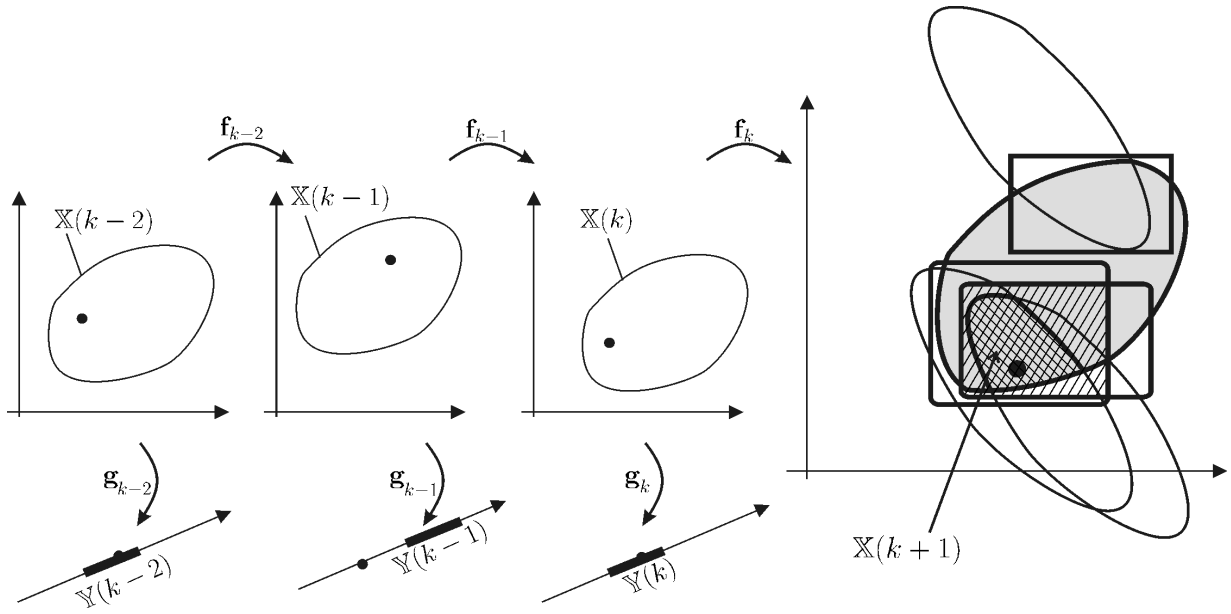

Figure 3.3: Computation of a the box $[\mathbf{x}](k+1)$ (hatched box) which encloses the feasible set $\mathbb{X}(k+1)$

type of robot in the presence of outlier is similar to that treated in [MLJW02] or [KJWM00], but, in these two papers, the outliers was treated with a static manner, *i.e.*, at each k a lot of measurements were collected (24 sensors were available for the application treated). The robot pose had to be consistent with all measurements made at time k except q of them. Here, the outliers have to be treated in a dynamic manner: the maximum number of allowed outliers is not defined for each k, but for all feasible time windows of a given length.

4.1. Observer

The system can be discretized as follows

$$\left\{\begin{array}{rcl} x(k+1) & = & x(k)+\delta.v(k).\cos\left(\theta(k)\right) \\ y(k+1) & = & y(k)+\delta.v(k).\sin\left(\theta(k)\right) \\ \theta(k+1) & = & \theta(k)+\delta.\left(u_2(k)-u_1(k)\right) \\ v(k+1) & = & v(k)+\delta.\left(u_1(k)+u_2(k)-v(k)\right) \end{array}\right. \tag{4.2}$$

or equivalently by

$$\mathbf{x}(k+1)=\mathbf{f}_k\left(\mathbf{x}(k)\right), \tag{4.3}$$

where δ is the sampling time, $\mathbf{x} = (x, y, \theta, v)$ is the state vector and

$$\mathbf{f}_k \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} x_1 + \delta.x_4.\cos(x_3) \\ x_2 + \delta.x_4.\sin(x_3) \\ x_3 + \delta.u_2(k) - \delta.u_1(k) \\ x_4 + \delta.u_1(k) + \delta.u_2(k) - \delta.x_4 \end{pmatrix}. \tag{4.4}$$

Note that the function $\mathbf{f}_k$ is invertible as required by the condition of the Theorem 2.1. The expression of $\mathbf{f}_k^{-1}$ is

$$\mathbf{f}_k^{-1} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} x_1 - \frac{\delta}{1-\delta}\left(x_4 - \delta.u_1(k) - \delta.u_2(k)\right).\cos\left(x_3 - \delta.u_2(k) + \delta.u_1(k)\right) \\ x_2 - \frac{\delta}{1-\delta}\left(x_4 - \delta.u_1(k) - \delta.u_2(k)\right).\sin\left(x_3 - \delta.u_2(k) + \delta.u_1(k)\right) \\ x_3 - \delta.u_2(k) + \delta.u_1(k) \\ \frac{1}{1-\delta}\left(x_4 - \delta.u_1(k) - \delta.u_2(k)\right) \end{pmatrix}.$$

The robot is assumed to move inside a swimming pool with a known shape. It is equipped with a sonar which makes it possible to measure the horizontal distance between the robot and the border of the pool following the direction pointed by the sonar. The sonar turns on itself and we shall denote by α the angle between the direction of the sonar and the axis of the robot. If the swimming pool is composed with planar vertical walls, the observation equation of the system is

$$d = g_k(\mathbf{x}), \tag{4.5}$$

where the observation function g_k is given by the algorithm of Table 4.1. When walls have a more complicated shape, such as cylinder, the algorithm could be adapted. In this algorithm, Step 1 computes the directional vector $\overrightarrow{\mathbf{u}}$ of the sonar. In Step 2, $\mathbf{m}$ is the position of the center of the robot. The loop of Step 3 is run for each segment $[\mathbf{a}_j, \mathbf{b}_j]$ with endpoints $\mathbf{a}_j$ and $\mathbf{b}_j$. The condition of Step 4, checks whether or not the line $(\mathbf{m}, \overrightarrow{\mathbf{u}})$ crosses the segment $[\mathbf{a}_j, \mathbf{b}_j]$. Step 5 computes the distance d_j between the robot and the corresponding wall with respect to the direction $\overrightarrow{\mathbf{u}}$. Only the minimum of all the distances d_i can be the distance between the robot and all the walls with respect to $\overrightarrow{\mathbf{u}}$. This is illustrated by Figure 4.1, left. As shown on Figure 4.1, right, the angle α between the sonar and the robot depends on k.

Even if the functions $\mathbf{f}_k$ and g_k are strongly nonlinear, the interval contraction methods required by RSIVIA can be used efficiently to compute a box $[\mathbf{x}](k)$ which encloses the feasible set $\mathbb{X}(k)$. The center $\hat{\mathbf{x}}(k)$ of this box is returned by the observer as an estimation of the actual state vector for the robot. It is this estimate that will be used by the controller to compute the control vector $\mathbf{u}(k)$.

4.2. Controller

The principle of the controller is described on Figure 4.2. First, a mission planner sends to the controller a waypoint (x_w, y_w) that has to be reached by the robot. When the current waypoint is

Table 4.1: Distance measured by the sonar between the robot and the wall (assumed to be planar and vertical), following the direction pointed by the sonar

Algorithm: g_k(in: x, y, θ, out: d)	
1	$\overrightarrow{\mathbf{u}} = (\cos(\theta + \alpha(k)); \sin(\theta + \alpha(k))), \quad d = \infty$
2	$\mathbf{m} = (x\ y)^{\mathrm{T}}$
3	for $j = 1$ to n_{walls}
4	if $\det(\mathbf{a}_j - \mathbf{m}, \overrightarrow{\mathbf{u}}) . \det(\mathbf{b}_j - \mathbf{m}, \overrightarrow{\mathbf{u}}) \geq 0$ then next j
5	$d_j := \frac{\det(\mathbf{m} - \mathbf{a}_j, \mathbf{b}_j - \mathbf{a}_j)}{\det(\overrightarrow{\mathbf{u}}, \mathbf{b}_j - \mathbf{a}_j)}$
6	if $d_j < 0$ then next j
7	$d := \min(d, d_j)$
8	next j
9	return (d).

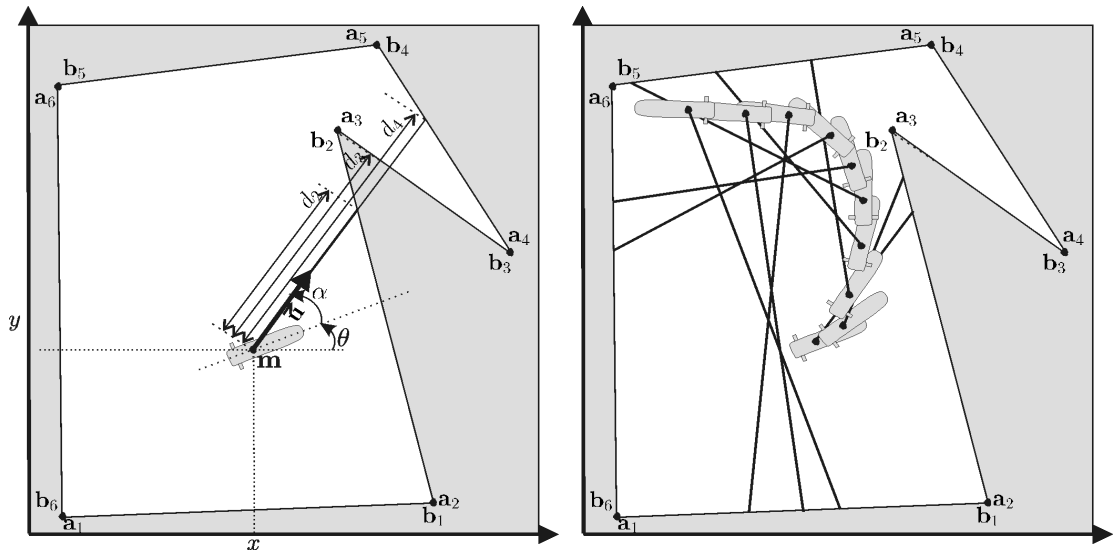

Figure 4.1: Left: distance that is supposed to be returned by the sonar; Right: when the robot is moving, the sonar turns around the robot

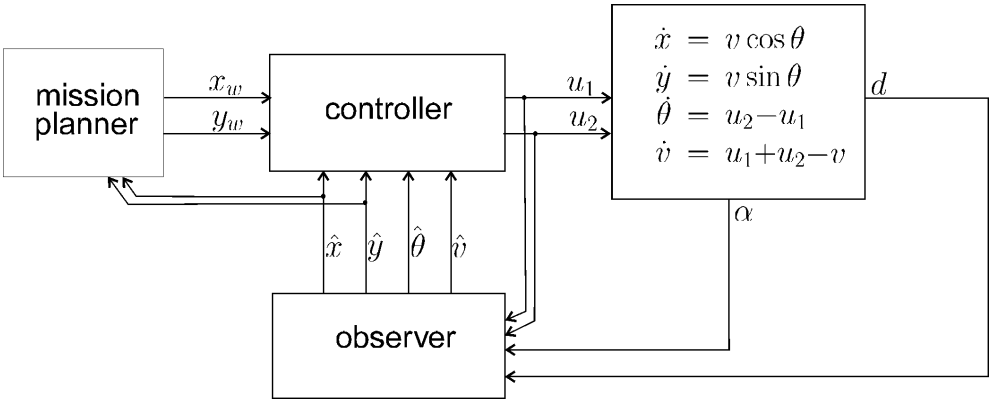

Figure 4.2: Principle of the control of the underwater robot

considered as reached with a given precision (*i.e.*, $(\hat{x} - x_w)^2 + (\hat{y} - y_w)^2 \leq \varepsilon$), the planner sends the next waypoint. The controller that has been chosen is given by the following expression:

$$\mathbf{u} = \begin{pmatrix} 1-\omega \\ 1+\omega \end{pmatrix}, \text{ where } \omega = \text{sign}\left(\det\begin{pmatrix} \cos\hat{\theta} & x_w - \hat{x} \\ \sin\hat{\theta} & y_w - \hat{y} \end{pmatrix}\right).$$

The main advantage of this controller is its simplicity. The direction to be followed by the robot is given by the vector $\mathbf{e} = (x_w - \hat{x},\ y_w - \hat{y})^{\mathrm{T}}$. The estimated orientation of the robot is given by the vector $\mathbf{v} = \left(\cos\hat{\theta},\ \sin\hat{\theta}\right)^{\mathrm{T}}$. If $\mathbf{v}$ is on the right of $\mathbf{e}$ (i.e., $\det(\mathbf{v}, \mathbf{e}) < 0$), we turn right ($\omega = 1$) otherwise, we turn left ($\omega = -1$).

4.3. Results

To illustrate the behavior of our observer and controller, we consider the problem of localization and control of an underwater robot moving inside a pool with four vertical planar walls and one vertical cylinder (which plays the role of an artificial island inside the pool). Since all walls are vertical, a projection onto the (x, y)-plane is sufficient to characterize them. The coordinates of the corners made by the vertical walls are given by

x	0	13	15	0
y	0	0	10	8

and the circle corresponding to the vertical cylinder has a center at $(10, 5)$ and a radius equal to 2.8m. The mission planner has to send the three following waypoints $(4, 5), (4, 2)$ and $(1.5, 1)$. Once a waypoint is thought to be reached with a precision less than 0.5m, the planner sends the next waypoint, until all waypoints have been sent. The sampling time is chosen as $\delta = 0.0625$s. The length of the sliding time window is chosen as $\ell = 40$, which corresponds to one complete

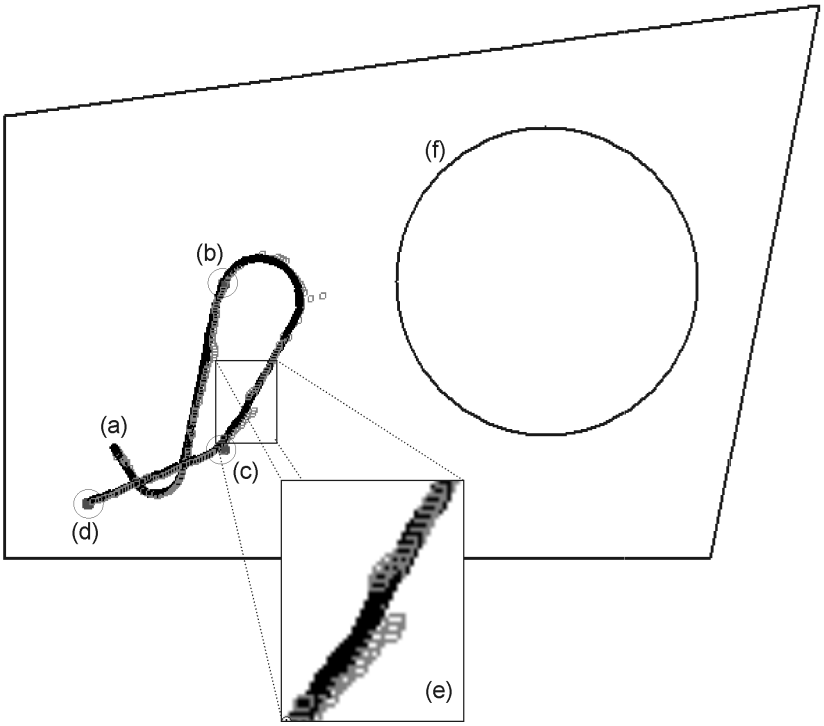

Figure 4.3: Actual (black) and estimated (gray) trajectory of the robot.

turn of the sonar. The number of allowed outliers inside a time window of length ℓ is chosen as $q = 10$. In our simulation, an outlier is generated with a probability of 0.1. In such a case, the measured distance returned by the simulated robot is fixed at 15m. This choice will facilitate the visualization of the results. Moreover, we added to the measured distance a white noise with a uniform distribution inside the interval $[-0.03, 0.03]$, which correspond to an error of $\pm$3cm.

The results obtained by our observer and controller are illustrated by Figures (4.3) and (4.4). The total computation time is less than 30sec on classical personal computer, which makes the approach consistent with real time applications. Figure (4.3) represents the shape of the pool, the trajectory performed by the robot (in black) and the estimated trajectory (small gray squares which represent the centers of the boxes that have been proven to enclose the actual position). On this figure, point (a) represents the initial position for the robot. Points (b),(c) and (d) are the waypoints that have to be reached by the robot. Note that once point (d) has been reached, the mission is finished. A zoom of a part of the trajectory is drawn at point (e). One may see the gray boxes representing the estimated position returned by our observer. At point (f) is represented the vertical cylinder which is inside the pool. To make the interval observer working, we had to adapt the observation function provided by Table 4.1, by taking into account the presence of the cylinder. This algorithm is much longer and is not given here for the sake of clarity.

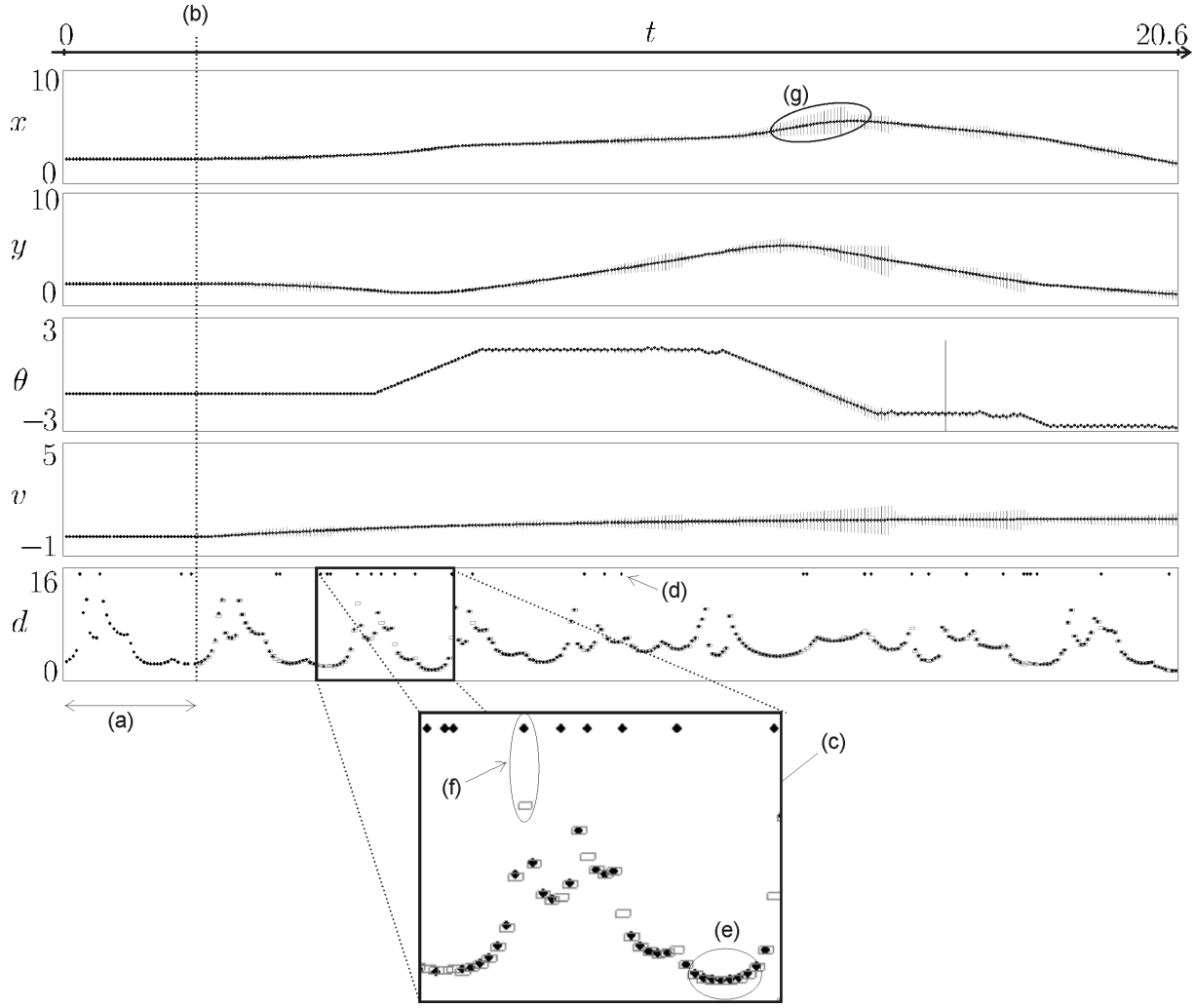

Figure 4.4: The interval observer computes intervals that are guaranteed to enclose all the feasible state variables x, y, θ, v. It also filters the measured distance d.

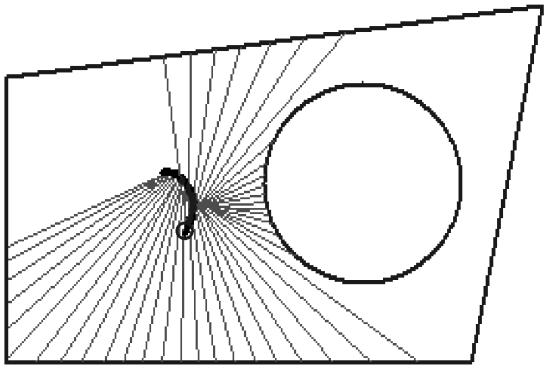

Figure 4.5: From the current estimated state vector, one is able to reconstruct the past. This can be used for instance to detect which data are outliers.

Figures (4.4) presents the temporal behavior of our interval observer. For each sampling time, the actual state vector $(x(t), y(t), \theta(t), v(t))$ and the measured distance $d(t)$ are depicted. Point (a) represents the initialization step. Its duration corresponds to the length of the sliding window (*i.e.*, $\ell.\delta = 40 * 0.0625 = 2.5$sec). The vertical bar on (b) indicates the end of the initialization step. A zoom of a sliding window is represented at point (c). The data collected are represented by the black points and the filtered distance, by the small gray squares. One can see that, inside this window, we have 9 outliers (recall that all of them correspond to a distance of 15m) which is consistent with the assumption that a maximum of 10 outliers could occur inside a window of length ℓ. A typical outlier is represented at point (d). At point (e), we can see a local minimum in the measured distance. This minimum usually corresponds to a situation where the sonar beam is orthogonal to one of the wall, but of course, we don't know a priori which wall it is. At point (f), an outlier has been detected and the estimated distance is far from the measured distance. Point (g) provides some intervals $[x(t)]$ enclosing $x(t)$. When this interval is large, the approximation is less accurate. However, for our testcase, the center is always a good approximation of $x(t)$ and it is the center that is used by the controller. Such enclosing intervals are also provided for $y(t), \theta(t)$ and $v(t)$. All these intervals are proven to to enclose the state variables as long as our MNO assumption is satisfied. Note that, for each t, from the current estimated state vector, one is able to reconstruct the past (see Figure 4.5). The small circle represents the position of the robot at the current time.

4.4. SAUC'ISSE

The principle of our observer and controller have been implemented on an actual submarine robot that participated to the SAUC'E (Student Autonomous Underwater Competition, European) competition that took place on July 2007 in Portsmouth, England. This competition has been

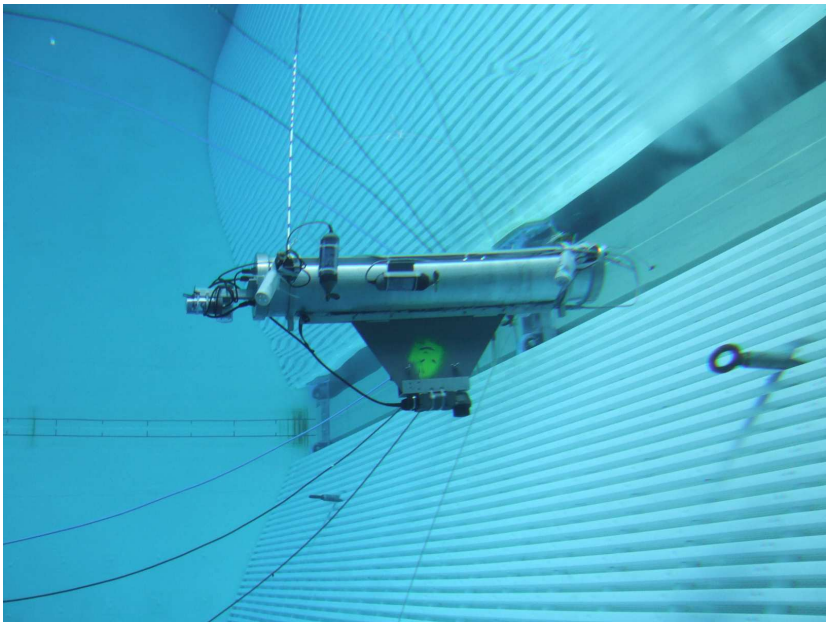

Figure 4.6: A photo of the underwater robot SAUC'ISSE

organized by DGA (France) and DSTL (UK). For more information, see

`http://www.dstl.gov.uk/news_events/competitions/sauce/`

Of course, many technical adaptations have to be done in order to take into account the real time requirement, a more complicated state space model, the depth, roll and pitch control, the informations given by the cameras, ... However, the principle of the estimation of (x, y, θ, v) remains the same. The robot, named SAUC'ISSE (SAUCe Interval Super Submarine of Ensieta), has been build by student of the ENSIETA Engineering school, Brest, France. A photo of this robot is given on Figure 4.6. Our team won the third place at the competition and got the price of innovation for control. The robot was able to localize itself and to go from one waypoint to another. Unfortunately, it was not able to find some targets and during its mission it suddenly sunk for some low-level software reasons. One should note that interval analysis has often been used for offline robotic applications [MLJW02], [DDEBC05], [Por05], [GB06], [LP03], [VSVJ06], [DJC06], ..., but to my knowledge, SAUC'ISSE is the first robot which embeds interval computation (*i.e.*, which control its trajectory using online interval methods).

5. Conclusion

In this paper, a new observer has been presented with several nice properties that we shall now recall.

- The observer *is robust with respect to outliers*. By propagating the assumptions on the possible outliers through time, we are able to be robust with respect to a bounded number of outliers, even if we have a small number of outputs in our system. To my knowledge, with existing methods, we were only able to detect outliers in a static way. It should be noted that the principle of propagating outlier assumptions could also be adapted to probabilistic observers (such as Kalman or particle filters [TBF05]) that are not based on some set membership assumptions.

- The observer *is reliable with respect to nonlinearities*. Thanks to interval analysis, we were able to deal with some nonlinear (or nondifferentiable and even noncontinuous) state equations, without linearizing or approximating them.

- The observer *can be used for real time applications*. Constraint propagation techniques, combined with interval analysis, have been used to contract the domains for the variables involved in our problem. These techniques are known to be very efficient even when the number of variables is high. An implementation on an actual robot has also demonstrated the feasibility of our approach when real time is required.

The principle of the approach is based on a new theorem that made it possible to compute recursively the set of all state vectors that are consistent with some bounded errors and a maximal number of outliers within any time window of a given length. A proof of this theorem has also been provided. A testcase illustrating the efficiency of the approach has been treated. The C++ source code of the simulation as well as an executable program are made available at the following address:

`http://www.ensieta.fr/e3i2/Jaulin/sauce_article_auto.zip`

Some limitations of our approach should also be mentioned.

- The evolution function of the state model is assumed to be invertible with respect to the state vector. But no answer has been given is the case where this function is not invertible anymore.

- When dealing with continuous systems, a discretization should be done and the guarantee is lost. Now, there exists some interval-based methods that discretize state equations in a guaranteed way (see *e.g.*, [RRC04]). It remains to study how such an interval discretization could be combined with the approach proposed here to keep guaranteed results.

- When the assumption on the maximum number of outliers is not satisfied anymore, the observer may return an emptyset. What should we do in such a situation ? Should we stop the robot to start a new initialization, and should we increase the number of allowed outliers, in an adaptative way ...

All these questions that have not been studied in this paper show that it remains a lot of room for improvements.

Acknowledgement: The author wants to thank Alexandre Goldsztejn, Gilles Chabert and Vladik Kreinovich for helpful comments and for giving the idea, the proof and the reference related to the fact that computing the q-relaxed intersection of p boxes was an NP hard problem that could be solved in a polynomial time when n was fixed. The author wants also to thank all students from ENSIETA that participated to the construction of the underwater robot.

References

[BR71] D. P. Bertsekas and I. B. Rhodes. Recursive state estimation for a set-membership description of uncertainty. *"IEEE Transactions on Automatic Control"* **16**(2), 117–128 (1971).

[BR07] L. Stewart B. Rosgen. Complexity results on graphs with few cliques. *Discrete Mathematics and Theoretical Computer Science* (2007).

[CGLP02] A. Caiti, A. Garulli, F. Livide, and D. Prattichizzo. Set-membership acoustic tracking of autonomous underwater vehicles. *Acta Acustica united with Acustica* **5**(88), 648–652 (2002).

[DDEBC05] C. Drocourt, L. Delahoche, B. Marhic E. Brassart, and A. Clerentin. Incremental construction of the robot's environmental map using interval analysis. *Global Optimization and Constraint Satisfaction: Second International Workshop, COCOS 2003* **3478**, 127–141 (2005).

[DJC06] N. Delanoue, L. Jaulin, and B. Cottenceau. Using interval arithmetic to prove that a set is path-connected. *Theoretical Computer Science, Special issue: Real Numbers and Computers* **351**(1), 119–128 (2006).

[GB06] A. Gning and Ph. Bonnifait. Constraints propagation techniques on intervals for a guaranteed localization using redundant data. *Automatica* **42**(7), 1167–1175 (2006).

[HM96] E. Halbwachs and D. Meizel. Bounded-error estimation for mobile vehicule localization. *CESA'96 IMACS Multiconference (Symposium on Modelling, Analysis and Simulation)* pages 1005–1010 (1996).

[JKBW01] L. Jaulin, M. Kieffer, I. Braems, and E. Walter. Guaranteed nonlinear estimation using constraint propagation on sets. *International Journal of Control* **74**(18), 1772–1782 (2001).

[KJWM00] M. Kieffer, L. Jaulin, E. Walter, and D. Meizel. Robust autonomous robot localization using interval analysis. *Reliable Computing* **6**(3), 337–362 (2000).

[KLP+03] V. Kreinovich, L. Longpré, P. Patangay, S. Ferson, and L. Ginzburg. Outlier detection under interval uncertainty: Algorithmic solvability and computational complexity. In I. Lirkov, S. Margenov, J. Wasniewski, and P. Yalamov, editors, "Large-Scale Scientific Computing", Proceedings of the 4th International Conference LSSC'2003 (2003).

[L. 01] L. Jaulin, M. Kieffer, O. Didrit, E. Walter. "Applied Interval Analysis, with Examples in Parameter and State Estimation, Robust Control and Robotics". Springer-Verlag, London (2001).

[LP03] F. Lydoire and P. Poignet. Nonlinear predictive control using constraint satisfaction. In "In 2nd International Workshop on Global Constrained Optimization and Constraint Satisfaction (COCOS)", pages 179–188 (2003).

[LWG87] H. Lahanier, E. Walter, and R. Gomeni. OMNE: a new robust membership-set estimator for the parameters of nonlinear models. *Journal of Pharmacokinetics and Biopharmaceutics* **15**, 203–219 (1987).

[MLJW02] D. Meizel, O. Lévêque, L. Jaulin, and E. Walter. Initial localization by set inversion. *IEEE transactions on robotics and Automation* **18**(6), 966–971 (2002).

[Moo66] R. E. Moore. "Interval Analysis". Prentice-Hall, Englewood Cliffs, NJ (1966).

[MPRH96] D. Meizel, A. Preciado-Ruiz, and E. Halbwachs. Estimation of mobile robot localization: geometric approaches. In M. Milanese, J. Norton, H. Piet-Lahanier, and E. Walter, editors, "Bounding Approaches to System Identification", pages 463–489. Plenum Press, New York, NY (1996).

[NV93] J.P. Norton and S.M. Verez. Outliers in bound-based state estimation and identification. *Circuits and Systems* **1**, 790–793 (1993).

[Por05] J.M. Porta. Cuikslam: A kinematics-based approach to slam. In "Proceedings of the 2005 IEEE International Conference on Robotics and Automation", pages 2436–2442, Barcelona (Spain) (2005).

[PW96] L. Pronzato and E. Walter. Robustness to outliers of bounded-error estimators and consequences on experiment design. In M. Milanese, J. Norton, H. Piet-Lahanier, and E. Walter, editors, "Bounding Approaches to System Identification", pages 199–212, New York (1996). Plenum.

[RRC04] T. Raissi, N. Ramdani, and Y. Candau. Set membership state and parameter estimation for systems described by nonlinear differential equations. *Automatica* **40**, 1771–1777 (2004).

[TBF05] S. Thrun, W. Bugard, and D. Fox. "Probabilistic Robotics". MIT Press, Cambridge, M.A. (2005).

[VSVJ06] P. Herrero Vinas, M. A. Sainz, J. Vehi, and L. Jaulin. Quantified set inversion algorithm with applications to control. *Reliable computing* **11**(5), 369–382 (2006).