



Set inversion for χ -algorithms, with application to guaranteed robot localization

Luc Jaulin[◊], Eric Walter^{*}, Olivier Lévéque⁺, Dominique Meizel⁺

[◊] *Laboratoire d'Ingénierie des Systèmes Automatisés,
Université d'Angers, Faculté des Sciences,
2 boulevard Lavoisier, 49045 Angers, France*

^{*} *Laboratoire des Signaux et Systèmes, CNRS-Supélec,
Plateau de Moulon, 91192 Gif-sur-Yvette, France*

⁺ *HEUDYASIC, UMR CNRS 6599,
Université de Technologie de Compiègne,
BP 529, 60205 Compiègne Cedex, France*

Abstract: Characterizing the set of all parameter vectors such that their image by a vector function belongs to a given set is a set-inversion problem. The algorithm SIVIA (Set Inversion Via Interval Analysis) makes it possible to perform this task in an approximate but guaranteed way. In the examples treated so far, the function to be inverted was given either explicitly or by a sequential algorithm. In this paper, this approach is extended to the case of branching algorithms involving *if* statements. As an illustration, the static localization of a robot from bounded-error range measurements is considered. The notion of remoteness, introduced for an archetypal but realistic sonar model, allows this problem to be cast into the set-inversion framework.

Keywords: Bounded errors, Interval analysis, Nonlinear estimation, Robot localization, Set-inversion.

1. Introduction

The problem considered in this paper is the characterization of the set

$$\mathcal{S} = \{\mathbf{p} \mid \mathbf{f}(\mathbf{p}) \in \mathcal{Y}\}, \quad (1.1)$$

where \mathbf{p} is a finite-dimensional parameter vector, \mathbf{f} a vector function and \mathcal{Y} a given set, for instance a box in some data space. \mathcal{S} may alternatively be defined as

$$\mathcal{S} = \mathbf{f}^{-1}(\mathcal{Y}), \quad (1.2)$$

so its characterization may be seen as a problem of set inversion [10]. The algorithm SIVIA (for Set Inversion Via Interval Analysis) has been proposed to allow an approximate but guaranteed characterization of \mathcal{S} by bracketing it between inner and outer sets of boxes in parameter space. Its complexity is analyzed in [11], and its convergence in [10]. It has been applied, *e.g.*, to guaranteed nonlinear parameter estimation [10] and robust stability analysis [24].

In the implementation of SIVIA, interval analysis was used to extend *sequential algorithms* on real numbers to intervals. By sequential algorithm, we mean an algorithm for which the sequencing of the execution of the instructions does not depend on the values of the input variables. Examples of non-sequential algorithms are *branching algorithms* which involve *if* statements (either explicitly or implicitly via *while do* or *repeat until* statements). Classical interval analysis no longer provides a ready-made methodology for the interval extension of branching algorithms. However, *if then else* statements can often be eliminated from the code by using Kearfott's function χ [13], for which interval extensions are available. We shall call a χ -algorithm any algorithm in which all *if then else* statements can thus be eliminated.

In this paper, we extend set inversion to χ -algorithms, and apply the resulting methodology to the guaranteed localization of a robot during a static phase, based on a finite number of range measurements by exteroceptive sensors. For a general presentation of robot navigation based on sonars, see [16]. It is well known that static localization is very difficult to perform automatically, because of the variety of the associations that can be made between measurements and landmarks of the environment. It is however a prerequisite to tracking displacements of the robot *via* recursive state estimation based, *e.g.*, on extended Kalman filtering [14] or bounded-error set estimation [18]. It is also required whenever the state estimator turns out to have failed, *e.g.*, after a sequence of collision-avoidance steps. For the time being, in the absence of specific beacons and additional sensors, no systematic, efficient and rigorous method exists to automatically estimate the initial configuration of the robot, *i.e.*, its position and orientation. We shall see that set inversion makes it possible to perform this task automatically and systematically, thereby

increasing the autonomy of the robot. The delicate problem of data association will be solved as a by-product of the procedure and the uncertainty associated with sonar measurements will be taken into account. The simulation algorithm used to compute the vector of measurements to be expected for a given configuration will be shown to be a χ -algorithm, so its inversion will be a direct application of the methodology advocated here.

The paper is organized as follows. Section 2 recalls the very few notions of interval computation needed and that of a χ function. A new version of SIVIA, more efficient and recursive, is presented in Section 3. Section 4 shows how the localization of a robot from on-board sonar measurements can be formulated as a problem of set inversion for a χ -algorithm and compares this approach with those available in the literature.

2. Interval analysis

Interval arithmetic was originally developed [21] to quantify the effect of finite-precision arithmetic on results obtained with a computer. It extends classical operators and functions on real numbers to intervals. In what follows, real variables will be denoted by lower-case letters, intervals by upper-case letters and vector intervals (or *boxes*) by bold upper-case letters. The notation used is explained in details at the end of the paper.

The next example shows how to compute the interval evaluation of the distance of a point to a line along a vector. This quantity will be useful for the localization problem of Section 4.

Example 1. *The distance from \mathbf{m} to (\mathbf{ab}) along the unit vector $\overrightarrow{\mathbf{u}}$ is*

$$\ell_{\overrightarrow{\mathbf{u}}}(\mathbf{m}, (\mathbf{ab})) = \left| \frac{(y_b - y_a)x_m + (x_a - x_b)y_m + x_b y_a - x_a y_b}{\langle \overrightarrow{\mathbf{ab}}, \overrightarrow{\mathbf{u}} \rangle} \right|. \quad (2.1)$$

If $\overrightarrow{\mathbf{U}} = [0.5, 1] \times [0.5, 1]$ and $\mathbf{M} = [1, 2] \times [1, 3]$ contain $\overrightarrow{\mathbf{u}}$ and \mathbf{m} , and if $\mathbf{a} = (6, 1)^T$ and $\mathbf{b} = (4, 8)^T$, then $\ell_{\overrightarrow{\mathbf{u}}}(\mathbf{m}, (\mathbf{ab}))$ belongs to the interval

$$L_{\overrightarrow{\mathbf{U}}}(\mathbf{M}, (\mathbf{ab})) = \left| \frac{7[1,2]+2[1,3]-44}{-2[0.5,1]+7[0.5,1]} \right| = \left| \frac{[7,14]+[2,6]-44}{[-2,-1]+[3.5,7]} \right| = \left| \frac{-[24,35]}{[1.5,6]} \right| = [4, 70/3]. \quad (2.2)$$

◊

To evaluate the interval counterpart of a function defined by a sequential algorithm it suffices to replace each statement by its interval counterpart. However, many functions incorporated in models used in control or in robotics are nonsequential, because of phenomena such as saturation or encounters with mechanical stops or because of the interplay between continuous and discrete parts of hybrid systems. As a result, branching appears in the algorithms describing the models, and a specific methodology is needed to deal with *if then else* statements in an interval context. Although this methodology is rather classical in interval analysis, it seems absent from the control literature, and will be presented before applying it to the localization problem.

Extending the Boolean test involved in the *if* statement to intervals requires a three-valued logic [20]. Denote the set of all Booleans by $\mathcal{B} = \{0, 1\}$, where 0 stands for *false* and 1 for *true*. An *interval Boolean* is an element of $\mathcal{IB} = \{0, [0, 1], 1\}$, where $[0, 1]$ means that the variable is *indeterminate*. All usual operations on sets, such as union or intersection and logical computations apply to interval Booleans. For instance, if $(A \text{ and } B)$ is denoted by $A \wedge B$, and $(A \text{ or } B)$ by $A \vee B$, then $([0, 1] \vee 1) \wedge ([0, 1] \vee 0) = 1 \wedge [0, 1] = [0, 1]$. Intersection and union should not be confused with the logical operators *and* and *or*. For instance, $[0, 1] \cap 1 = 1$, whereas $[0, 1] \wedge 1 = [0, 1]$. Three-valued logic makes it possible to extend tests involving points to intervals. Let \mathcal{IR}^n be the set of all n -dimensional real boxes (or vectors of real intervals). An *inclusion test* for the Boolean function (or *test*) $t : \mathcal{R}^n \rightarrow \mathcal{B}$ is a function $T : \mathcal{IR}^n \rightarrow \mathcal{IB}$ such that $T(\mathbf{P}) = a$, with $a \in \mathcal{B}$, implies that $\forall \mathbf{p} \in \mathbf{P}, t(\mathbf{p}) = a$. The inclusion test can be seen as a counterpart for Boolean functions of the inclusion function [21] for real functions. An inclusion test is *inclusion monotonic* if $\mathbf{P} \subset \mathbf{Q} \implies T(\mathbf{P}) \subset T(\mathbf{Q})$.

Example 2. Consider the test $t(\mathbf{p}) = (\mathbf{p} \in \mathcal{D})$, where \mathcal{D} is a given set. An *inclusion-monotonic inclusion test* associated to $t(\mathbf{p})$, and denoted by $T(\mathbf{P}) = (\mathbf{P} \in \mathcal{D})$, is

$$\begin{aligned} & \text{if} && \mathbf{P} \subset \mathcal{D}, && \text{then } (\mathbf{P} \in \mathcal{D}) = 1, \\ & \text{if} && \mathbf{P} \cap \mathcal{D} = \emptyset, && \text{then } (\mathbf{P} \in \mathcal{D}) = 0, \\ & \text{otherwise} && && (\mathbf{P} \in \mathcal{D}) = [0, 1]. \end{aligned} \tag{2.3}$$

◊

The inclusion test T' is *stronger* than the inclusion test T if $\forall \mathbf{P} \in \mathcal{IR}^n, T'(\mathbf{P}) \subset T(\mathbf{P})$.

Example 3. Consider a test $t(\mathbf{p}) = (f(\mathbf{p}) \geq 0)$. Assume that two interval counterparts

F_1 and F_2 are available for f . They are associated with the two inclusion tests $T_1(\mathbf{P}) = (F_1(\mathbf{P}) \in \mathcal{R}^+)$ and $T_2(\mathbf{P}) = (F_2(\mathbf{P}) \in \mathcal{R}^+)$. A stronger inclusion test is $T_3(\mathbf{P}) = T_1(\mathbf{P}) \cap T_2(\mathbf{P})$. This illustrates the potential advantage of considering several tests in parallel. \diamond

Example 4. To bisect a box \mathbf{P} means to cut it along a symmetry plane normal to a side of maximum length. This generates two boxes \mathbf{P}_1 and \mathbf{P}_2 such that $\mathbf{P} = \mathbf{P}_1 \cup \mathbf{P}_2$. If T is inclusion monotonic, then the test T' defined by $T'(\mathbf{P}) := T(\mathbf{P}_1) \cup T(\mathbf{P}_2)$ is stronger than T , because $T(\mathbf{P}_1)$ and $T(\mathbf{P}_2)$ may both be equal to 0 or to 1 even if $T(\mathbf{P}) = [0, 1]$. This suggests that bisecting boxes may help resolve ambiguous situations, an idea that is the corner-stone of SIVIA. \diamond

Once Boolean tests have been extended to intervals, it remains to be decided which branch of the algorithm should be executed. Kearfott's χ function, first mentioned in [13], is a possible way of eliminating the problem. If \mathbf{x} is a real vector and y and z are two real numbers, then $\chi(\mathbf{x}, y, z)$ is equal to y if $\mathbf{x} \leq \mathbf{0}$ and to z otherwise. Its interval counterpart is given by

$$\chi(\mathbf{X}, Y, Z) = \begin{cases} Y & \text{if } (\mathbf{X} \leq \mathbf{0}) = 1 \\ Z & \text{if } (\mathbf{X} \leq \mathbf{0}) = 0 \\ Y \cup Z & \text{otherwise} \end{cases} . \quad (2.4)$$

3. SIVIA

SIVIA (Set Inverter Via Interval Analysis) [10] is a branch-and-bound algorithm that characterizes the set \mathcal{S} of all feasible parameter vectors in some prior box of interest \mathbf{P}_0 by partitioning \mathbf{P}_0 into three subpavings (list of nonoverlapping boxes). The first one \mathcal{S}^- consists of boxes proved to be inside \mathcal{S} , the second one of boxes proved to be outside \mathcal{S} , and the last one $\Delta\mathcal{S}$ of boxes for which no conclusion could be reached. The solution set can thus be bracketed between inner and outer sets: $\mathcal{S}^- \subset \mathcal{S} \subset \mathcal{S}^- \cup \Delta\mathcal{S}$.

The main advantage of the new version to be presented below is that it drastically decreases the number of boxes to be stored. As the multiplication of these boxes is the main factor limiting the complexity of the problems that can be handled, this is a major achievement. Recursive implementation makes it possible to obtain this result without increasing computing time. The new version also takes advantage of the notion of inclusion tests

introduced in the previous section. \mathcal{S} is assumed to be defined by a Boolean function $t(\mathbf{p})$ that takes the value *true* if and only if \mathbf{p} is acceptable, i.e., $\mathcal{S} = \{\mathbf{p} \mid t(\mathbf{p}) = 1\} = t^{-1}(1)$. Assume that an inclusion test T for t is available. If a box \mathbf{P} satisfies $T(\mathbf{P}) = 1$, then it is inside \mathcal{S} and stored in \mathcal{S}^- . If $T(\mathbf{P}) = 0$, then \mathbf{P} is outside \mathcal{S} and discarded. Otherwise, it will be split into subboxes, unless it is smaller than a given required accuracy parameter ε , in which case it will be put in $\Delta\mathcal{S}$. SIVIA, presented on Table 1, calls a recursive function CLASSIFY, which performs most of the work. In what follows, the *width* $w(\mathbf{P})$ of a box is the length of its largest side(s).

SIVIA
Step 0 $\Delta\mathcal{S} := \emptyset; \mathcal{S}^- := \emptyset;$
Step 1 $c_0 := \text{CLASSIFY}(\mathbf{P}_0);$
Step 2 If $c_0 = 1$ then $\mathcal{S}^- := \{\mathbf{P}_0\};$
Step 3 If $c_0 = [0, 1]$ then $\Delta\mathcal{S} := \{\mathbf{P}_0\};$
Step 4 return $\Delta\mathcal{S}, \mathcal{S}^-;$
CLASSIFY(\mathbf{P})
Step 1 If $T(\mathbf{P}) = 0$ or 1 , return($T(\mathbf{P})$);
Step 2 If $w(\mathbf{P}) < \varepsilon$, return($[0, 1]$);
Step 3 Bisect \mathbf{P} to get \mathbf{P}_1 and \mathbf{P}_2 ;
Step 4 $c_1 := \text{CLASSIFY}(\mathbf{P}_1); c_2 := \text{CLASSIFY}(\mathbf{P}_2);$
Step 5 If $c_1 = c_2$, return(c_1);
Step 6 If $c_1 = [0, 1]$, store \mathbf{P}_1 into $\Delta\mathcal{S}$;
Step 7 If $c_2 = [0, 1]$, store \mathbf{P}_2 into $\Delta\mathcal{S}$;
Step 8 If $c_1 = 1$, store \mathbf{P}_1 into \mathcal{S}^- ;
Step 9 If $c_2 = 1$, store \mathbf{P}_2 into \mathcal{S}^- ;
Step 10 return(\emptyset).

Table 1: New recursive version of SIVIA.

CLASSIFY has two roles. The first one is to improve the strength of the test $T(\mathbf{P})$ by considering smaller boxes and taking advantage of inclusion monotonicity. This role has been fulfilled when return takes place at Step 1, 2 or 5. In either case, CLASSIFY refrains from storing \mathbf{P} in the corresponding subpaving, in the hope that it may be possible to reunite it later with another box via Step 5, thereby decreasing the number of boxes to be stored. Numerical experiments have shown that this reduction may be very substantial indeed. The second role of CLASSIFY is to store boxes in the appropriate subpavings.

This is performed by Steps 6 to 9. Note that when $c_1 = 0$ (resp. $c_2 = 0$), the box \mathbf{P}_1 (resp. \mathbf{P}_2) is eliminated. Returning \emptyset at Step 10 indicates to the calling program that \mathbf{P} or its subboxes have all been classified and need no longer be considered. Step 2 makes the algorithm finite by forbidding bisection *ad infinitum*.

To illustrate the procedure, assume that at Step 5, $c_1 = c_2 = 1$. Rather than storing \mathbf{P}_1 and \mathbf{P}_2 individually in \mathcal{S}^- , it is much more efficient to keep them united in their father box \mathbf{P} . This reasoning can be carried over several generations of subboxes obtained by bisection; in the limit, \mathbf{P}_0 may end up as a single box in \mathcal{S}^- . Assume now that at Step 5 $c_1 = [0, 1]$ and $c_2 = \emptyset$. This means that \mathbf{P}_2 has been classified and that \mathbf{P}_1 has been partitioned into (possibly many) ambiguous boxes, the widths of which are smaller than ε . \mathbf{P}_1 cannot be reunited with \mathbf{P}_2 and must be stored (as a single box) into $\Delta\mathcal{S}$. The value \emptyset is then returned to the calling program, which indicates that the current box \mathbf{P} can be dropped from further consideration. If the calling program was CLASSIFY (at Step 4), then \mathbf{P} will become either \mathbf{P}_1 or \mathbf{P}_2 in the calling program and c_1 or c_2 will take the value \emptyset .

In this new version, the stack that was explicit in the previous nonrecursive version of SIVIA is made implicit by using recursivity.

4. Robot localization

The problem of the guaranteed localization of a robot considered in this section is similar to that described in [17], where much less detail was provided on the methodology. Localizing a robot in a 2D environment means estimating its configuration $\mathbf{p} = (x, y, \theta)^T$, where (x, y) is the position of the origin of the robot frame \mathcal{M} and θ is the heading angle of the robot, both in the world frame \mathcal{W} (see Figure 4.1).

Various approaches have been proposed in the literature to deal with the localization problem where the distances from some robot sensors to some landmarks are measured, and we shall only present recent contributions. If it can be assumed that data association has already been performed, *i.e.*, that the correspondence between the landmarks and the sensors is known, then algorithms running in time linear in the number of landmarks can be found [2]. Local methods especially suitable for processing sonar scans obtained by ultrasonic sensors are proposed in [25] and [26], they also require the robot to be already approximately localized. These assumptions seems however fairly unrealistic when

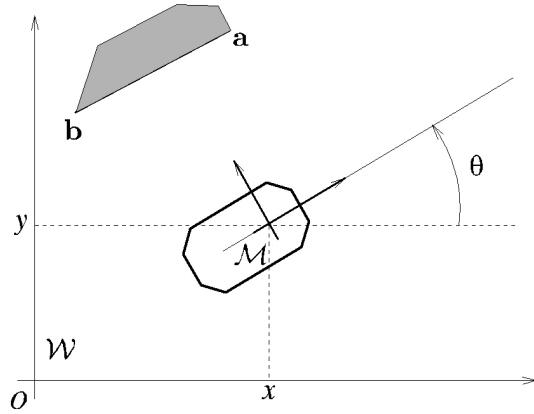


Figure 4.1: World frame \mathcal{W} and robot frame \mathcal{M} . The parameters to be estimated are those of the configuration of the robot, *i.e.*, x , y and θ . The segment with endpoints **a** and **b** represents a given landmark.

nothing is known *a priori* of the actual position of the robot. They are not required by [9], where a clustering technique maximizing the number of measurements consistent with a given configuration of the robot is proposed. Nor are they required by [4], where a global Monte Carlo method is shown to be more accurate and less memory-intensive than more classical grid-based methods. The main limitation of these two approaches is that no guarantee can be provided about their results. An approach to providing such a guarantee can be found in [8], where a combination of stochastic and set-theoretic uncertainty is considered. Although it is mentioned that extension to multivariable cases would be possible, the method is developed for a one dimensional problem and thus not applicable to our problem where $\dim \mathbf{p}$ is equal to three.

The set-inversion approach advocated in this paper is able to localize the robot in a global and guaranteed way, without requiring any prior data association. We are not aware of any other method that could make similar claims in a multidimensional case. The approach is illustrated in the case where the distance measurements are computed from the time-lag between reception and emission of ultrasonic waves emitted by sonars.

4.1. Localization as a problem of set inversion

The notation to be used for the various geometrical entities involved is summarized at the bottom of the paper. The configuration of the robot is assumed to be in some prior box of

interest \mathbf{P}_0 in configuration space, large enough to contain all configurations of interest. The landmarks of the environment are assumed to be j_{\max} oriented segments $[\mathbf{a}_j \mathbf{b}_j]$, the collection of which constitutes the *map*. Each of these segments has a fixed location in \mathcal{W} . By convention, when going from \mathbf{a}_j to \mathbf{b}_j , the reflecting face of the segment is on the left. A necessary condition for a signal sent by a sonar located at \mathbf{s} to be reflected by the segment $[\mathbf{a}_j \mathbf{b}_j]$ is then that \mathbf{s} be on the left side of $[\mathbf{a}_j \mathbf{b}_j]$, *i.e.*,

$$\mathbf{s} \in \Delta_{a_j b_j} = \{\mathbf{m} \mid \det(\overrightarrow{\mathbf{ma}}_j, \overrightarrow{\mathbf{mb}}_j) > 0\}. \quad (4.1)$$

The robot is equipped with i_{\max} on-board sonars. Each sonar emits a ultrasonic wave, measures the time-lag between emission and reception of the wave reflected or diffracted by the environment and converts it into a distance, assuming knowledge of the speed of sound. As illustrated by Figure 4.2, the i th sonar is installed on the vehicle at the coordinates $(\tilde{x}_s(i), \tilde{y}_s(i))$ in \mathcal{M} , and the orientation of its emission axis is specified by the angle $\tilde{\theta}_s(i)$. The i th sonar returns a measurement $d(i)$ of the distance from $\mathbf{s}(i)$, the location of the i th sonar in \mathcal{W} , to some unknown landmark at least partly located in its emission cone. To take measurement inaccuracy into account, with each data point $d(i)$ is associated a feasible interval $D_i = [d(i)(1 - \alpha_i), d(i)(1 + \alpha_i)]$, where α_i is the relative precision of the i th measurement, assumed to be known. Each measurement is assumed to be the result of a single specular reflection (multiple reflections are not taken into account for the time being).

With each sonar, an emission cone \mathcal{C} can be associated, characterized as

$$\mathcal{C}(\mathbf{s}, \overrightarrow{\mathbf{u}_1}, \overrightarrow{\mathbf{u}_2}) = \{\mathbf{m} \in \mathcal{R}^2 \mid \det(\overrightarrow{\mathbf{u}_1}, \overrightarrow{\mathbf{sm}}) \geq 0 \text{ and } \det(\overrightarrow{\mathbf{u}_2}, \overrightarrow{\mathbf{sm}}) \leq 0\}, \quad (4.2)$$

where \mathbf{s} is its vertex, with coordinates those of the sonar, and $\overrightarrow{\mathbf{u}_1}$ and $\overrightarrow{\mathbf{u}_2}$ are unit direction vectors. Each sonar is such that the aperture of the emission cone is smaller than $\pi/2$. This implies that $\langle \overrightarrow{\mathbf{u}_1}, \overrightarrow{\mathbf{u}_2} \rangle > 0$. Moreover, by convention, $\overrightarrow{\mathbf{u}_1}$ and $\overrightarrow{\mathbf{u}_2}$ are chosen such that $\det(\overrightarrow{\mathbf{u}_1}, \overrightarrow{\mathbf{u}_2}) > 0$. For any given configuration $\mathbf{p} = (x, y, \theta)^T$, and any given sonar with coordinates \tilde{x}_s and \tilde{y}_s and orientation $\tilde{\theta}_s$, the coordinates of the vertex of \mathcal{C} in \mathcal{W} are

$$\mathbf{s} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} \tilde{x}_s \\ \tilde{y}_s \end{bmatrix}, \quad (4.3)$$

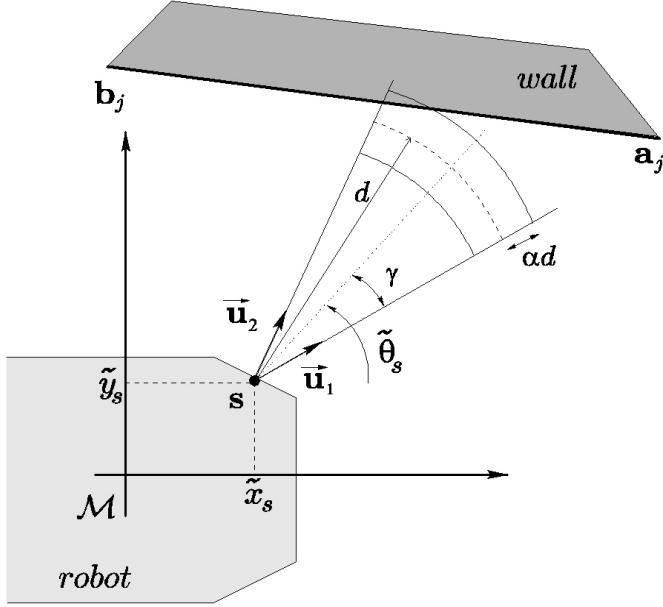


Figure 4.2: Characterization of a measurement performed by a given sonar. The sonar \mathbf{s} returns the measurement d of the distance from \mathbf{s} to an unknown landmark $a_j b_j$ at least partly located in its emission cone.

and the coordinates of the unit direction vectors in \mathcal{W} are

$$\vec{\mathbf{u}}_1 = \begin{bmatrix} \cos(\theta + \tilde{\theta}_s - \gamma) \\ \sin(\theta + \tilde{\theta}_s - \gamma) \end{bmatrix} \text{ and } \vec{\mathbf{u}}_2 = \begin{bmatrix} \cos(\theta + \tilde{\theta}_s + \gamma) \\ \sin(\theta + \tilde{\theta}_s + \gamma) \end{bmatrix}, \quad (4.4)$$

where γ denotes the half aperture of \mathcal{C} . These characteristics of \mathcal{C} depend on the configuration \mathbf{p} and sonar i considered. We should thus have written $\tilde{x}_s(i)$, $\tilde{y}_s(i)$, $\tilde{\theta}_s(i)$, $\mathbf{s}(i, \mathbf{p})$, $\vec{\mathbf{u}}_1(i, \mathbf{p})$ and $\vec{\mathbf{u}}_2(i, \mathbf{p})$. In what follows, however, this dependence will be omitted wherever possible to simplify notation.

Define the *remoteness* of a cone $\mathcal{C} = \mathcal{C}(\mathbf{s}, \vec{\mathbf{u}}_1, \vec{\mathbf{u}}_2)$ from an oriented segment $[\mathbf{ab}]$ by

$$\begin{aligned} r(\mathbf{s}, \vec{\mathbf{u}}_1, \vec{\mathbf{u}}_2, \mathbf{a}, \mathbf{b}) &= \infty && \text{if } \mathbf{s} \notin \Delta_{ab} \text{ or } [\mathbf{ab}] \cap \mathcal{C} = \emptyset, \\ &= \min_{\mathbf{m} \in [\mathbf{ab}] \cap \mathcal{C}} \|\overrightarrow{\mathbf{sm}}\| && \text{otherwise.} \end{aligned} \quad (4.5)$$

The remoteness is thus the distance that would be reported by the sonar \mathbf{s} if the environment consisted only of $[\mathbf{ab}]$. When $[\mathbf{ab}]$ is outside the emission cone \mathcal{C} , or when it is not properly oriented, the distance is considered as infinite. It would be easy to modify the definition of the remoteness to take into account additional knowledge such as the existence of blind zones.

Given the error bounds, a configuration \mathbf{p} is consistent with the measurement interval D_i associated with the i th sonar if and only if

$$f_i(\mathbf{p}) \triangleq \min_{j \in \{1, \dots, j_{\max}\}} r(\mathbf{s}(\mathbf{p}), \vec{\mathbf{u}}_1(\mathbf{p}), \vec{\mathbf{u}}_2(\mathbf{p}), \mathbf{a}_j, \mathbf{b}_j) \in D_i. \quad (4.6)$$

This amounts to saying that there exists a landmark consistent with the measurement and that no other landmark would have led to a shorter range measurement.

Remark 1. If the segment $[\mathbf{a}_j, \mathbf{b}_j]$ illuminated by the i th sonar was known a priori, as assumed by most classical localization methods, the minimization involved in (4.6) could be dropped, and the localization algorithm to be presented could still be used, while requiring much less computation. The minimization in (4.6) is the key to a global resolution of the problem of data association. \diamond

If \mathbf{D} is the box with components D_i and $\mathbf{f}(\mathbf{p})$ the vector function with coordinates $f_i(\mathbf{p})$, the feasible set for the configuration vector can be defined by

$$\mathcal{S} = \{\mathbf{p} \mid \mathbf{f}(\mathbf{p}) \in \mathbf{D}\} = \mathbf{f}^{-1}(\mathbf{D}). \quad (4.7)$$

Characterizing \mathcal{S} is therefore a set-inversion problem, which can be solved using SIVIA, with $T(\mathbf{P})$ an inclusion test for $t(\mathbf{p}) = (\mathbf{f}(\mathbf{p}) \in \mathbf{D})$. Provided that an algorithm is available for evaluating $r(\mathbf{s}, \vec{\mathbf{u}}_1, \vec{\mathbf{u}}_2, \mathbf{a}_j, \mathbf{b}_j)$, $t(\mathbf{p})$ and its interval counterpart $T(\mathbf{P})$ can be computed using (4.6), (4.4) and (4.3).

We have seen in Example 2 that the Boolean operator \in can be given an interval meaning. The use of a programming language that allows operator overloading¹ (such as C++, FORTRAN 90 and ADA) makes it possible to use exactly the same code to compute $T(\mathbf{P})$, provided that the point variables \mathbf{s} , $\vec{\mathbf{u}}_1$, $\vec{\mathbf{u}}_2$, x , y , θ , f_i and \mathbf{f} are replaced by their interval counterparts \mathbf{S} , $\vec{\mathbf{U}}_1$, $\vec{\mathbf{U}}_2$, X , Y , Θ , F_i and \mathbf{F} and that all the necessary operators and functions $+$, \cos , \sin , \min , \in and $r(., ., ., ., .)$ have been suitably overloaded.

Let us now describe an algorithm for computing r . When used with interval arguments in a context of operator overloading, the same algorithm will be used to compute R . Let

¹Such languages allow the redefinition of classical operators such as $+, -, *, /$ for structures (such as matrices, vectors, complex numbers, intervals, ...) that have not been initially defined in the language.

\mathbf{h} be the orthogonal projection of \mathbf{s} onto (\mathbf{ab}) , \mathbf{h}_1 be the intersection of the lines $(\mathbf{s}, \overrightarrow{\mathbf{u}_1})$ and (\mathbf{ab}) , and \mathbf{h}_2 be the intersection of the lines $(\mathbf{s}, \overrightarrow{\mathbf{u}_2})$ and (\mathbf{ab}) .

The minimizer \mathbf{m}^* of $\|\overrightarrow{\mathbf{sm}}\|$, if any, belongs to the finite set $\mathcal{K} = \{\mathbf{h}, \mathbf{h}_1, \mathbf{h}_2, \mathbf{a}, \mathbf{b}\}$. Therefore

$$\begin{aligned} r(\mathbf{s}, \overrightarrow{\mathbf{u}_1}, \overrightarrow{\mathbf{u}_2}, \mathbf{a}, \mathbf{b}) &= \infty && \text{if } \mathbf{s} \notin \Delta_{ab}, \\ &= \min_{\mathbf{m} \in \mathcal{K} \cap [\mathbf{ab}] \cap \mathcal{C}} \|\overrightarrow{\mathbf{sm}}\| && \text{otherwise.} \end{aligned} \quad (4.8)$$

Assume that $\mathbf{s} \in \Delta_{ab}$, so that $\det(\overrightarrow{\mathbf{s}\mathbf{a}}, \overrightarrow{\mathbf{s}\mathbf{b}}) > 0$. In order to express $r(\mathbf{s}, \overrightarrow{\mathbf{u}_1}, \overrightarrow{\mathbf{u}_2}, \mathbf{a}, \mathbf{b})$ as a χ -algorithm, introduce the following vector functions:

$$\begin{aligned} \mathbf{f}_h(\mathbf{s}, \overrightarrow{\mathbf{u}_1}, \overrightarrow{\mathbf{u}_2}, \mathbf{a}, \mathbf{b}) &= \left(\begin{array}{cccc} \langle \overrightarrow{\mathbf{ab}}, \overrightarrow{\mathbf{sa}} \rangle & -\langle \overrightarrow{\mathbf{ab}}, \overrightarrow{\mathbf{sb}} \rangle & \langle \overrightarrow{\mathbf{u}_1}, \overrightarrow{\mathbf{ab}} \rangle & -\langle \overrightarrow{\mathbf{u}_2}, \overrightarrow{\mathbf{ab}} \rangle \end{array} \right)^T, \\ \mathbf{f}_{h_i}(\mathbf{s}, \overrightarrow{\mathbf{u}_i}, \mathbf{a}, \mathbf{b}) &= \left(\begin{array}{cc} \det(\overrightarrow{\mathbf{u}_i}, \overrightarrow{\mathbf{s}\mathbf{a}}) & -\det(\overrightarrow{\mathbf{u}_i}, \overrightarrow{\mathbf{s}\mathbf{b}}) \end{array} \right)^T, i = 1, 2, \\ \mathbf{f}_v(\mathbf{s}, \overrightarrow{\mathbf{u}_1}, \overrightarrow{\mathbf{u}_2}, \mathbf{v}) &= \left(\begin{array}{cc} -\det(\overrightarrow{\mathbf{u}_1}, \overrightarrow{\mathbf{s}\mathbf{v}}) & \det(\overrightarrow{\mathbf{u}_2}, \overrightarrow{\mathbf{s}\mathbf{v}}) \end{array} \right)^T, \mathbf{v} = \mathbf{a}, \mathbf{b}. \end{aligned} \quad (4.9)$$

It is trivial to show that

$$\begin{aligned} \mathbf{h} \in [\mathbf{ab}] \cap \mathcal{C} &\iff \mathbf{f}_h(\mathbf{s}, \overrightarrow{\mathbf{u}_1}, \overrightarrow{\mathbf{u}_2}, \mathbf{a}, \mathbf{b}) \leq \mathbf{0}, \\ \mathbf{h}_i \in [\mathbf{ab}] \cap \mathcal{C} &\iff \mathbf{f}_{h_i}(\mathbf{s}, \overrightarrow{\mathbf{u}_i}, \mathbf{a}, \mathbf{b}) \leq \mathbf{0}, \quad i = 1, 2, \\ \mathbf{v} \in [\mathbf{ab}] \cap \mathcal{C} &\iff \mathbf{f}_v(\mathbf{s}, \overrightarrow{\mathbf{u}_1}, \overrightarrow{\mathbf{u}_2}, \mathbf{v}) \leq \mathbf{0}, \quad \mathbf{v} = \mathbf{a}, \mathbf{b}. \end{aligned} \quad (4.10)$$

The algorithm described in Table 2 computes $r(\mathbf{s}, \overrightarrow{\mathbf{u}_1}, \overrightarrow{\mathbf{u}_2}, \mathbf{a}, \mathbf{b})$, based on (4.10), (4.8) and (4.1). In its description, $\ell(\mathbf{s}, (\mathbf{ab}))$ denotes the distance from the point \mathbf{s} to the line (\mathbf{ab}) and $\ell_{\overrightarrow{\mathbf{u}}}(\mathbf{s}, (\mathbf{ab}))$ the distance from \mathbf{s} to (\mathbf{ab}) along the unit vector $\overrightarrow{\mathbf{u}}$.

$r(\mathbf{s}, \overrightarrow{\mathbf{u}_1}, \overrightarrow{\mathbf{u}_2}, \mathbf{a}, \mathbf{b})$
Input : $\mathbf{a}, \mathbf{b}, \mathbf{s}, \overrightarrow{\mathbf{u}_1}, \overrightarrow{\mathbf{u}_2}$;
Step 1 $r_h = \chi(\mathbf{f}_h(\mathbf{s}, \overrightarrow{\mathbf{u}_1}, \overrightarrow{\mathbf{u}_2}, \mathbf{a}, \mathbf{b}), \ell(\mathbf{s}, (\mathbf{ab})), \infty)$
Step 2 $r_{h_1} = \chi(\mathbf{f}_{h_1}(\mathbf{s}, \overrightarrow{\mathbf{u}_1}, \mathbf{a}, \mathbf{b}), \ell_{\overrightarrow{\mathbf{u}_1}}(\mathbf{s}, (\mathbf{ab})), \infty)$
Step 3 $r_{h_2} = \chi(\mathbf{f}_{h_2}(\mathbf{s}, \overrightarrow{\mathbf{u}_2}, \mathbf{a}, \mathbf{b}), \ell_{\overrightarrow{\mathbf{u}_2}}(\mathbf{s}, (\mathbf{ab})), \infty)$
Step 4 $r_a = \chi(\mathbf{f}_a(\mathbf{s}, \overrightarrow{\mathbf{u}_1}, \overrightarrow{\mathbf{u}_2}, \mathbf{a}), \ \overrightarrow{\mathbf{s}\mathbf{a}}\ , \infty)$
Step 5 $r_b = \chi(\mathbf{f}_b(\mathbf{s}, \overrightarrow{\mathbf{u}_1}, \overrightarrow{\mathbf{u}_2}, \mathbf{b}), \ \overrightarrow{\mathbf{s}\mathbf{b}}\ , \infty)$
Step 6 $r = \chi(\det(\overrightarrow{\mathbf{s}\mathbf{a}}, \overrightarrow{\mathbf{s}\mathbf{b}}), \infty, \min\{r_h, r_{h_1}, r_{h_2}, r_a, r_b\})$
Output: return (r)

Table 2: Computation of the remoteness $r(\mathbf{s}, \overrightarrow{\mathbf{u}_1}, \overrightarrow{\mathbf{u}_2}, \mathbf{a}, \mathbf{b})$ of a landmark $[\mathbf{a}, \mathbf{b}]$ from a cone $\mathcal{C}(\mathbf{s}, \overrightarrow{\mathbf{u}_1}, \overrightarrow{\mathbf{u}_2})$.

Remark 2. This algorithm has been kept simple on purpose. The performance of its interval evaluation could be improved by adding more tests. For example, taking into account that

$$\det(\overrightarrow{\mathbf{u}_1}, \overrightarrow{\mathbf{ab}}) \leq 0 \quad \text{and} \quad \det(\overrightarrow{\mathbf{u}_2}, \overrightarrow{\mathbf{ab}}) \leq 0 \implies r(\mathbf{s}, \overrightarrow{\mathbf{u}_1}, \overrightarrow{\mathbf{u}_2}, \mathbf{a}, \mathbf{b}) = \infty, \quad (4.11)$$

the interval evaluation of r becomes more efficient if the following instruction is inserted after Step 6:

$$r = \chi \left(\begin{pmatrix} \det(\overrightarrow{\mathbf{u}_1}, \overrightarrow{\mathbf{ab}}) & \det(\overrightarrow{\mathbf{u}_2}, \overrightarrow{\mathbf{ab}}) \end{pmatrix}^T, \infty, r \right). \quad (4.12)$$

◇

4.2. Comparison with other approaches

The existing methods for the static localization of a robot from onboard sonar measurements can be classified depending on how noise is characterized (by bounds or by probability distributions) and on whether they handle the problem of associating the range data with segments of the map. Most methods assume the data association to be

given *a priori*. Based on a statistical description of noise by probability distributions and a linearization of the model, a static version of extended Kalman filtering (EKF) can then be used, which leads to very simple computations, but the results are only local, and disconnected solutions due, *e.g.*, to map symmetries (see Figure 4.5 below) are not dealt with. When bounds are available instead of noise distributions, bounded-error estimation can be used, see, *e.g.*, [22], [23] and [19], and the references therein. For models linear in their parameters, the most commonly employed method, known in the literature as OBE (outer bounding ellipsoid) [1], recursively computes an ellipsoid guaranteed to contain all possible values of the parameter vector that are consistent with the measurements and error bounds [18]. Computation is almost as simple as with EKF and, as for EKF, a linearization is necessary to make OBE applicable to robot localization and a prior data association is usually assumed available.

The solution classically used to perform data association when it is not given *a priori* is to enumerate all possible associations between measurements and segments before eliminating as many of them as possible. This multiple hypothesis testing (MHT) can be combined with EKF [3][14][15] or OBE [6][7], which gives these methods the ability to deal with ambiguity. Provided that the bounds used for OBE take into account the linearization error, the results obtained by combining MHT and OBE can even be made global and guaranteed. However, the complexity of MHT increases so quickly with the number of segments in the map and the number of sonars that it can only be used on fairly simple examples. Moreover, the results obtained by combining MHT and OBE are usually very pessimistic, because the error bounds have to be taken large and because of the ellipsoidal approximation committed each time a new sensor datum is incorporated. By contrast, the set-inversion method advocated here uses an exact description of the set of all configurations that are consistent with the measurements and hypotheses on the error bounds. It does not perform any linearization. The data association does not need to be available *a priori*, and all feasible associations are obtained as a by-product of the algorithm. The set \mathcal{S} of all feasible configurations is enclosed in the set of boxes computed by the algorithm. Precision in the description of \mathcal{S} can be increased by decreasing ε at the cost of increasing the volume of computation. The complexity of the method is exponential in the number of parameters to be estimated. However, here, $\dim(\mathbf{p}) = 3$, small enough for the method to remain tractable. Table 3 summarizes the properties of various approaches considered.

Estimation method	Data association	Validity of results	Model	Error description	Disconnected solutions	Complexity
EKF	a priori	local	linearized	distribution	ignored	low
OBE	a priori	local	linearized	bounds	ignored	low
MHT + EKF	generated	local	linearized	distribution	handled	huge
MHT + OBE	generated	global	linearized	bounds	handled	huge
SIVIA	generated	global	nonlinear	bounds	handled	high

Table 3: Comparison of approaches for static localization.

4.3. Examples

Consider first the situation displayed on Figure 4.3, in which five sonars have reported measurements. For each of them, the half aperture of the emission cone is taken as $\gamma = 11.3^\circ$ and the relative precision of the measurement as $\alpha = 0.02$. With each data point $d(i)$, a thick uncertainty arc can be associated, resulting from the intersection of the corresponding emission cone with the ring centered on $s(i)$ with interval radius $D_i = [d(i)(1 - \alpha), d(i)(1 + \alpha)]$. A configuration is feasible if each of these thick arcs intersects a landmark and no other landmark is located between one arc and the corresponding sonar.

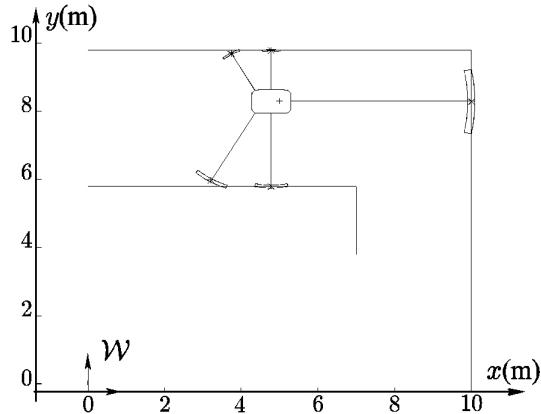


Figure 4.3: Actual configuration of the robot, symmetrical test case.

Figure 4.4 describes the set $\mathcal{S}^- \cup \Delta\mathcal{S}$, which encloses \mathcal{S} . It consists of two connected sets; each corresponding to a possible association between the data and the landmarks of the environment, as shown on Figure 4.5.

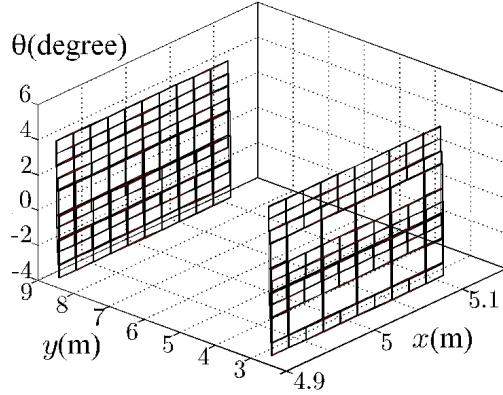


Figure 4.4: Outer approximation of \mathcal{S} , symmetrical test case. The configuration of the robot is guaranteed to be in one of the boxes. The two disconnected groups of boxes correspond to radically different data associations. On a given component the precision for x and y is similar. The apparent plate shape of these two components is due to the scales.

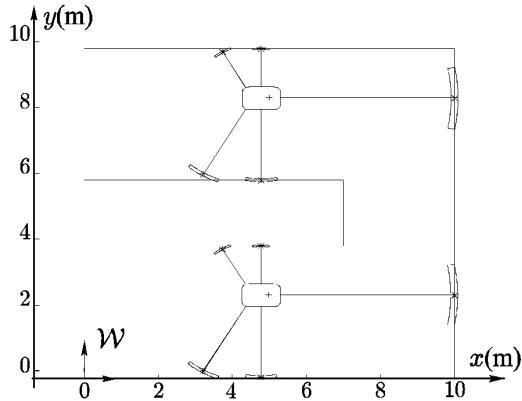


Figure 4.5: Configurations belonging to disconnected components of \mathcal{S} .

Consider now the situation displayed on Figure 4.6, where the same sonars have reported measurements but where the environment is no longer symmetrical with respect to an axis parallel to x . $\mathcal{S}^- \cup \Delta\mathcal{S}$, as estimated by SIVIA, is presented on Figure 4.7. It has only one connected component, because the matching between measurements and landmarks is now unique.

In both cases, for $\varepsilon = 0.02$, the result is obtained in less than one minute with a P166MMX processor.

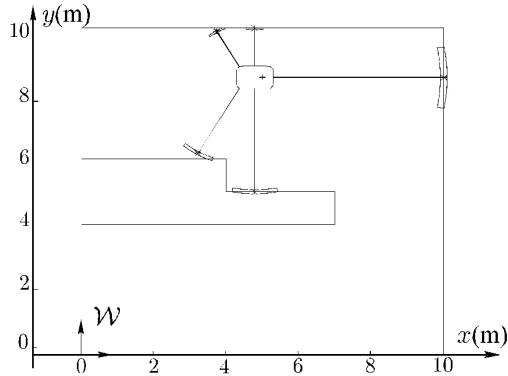


Figure 4.6: Actual configuration of the robot, asymmetrical test case.

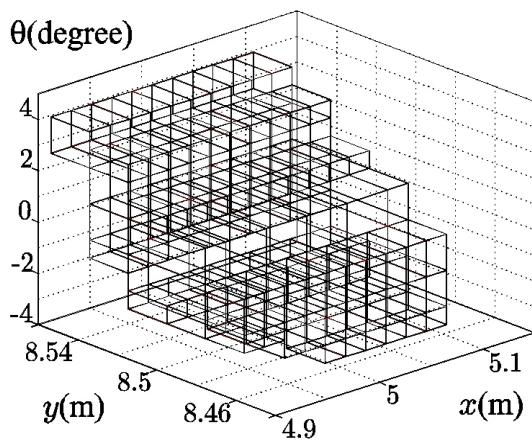


Figure 4.7: Outer approximation of \mathcal{S} , asymmetrical test case. There is now only one connected component. The uncertainty on the configuration of the robot is less than ± 10 cm for x and y , and less than ± 4 degrees for θ .

5. Conclusions

Two contributions to the methodology of nonlinear bounded-error estimation have been presented. The first one is a new version of SIVIA, which drastically decreases the number of boxes to be stored, thereby significantly increasing the complexity of the problems that can be considered. The second one is an extension of the class of models that can be considered to models involving functions computed by branching algorithms, implicitly or explicitly involving *if* statements. Such models include those of physical phenomena involving saturation or mechanical stops and of systems that change their dynamical behavior under conditions described by logical tests.

A third contribution of the paper is a detailed formalization of the guaranteed localization of a robot from bounded-error range data as a set-inversion problem. Because of the many different situations that have to be accounted for, computation relies on a branching algorithm. Trying all *a priori* possible associations of landmarks to the distances measured by the sonar belt, which is a bottleneck of global autonomous robot localization because of its combinatorial complexity, is avoided, and a posterior association is provided by the algorithm.

These three contributions have been put at work to enclose all feasible configurations of a robot in a reasonably small (but possibly disconnected) set. The method can deal with a large number of sonars and with a map consisting of a large number of segments, and the results provided are global and guaranteed. To the best of our knowledge, no other method can make similar claims. Extension to the tracking of a moving robot and to the case where the data contain outliers due, *e.g.*, to faulty sensors or to an outdated map is under consideration.

Notation

Vectors are in bold with an arrow on top: $\vec{\mathbf{u}}$. Points are in bold: $\mathbf{a}, \mathbf{b}, \mathbf{c}$. Coordinates for two-dimensional vectors $\vec{\mathbf{u}}$ and points \mathbf{a} are denoted by x_u, y_u and x_a, y_a .

$\langle \vec{\mathbf{u}}, \vec{\mathbf{v}} \rangle$: scalar product of $\vec{\mathbf{u}}$ and $\vec{\mathbf{v}}$,
(\mathbf{ab})	: line supported by \mathbf{a} and \mathbf{b} ,
$\vec{\mathbf{ab}}$: vector from \mathbf{a} to \mathbf{b} ,
$(\mathbf{s}, \vec{\mathbf{u}})$: line supported by \mathbf{s} and with direction vector $\vec{\mathbf{u}}$,
γ	: half aperture of the emission cone,
i	: sonar index, ($i = 1, \dots, i_{\max}$),
j	: landmark index, ($j = 1, \dots, j_{\max}$),
$\ell(\mathbf{s}, (\mathbf{ab}))$: distance from \mathbf{s} to (\mathbf{ab}) ,
$\ell_{\vec{\mathbf{u}}}(\mathbf{s}, (\mathbf{ab}))$: distance from \mathbf{s} to (\mathbf{ab}) along the unit vector $\vec{\mathbf{u}}$,
$\mathbf{p} = (x, y, \theta)^T$: robot configuration,
\mathcal{S}	: Set of all feasible robot configurations.

References

- [1] Belforte, G., B. Bona and V. Cerone (1990). Parameter estimation algorithms for a set-membership description of uncertainty. *Automatica*, **26**, 887-898.
- [2] Betke, M. and L. Gurvits (1997). Mobile robot localization using landmarks, *IEEE Transactions on Robotics and Automation*, **13**(2), 251 -263.
- [3] Crowley, J.L. (1989), World modeling and position estimation for a mobile robot using ultrasonic ranging, *Proc. Int. Conf. on Robotics and Automation*, Scottsdale, 674-680.
- [4] Dellaert, F., D. Fox, W. Burgard and S. Thrun (1999). Monte Carlo localization for mobile robots. *IEEE International Conference on Robotics and Automation*, **2**, 1322 -1328.
- [5] Drumheller, M. (1987). Mobile robot localization using sonar, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **9**(2), 325-332.
- [6] Halbwachs, E. and D. Meizel (1996). Bounded-error estimation for mobile vehicle localization. *CESA'96 IMACS Multiconference (Symposium on Modelling, Analysis and Simulation)*, Lille, **2**, 1005-1010.
- [7] Halbwachs, E. and D. Meizel (1997). Multiple hypotheses management for mobile vehicles localization , *CD-ROM of the European Control Conference on ECC'97*, Belgium.
- [8] Hanebeck, U.D. and J. Horn (1999). A new estimator for mixed stochastic and set theoretic uncertainty models applied to mobile robot localization. *IEEE International Conference on Robotics and Automation*, **2**, 1335 -1340.
- [9] Holenstein, A.A., M.A. Muller and E. Badreddin (1992). Mobile robot localization in a structured environment cluttered with obstacles. In Proceedings of IEEE International Conference on Robotics and Automation, **3**, 2576 -2581.
- [10] Jaulin, L. and E. Walter (1993). Set inversion via interval analysis, *Automatica*, **29**(4), 1053-1064,
- [11] Jaulin, L. and E. Walter (1993). Guaranteed nonlinear parameter estimation from bounded-error data via interval analysis, *Math. Comput. Simulation*, **35**, 1923-1937.

- [12] Jaulin, L., E. Walter and O. Didrit (1996). Guaranteed robust nonlinear parameter bounding, *CESA '96 IMACS Multiconference (Symposium on Modelling, Analysis and Simulation)*, Lille, **2**, 1156-1161.
- [13] Kearfott, R.B. (1995). A Fortran 90 environment for research and prototyping of enclosure algorithms for nonlinear equations and global optimization. *ACM Trans. Math. Software*, **21**(1), 63-78.
- [14] Leonard, J.J. and H. F. Durrant-Whyte (1991). Mobile robot localization by tracking geometric beacons, *IEEE Trans. on Robotics and Automation*, **7**(3), 376–382.
- [15] Leonard, J.J. and H. F. Durrant-Whyte (1991). Simultaneous map building and localization for an autonomous mobile robot, *Proc. Int. Workshop on Intelligent Robots and Systems*, Osaka, 1442-1447.
- [16] Leonard, J.J. and H. F. Durrant-Whyte (1992). *Directed Sonar Sensing for Mobile Robot Navigation*, Kluwer Academic Publishers, Boston.
- [17] Lévéque, O., L. Jaulin, D. Meizel and E. Walter (1997). Vehicle localization from inaccurate telemetric data: a set inversion approach, *Proc. 5th IFAC Symp. on Robot Control SY.RO.CO. '97*, Nantes, France, **1**, 179-186.
- [18] Meizel, D., A. Preciado-Ruiz and E. Halbwachs (1996). *Estimation of the location of a mobile robot: geometric approaches*. In *Bounding Approaches to System Identification*, M. Milanese, J. Norton, H. Piet-Lahanier and E. Walter, (eds.), Plenum Press, New York, pp. 463-489.
- [19] Milanese, M., J.P. Norton, H. Piet-Lahanier and E. Walter, Eds. (1996). *Bounding Approaches to System Identification*, Plenum, New York.
- [20] Moore, R.E. (1968). Practical aspects of interval computation, *Appl. Math.*, **13**, 52-92.
- [21] Moore, R.E. (1979). *Methods and Applications of Interval Analysis*, SIAM, Philadelphia.
- [22] Norton J. (Ed.) (1994). Special issue on bounded-error estimation: Issue 1, *Int. J. of Adaptive Control and Signal Processing*, **8**(1), 1-118.
- [23] Norton J. (Ed.) (1995). Special issue on bounded-error estimation: Issue 2, *Int. J. of Adaptive Control and Signal Processing*, **9**(1), 1-132.

- [24] Walter, E. and L. Jaulin (1994). Guaranteed characterization of stability domains via set inversion, *IEEE Trans. on Autom. Control*, **39**(4), 886-889.
- [25] Zhao Feng-Ji, Guo Hai-Jiao and K. Abe (1997), A mobile robot localization using ultrasonic sensors in indoor environment. *6th IEEE International Workshop on Robot and Human Communication, RO-MAN '97*, 52 -57
- [26] Zhao Feng-Ji, Guo Hai-Jiao and K. Abe (1998), Mobile robot localization using two sonar sensors and one natural landmark. *37th SICE Annual Conference SICE '98*, 893 -898