Integrity Zone Computation using Interval Analysis

Vincent Drevelle, Philippe Bonnifait

Integrity Zone Computation using Interval Analysis

Vincent Drevelle, Philippe Bonnifait,
Heudiasyc UMR 6599, Université de Technologie de Compiègne

BIOGRAPHY

Vincent Drevelle (1985) graduated as a computer science engineer from the Université de Technologie de Compiègne (UTC), France, in 2007; then he obtained a master degree in system and information technologies in 2008. Since September 2008 he has been at Heudiasyc UMR 6599, France, as a Ph.D. student in Computer Science and Engineering at the UTC. His current research interests are in set-theoretic methods and interval analysis, for multi-sensor localization of vehicles.

Philippe Bonnifait (1969) graduated from the Ecole Superieure d'Electronique de l'Ouest, France, in 1992 and obtained his Ph.D. degree in automatic control and computer science from the Ecole Centrale de Nantes, France, in 1997. In December 2005, he obtained his Habilitation Diriger des Recherches from the Université de Technologie de Compiègne (UTC). Since September 1998 he has been at Heudiasyc UMR 6599, France, and he is currently professor in the UTC's Computer Science and Engineering department. His current research interests are in Intelligent Vehicles and Advanced Driving Assistance Systems, with particular emphasis on dynamic ego-localization based on multisensor-fusion (GNSS, dead-reckoning and GIS).

ABSTRACT

Our approach, using a set-inversion method in a bounded-error context, enables to compute a position fix in the form of a localization area taking into account all sources of error (notably space vehicles position errors) without linearizing. The method computes an integrity zone recursively using bisection. Robustness can also be achieved with several simultaneous faulty measurements. A way to set bounds on measurements to meet an integrity risk requirement is presented. An experimental validation using real L1 code measurements is also presented in the paper.

1 INTRODUCTION

GNSS is broadly used in applications requiring a positioning service. In addition to a punctual estimate of the position, uncertainty (precision and accuracy) indicators are crucial when the position estimate leads to a decision linked to safety.

The snapshot GNSS localization problem consists in inverting a non-linear pseudo-range observation function, with often redundant measurements. This is generally solved as a non-linear Least Squares problem, using a Gauss-Newton algorithm. Alternative techniques (like Bancroft's method [1, 2]) also provide a non-iterative computation of position.

Then, localization uncertainty can be roughly estimated from User Equivalent Range Error (UERE) and Dilution Of Precision (DOP). This however only provides estimates of horizontal and vertical error standard deviations, *i.e.,* how measurement noise is reported on position. To deal with outlier measurements, Fault Detection and Exclusion (FDE) can be implemented if data redundancy exists. M-estimation or Range Consensus algorithms [3] are examples to do FDE. Unfortunately, FDE can miss small faults and this minimal detectable bias is usually added to the effect of the noise to characterize an integrity zone under the hypothesis of only one fault at a time.

In this paper, a new framework to address these issues is presented. It relies on a set-inversion method in a bounded-error context. It computes a position fix in the form of a localization area taking into account all sources of error (notably space vehicles position errors) without linearizing. The method characterize an outer-bound integrity zone recursively using bisection. Using a relaxed constraint version of the method, robustness can also be achieved with several simultaneous faulty measurements.

After a presentation of set-theoretic localization, around a simple example, we will introduce software tools based on interval analysis allowing to perform

set-inversion. As we deal with bounded error models, a method to set bounds on measurement intervals so as to meet specific risk requirements is presented. The computation of location zones is performed with real GPS L1 code measurements. An evaluation of the experimental results is done, as well as a comparison between robust and non-robust set-inversion results.

2 SET-THEORETIC LOCALIZATION

Let's consider a pedagogical example having similarities with GNSS navigation. A robot and three beacons in a planar world communicate altogether via a radio link. The robot is equipped with an ultrasonic emitter, while the beacons feature a receptor. To range itself to the beacons, the robot simultaneously emits a radio message and an ultrasound at time t_e. Beacons start timing, and stop when they receive the ultrasound at t_r. Since radio propagation time is negligible compared to sound travel time, the time of flight measurement is $t_r - t_e$. Knowing the speed of sound c_s, distance measurements to each beacon can easily be determined $d^i = c_s \cdot (t_r^i - t_e)$.

Each measurement has to be represented as the set of possible values, taking uncertainty into account. Intervals are commonly used to express measurement inaccuracy. In the case of time of flight measurements based on ultrasound transducers, a proportional error is expected, due to the variation of the speed of sound. Measurements can be represented as intervals

$$[d^i] = [(1-\alpha)\cdot d^i, (1+\alpha)\cdot d^i]$$

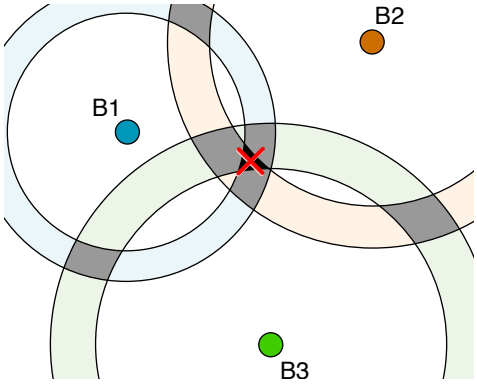
Each measurement acts as a constraint on the robot location, setting bounds on the distance between the robot and the beacon. Thus we have the membership relation:

$$\sqrt{(x_R - x_{B^i})^2 + (y_R - y_{B^i})^2} \in [d^i] \qquad (1)$$

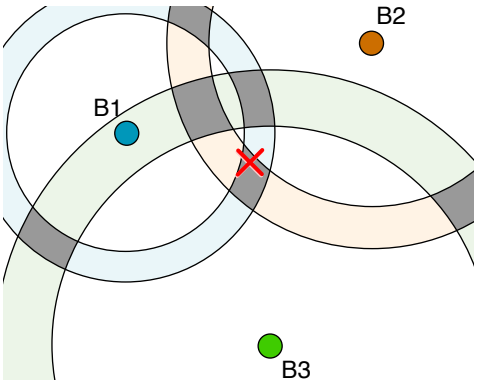
Given equation 1, each measurement constrains the robot location inside a ring, whose inner and outer radii are respectively the lower and outer bounds of the measurement interval $[d^i]$.

Since there are three beacons, equation 1 has to be verified for all three measurements. As a consequence, the robot is located inside the intersection of the three measurement annuli (Fig. 1a), given the measurement error bounds.

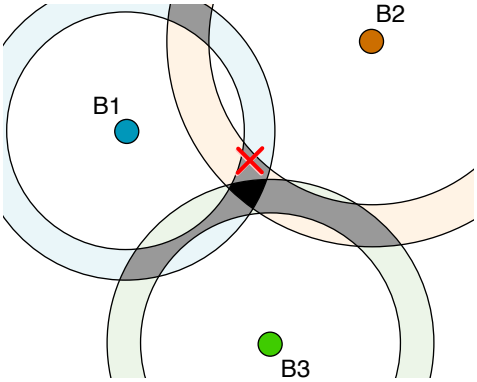
Now let's consider the case of a beacon measurement inconsistent with respect to the bounded-error model,



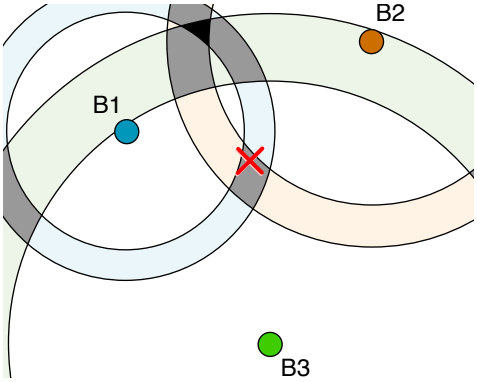
(a) No erroneous measurement



(b) Erroneous measurement from green beacon *B3*



(c) Non-detectable small error



(d) Non-detectable large error

Figure 1: Localization using 3 beacons.
Solution set in black. 1-relaxed solution set in black and grey. The red cross represents the actual robot position.

i.e., the geometrical distance between the robot and the beacon is not in the measurement interval. This kind of measurement is often called "outlier" or "fault". There are two possible consequences:

- There is no common intersection for the three measurement annuli (Fig. 1b). In this case, the solution is the empty set, which indicates an inconsistency between model and measurements. One can thus easily detect the presence of an erroneous measurement.
- There is a non-empty solution. This is the case in Fig. 1c and 1d, where the true robot location is not inside the measurement annulus generated by beacon *B3*, but the intersection of the three annuli is not the empty set. The set-inversion method is unable to detect the erroneous measurement in such a case, and returns a solution set which does not include the actual location.

Robustification of the method against erroneous measurements can be achieved by allowing the presence of at most one aberrant measurement (in this example case). This is done by relaxing the constraints intersection: instead of returning the set of solutions compatible with three measurements, we will consider the set of solutions compatible with at least two measurements. In the case of Fig. 1, it corresponds to the gray and black surfaces. This method can be generalized to allowing q erroneous measurements, using the q-relaxed intersection of the constraints [4]. The main drawback of allowing erroneous measurements is that it generates larger solution sets.

Since robust set-theoretic localization is done by relaxing constraints, one should ensure than the problem remains constrained enough to get usable results. In the example, three non aligned beacons are needed for an non ambiguous localization. If one constraint is relaxed, two-beacon solutions are considered. Two beacons are not enough to get a non ambiguous location: such a configuration generally generates two location zones. As a consequence, the 1-relaxed solution sets of Fig. 1 are made of several disjoint subsets. Thus, robust set-inversion methods should only be used when there is enough data redundancy to keep the problem well constrained.

Another robust scheme called *Guaranteed Minimum Outlier Number Estimator* [5] (GOMNE) consists in adaptively relaxing a growing number of constraints, as long as the solution set is empty. The main advantage is to keep the solution set small as long as there is no inconsistency between measurements. However, since we already shown that erroneous measurements may lead to a non-empty solution set, GOMNE does not ensures that the true solution is inside the solution set. In figure 1a, GOMNE will return the black zone; in figure 1b, the gray zone will be returned. In these two cases, the returned location zone is consistent with ground truth. On the contrary, in the cases of figures 1c and 1d, GOMNE will return the black zone, which does not contain the actual robot location. Therefore, GOMNE is not adapted to protection zone computation.

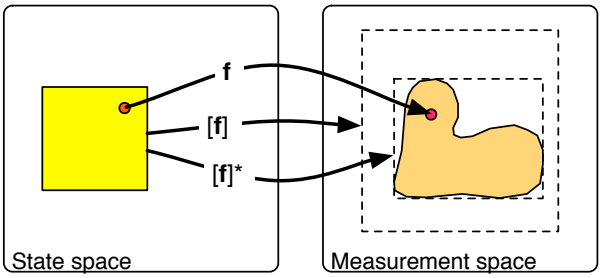


Figure 2: Inclusion functions. $[\mathbf{f}]$ *is an inclusion function for* $\mathbf{f}$. $[\mathbf{f}]^*$ *is the minimal inclusion function for* $\mathbf{f}$.

3 SET INVERSION VIA INTERVAL ANALYSIS

3.1 Interval analysis

Computing and handling exact representations of arbitrary sets is not tractable on a computer. An efficient representation is to consider intervals, and their multidimensional extension: *boxes* (also called *interval vectors*). $\mathbb{IR}$ will denote the set of real intervals, and $\mathbb{IR}^n$ is the set of n-dimensional boxes.

The classical real arithmetic operations ($+, -, \times$ and $\div$) can be extended to intervals. They are defined such as ($\diamond$ being a binary operator)

$$[x] \diamond [y] = [\{x \diamond y \in \mathbb{R} | x \in [x], y \in [y]\}].$$

In other words, the interval extension of an operator returns the smallest interval containing all the results of the operation when the two operands cover their respective intervals.

Elementary functions such as $\tan, \sin, \exp \ldots$ also extend to intervals. Given a function $f : \mathbb{R} \to \mathbb{R}$, its interval extension $[f]$ satisfies

$$[f]([x]) = [\{f(x) | x \in [x]\}].$$

Let's consider a function $\mathbf{f}$ from $\mathbb{R}^n$ to $\mathbb{R}^m$. The interval function $[\mathbf{f}]$ from $\mathbb{IR}^n$ to $\mathbb{IR}^m$ is an *inclusion*

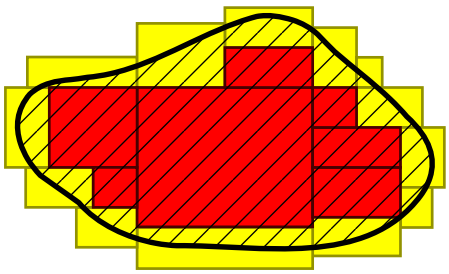


Figure 3: Bracketing of the hatched set between two subpavings. *Red boxes: inner subpaving, Red and yellow: outer subpaving*

function (Fig. 2) for $\mathbf{f}$ if

$$\forall [\mathbf{x}] \in \mathbb{IR}^n, \mathbf{f}([\mathbf{x}]) \subset [\mathbf{f}]([\mathbf{x}]).$$

The minimal inclusion function for $\mathbf{f}$, $[\mathbf{f}]^*$ is defined such as for any $[\mathbf{x}]$, $[\mathbf{f}]^*([\mathbf{x}])$ is the smallest box that contains $\mathbf{f}([\mathbf{x}])$ — *i.e.,* the *interval hull* of $\mathbf{f}([\mathbf{x}])$.

If a function $f : \mathbb{R}^n \to \mathbb{R}$ can be expressed as a finite composition of operators $+, -, \times, \div$ and elementary functions (sin, cos, sqr...), an inclusion function $[f]$ for f is obtained by replacing each variable and each operator or function by their interval counterparts. This inclusion function is called the *natural inclusion function* of f. If f only involves continuous operators and functions, and if moreover each variable appears at most once in the expression of f, then the natural inclusion function is minimal.

The natural inclusion function for a vector function $\mathbf{f}$ can be obtained by taking natural inclusion functions for each of its coordinate functions f_i.

A *subpaving* of a box $[\mathbf{x}]$ is the union of non-empty non-overlapping subboxes of $[\mathbf{x}]$. They can be used to approximate compact sets in a guaranteed way. An guaranteed approximation of the compact set $\mathbb{X}$ can be done by bracketing it between an inner subpaving $\underline{\mathbb{X}}$ and an outer subpaving $\overline{\mathbb{X}}$ such as $\underline{\mathbb{X}} \subset \mathbb{X} \subset \overline{\mathbb{X}}$. Fig. 3 shows the approximation of the set delimited by the black boundary: red boxes constitute an inner subpaving, while the union of red and yellow boxes is an outer subpaving for the set.

3.2 Algorithms for set inversion

Determining the set $\mathbb{X}$ such as $\mathbf{f}(\mathbb{X}) = \mathbb{Y}$ when $\mathbb{Y}$ is known is a set inversion problem.

A box $[\mathbf{x}]$ of $\mathbb{IR}^n$ will be feasible if $[\mathbf{x}] \in \mathbb{X}$ and unfeasible if $[\mathbf{x}] \cap \mathbb{X} = \emptyset$, otherwise $[\mathbf{x}]$ is ambiguous. Using an inclusion function $[\mathbf{f}]$, we can identify feasibility of boxes:

- If $[\mathbf{f}]([\mathbf{x}]) \subset \mathbb{Y}$ then $[\mathbf{x}]$ is feasible
- If $[\mathbf{f}]([\mathbf{x}]) \cap \mathbb{Y} = \emptyset$ then $[\mathbf{x}]$ is unfeasible
- Else $[\mathbf{x}]$ is indeterminate, meaning it can be feasible, unfeasible or ambiguous.

The Set Inversion Via Interval Analysis [6] algorithm (SIVIA) works by testing feasibility of boxes, starting from an arbitrarily big prior searching box $[\mathbf{x}_0]$. If a box is feasible, it is added to the inner subpaving of solution $\underline{\mathbb{X}}$. If a box is unfeasible, it is discarded. Finally, if a box is indeterminate, it is bisected into two sub-boxes, which are enqueued in the list of boxes to examine $\mathfrak{L}$. Indeterminate boxes whose width is too small are added to the $\Delta\mathbb{X}$ subpaving of indeterminate boxes. The outer subpaving is then $\overline{\mathbb{X}} = \underline{\mathbb{X}} \cup \Delta\mathbb{X}$.

Algorithm 1 SIVIA(in: $[\mathbf{x}_0], \mathbb{Y}$; out: $\underline{\mathbb{X}}$, $\Delta\mathbb{X}$)
Original algorithm

```
1:  push([x_0], L)
2:  while L ≠ ∅ do
3:     [x] = pull(L)
4:     if [f]([x]) ⊂ Y then
5:        X = X ∪ [x]
6:     else if [f]([x]) ∩ Y = ∅ then
7:        discard [x]
8:     else if w([x]) < ε then
9:        ΔX = ΔX ∪ [x]
10:    else
11:       ([x_1], [x_2]) = bisect([x])
12:       push([x_1], L); push([x_2], L)
13:    end if
14: end while
```

The working list of boxes $\mathfrak{L}$ used in the algorithm can be implemented in various ways, each having its advantages and drawbacks :

- $\mathfrak{L}$ as a stack : the memory occupation of the algorithm is minimal and can be bounded as a function of n and ε. It is an in depth first implementation, requiring the algorithm to complete to get a usable result.
- $\mathfrak{L}$ as a queue : search over the state space is done in width first, allowing a similar paving width over the whole solution set. On the one hand, this enables the computation to be stopped at any time to get a result. This approach is thus compatible with real time applications. On the other hand, memory occupation is a lot larger than using a stack.

SIVIA is described in algorithm 1, where the *push* function adds a box to the list, and the *pull* function extracts the next box from the list.

As the dimension of the problem gets bigger, the number of needed bisections gets exponentially bigger and the computational burden becomes intractable. A solution to this problem is to add a *contraction* stage to the algorithm, which shrinks the boxes without loosing any solution (Alg. 2). Such a contractor can be built using a constraint propagation algorithm [7].

Algorithm 2 SIVIAP(in: $[\mathbf{x}_0]$,$\mathbb{Y}$; out: $\underline{\mathbb{X}}$, $\Delta\mathbb{X}$)
SIVIA with constraint propagation

push($[\mathbf{x_0}], \mathfrak{L}$)
while $\mathfrak{L} \neq \emptyset$ **do**
  $[\mathbf{x}] = \text{pull}(\mathfrak{L})$
  contract($[\mathbf{x}]$)
  if $[\mathbf{f}]([\mathbf{x}]) \subset \mathbb{Y}$ **then**
    $\underline{\mathbb{X}} = \underline{\mathbb{X}} \cup [\mathbf{x}]$
  else if $[\mathbf{f}]([\mathbf{x}]) \cap \mathbb{Y} = \emptyset$ **then**
    discard $[\mathbf{x}]$
  else if $w([\mathbf{x}]) < \varepsilon$ **then**
    $\Delta\mathbb{X} = \Delta\mathbb{X} \cup [\mathbf{x}]$
  else
    $([\mathbf{x_1}], [\mathbf{x_2}]) = \text{bisect}([\mathbf{x}])$
    push($[\mathbf{x_1}], \mathfrak{L}$); push($[\mathbf{x_2}], \mathfrak{L}$)
  end if
end while

When a contraction step is included in the algorithm, the generated subpavings are no longer regular, and size of boxes is unpredictable. The queue implementation of the working list of boxes may thus be replaced by an *sorted list* implementation, allowing to bisect the largest boxes first.

Adding robustness can be done by relaxing a given number q of constraints. The solver will then compute a sub-paving of the state space consistent with at least $m-q$ measurements (Alg. 3).

The component functions f_i of $\mathbf{f}$ are considered independently, with their inclusion functions $[f_i]$. If feasibility of $[\mathbf{x}]$ is achieved with at least $m-q$ components of $\mathbb{Y}$ via component inclusion functions $[f_i]$, $[\mathbf{x}]$ will be considered as feasible. Else, if infeasibility of $[\mathbf{x}]$ is concluded for more than q components, $[\mathbf{x}]$ will be unfeasible. Otherwise, $[\mathbf{x}]$ will be indeterminate.

Algorithm 3 RSIVIA(in: $[\mathbf{x}_0]$, $\mathbb{Y}$, q; out: $\underline{\mathbb{X}}$, $\Delta\mathbb{X}$)
Robustified SIVIA

push($[\mathbf{x_0}], \mathfrak{L}$)
while $\mathfrak{L} \neq \emptyset$ **do**
  $[\mathbf{x}] = \text{pull}(\mathfrak{L})$
  $incl := 0; sep := 0$
  for $i = 1 \ldots m$ **do**
    if $[f_i]([\mathbf{x}]) \subset [y_i]$ **then**
      $incl := incl + 1$
    else if $[f_i]([\mathbf{x}]) \cap [y_i] = \emptyset$ **then**
      $sep := sep + 1$
    end if
  end for
  if $incl \geq m - q$ **then**
    $\underline{\mathbb{X}} = \underline{\mathbb{X}} \cup [\mathbf{x}]$
  else if $sep > q$ **then**
    discard $[\mathbf{x}]$
  else if $w([\mathbf{x}]) < \varepsilon$ **then**
    $\Delta\mathbb{X} = \Delta\mathbb{X} \cup [\mathbf{x}]$
  else
    $([\mathbf{x_1}], [\mathbf{x_2}]) = \text{bisect}([\mathbf{x}])$
    push($[\mathbf{x_1}], \mathfrak{L}$); push($[\mathbf{x_2}], \mathfrak{L}$)
  end if
end while

4 SETTING BOUNDS ON MEASUREMENTS

A nice characteristic of bounded-error models is that a risk is taken when the measurement bounds are chosen. In an opposite manner, other methods (like Kalman filtering or weighted least squares for instance) often consists in propagating measurement errors using the two first statistical moments (mean and covariance). Then, the risk arises when trying to bound the obtained estimation error. An essential quality of a set-inversion solver is than since the computation and the inversion are guaranteed, it does not add any risk. This way, the integrity risk of the solution set not to include the ground truth is only linked to the risk taken when formulating initial assumptions on measurements (bounds parameters and maximum number of outliers).

In the following, we present a method to choose the bounds of the measurement errors given an integrity risk.

Each measurement can be considered as a Bernoulli trial "Is the true value inside the measurement bounds?". Knowing the probability density function f_{e_y} of the measurement error e_y and the error bounds $[a,b]$ such as a measurement y_{meas} is represented by the interval $[y_{meas}] = [y_{meas}+a, y_{meas}+b]$, we can compute the probability $p = P(y \in [y_{meas}])$ of the true

y being inside $[y_{meas}]$.

$$p = P(y \in [y_{meas}]) = \int_a^b f_{e_y}(\alpha) d\alpha$$

Let n_{ok} be the number of measurements consistent with the bounded error model (in other words there are $m - n_{ok}$ outliers). Under the assumption of independence, if m measurements are made with a probability p of the true value being inside each measurement interval, the probability of all the m measurements being consistent with the true value is

$$P(n_{ok} = m) = p^m$$

If a number q of faulty measurements is tolerated, we have to compute the probability of having at least $m-q$ good measurements. The probability of having exactly k good measurements out of m is given by binomial law:

$$P(n_{ok} = k) = \binom{m}{k} p^k (1-p)^{m-k} \qquad (2)$$

$$\text{where } \binom{m}{k} = \frac{m!}{k!(m-k)!}$$

Thus, by summing equation 2 over successive k values, the probability of having at least $m-q$ good measurements is

$$P(n_{ok} \geq m-q) = \sum_{k=m-q}^{m} \binom{m}{k} p^k (1-p)^{m-k} \qquad (3)$$

SIVIA algorithm (see section 3.2) computes a conservative approximation $\overline{\mathbb{X}}$ of the solution set $\mathbb{X}$. Moreover, if the hypotheses made on the measurements are verified, the solution set is consistent with the truth. This way,

$$n_{ok} \geq m-q \Rightarrow x \in \mathbb{X} \Rightarrow x \in \overline{\mathbb{X}}$$

which leads to

$$P(x \in \overline{\mathbb{X}}) \geq P(x \in \mathbb{X}) \geq P(n_{ok} \geq m-q)$$

Equivalently, the risk taken when formulating assumptions on measurements is an upper-bound of the risk than the solution set does not include the ground truth.

$$P(x \notin \overline{\mathbb{X}}) \leq P(x \notin \mathbb{X}) \leq 1 - P(n_{ok} \geq m-q)$$

One can try to tune the measurement error bounds to meet a global risk requirement. Equation 3 has to be inverted to compute the required probability

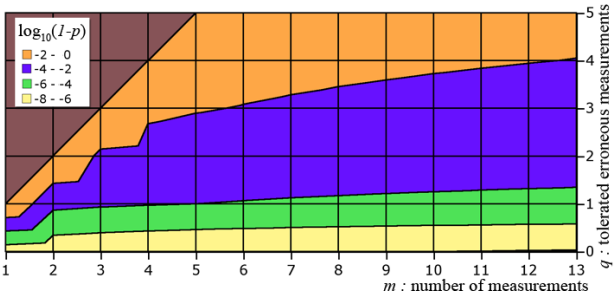


Figure 4: $\log_{10}$ of individual measurement risk $(1-p)$ to achieve a global risk of 10^{-7}

Table 1: Computation of $(1-p)$ to achieve a risk of 10^{-7} using m measurements and allowing q erroneous measurements

	$m=4$	$m=5$	$m=6$	$m=7$
$q=0$	$2.5 \cdot 10^{-8}$	$2 \cdot 10^{-8}$	$1.66 \cdot 10^{-8}$	$1.42 \cdot 10^{-8}$
$q=1$	$1.29 \cdot 10^{-4}$	$1.0 \cdot 10^{-4}$	$8.16 \cdot 10^{-5}$	$6.90 \cdot 10^{-5}$
$q=2$	$2.93 \cdot 10^{-3}$	$2.16 \cdot 10^{-3}$	$2.71 \cdot 10^{-3}$	$1.42 \cdot 10^{-3}$

p of each measurement, for a given number m of measurements and q tolerated outliers. Using a bisection method, we computed the required value of p to achieve a global risk of 10^{-7} for different (m,q) combinations (Fig. 4).

Table 1 contains the value of probabilities for GPS-like application.

Once the required probability p of each measurement interval to contain the actual value is computed, the measurement error bounds can be set to meet this requirement. In the case of a centered Gaussian measurement error $e_y \sim \mathcal{N}(0, \sigma_y)$, if Φ represents the cumulative distribution function of the standard normal distribution, the measurement interval should be set to

$$[y_{meas}] = [y_{meas} - K\sigma_y, y_{meas} + K\sigma_y]$$
$$\text{with } K = -\Phi^{-1}\left(\frac{1-p}{2}\right)$$

This way, the probability of a wrong measurement is evenly distributed on the two tails of the Gaussian error distribution (Fig. 5). Table 2 contains values of K to achieve a global risk of 10^{-7} for different (m,q) combinations, in the case of Gaussian measurement errors.

5 GPS INTEGRITY ZONE COMPUTATION

Using L1 code pseudo-range measurements and a bounded error model, we here compute a location zone with outliers. The risk of the actual user

Table 2: Computation of K to achieve a risk of 10^{-7} using m measurements and allowing q erroneous measurements

	$m=4$	$m=5$	$m=6$	$m=7$
$q=0$	5.57	5.61	5.64	5.67
$q=1$	3.83	3.89	3.94	3.98
$q=2$	2.98	3.07	3.14	3.19

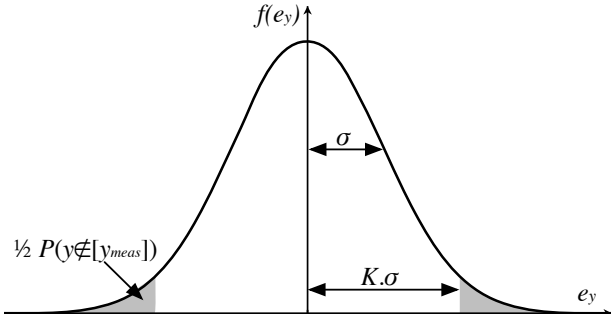


Figure 5: Risk when choosing a $K.\sigma$ bound associated to Gaussian noise

position not being inside the computed zone is an input of the method.

5.1 Set-theoretic GPS localization

GPS localization using pseudoranges is a four-dimensional problem: along with space coordinates (x,y,z) of the user, the user clock offset dtu has to be estimated. With ρ_i being the corrected pseudo-ranges, the GPS code observation model is :

$$\begin{pmatrix}\rho_1\\ \rho_2\\ \vdots\\ \rho_m\end{pmatrix} = \begin{pmatrix}\sqrt{(x-x_1^s)^2+(y-y_1^s)^2+(z-z_1^s)^2}+cdtu\\ \sqrt{(x-x_2^s)^2+(y-y_2^s)^2+(z-z_2^s)^2}+cdtu\\ \vdots\\ \sqrt{(x-x_m^s)^2+(y-y_m^s)^2+(z-z_m^s)^2}+cdtu\end{pmatrix}$$

Satellite positions (x_i^s,y_i^s,z_i^s) are known with uncertainty due to the inaccuracy of the broadcast ephemeris information. For each satellite, we consider a box $[\mathbf{x}_i^s] = ([x_i^s],[y_i^s],[z_i^s])$ whose bounds are chosen to contain the true satellite position at a given confidence level.

Measured pseudo-ranges are compensated from relativistic effects, ionosphere and troposphere propagation delays using EGNOS to get corrected pseudoranges ρ_i. These corrections are imprecise due to model and parameters errors. Moreover, the receiver makes also measurement errors. Therefore, we model the pseudo-range measurements as intervals $[\rho_i]$ whose bounds will be determined given an integrity risk.

The location zone computation consists in characterizing the set $\mathbb{X}$ of all locations compatible with the measurements and the satellite positions intervals:

$$\begin{aligned}\mathbb{X} = \Big\{&(x,y,z,cdtu)\in\mathbb{R}^4, \forall i = 1\ldots m,\\ &\exists\rho_i\in[\rho_i], \exists(x_i^s,y_i^s,z_i^s)\in[\mathbf{x}_i^s],\\ &\rho_i = \sqrt{(x-x_i^s)^2+(y-y_i^s)^2+(x-y_i^s)^2}+cdtu\Big\}\end{aligned}$$

The solution set is the set of locations for which a pseudorange and a satellite position can be found inside the measurement and satellite position intervals for every satellite.

To solve this set inversion problem, SIVIA is employed (see section 3.2). By recursively bisecting an arbitrarily big initial box, this algorithm returns a subpaving of the state-space (user position and clock offset) guaranteed to include the solution set — *i.e.,* an outer approximation of the solution set by a set of boxes. As long as the errors on measurements and satellite positions stay inside their bounds, the true receiver position is guaranteed to be inside the computed localization zone.

We use a real-time oriented implementation, based on an sorted list, that gives usable results even if computation time is bounded. To reduce the practical computational complexity, a contractor based on Waltz constraint propagation algorithm is used at each step to shrink the boxes. This allows to get a good characterization of the localization zone in less than one second.

To speed up computation, the prior position and clock offset box at time t_k is computed from the interval-hull $[\mathbb{X}^{k-1}]$ of the location zone at time t_{k-1}, taking into account the physical limits of the system. We assume that the user speed is no more than v_{max} and that the receiver clock drift is no more than d_{max}. This way, the prior box $[\mathbf{x}_0^k]$ at time t_k will be

$$[\mathbf{x}_0^k] = [\mathbb{X}^{k-1}] + \begin{pmatrix}[-v_{max},+v_{max}]\\ [-v_{max},+v_{max}]\\ [-v_{max},+v_{max}]\\ [-d_{max},+d_{max}]\end{pmatrix}\cdot(t_k-t_{k-1}) \quad (4)$$

5.2 Experimentation with real data

GPS localization by set inversion has been tested with data recorded using a *Septentrio PolaRx* receiver and the experimental vehicle *Strada* of Fig. 6. Ground truth was provided by a post-processed *Trimble 5700* receiver with a local base. Results presented in the following are obtained from to a

Figure 6: Our experimental vehicles: CARMEN and STRADA.

170 second long sequence of data, recorded during a 1800 m trip near the lab.

Error parameters have been obtained using the EGNOS satellite based augmentation system decoded by the PolaRx receiver. This allows computing overbounding Gaussian distributions of the pseudorange measurement errors.

We report here location zones with an integrity risk of $0.5 \cdot 10^{-7}$ per hour. Based on ionospheric corrections, 360 s seems to be a reasonable delay to ensure independence of measurements. Thus, we consider 10 independent samples per hour [8]. It leads to a probability of inconsistent location zone of $5 \cdot 10^{9}$ per sample. Following the method described in section 4, if six pseudorange measurements are available at each time, the probability of an out-of-bounds measurement should then be at most $8.33 \cdot 10^{-10}$, which translates into $\pm 6.14\sigma$ bounds, assuming a Gaussian distribution of pseudo-range error.

The location zone is computed in a local tangent plane, with East-North-Up coordinates. Four to six satellites were in view during the trial. Figure 7 shows the evolution of location zone bounds during the experiment. Figure 8 shows the bounding boxes of the computed location zones and the ground truth trajectory.

When six measurements are available, the location zone radius stays within 15 to 20 meters in the horizontal plane. This radius is about 35 to 40 meters on the vertical axis; vertical dilution of precision is usually higher than horizontal dilution of precision due to the geometrical configuration of the GPS localization problem.

The center of gravity of the computed location subpaving is used as a punctual estimate of position (*red line in Fig. 7*). The RMS error of this 3-dimensional estimation of position is 0.84 m for the experiment, and the absolute error stays within 3 meters.

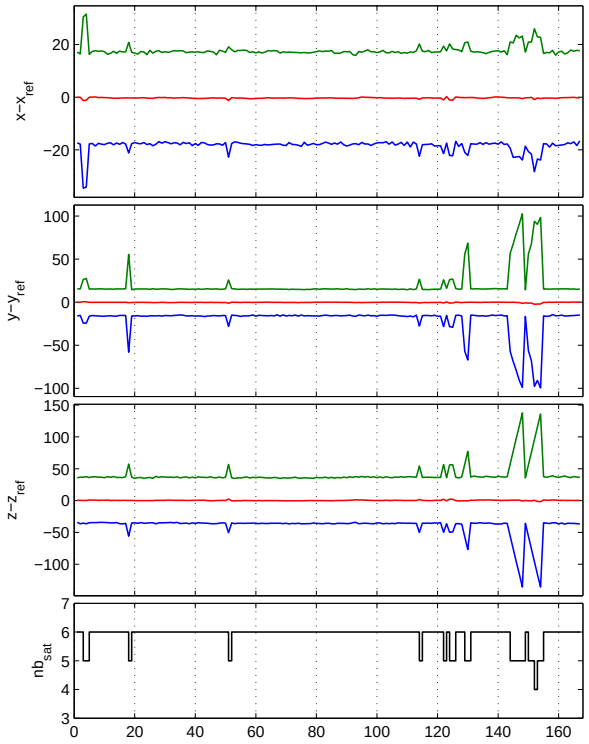


Figure 7: Upper *(green)* and lower *(blue)* bounds of the location zone along the East, North and Up axis, with respect to ground truth. *Red line is the center of gravity of the computed location zone.*

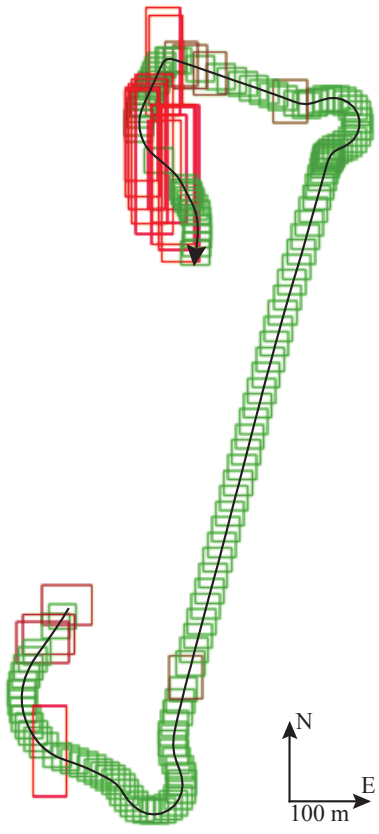
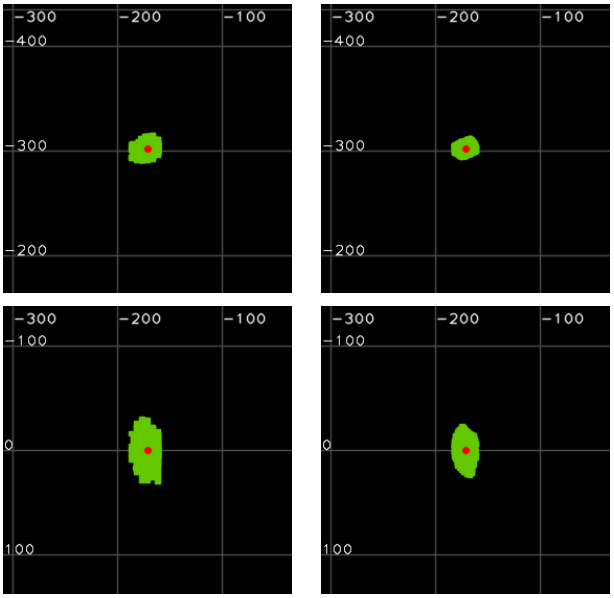


Figure 8: Trajectory. *Boxes are bounding boxes of solution sets. Reference trajectory is in black.*



(a) 50 ms computing time (b) 1 s computing time

Figure 9: X,Y projection *(top)* and X,Z projection *(bottom)* of computed location zone. *Actual location in red.*

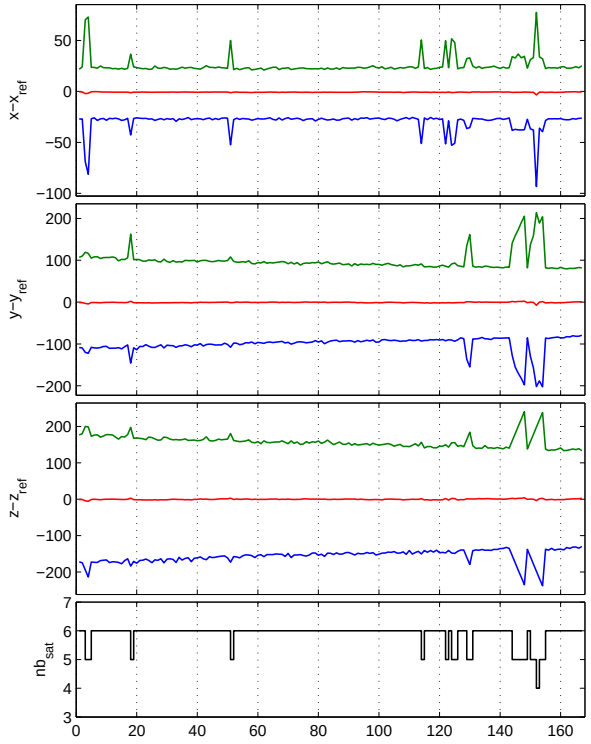


Figure 10: Upper *(green)* and lower *(blue)* bounds of the location zone along the East, North and Up axis, with respect to ground truth. One erroneous measurement is tolerated.

Variations in the computed location area width are due to the loss of satellites in view. Each measurement acts as a constraint on the location zone, consequently, when fewer measurement are available, the location zone is less constrained.

Algorithm is implemented in C++ and works in a time-bounded fashion, allowing devoted computed time to be specified. The computation is parallelized so as to take advantage of multicore and multiprocessor computers. The computed subpaving approximation of solution gets more precise and less pessimistic as more time is alloted to computation (Fig 9). One has to notice that more time will be needed to precisely characterize a wide location zone compared to a narrow zone, since more boxes will be needed. Zones of less than 100 meters wide are well approximated in a few tenths of second.

5.3 Results with the robust set-inversion solver

If the computation takes into account the presence of at most one erroneous measurement at a time (like often done in any RAIM algorithm), a risk of $5 \cdot 10^9$ per sample can be achieved if the probability of a wrong measurement is $1.83 \cdot 10^{-5}$ per sample (with six satellites in view). The bounds will then be set to $\pm 4.29\sigma$, assuming a Gaussian measurement error. The variances of measurement errors are obtained using EGNOS augmentation system.

Computation of location zone under those hypotheses require a robust solver, which can be based on RSIVIA algorithm (see Alg. 3 in section 3.2). Precise determination of location zones with the robust solver takes more time than the non robust algorithm. This is in part due to the overhead of additional computations for relaxed intersection, but the main factor is that relaxed solution sets are bigger.

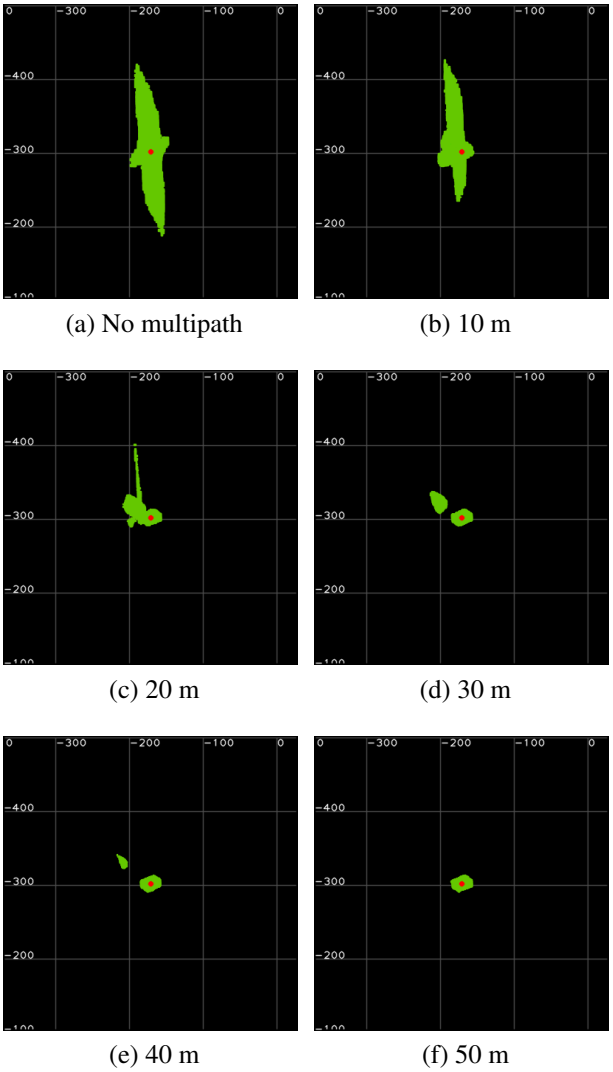
Location zones computed under the hypothesis of at most one outlier are larger that zones computed without taking into account the presence of an erroneous measurement. Given six satellites in view, these zones are equivalent to the union of the zones generated by all five-satellite combinations. Since five-satellite solutions are more likely to be ill-conditioned geometrically (with higher dilution of precision) than a six-satellite solution, the computation of a relaxed solution-set will generate larger location zones, despite the narrower measurement bounds.

One may notice that since one constraint is relaxed, at least five pseudo-ranges instead of four are needed to compute a location zone with the robust solver. The reason why the location zone does not become huge when only four measurements are available is the assumption of bounded vehicle speed and clock drift (Eq. 4). This assumption roughly constrains

the expansion of the location zone. This emphasises the need for data redundancy when using a robust method. Such a data redundancy can be obtained by using more than four GPS pseudorange measurements. In difficult environments like urban areas, redundancy is not always available from GPS alone, thus other sources like geographical information [9] can be employed.

Fig. 10 shows the bounds of the computed location zones. One can notice than bounds on the x-axis (East) remain the same order of magnitude than for the solution allowing no outlier. However, y and z axis (North and Up) suffer from the dilution of precision of 5-satellite solutions.

Robust computation of location zone allows to keep integrity in the presence of an outlier. Outliers can frequently appear in urban environments, due to unmodeled multipaths. When no outlier is present, computed zone can be very wide if there is not enough data redundancy to limit the effect of relaxing intersections of constraints. However, when outliers occur, location zone remains consistent with ground truth as long as the number of tolerated outliers is not exceeded, and moreover, the location zone tends to get narrower (Fig 11). If the outlier is incompatible with all other measurements, it is automatically discarded without any additional computation (Fig 11f). If needed, the identification of the erroneous measurement can be done by analysing compatibility of the location zone with each measurement.



(a) No multipath (b) 10 m

(c) 20 m (d) 30 m

(e) 40 m (f) 50 m

Figure 11: Computed location zone (X,Y projection) with a solver robust to one outlier. A multipath of growing intensity is simulated on the first pseudorange. *Ground truth in red*

6 CONCLUSION

In this paper, a new way to compute integrity zone has been proposed, using robust bounded error solvers. The integrity risk is expressed on the measurements, using a bounded error model, and the number of outliers tolerated without affecting integrity can be specified. A method to set bounds on measurement to achieve a specific integrity risk requirement has also been presented and developed in the case of Gaussian error models.

An experimental validation has been carried out to compute location zones with GPS pseudorange measurements and EGNOS. A parallelized and time bounded implementation in C++ has been done for a real-time location zone computation application. Robustness to outliers was also tested by altering data.

Future work will be focused on data fusion for multi-sensor localization, since the robust solver needs data redundancy to keep location zones narrow. Altitude information from altimeters or digital elevation models could be used to constrain solution. A kinematic model of vehicle will also be introduced, and proprioceptive sensors could help in localization, especially during GPS outages. We plan also to study difficult situations like urban canyons where several multipath can occur simultaneously.

ACKNOWLEDGMENTS

The dataset and the reference trajectory used in this work were provided by Clément Fouque (*Heudiasyc UMR 6599, Université de Technologie de Compiègne*).

REFERENCES

[1] Bancroft, S. (1985) An algebraic solution of the GPS equations. *IEEE Trans. Aerosp. and Elec. Systems*, **21**(7)

[2] Yang M. and K-H. Chen (2001). Performance Assessment of a Noniterative Algorithm for Global Positioning System (GPS) Absolute Positioning. *Proc. Natl. Sci. Counc. ROC (A)*, **25**(2), 102–106.

[3] Schroth, G., A. Ene, J. Blanch, T. Walter, and P. Enge (2008). Failure Detection and Exclusion via Range Consensus. *European Navigation Conference*.

[4] Jaulin, L. (2009), Robust set-membership state estimation; application to underwater robotics. *Automatica*, **45**(1), 202–206.

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

[6] Jaulin, L. and E. Walter (1993). Set inversion via interval analysis for nonlinear bounded-error estimation. *Automatica*, **29**(4), 1053–1064.

[7] Jaulin, L., M. Kieffer, O. Didrit, and E. Walter (2001). *Applied Interval Analysis*. Springer-Verlag, London.

[8] Roturier, B., E. Chatre and J. Ventura-Traveset (2001). The SBAS Integrity Concept Standardised by ICAO-Application to EGNOS.

[9] Fouque, C. and Ph. Bonnifait (2008). Tightly coupled GIS data in GNSS fix computation with integrity test. *International Journal of Intelligent Information and Database Systems*, **2**(2), 167–186.