Projet de fin d'étude
ENSI 2014

Set-Membership Approach to Map-Based Localization

Benoît DESROCHERS

Supervisors :
Simon Lacroix, Luc Jaulin

August 22, 2014

Abstract

Absolute position estimation is a key function of autonomous systems and is often required at the mission level. One way the get such information is to use an initial model of the environment provided for instance, by a geographic database or built by an other robot. This study aims at providing an algorithm which matches the output of a sensor with an initial model to estimate the pose of the robot in a guaranteed way, using the intervals analysis framework. By construction, initial models do not perfectly fit the reality and the acquire data set can contains an unknown, and potentially large, percentage of outliers. When the environment is described by a surface, the set membership estimator *GOMNE* can be used to concurrently estimate the number of outliers and the localisation parameters. However, in the general case, with full 3D representation and partially mapped objects, it can not be used. To cope this issue, a new algorithm called Outer-GOMNE is proposed. By combining intervals methods with local estimation algorithms, it have been applied to the localization problem. After a simulated test case with a 2D map, an experimental validation using real laser data and different 3D models is reported to illustrate the performance of the method. Results are compared with ground truth provided by a differential GPS. Outer-GOMNE is able to robustly enclose the ground truth in a sub-paving (union of non overlapping boxes)

Résumé

La localisation d'un robot mobile dans son environnement est une fonction essentielle des systèmes autonomes. C'est l'une des briques de base permettant d'effectuer des actions de plus haut niveau, tel que la cartographie ou la planification de trajectoire. L'objectif de ce stage était d'élaborer une méthode de localisation absolue, c'est-à-dire sans connaissance a priori de la position du robot, à partir d'un modèle numérique de terrain initial et des données extéroceptives fournies par les capteurs du robot. Cependant, par construction, ces modèles contiennent de nombreuses erreurs qui amènent à estimer, conjointement à la position du robot, le nombre de mesures aberrantes présentes dans les données. Dès lors que le modèle de terrain n'est pas surfacique, les techniques d'estimations du nombre de mesures aberrantes ne fonctionnent plus. Outer-GOMNE a donc été développé pour palier à ce manque. En combinant les méthodes ensemblistes avec des algorithmes d'estimation locaux (optimisation de fonction, Monte-Carlo, ...) il s'est avéré efficace et robuste pour ce type de problème. Une expérience avec des données réelles, et différents modèles de terrain, a été conduite pour évaluer et valider l'algorithme.

Keywords

Absolute localization, Intervals analysis, Digital Elevation Map, autonomous vehicle

Thanks:

We would first like to thank our supervisor Mr Simon Lacroix who has enabled us to realize this internship, and who helped us write this report. We would also like to thank all the team of Robotics and Interactions at LAAS-CNRS, taking their turn to give us explanation and to integrate us to their projects, especially Mr Koch and Mr Maligo. At the end, we would like to thank M. Jaulin for his help during the internship.

Contents

1 Introduction

Metric maps of buildings, urban and natural environments are becoming widely available, and make map-based localization a key function that perfectly complements dead-reckoning and SLAM approaches to solve the overall robot localisation problem. By providing an "absolute" position estimate (actually relative to the map frame), map-based localization can indeed be exploited either as a solution to the kidnapped robot problem, to palliate the inherent drift of dead-reckoning, in SLAM approaches, or in multi-robot contexts where one robot localizes itself within a map built by an other.

Related work The literature provides various approaches to this problem, which can be characterized by the way the acquired data are matched to the a priori map (data association), and the algorithms that compute the position from the matches (estimation). Skyline localisation approaches match the skyline detected on images with a predicted skyline derived from a 3D model in urban environments [23], or from a digital elevation map (DEM) in planetary environments [27, 10]. In this latter context, the absence of absolute localization means and the availability of sub-meter accuracy orthoimages and DEMs has motivated the development of various approaches, that register orthoimages built from camera and LIDAR images [25], correlate locally built DEMs with the global one [13], or matches features (peaks) extracted from LIDAR scans and the global DEM [7]. An other relevant approach is [30], where spin-images are associated between aerial and ground data. Note that the "pinpoint landing" problem, which consists in localizing a lander within existing orbiter DEMs and orthoimages, also call for similar map-based techniques [29].

In urban environments, the building of maps dedicated to vehicles localization can yield excellent precision: for instance [22] propose algorithms to build an orthoimage of ground reflectivity, which is later used to augment the precision of GPS/IMU/odometry solution. Particle filters are often used as the estimation engine for map-based localisation [11]. One of their advantage is that they do not rely on explicit data association: only a likelihood measure is required between the acquired data and the initial map for a given position hypothesis, and the estimation scheme progressively discards wrong hypotheses. Yes this approach need to be complemented with additional motion estimation techniques for the prediction step, and depending on the environment, can require lengthy displacements to provide a precise localisation estimate when starting without any prior information on this estimate.

Motivations and approach In most of the outdoor and indoor environments, map-based localisation approaches are challenged by the presence of dynamic elements, which generate numerous outliers, and hence considerably hinder the data association process or the likelihood measure of a position hypothesis. The environment dynamics have a variety of frequencies, from fast moving people or vehicles, to slowly seasonal changes or infrastructure evolution, via changes in the illumination conditions. Our aim is to define an approach to estimate the robot position that is *robust* with respect to these changes. For that purpose, we rely on the following two choices:

- Exploit *geometric information*: the geometry of the environment is an intrinsic property that does not depend on illumination conditions, neither on the sensors that capture it

and their viewpoints. Furthermore, most of initial maps are geometric, and in particular Digital Elevation Maps are easily built with either aerial or ground means;

- Exploit interval analysis, that cast the localisation problem into a set inversion problem. The literature has shown that interval analysis defines *robust* estimation solutions, while ensuring *integrity*: when a estimation is provided, the actual position is guaranteed to lie within the associated bounds.

Yet, the potential large number of outliers, the facts that initial maps are always partial and that the initial position can very poorly known challenge the existing set membership approaches. The main contribution of this work is the proposal of an interval analysis algorithm that can cope with these problem characteristics.

Outline The next section briefly recall the basics of interval artihmetics and set inversion, so as to familiarize the reader with this formalism. Section 3 depicts the GOMNE algorithm, an essential contribution of the interval analysis literature to achieve robust set inversion in the presence of an unknown proportion of outliers. Our extension to this algorithm is presented in section 4, with an extensive analysis of its characteristics in simulation. An extension to the estimate of the whole trajectory is presented in section 5, and finally section 6 presents its application to the robot localization problem using range data and initial Digital Elevation Maps. A discussion concludes the report.

2 Background On Interval Analysis For Estimation

This section summarizes the basics of interval analysis, for a more detailed introduction see [16].

2.1 Interval Arithmetics

An *interval*, denoted by $[x]$ is a closed and connected subset of $\mathbb{R}$ defined by a lower and upper bound such as:

$$[x] = \left[x^-, x^+\right] = \left\{x \in \mathbb{R}, x^- \leq x \leq x^+\right\} \tag{1}$$

The width w of an interval is defined by $w\left([x]\right) = x^+ - x^-$. A *box* $[\mathbf{x}]$ of $\mathbb{R}^n$ is a Cartesian product of n intervals. The set of all boxes of $\mathbb{R}^n$ is denoted $\mathbb{IR}^n$. Basic operations on real numbers or vectors can be extended to intervals in a natural way. For two intervals $[x]$ and $[y]$ of $\mathbb{R}^n$ and an operator $\diamond \in \{\cap, \cup, +, -, *, /\}$, we define $[x] \diamond [y]$ as the smallest interval containing all feasible values for $x \diamond y$ when $x \in [x]$ and $y \in [y]$:

$$[x] \diamond [y] = [\{x \diamond y | x \in [x], y \in [y]\}] \tag{2}$$

where [.] denotes the convex hull. For instance:

$$\begin{aligned} [-1, 5] \cap [-3, 2] &= [-1, 2] \\ [-3, 3] \cup [4, 10] &= [-3, 10] \\ [-3, 3] - [4, 10] &= [-13, 1] \end{aligned} \tag{3}$$

Let $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$. An *inclusion function* denoted $[\mathbf{f}]$ of f is defined by:

$$\forall [\mathbf{x}] \in \mathbb{IR}^n, [\mathbf{f}]([\mathbf{x}]) \triangleq [\{\mathbf{f}(\mathbf{x}), \mathbf{x} \in [\mathbf{x}]\}] \tag{4}$$

An inclusion function $[\mathbf{f}]$ is *convergent* if, for any sequence of boxes $[\mathbf{x}](k)$

$$\lim_{k\to\infty} w([\mathbf{x}](k)) = 0 \Rightarrow \lim_{k\to\infty} w([\mathbf{f}]([\mathbf{x}](k))) \tag{5}$$

For any function obtained by the composition of elementary operators such as +, -, *, /, cos, sin, exp, ..., a natural inclusion function can be built by replacing these operators by their interval counterpart. For instance, let $f(x_1, x_2) = x_1 * \cos(x_2)$. The natural inclusion function $[\mathbf{f}]$ is given by:

$$[\mathbf{f}]([x_1], [x_2]) = [x_1] * \cos([x_2]) \tag{6}$$

Its evaluation is done by applying interval arithmetic:

$$\begin{array}{rcl} [f]([-2,1],[\frac{\pi}{6},\frac{\pi}{3}]) & = & [-2,1] * \cos([\frac{\pi}{6},\frac{\pi}{3}]) \\ & = & [-2,1] * [\frac{1}{2},\frac{\sqrt{3}}{2}] \\ & = & [-\sqrt{3},\frac{\sqrt{3}}{2}] \end{array} \tag{7}$$

2.2 Constraint Propagation

A *contractor* $\mathcal{C}$ is an operator from $\mathbb{IR}^n$ to $\mathbb{IR}^n$ such as:

$$\begin{array}{ll} \mathcal{C}([\mathbf{x}]) \subset [\mathbf{x}] & (\textit{contractance}) \\ [\mathbf{x}] \subseteq [\mathbf{y}] \Rightarrow \mathcal{C}([\mathbf{x}]) \subseteq \mathcal{C}([\mathbf{y}]) & (\textit{monotonicity}) \end{array} \tag{8}$$

A set $\mathbb{X}$ is *consistent* with the contractor $\mathcal{C}$ (denoted as $\mathbb{X} \sim \mathcal{C}$) if for all $[\mathbf{x}]$, we have

$$\mathcal{C}([\mathbf{x}]) \cap \mathbb{X} = [\mathbf{x}] \cap \mathbb{X} \tag{9}$$

Let $\mathbb{X}$ be a set defined with a constraint $\mathscr{C}$ such as an equality or an inequality. The contractor $\mathcal{C}_{out}$ associated with $\mathscr{C}$ makes it possible to remove part of a arbitrary initial box $[\mathbf{x}]$ which are not consistent with $\mathscr{C}$ without removing any feasible values. This contractor is called the outer contractor. If $\mathbb{X}$ has a non empty volume (there exist points in $\mathbb{X}$ which do not belong to the border of $\mathbb{X}$), based on the *De Morgan* rules, it is also possible to define the inner contraction which removes parts of $[\mathbf{x}]$ which belong to the $\mathbb{X}$ set. For instance, a curve such as a line has an empty volume because it is only described by its border ($\mathbb{X} = \delta\mathbb{X}$) and no inner contractor can be defined. On the contrary, it can be defined for a disk or a ring. These two example are depicted on figure 1. When available, these two contractors have complementary actions and from the pair $\{\mathcal{C}_{in}, \mathcal{C}_{out}\}$, a *separator* [20] can be built. In this case the separator $\mathcal{S}$ associated with the set $\mathbb{X}$ is an application such as:

$$\begin{array}{rccl} \mathcal{S}: & \mathbb{IR}^n & \longrightarrow & \mathbb{IR}^n \times \mathbb{IR}^n \\ & [\mathbf{x}] & \longmapsto & \{[\mathbf{x_{in}}], [\mathbf{x_{out}}]\} = \{\mathcal{C}_{in}([\mathbf{x}]), \mathcal{C}_{out}([\mathbf{x}])\} \end{array}$$

We define the remainder $\delta\mathcal{S}$ of a separator S as $\delta\mathcal{S} = \mathcal{C}_{in}([\mathbf{x}]) \cap \mathcal{C}_{out}([\mathbf{x}])$. $\delta\mathcal{S}$ is a contractor on the border of the set (see figure 2).

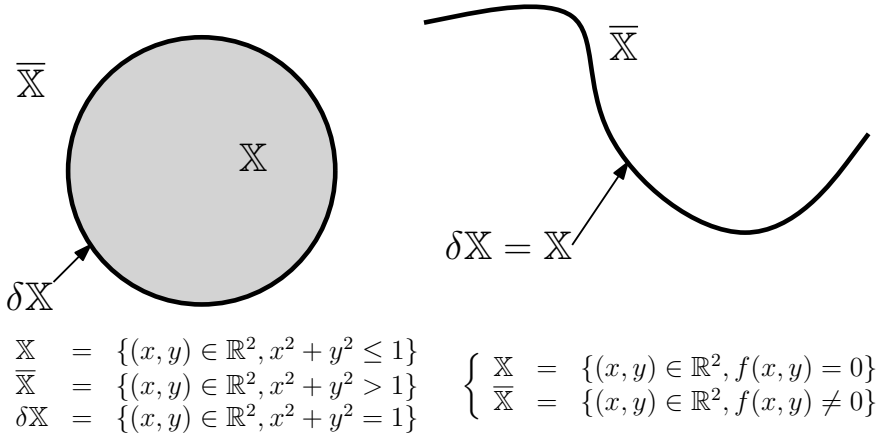


Figure 1: The disk, described by an equality admits an inner contractor. On the contrary, only an outer contractor can be build for the curve.

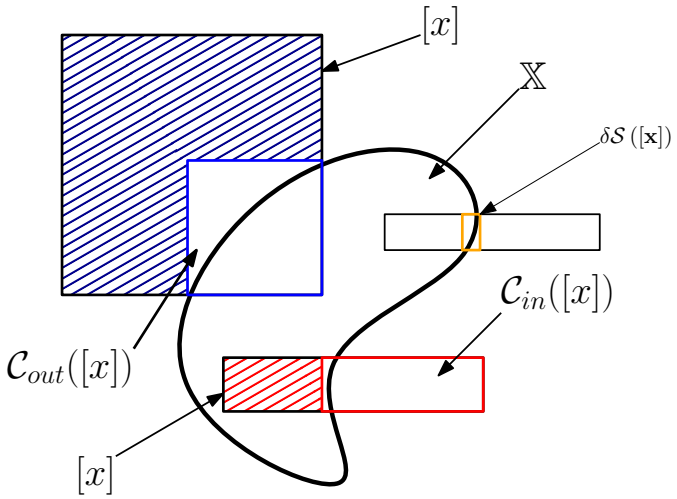


Figure 2: Illustration of a separator on three different initial intervals. The outer contractor removed the blue dashed area and the red dashed area was removed by the inner contractor. Their combined action leads to the yellow box.

2.3 Set Inversion

Let $\mathbf{f}$ be a function from $\mathbb{R}^n$ to $\mathbb{R}^m$ (possibly non-linear) and let $\mathbb{Y}$ be a subset of $\mathbb{R}^m$ such as for instance a sub-paving (finite union of non-overlapping boxes). The Set Inversion is the characterization of :

$$\mathbb{X} = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{f}(\mathbf{x}) \in \mathbb{Y}\} = \mathbf{f}^{-1}(\mathbb{Y}) \tag{10}$$

For any $\mathbb{Y} \subset \mathbb{R}^m$ and for any function $\mathbf{f}$ admitting a convergent inclusion function $[\mathbf{f}](.)$, two regular sub-pavings $\mathbb{X}^-$ and $\mathbb{X}^+$ can be obtained with an arbitrary precision using the algorithm SIVIA (Set Inversion Via Interval Analysis) [18] such that :

$$\mathbb{X}^- \subset \mathbb{X} \subset \mathbb{X}^+ \tag{11}$$

SIVIA is a *branch-and-bound* algorithm with several implementations. It takes an initial box $[\mathbf{x}_0]$ large enough to contain the solution, and applies on it contractors (or separators) to remove parts which can be classified into $\mathbb{X}$ or $\overline{\mathbb{X}}$. If the width of the resulting box is larger than a threshold ϵ the box is bisected (cut into 2 sub-boxes) and the algorithm is run on each new box. The complexity of the algorithm is exponential with respect to the dimension n of the initial box, and terminate after less than $(\frac{w([\mathbf{x}_0])}{\epsilon} + 1)^n$ iterations. A recursive version of the algorithm is given in algorithm 1 using separators and, another using a list and only an outer contractors is given in algorithm 2.

Algorithm 1: SIVIAS(in: $[\mathbf{x}], \mathcal{S}, \epsilon$, InOut: $\mathbb{X}^-, \mathbb{X}^+$)

```
{[x_in], [x_out]} = S([x]);
Store [x_in]\[x] into X⁻ and also into X⁺ ;
[x] = [x_in] ∩ [x_out];
max ← a_1;
if w([x])) ≤ ε) then
    store [x] into X⁺
else
    bisect [x] into [x_1] and [x_2];
    SIVIAS([x_1], S, ε, X⁻, X⁺);
    SIVIAS([x_2], S, ε, X⁻, X⁺);
```

Algorithm 2: SIVIA(in: $[\mathbf{x_0}], \mathcal{C}_{out}, \epsilon$; Out:$\mathbb{X}^+$)

```
L ← [x_0];
while L ≠ ∅ do
    Pull [x] from L;
    [x] = C_out([x]);
    if w([x])) ≤ ε) then
        store [x] into X⁺
    else
        bisect [x] and push into L the two resulting boxes
```

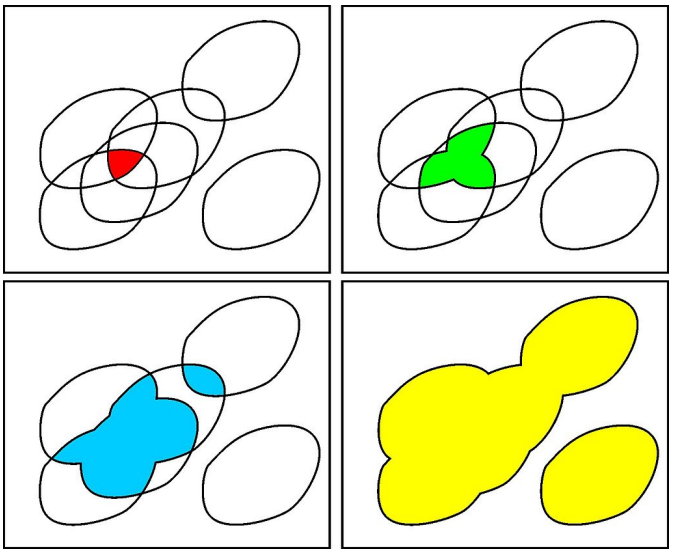


Figure 3: q-relaxed intersection of 6 sets for q=2 (red), q=3 (green), q= 4 (blue) , q= 5 (yellow)

2.4 Relaxed Intersection

Parameters estimation using interval analysis is more efficient if the number of sets involved is greater than the number of unknowns. However the presence of outliers will yield an empty solution set. To solve this issue, the relaxed-intersection [15] has been introduced.

Consider N sets $\mathbb{X}_1, \ldots, \mathbb{X}_n$ of $\mathbb{IR}^n$. 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 3 illustrates this notion for N = 6 and q = 2, 3, 4.

Since the relaxed-intersection can be written as a combination of unions and intersections, it is monotonic with respect to $\mathbb{X}_i$ and q:

$$\begin{array}{rcl} q_1, q_2 \in \mathbb{N}^*, q_1 \leq q_2 & \Rightarrow & \bigcap^{\{q_1\}} \mathbb{X}_i \subseteq \bigcap^{\{q_2\}} \mathbb{X}_i \\ (\mathbb{X}_1 \subset \mathbb{Y}_1, \ldots, \mathbb{X}_N \subset \mathbb{Y}_N) & \Rightarrow & \bigcap^{\{q\}} \mathbb{X}_i \subset \bigcap^{\{q\}} \mathbb{Y}_i \end{array} \tag{12}$$

Computing the exact q-intersection of p n-dimensional boxes has a complexity in $\mathcal{O}(p^n)$. However, using the Marzullo's algorithm ([9]), an approximation can be computed in $\mathcal{O}(np\log(p))$ by calculating the intersection along each dimension independently. A study of the relaxed-intersection can be found in [6] including a better approximation in $\mathcal{O}(np^2)$.

From a set of N contractors $\mathcal{C}^1, \ldots, \mathcal{C}^N$ associated with sets $\mathbb{X}_1, \ldots, X_N$, the contractor $\mathcal{C}_q$ is defined by algorithm 3.

Algorithm 3: C_q(in: $q, \mathcal{C}^1, \ldots, \mathcal{C}^N$, inout: $[\mathbf{x}]$)

1 **for** *k from 1 to N* **do**
2 $\quad [\mathbf{x}_k] = \mathcal{C}^k([\mathbf{x}])$;
3 $[\mathbf{x}] = [\mathbf{x}] \cap \bigcap_{i \in \{0,\ldots,N\}}^{\{q\}} [\mathbf{x}_i]$;

Knowing q, the relaxed-intersection is mainly used inside the set inversion algorithm instead of the classical intersection to take into account at most q outliers.

3 Guaranteed Outlier Minimal Number Estimator

In actual cases, the percentage of outliers is not known and needs to be estimated concurrently with the parameters. The Guaranteed Outlier Minimal Number Estimator (*GOMNE*) algorithm, introduced in [19] and [17], has been designed for this purpose.

3.1 GOMNE Algorithm

Let's take N sets $(\mathbb{X}_1, \ldots, \mathbb{X}_N)$ defined by constraints such as, for instance, $f_i(\mathbf{x}) \in [\mathbf{y_i}]$. Let j be a cost function which returns the number of measurements inconsistent with $\mathbf{x}$.

$$\begin{array}{rccl} j: & \mathbb{R}^n & \longrightarrow & [\![1, N]\!] \\ & \mathbf{x} & \longmapsto & card\{i | f_i(\mathbf{x}) \notin [\mathbf{y}_i]\} \end{array} \tag{13}$$

The *GOMNE* aims at characterizing the set $\mathbb{S}^*$ of feasible parameters which minimizes the number of outliers :

$$\mathbb{S}^* = \arg\min_{\mathbf{x} \in \mathbb{R}^n} j(\mathbf{x}) \tag{14}$$

Using the relaxed intersection, the set of all parameters that are consistent with at least $N - q$ measurements can then be seen as a set inversion :

$$\mathbb{S}_q = \{\mathbf{x} \in \mathbb{R}^n | j(\mathbf{x}) \leq q\} = j^{-1}([0, q]) \tag{15}$$

GOMNE runs a finite sequence of set inversions with different values of q in order to enclose $\mathbb{S}_{q^*}$ between two sub-pavings $\mathbb{S}^-_{q^*}$ and $\mathbb{S}^+_{q^*}$ in a guaranteed way. With an initial box $[\mathbf{x}_0]$, it starts with $q = 0$ and calculates $\mathbb{S}^-_q$ and $\mathbb{S}^+_q$ by running SIVIA($[\mathbf{x}_0]$,$\mathcal{C}_q$,ϵ). If the outer sub-paving $\mathbb{S}^+_q$ is empty, there does not exist any solution consistent with at least $N - q$ data. The number of outliers q is incremented and SIVIA is run again. If $\mathbb{S}^+_q$ is not empty and $\mathbb{S}^-_q$ is, a smaller value of ϵ is used to find a non empty inner set or an empty outer set. In the first case, the algorithm ends and $q^* = q$, otherwise SIVIA is run again with a greater value of q. The full GOMNE algorithm is given in listing 4. A minimal value ϵ_0 of ϵ has been introduced in order to avoid infinite loop when the interior of $\mathbb{S}^*$ is empty which could happen when, for instance, $\mathbb{S}^*$ is a singleton.

Algorithm 4: GOMNE(in: $[\mathbf{x}_0]$, ϵ_0, ϵ_{min}, out: $q^*, \mathbb{S}^-_{q^*}, \mathbb{S}^+_{q^*}$)

```
q = −1 ;
repeat
    ϵ = ϵ0; q = q + 1;
    repeat
        [S_q^-, S_q^+] = SIVIA([x0],q,ϵ);
        ϵ = ϵ/2;
    until (S_q^- ≠ ∅ or S_q^+ = ∅ or ϵ < ϵ_min) ;
until S_q^- ≠ ∅ ;
```

3.2 Illustration : Simple Map-Based Localization

A simple localization problem within an initial 2D map illustrates how GOMNE can be used and its limitations. In order to build an inner and outer contractor, the map must divide the space

into two complementary sets, with non-zero volume: $\mathbb{M}$ (gray in figure 4a) and $\overline{\mathbb{M}}$. Measurements will belong to the border of $\mathbb{M}$. In our case, the map is composed of a non-convex polygon with a hole, which for instance represents a room. The robot $\mathcal{R}$ is equipped with a laser sensor which returns the distance $d_i \in [d_i^-, d_i^+]$ between $\mathcal{R}$ and the nearest obstacle in a given direction ($\alpha_i \in [\alpha_i - \epsilon_\alpha, \alpha_i + \epsilon_\alpha]$). We also denote by $\mathbf{a}_i$ (resp. $\mathbf{b}_i$) the nearest (resp. furthest) impact point associated with the distance d_i^- (resp. d_i^+). The parametrization is shown in figure 4b. The goal of this problem is to estimate the position $\mathbf{x} = (p_x, p_y)$ of the robot on the map. Its heading θ is assumed to be known and is not estimated in this example.

Let $f(\mathbf{x}, \alpha, d)$ be the function which translates the point $\mathbf{x}$ by a distance d in the direction given by $\alpha + \theta$. Let $f_{i^-}(\mathbf{x}) = f\left(\mathbf{x}, \alpha_i, d_i^-\right)$ (resp. $f_{i^+}(\mathbf{x}) = f\left(\mathbf{x}, \alpha_i, d_i^+\right)$) be the function which transforms the $\mathbf{x}$ into $\mathbf{a}_i$ (resp. $\mathbf{b}_i$). The set $\mathbb{X}_i$ of feasible configurations consistent with the i^{th} measurement is the set of all $\mathbf{x}$ such that the point $\mathbf{a}_i$ belongs to $\mathbb{M}$ and $\mathbf{b}_i$ belongs to $\overline{\mathbb{M}}$ given by:

$$\begin{array}{rcl} \mathbb{X}_i & = & \{\mathbf{x} \in \mathbb{R}^n | f_{i^-}(\mathbf{x}) \in \mathbb{M} \text{ and } f_{i^+}(\mathbf{x}) \notin \mathbb{M}\} \\ ie. \quad \mathbb{X}_i & = & f_{i^-}^{-1}(\mathbb{M}) \cap f_{i^+}^{-1}\left(\overline{\mathbb{M}}\right) \end{array} \tag{16}$$

The complementary set $\overline{\mathbb{X}_i}$ is defined by :

$$\overline{\mathbb{X}_i} = f_{i^-}^{-1}\left(\overline{\mathbb{M}}\right) \cup f_{i^+}^{-1}(\mathbb{M}) \tag{17}$$

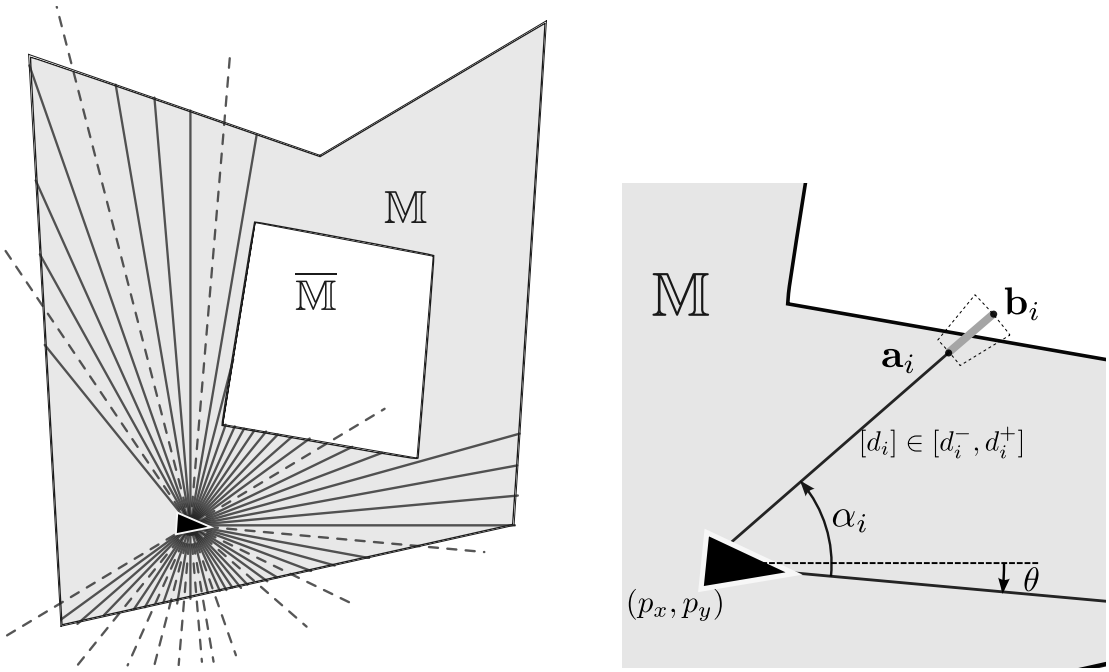
With the contractor arithmetics it is possible to build the separator $\mathcal{S} = \{\mathcal{C}_{in}, \mathcal{C}_{out}\}$ such as $\mathcal{C}_{in} \sim \overline{\mathbb{X}_i}$ and $\mathcal{C}_{out} \sim \mathbb{X}_i$. The GOMNE algorithm is then applied to find the solution set $\mathbb{X}^*$ which minimizes the number of outliers:

$$\mathbb{X}^* = \bigcap_i^{\{q^*\}} \mathbb{X}_i \tag{18}$$

Results of the set inversion are shown in figure 5 with two values of bounds for α_i. In the top right sub-figure, the value of α_i is assumed to be precisely known ($\epsilon_\alpha = 1e^{-4}$ rad.) and the resulting solution set has an inner part. On the contrary, in the bottom right sub-figure, the interval for α is larger ($\epsilon_\alpha = 1e^{-2}$ rad.) and the inner part disappears. This situation could occur frequently if a sonar, which has a larger emission cone than a laser, is for instance used. In this case, the GOMNE algorithm fails to find a non empty $\mathbb{X}^-$ even if a smaller value of ϵ is used. Maybe, the condition for $\mathbf{x}$ to belong to $\mathbb{X}_i$ is too restrictive because the equation (16) must be satisfied for all α_i in $[\alpha_i]$. For that purpose the projection algorithm [8] should be used to find the set $\mathbb{X}_i^{\cap[\alpha_i]}$ defined by:

$$\mathbb{X}_i^{\cap[\alpha_i]} = \left\{\mathbf{x} \in \mathbb{R}^n | \exists \alpha_i \in [\alpha_i], f_{i^-}^{-1}(\mathbb{M}) \cap f_{i^+}^{-1}\left(\overline{\mathbb{M}}\right)\right\} \tag{19}$$

But this requires additional bisections on α_i and hence increases the complexity of the algorithm. However, depending on the problem, it may be useless to precisely characterize the inner set. For instance, with the localization problem, in order to ensure a guarantee on the solution, only $\mathbb{X}^+$ is taken into account. Finally, the inner contractor may not exist and the GOMNE algorithm can not be used. This is main reason why a new method is introduced.



(a) Initial situation (b) Modeling and parameters

Figure 4: Initial situation, the laser sensor returns 56 measurements with 15 outliers (dashed lines)

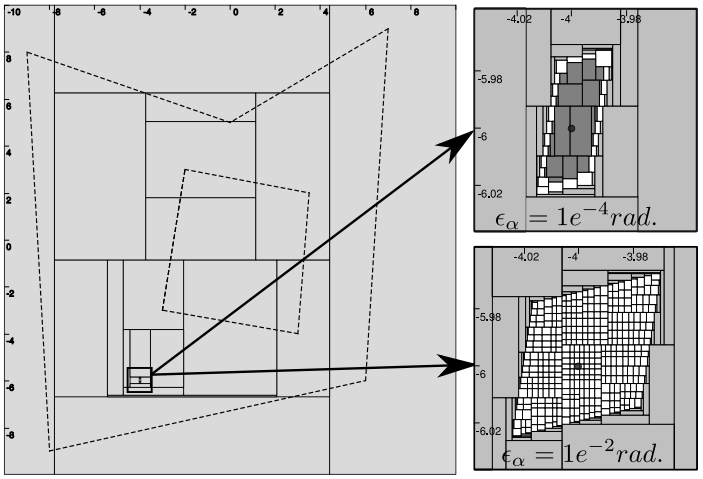


Figure 5: Resulting sub-pavings of the set inversion for $q = 11$. Dark gray boxes belong to $\mathbb{X}^-$ and light gray boxes are in $\Delta\mathbb{X}$. The black circle is the theoretical position of the robot

4 Outer-GOMNE

4.1 Principle

With only an outer contractor, the classical GOMNE approach fails to find a guaranteed result because the stopping condition ($\mathbb{S}^- \neq \emptyset$) can not be satisfied. To cope this lack of constraint, we propose to use the local information given by j (equation 13) to find the smallest majorant of q^* as possible. The outer contractor will remove inconsistent parts of the initial space while a local method looks for q^*.

Let's take a set of N measurements, each of them defining a set $\mathbb{X}_i$. Let $\mathcal{C}^i_{out}$ be the outer contractor associated with the set $\mathbb{X}_i$. The method is based on the following propositions:

proposition 1:

Based on the definition of the outer contractor, we have :

$$\forall [\mathbf{x}] \in \mathbb{IR}^n, \mathcal{C}^i_{out}([\mathbf{x}]) = \emptyset \Leftrightarrow [\mathbf{x}] \notin \mathbb{X}_i \tag{20}$$

which means that if $\mathcal{C}^i_{out}([\mathbf{x}]) = \emptyset$, $[\mathbf{x}]$ is not consistent with the i^{th} measurement. In the special case where $[\mathbf{x}]$ is a singleton denoted $\{\mathbf{x}\}$, we define the following function :

$$\begin{array}{rcl} \mathbb{R}^n & \rightarrow & [0,1] \\ \mathbf{x} & \mapsto & \mu_i(\mathbf{x}) = \left\{ \begin{array}{ll} 1 & \text{if } \mathcal{C}^i_{out}(\{\mathbf{x}\}) = \emptyset \\ 0 & \text{otherwise} \end{array} \right. \end{array} \tag{21}$$

proposition 2:

An explicit expression for $j : \mathbb{R}^n \rightarrow [\![0, N-1]\!]$ is given by :

$$j(\mathbf{x}) = \sum_{i=1}^{N} \mu_i(\mathbf{x}) \tag{22}$$

remark 1: Punctual values are used in j instead of intervals in order to limit the pessimism during the evaluation process.

remark 2: Instead of having additional information, properties of j such as continuity, variations, monotonicity are unknown: j can only be evaluated at a given value.

Properties: Based on the construction of j and on the monotonicity of the relaxed intersection we have :

$$\mathbb{S}^* \neq \emptyset \Leftrightarrow \exists \mathbf{x} \in \mathbb{R}^n \ , \ j(\mathbf{x}) = q^* \tag{23}$$

$$\forall \mathbf{x} \in \mathbb{R}^n, \mathbb{S}_{q^*} \subseteq \mathbb{S}_{j(\mathbf{x})} \tag{24}$$

$$\forall \mathbf{x} \notin \mathbb{S}^*, j(\mathbf{x}) > q^* \tag{25}$$

With equation (23), the minimum of j can always be found if the search algorithm has no time limit. Otherwise, with the property (24), it can be stopped at any time and the guarantee is preserved. With the last property, algorithms can be run safely with any $\mathbf{x} \in \mathbb{R}^n$. It is now possible to define a contractor for $[q]$. Given a subset $\mathbb{A}$ of $\mathbb{R}^n$:

$$\mathcal{C}^j_{\mathbb{A}}([q]) = [q] \cap [0, \min_{\mathbf{x} \in \mathbb{A}} j(\mathbf{x})] \tag{26}$$

By construction $\mathcal{C}^j_{\mathbb{A}}$ is monotonic, but not always minimal, depending on the result of the minimization algorithm used. If the minimum is very hard to find, q^* will be overestimated.

To minimize the function j several strategies can be used. A trivial way is to randomly test N points in $\mathbb{A}$ and keep the minimal value. The probability of finding the minimum increases when N tend towards infinity or when the volume of $\mathbb{A}$ approaches towards 0. A smarter way is to consider j as a likelihood function and use the particle filter framework to estimate $\mathbb{S}^*$. Alternatively, the issue of finding the best value of j could be seen as a black-box optimization problem because no information are available on j. We choose to use the NOMAD library [21, 1] which is a C++ implementation of the Mesh Adaptive Direct Search (MADS) algorithm [2, 4] designed for constrained optimization of black-box functions.

4.2 Algorithm

The initial GOMNE algorithm is modified in order to take into account the new way to estimate q^*. At the beginning, $[q] = [q^-, q^+]$ is unknown and ranges between 0 and $N-1$. Similarly to the GOMNE algorithm a set inversion is performed. If the resulting sub-paving S^+ is empty, there are at least q outliers and $q^- = q+1$ else the contractor $\mathcal{C}^j_{\mathbb{S}^+}$ is called to find a better value for q^+. The algorithm stops when $q^- \geq q^+$, and then $q^* = q^+$.
Remarks:

- The returned value is not always optimal and depends on the result of the minimization. However, if $\mathbb{S}^*$ is small, the initial search space is also small and an optimal value is found quickly. So, after the optimization, if no improvement of the value of q is found (line 8), a smaller accuracy (ϵ) is used to reduce the size of boxes which composed the $\mathbb{S}^+$. Then SIVIA is called until $\mathbb{S}^*$ becomes empty or $\epsilon < \epsilon_{lim}$ where ϵ_{lim} is chosen by the user.
- When the number of outliers is greater than 50% testing each value of q could be very slow. To solve this issue, when $\mathbb{S}^*$ is empty, q can be incremented by Δq and the best value which satisfies $q^- >= q^+$ will be found quickly. However, this value will not always be optimal, but satisfies $q^+ - q^* \leq \Delta q$.
- At any time, the algorithm could be stopped and the solution set $\mathbb{S}^+_{q^+}$ will contain the solution in a guaranteed way.

Algorithm 5: Outer-GOMNE(in: X_0 , $\epsilon_0, \epsilon_{lim}$, out: q^+)

```
1  q⁻ = 0 , q⁺ = N − 1, ϵ = ϵ0;
2  while (q⁺ > q⁻) do
3  |  S⁺ = SIVIA([x0], C_{q⁻}, ϵ);
4  |  if (S⁺ = ∅ or ϵ < ϵ_lim) then
5  |  |  q⁻ = q⁻ + 1, ϵ = ϵ0
6  |  else
7  |  |  q = min_{x∈S⁺} j(x);
8  |  |  if (q < q⁺) then
9  |  |  |  q⁺ = q
10 |  |  else
11 |  |  |  ϵ = ϵ/2
```

Results of this algorithm are shown in the next section. A compromise needs to be found between the time allocated to set inversion and to optimization.

4.3 Test Case and Results

4.3.1 Introduction

Let's take a robot $\mathcal{R}$ in an environment composed of lines and circles, which could for instance represent trees and buildings. An initial map containing only the border of obstacles is provided but in this case, no inner and outer parts can be defined (the volume of $\mathbb{M}$ is null). To our knowledge, no inner contractor has been found for this kind of situation and only an outer contractor is available. The robot is equipped with a laser sensor which returns the distance d_i between $\mathcal{R}$ and the nearest obstacle in a given direction (α_i). Let $\mathbf{y}_i = (d_i, \alpha_i)$ a measurement. The pose $\mathbf{x} = (p_x, p_y, \theta)$ of $\mathcal{R}$ must satisfy :

$$\mathbf{g}\left(\mathbf{x}, \mathbf{y}_i\right) \in \mathbb{M} \tag{27}$$

where $\mathbf{g}$ is the function which translates the point (p_x, p_y) by a distance d in the direction given by $\alpha + \theta$ defined by:

$$\begin{array}{rccl} \mathbf{g}: & \mathbb{R}^3 \times \mathbb{R}^2 & \longrightarrow & \mathbb{R}^2 \\ & (p_x, p_y, \theta, d_i, \alpha_i) & \longmapsto & \begin{pmatrix} q_x \\ q_y \end{pmatrix} = d_i . \begin{pmatrix} \cos\left(\alpha_i + \theta\right) \\ \sin\left(\alpha_i + \theta\right) \end{pmatrix} + \begin{pmatrix} p_x \\ p_y \end{pmatrix} \end{array}$$

To ease the reading, $\mathbf{g}(\mathbf{x}, \mathbf{y}_i)$ is denoted by $\mathbf{g}_i(\mathbf{x})$. The map is represented by a binary occupancy grid and the constraint $\mathbf{g}_i(\mathbf{x}) \in \mathbb{M}$ is implemented with the image contractor (see 4.3.2 for a short introduction), which is very efficient for this kind of application. In addition, the localization algorithm has to deal with erroneous measurements and detection of unmapped objects such as people, trees or cars. The situation is depicted in figure 6. The solution set $\mathbb{X}$ is then defined by:

$$\mathbb{X} = \bigcap^{\{q^*\}} g_i^{-1}\left(\mathbb{M}\right) \tag{28}$$

where $q^* = \min_{\mathbf{x} \in \mathbb{R}^3} j(\mathbf{x})$.

4.3.2 Map contractor

Unstructured environments can be described by an occupancy grid represented as a binary image i in which the value of each pixel is 1 if it is occupied, 0 otherwise. The *image contractor* introduced in [26, 12] can then be used. This contractor is based on the summed area table (also known as integral image), widely used in computer vision, defined by:

$$I\left(x, y\right) = \sum_{\substack{x' \leq x \\ y' \leq y}} i\left(x', y'\right) \tag{29}$$

Using the integral image, the number of 1-valued pixel contained in any rectangular region can be computed in four operations. Let ϕ be the function which returns the number of occupied cells in a given box $[\mathbf{x}] = [x_1^-, x_1^+] \times [x_2^-, x_2^+]$ aligned on the grid. Let $A(x^-, y^-)$, $B(x^+, y^-)$,

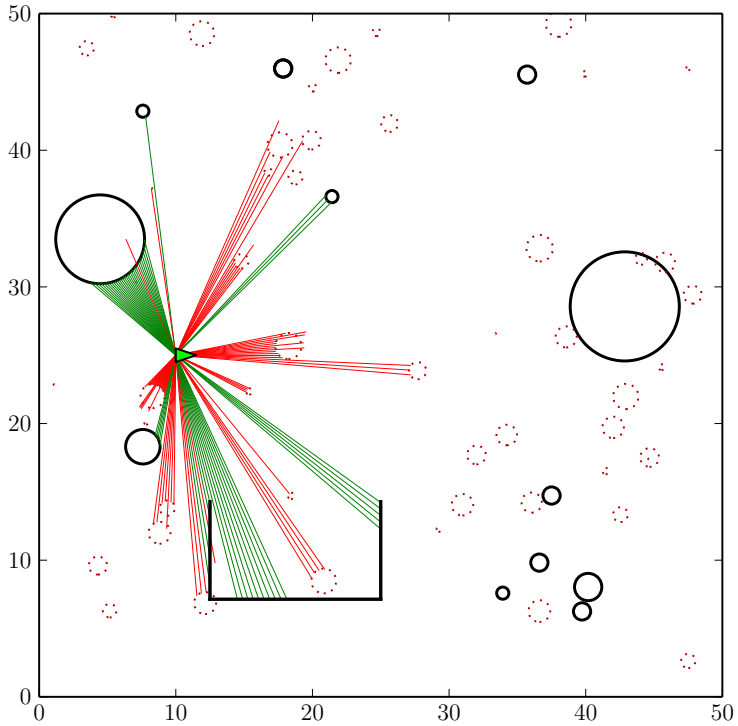


Figure 6: The map $\mathbb{M}$, in black, is composed of lines and circles. The opened polygon does not define a closed area and consequently the map has an empty volume. Dashed red circles are unmapped objects which generated false detection. Consistent measurements are in green, while outliers are in red.

$C(x^+, y^+)$, $D(x^-, y^+)$ be the coordinates of the rectangle corners, we have :

$$\begin{array}{rcl} \mathbb{IN}^2 & \longrightarrow & \mathbb{N} \\ \phi\left([\mathbf{x}]\right) & \longmapsto & I(A) + I(C) - I(B) - I(D) \end{array} \tag{30}$$

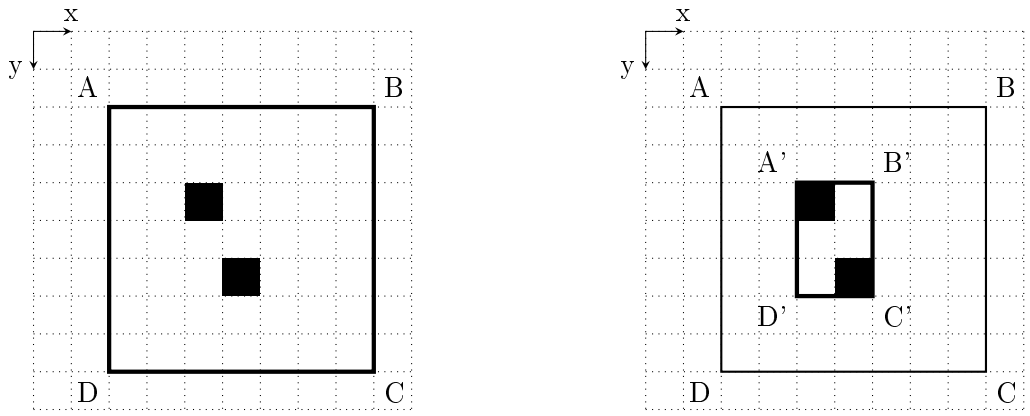
On figure 7a the box contains two black pixel, so $\phi\left([\mathbf{x}]\right) = 2$. From an initial box $[\mathbf{x}_0]$, the image contractor aims at finding the smallest box $[\mathbf{x}]$ included in $[\mathbf{x}_0]$ which contains exactly the same number of 1-valued pixels, ie. $\phi([\mathbf{x}_0]) = \phi([\mathbf{x}])$.
Denote by $\mathcal{C}$ the image contractor. Consider $[\mathbf{x}] \in \mathbb{IN}^2$ and $\mathcal{C}([\mathbf{x}]) = [\mathbf{y}]$. We have:

$$\begin{cases} y_1^- = \max(x \in [x_1], \phi([x_1^-; x] \times [x_2]) = 0) \\ y_1^+ = \min(x \in [x_1], \phi([x; x_1^+] \times [x_2]) = 0) \\ y_2^- = \max(x \in [x_2], \phi([x_1] \times [x_2^-; x]) = 0) \\ y_2^+ = \min(x \in [x_2], \phi([x_1] \times [x; x_2^+]) = 0) \end{cases} \tag{31}$$

The min and max can be computed using a dichotomy, which has a logarithmic complexity.

4.3.3 Results

A trajectory of 20 configurations is generated, for each of which the number and the position of unmapped objects changes (case of a very dynamic environment). All different configurations are shown in figure 10b. The heading θ_{th} is assumed to be known with an accuracy of $\pm$ 30 degrees while their position is completely unknown. The optimization procedure use MADS algorithm

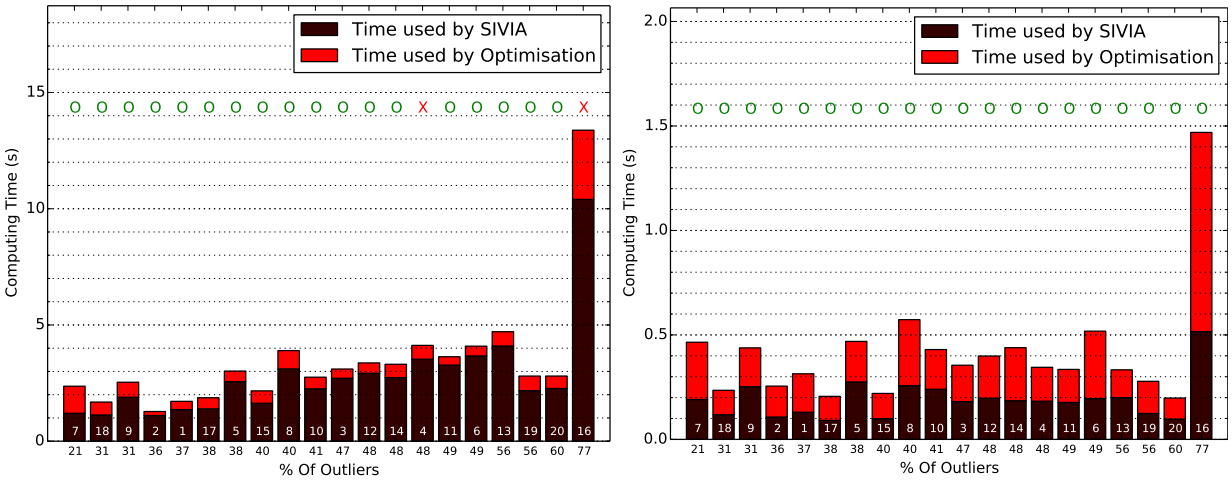


(a) ϕ function evaluation (b) Contraction of the box

Figure 7: Image contractor

coupled with a Various Neighborhood Search (VNS) algorithm [3] to escape local minima. For each box of the resulting sub-paving $\mathbb{S}^+$ an optimization is run with a maximum of 100 tries. Finally the ϵ used in SIVIA is $\epsilon = (1m, 1m, 3°)$. The algorithm is run on a Intel(R) Core(TM) i5-2450M CPU at 2.50GHz and the computing time for each situation is shown in figure 8a. The width of the box which enclosed $\mathbb{S}_{q^*}$ and the error between its center and x_{th} are given in table 1.

Outer-GOMNE retrieves, in a guaranteed way, the set $\mathbb{S}_{q^*}$. Nevertheless, due to undetected



(a) $w([\mathbf{x}]) = (50m,\ 50m,\ 60°)$ (b) $w([\mathbf{x}]) = (5m,\ 5m,\ 4°)$

Figure 8: Computing time needed to find $\mathbb{S}_{q^*}$ with different initial estimates, sorted by the percentage of outliers. The configuration's number is in white. Resulting sub-pavings, which does not contain the true solution, are marked with a red cross.

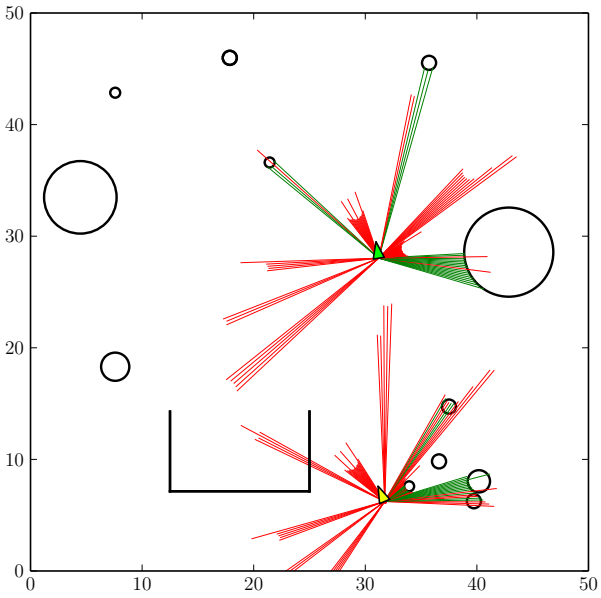
and spurious outliers, the true solution $\mathbf{x}_{th}$ sometimes do not belong to the resulting solution set $\mathbb{S}_{q^*}$. This is the case for the 4^th configuration in figure 9a. To avoid this, like the original GOMNE algorithm, an arbitrary number of undetected outliers r can be added to the returned value q^* to characterize the set $\mathbb{S}_{q^*+r}$, which is more likely to contain $\mathbf{x}_{th}$. However, when the proposition of outliers is higher than 50%, another configuration consistent with a set of outliers may exist in the initial box. This is the case of the 15^{th} situation in figure 9a, localized as shown in figure 9b. An additional constraint such as "a beam can not cross the map" can be added to avoid this situation but, in the general case, without additional information, such situation

#	$w([x])$ m	$w([y])$ m	$w([\theta])$ rad	err_{xy} m	err_{θ} rad
1	0.092	0.154	0.436	0.013	2.33e-02
2	0.106	0.131	0.432	0.018	2.44e-02
3	0.139	0.185	0.698	0.016	1.53e-02
4	**0.129**	**0.095**	**0.428**	**0.033**	**1.33e-01**
5	0.204	0.172	0.721	0.023	3.21e-02
6	0.172	0.197	0.800	0.035	1.59e-01
7	0.130	0.190	0.500	0.025	7.46e-02
8	0.099	0.192	0.439	0.014	7.10e-02
9	0.109	0.181	0.693	0.002	4.21e-02
10	0.093	0.166	0.545	0.015	1.46e-02
11	0.127	0.218	0.626	0.007	2.74e-02
12	0.178	0.236	0.702	0.012	9.57e-02
13	0.214	0.227	1.011	0.025	3.64e-02
14	0.231	0.199	0.950	0.016	1.89e-02
15	0.215	0.288	1.031	0.029	5.47e-02
16	**5.358**	**0.407**	**37.885**	**21.959**	**4.99e+00**
17	0.190	0.247	1.120	0.016	7.45e-02
18	0.228	0.197	1.040	0.015	3.15e-02
19	0.279	0.193	0.812	0.007	7.56e-02
20	0.258	0.204	1.123	0.020	2.35e-02

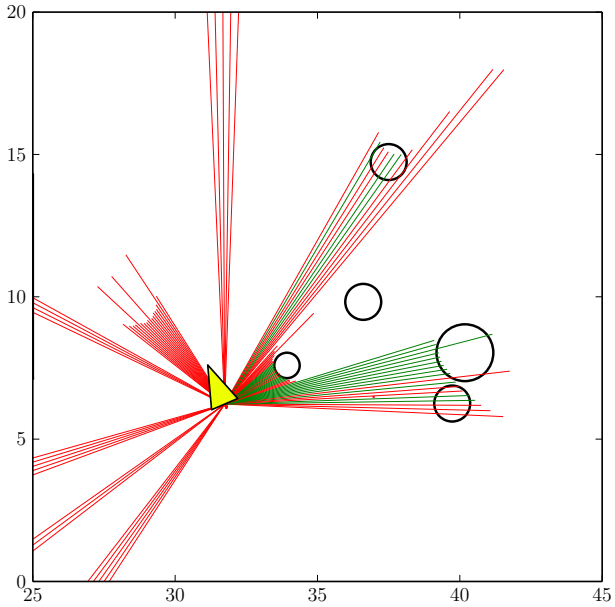
Table 1: Results of each configuration. The size of the union of all boxes of the sub-paving is given and the error between the theoretical position and the center of this union of the sub-paving. Bold lines are configuration inconsistent with the theoretical position.

can not be avoided. Results of the set inversion with a smaller initial box are shown in figure 8b with $r = 1$. Here the computing time is lower and the algorithm never fails.

During our tests, Outer-GOMNE was able to cope with more than 70% outliers. Its main drawback, like its original version, is that it looks for the set which minimizes the number of outliers and, if only static information are used, this set may not contain the true solution. To address this issue, the fact that successive configuration constitute a trajectory can be considered.

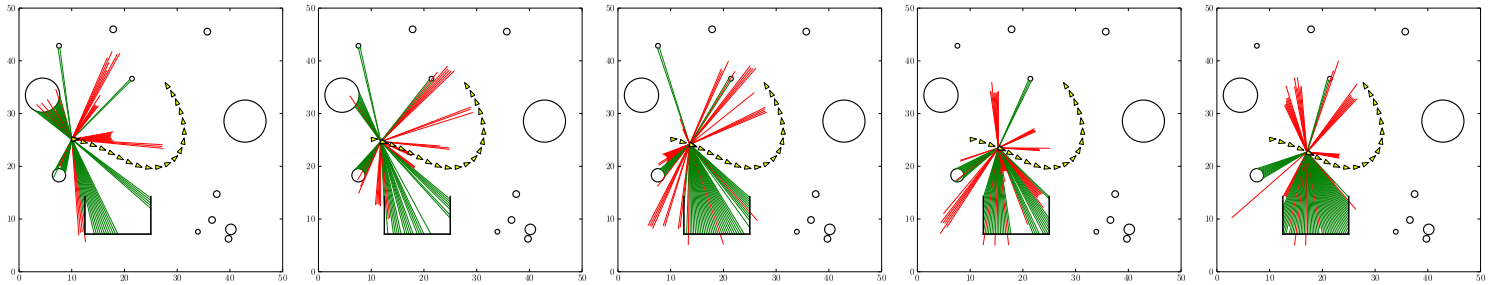


(a) Actual position (green robot) and estimated position (red robot) of the 15^{th} configuration

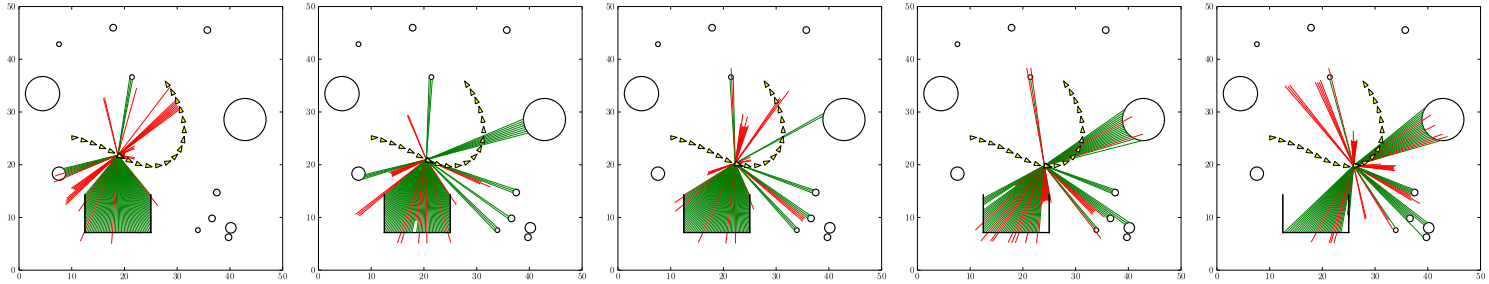


(b) Close-up of the estimated position

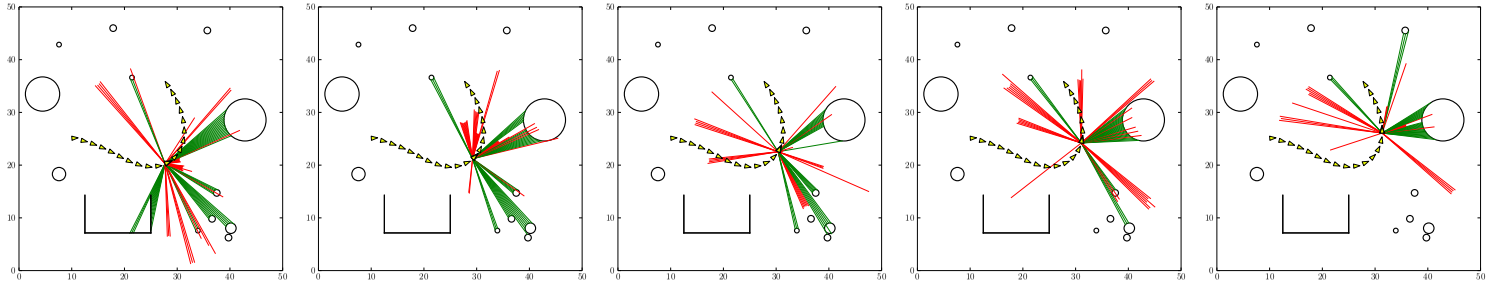
Figure 9: Result of the 15^{th} ambiguous situation. In figure 9a, the initial configuration (green robot) has 73 outliers. Outer-GOMNE finds another configuration (in yellow) with only 67 outliers (figure 9b). The constraint used allows beams to cross the map and leads an erroneous estimation.



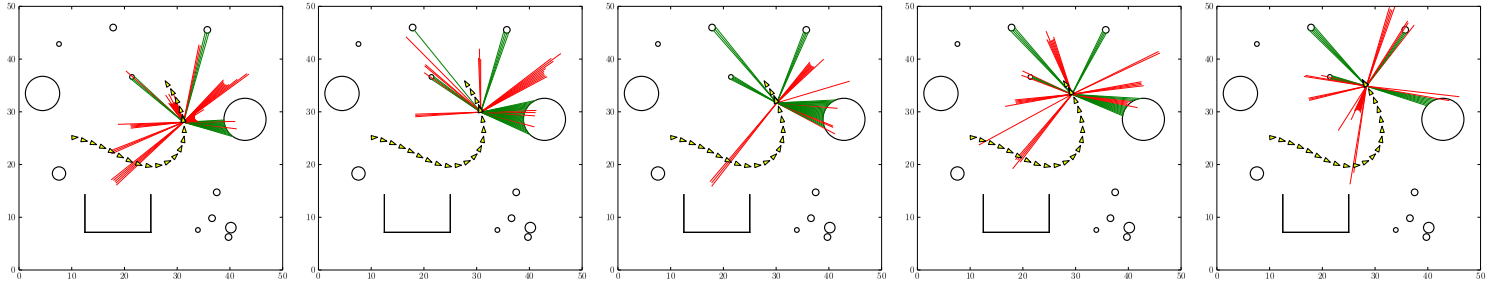
(1) 37% (106) (2) 36% (85) (3) 47% (102) (4) 48% (97) (5) 38% (108)



(6) 49% (150) (7) 21% (111) (8) 40% (134) (9) 31% (103) (10) 41% (109)



(11) 49% (96) (12) 48% (75) (13) 56% (66) (14) 48% (69) (15) 40% (58)



(16) 77% (95) (17) 38% (60) (18) 31% (52) (19) 56% (54) (20) 60% (50)

5 Extension to Trajectories

5.1 Principle and Algorithm

The previous method can be applied as contractor for *tubes* (which is an interval of functions [5]) to retrieve the trajectory of the robot by using its dynamics or any independent relative motion estimation measure such as provided by odometers or inertial navigation systems. Trajectories contain much more information which can deal with symmetric and ambiguous situations, especially when the true solution does not belong to $\mathbb{S}_{q^*+r}$. Let's take a system which provides N_k outputs at each time step k described by :

$$\begin{cases} \mathbf{x}_k &= f\left(\mathbf{x}_{k-1}, \mathbf{u}_k\right) + \mathbf{w}_k \\ \mathbf{y}_k &= \mathbf{g}_k\left(\mathbf{x}_k\right) \end{cases} \tag{32}$$

where $\mathbf{y}_k = (\mathbf{y}_k^1, \ldots, \mathbf{y}_k^{N_k})$ and $\mathbf{g}_k\left(\mathbf{x}_k\right) = \left(\mathbf{g}_1^1(\mathbf{x}_k), \ldots, \mathbf{g}_k^{N_k}(\mathbf{x}_k)\right)$. A state noise $\mathbf{w}_k$ is assumed to belong to the box $[\mathbf{w}(k)]$. Each output contains q_k outliers. Let $\mathbf{p}_k = (\mathbf{x}_k, q_k)$, the vector of parameters to be estimated for time k.

Constraint propagation can be used to estimate the trajectory $\mathbf{z} = (\mathbf{p}_0, \ldots, \mathbf{p}_{N_k})$ of the system. Let $\mathcal{C}^{ik}$ be the outer contractor associated to the constraint $\mathbf{g}_k^i(\mathbf{p}_k) \in [\mathbf{y}_k^i]$. We define the contractor $\mathcal{C}^k$ which contract a box $[\mathbf{p}_k] = [\mathbf{x}] \times [q_k]$:

$$\begin{array}{rccl} \mathcal{C}^k : & \mathbb{IR}^n \times \mathbb{IN} & \longrightarrow & \mathbb{IR}^n \times \mathbb{IN} \\ & ([\mathbf{x}], [q]) & \longrightarrow & \left\{ \begin{array}{rcl} [q] &=& [q] \cap [0, \min\limits_{\mathbf{x} \in [\mathbf{x}]} j(\mathbf{x})] \\ [\mathbf{x}] &=& [\mathbf{x}] \cap \bigcap\limits^{q^+} \mathcal{C}^{ik}([\mathbf{x}]) \end{array} \right. \end{array} \tag{33}$$

The algorithm 6 inspired from [14] contracts an unknown trajectory $[\mathbf{z}] = ([\mathbf{x}_0, \ldots, \mathbf{x}_N])$. For each time step k, $[\mathbf{x}_k]$ is predicted from the previous $[\mathbf{x}_{k-1}]$ using equation (32). Then, $\mathcal{C}^k$ is called to estimate the number of outliers and contract $[\mathbf{x}_k]$ with this value. Using interval arithmetic, a backward propagation is then done. This contractor procedure can be called several times until it has no effect on $[\mathbf{z}]$ (the fixed point is then reached).

Algorithm 6: C_{traj}(in: $\mathcal{C}^k$, $\mathbf{f}$ inout :$[\mathbf{z}] = (\mathbf{p}_0 = [\mathbf{x}_0] \times [q_0], \ldots, \mathbf{p}_k = [\mathbf{x}_k] \times [q_k])$)

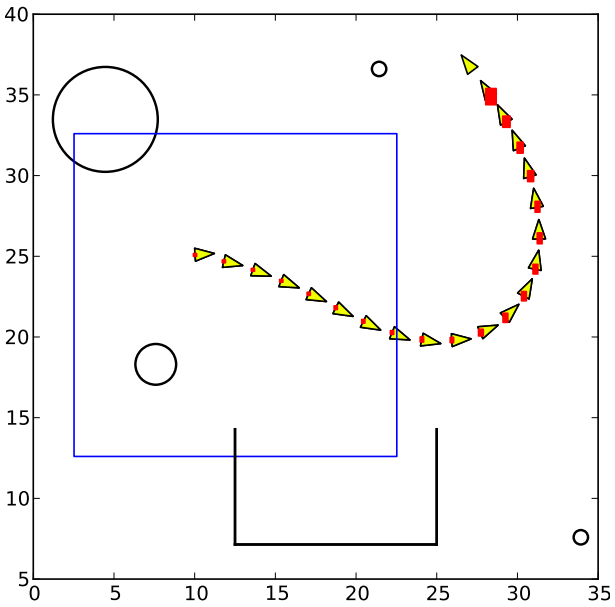
```
for k from 1 to N do
    [x_k] = f([x_{k-1}]);
    C^k([p_k]) see equation 33;
for k from N-1 to 0 do
    [x_k] = [x_k] ∩ f^{-1}(x_{k+1})
```

In the end, the trajectory can be retrieved only with contractions and optimizations. However, if one step becomes inconsistent with the other the resulting trajectory will be empty. To avoid this, the relaxed intersection can be used on the trajectory space. Let $\mathbf{z}_i$ be the trajectory corrected only with the output of the system at time k. The final trajectory is now defined by

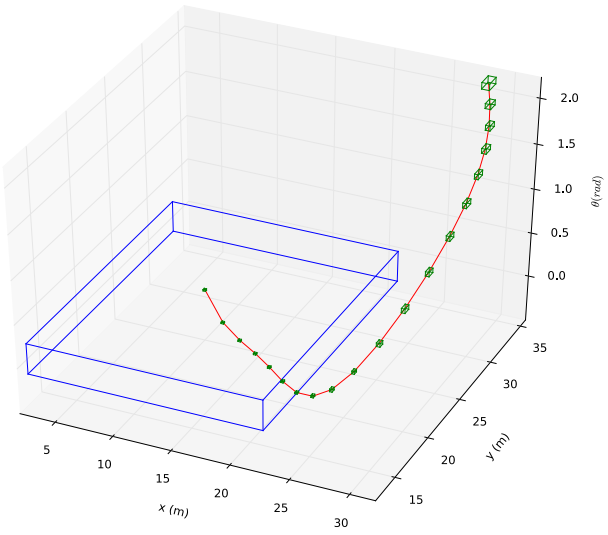
$$\mathbf{z} = \bigcap^{q_z} \mathbf{z}_i \tag{34}$$

$w_0([x])$(m)	$w_0([y])$(m)	$w_0([\theta])$(rad)	number of run	time(s)
20	20	0.35	500	165
10	10	0.035	500	120
10	10	0.035	100	33
5	5	0.035	50	20

Table 2: Computing time needed to retrieve the trajectory with different parameters



(a) Recovered trajectory with the dynamic algorithm



(b) Projection on the (x,y) plan

5.2 Test Case

Consider the whole trajectory of the previous test case. The observation model remains the same and the motion model is defined by:

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \mathbf{u}_k + \mathbf{w}_k \tag{35}$$

where $\mathbf{u}_k$ is the movement of the robot between time t_{k-1} and t_k. An additive noise of more or less 10% of the input vector value is used for the prediction : $[\mathbf{w}_k] = [-0.1 * \mathbf{u}_k, 0.1 * \mathbf{u}_k]$. We allow also 10% of spurious states ($q_z = 2$). Table 2 shows the computing time needed to retrieve a 65 meter trajectory long with different sizes of the initial box, and different numbers of run used by the optimization algorithm. Figures 10a and 10b show the estimated trajectory. Last steps are less accurate because less information constrains the position.

Instead of processing the whole trajectory in one run, it is possible to apply the algorithm on a sliding temporal window and compute the trajectory while the robot is moving. With a 10 time steps window, and a small box containing the true position, the computing time is less than 12 seconds. Outer-GOMNE can be used instead of $\mathcal{C}^k$.

6 Localization in a Digital Elevation Map

DEM's represent a surface $z = f(x, y)$ on a regular Cartesian grid where each cell contains the value of the elevation z. Depending on the data and algorithms used to build such a structure, additional information such as a standard deviation and extrema values for z can be encoded in each cell. DEM's are widely used, either in geographic data systems because they present a good compromise between expressiveness, simplicity and compactness[1], or in robotics, where they are easily built from range data and exploited to assess traversability, to plan itineraries or trajectories, or to assess visibilities between positions.

If a DEM is a nearly perfect structure to represent surfaces, it fails to represent verticals and overhangs, and this has numerous consequences, especially when it comes to associate range data acquired by the robot to the map. Furthermore, the presence of unmapped elements in the environment, and the fact that some objects may only be partially mapped make the localization challenging. This section presents how Outer-GOMNE can cope with these issues, using range data acquired by the robot.

6.1 Initial Models

6.1.1 Map Structures

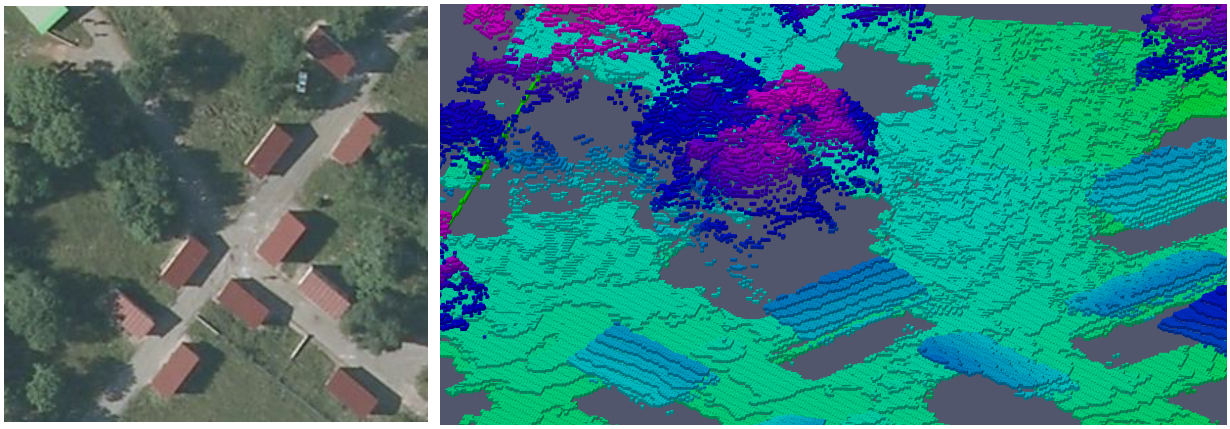
With a DEM, the space can be split into two sets: the one over the surface ($z - f(x, y) > 0$) and the one below ($z - f(x, y) < 0$). In such cases, an inner and an outer contractor can be defined, and the same formalism of section 3.2 can be applied to localize the robot using range data, using the classical GNOME approach to cope with the presence of outliers. DEMs are particularly adapted to open terrains for which a surfacic representation is well suited, or in more complex (non-surfacic) terrains, in the case where the range data used to localize the robot are gathered in conditions similar to the data from which the DTM has been built.

For instance, if a DEM is built from an Unmanned Aerial Vehicle (UAV) using a downwards oriented sensor (see figure 10b), range data acquired by other UAVs using a downwards oriented sensor can be well matched with the DEM structure. But on the contrary, range data acquired by a ground robot will hardly match such a map: *e.g.* the tree canopy is mapped by the UAV, whereas the ground robot only perceives tree trunks and lower branches and leaves. From the point of view of a UAV, the representation as a surface (convex world) is a good approximation. However from the point of view of the ground robot, the elevation between two adjacent cells is assumed to be continuous, which leads to a poor representation of verticals and overhangs.

As a compromise, the DEM can be voxelized, by filling the gaps between two different elevations of neighboring cells. The figure 11 shows the resulting environment model after applying this process to the DEM of figure 10b. Note that this representation also allows the use of an image contractor extended to the 3D case. But if this copes for verticals such as walls, it also clearly introduces erroneous information: the extension to the ground of the tree canopy does not fit the reality and hides the tree trunks for instance.

To better solve this issue, a volumetric representation using a three-dimensional voxel grid can be used: non surfacic elements are then accurately represented. However, the building of such models has a high computational cost and requires a lot of memory. And still, some objects may not be entirely mapped in the model: *e.g.* sometimes only parts of buildings, cars, trees, are

[1] For instance, most commercial aerial mapping systems generate DEMs along with orthoimages



(a) Aerial orthoimage of a test site (b) DEM

Figure 10: Aerial view of a 80m x 80m test site, and DEM built from a UAV with a downwards looking Lidar. Note that buildings roofs and tree canopy are disconnected from the ground surface.

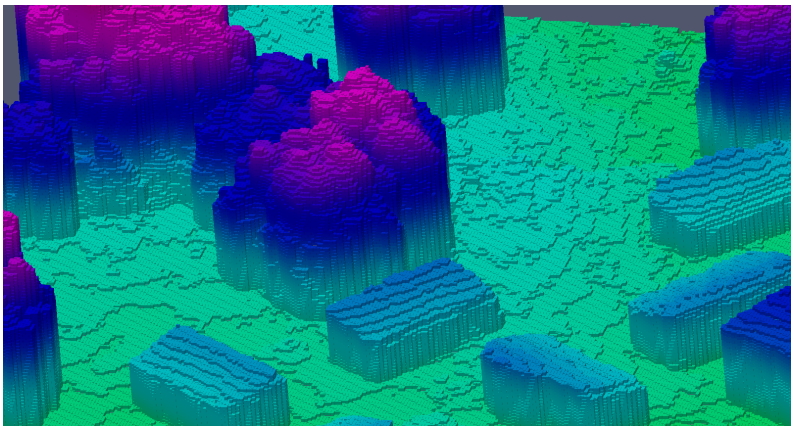


Figure 11: View of the DEM of figure 10b after its "voxelisation".

modeled and the resulting model does not define closed connected components, as can be seen in the model of figure 12, where only walls perceived by the robot are modeled – and not the entire buildings. The hypothesis of a convex world then does not hold, and no inner contractor are available: in such a situation, Outer-GOMNE needs to be used.

6.1.2 Test Maps

.

For our localisation tests, three different models of the same area (training camp of Caylus, figure 10a) are used. The first model, denoted "atLaas", is built from a set of 3D scans acquired by a Velodyne Lidar mounted on a ground robot (see figure 12). The second and third models are built from data acquired by an UAV equipped with a camera (13a) and a Lidar (figure 13b), and are respectively denoted "UAV-Vision" and "UAV-Lidar".

The voxelization is done by marking as occupied all cases such as $z = f(x, y)$ and filling gaps.

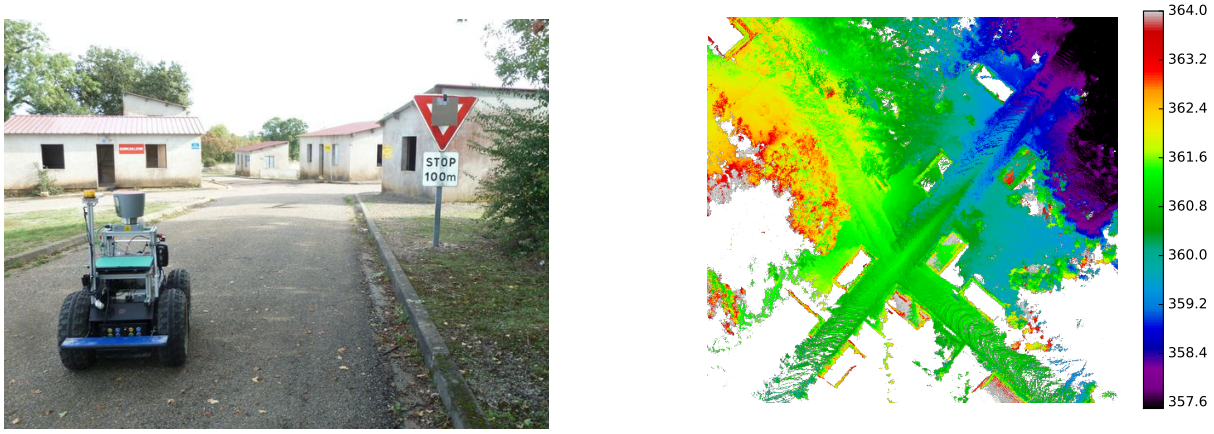


Figure 12: The ground robot with a Velodyne Lidar in the site of figure 10a, and the corresponding "atLaas" DEM (0.1m resolution).

The formulation is :

$$i(x_i, y_i, z_i) = \begin{cases} 1 & \text{if } z_i \in \left[\min_{\substack{x \in x_{i-1}, x_i \\ y \in y_{i-1}, y_i}} f(x,y), \max_{\substack{x \in x_{i-1}, x_i \\ y \in y_{i-1}, y_i}} f(x,y) \right] \\ 0 & \text{otherwise} \end{cases} \tag{36}$$

6.2 Data Pre-processing

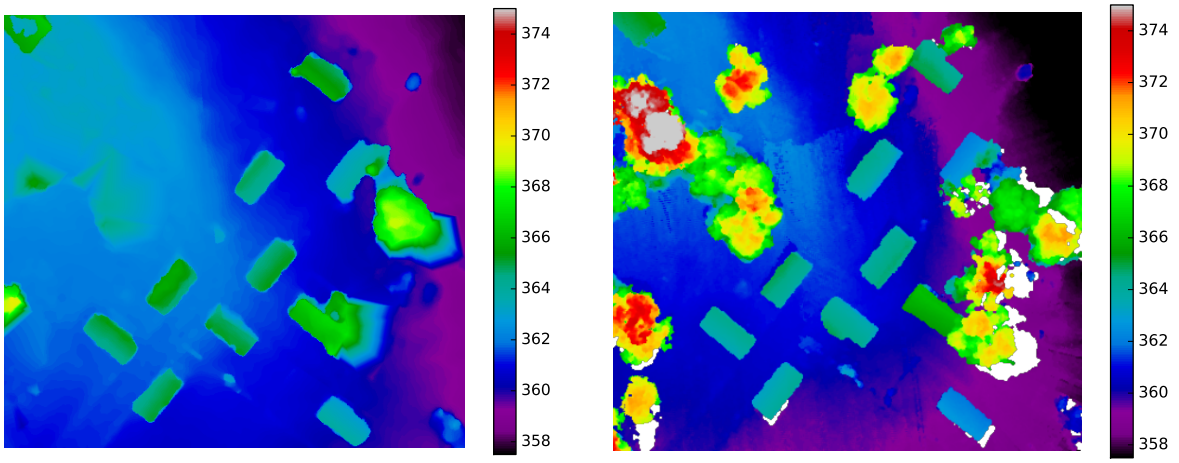
Before using the Outer-GOMNE algorithm, the range data acquired by the ground robot Lidar need to be preprocessed. Indeed, the complexity of the algorithm is linear with respect to the number of measurements, and a 360° panoramic scan indeed contains up to 288.000 points. Also, in the context of our localization problem where only p_x, p_y, φ need to be retrieved, measurements which are likely to correspond to the ground are removed.

First, the density of the point cloud dramatically decreases with the distance to the sensor: the initial point cloud (figure 14a) is down-sampled to homogenize its spatial density (see figure 14b). To select out the points corresponding to the ground, the normal vector of each point is estimated using a circular neighborhood support. We assume that a point belongs to the ground if the angle between the vertical $\vec{z}$ axis and the estimated normal vector is lower than a given value (based on the analysis of histograms of normal vectors, the threshold is set to 30°). The resulting point cloud is shown in figure 14c. Finally, based on the distance between points, an euclidean clustering is performed to split the cloud onto spatially coherent sub-clouds (figure 14d). Table 3 indicate the time required by these processes, which are implemented with the Point Cloud Library [24].

6.3 Algorithm Extension to 3D Data

The algorithms presented in section 4 can readily be extended to the case of localization in the 3D world, as only the contractor on the DEM and the constraint $\mathbf{g}$ (equation 27) need to be adapted.

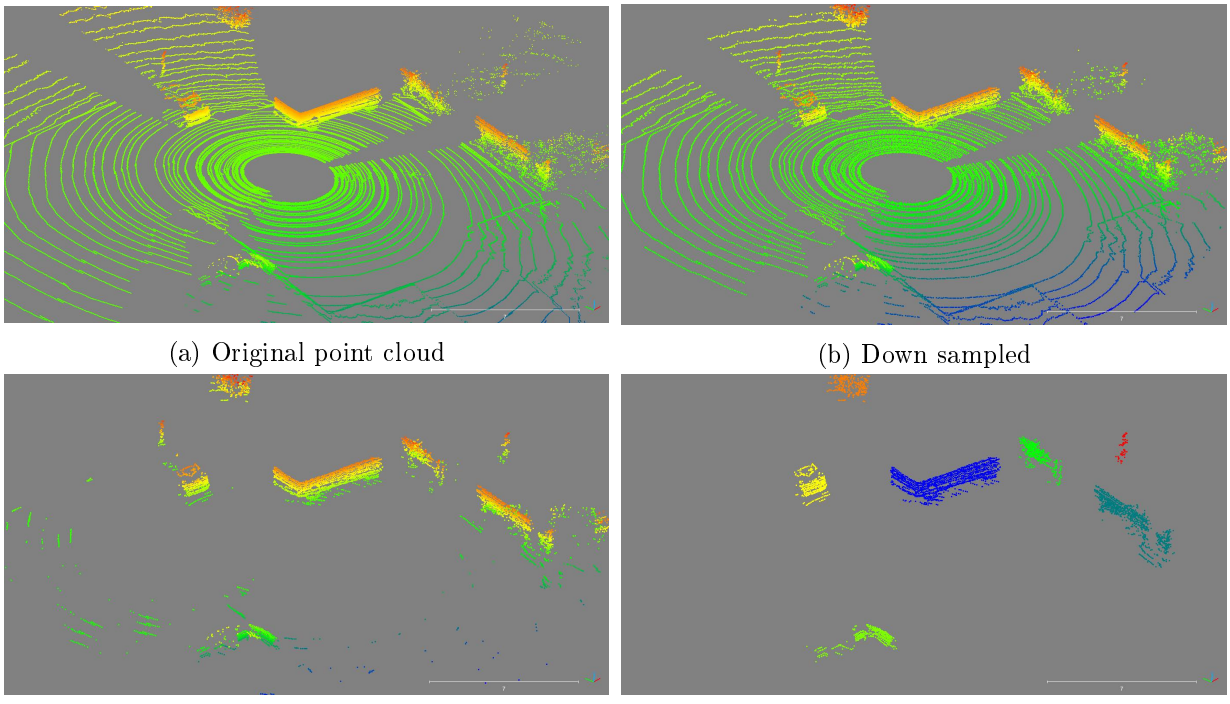
Let $\mathbf{y}^i_{B_s} = (x_i, y_i, z_i)$ be a point acquired by the sensor in its own frame (B_s). The goal is to estimate parameters $\mathbf{x} = (p_x, p_y, p_z, \theta, \psi, \phi)^T$ of the transformation matrix $\mathcal{T}_{B_0 \to B_s}(\mathbf{x})$ between



(a) "UAV-Vision" DEM built from images (0.04m resolution)

(b) "UAV-Lidar" DEM built from Lidar data (0.2m resolution)

Figure 13: DEMs of the test area of figure 10a generated by UAV data. Note that the DEM built from images contains gross errors, due to the absence of visual features matches: the elevation has been interpolated by the photogrametric algorithms to the ground elevation for most of the trees.



(a) Original point cloud

(b) Down sampled

(c) Filtered by normal direction

(d) Euclidean clustering

Figure 14: Point cloud filtering: The raw point cloud (14a) is first down-sampled (14b), and based on the normal direction, points which are likely to belong to the ground are removed (14c). Then, the remaining point cloud is split into smaller but spatially coherent subsets (14d).

Step Name	Number of points		time used
	initial	final	
Down-sampled	205 417	22 638	92 ms
Normal Estimation	22 638	22 638	239 ms
Thresholding	22 638	4 484	78 ms
Clustering	4 484	3 677	59 ms
		Total	648 ms

Table 3: Computing time of the processes that segment the point cloud of figure 14a.

the global frame B_0 and the sensor frame B_s (φ, θ, ψ respectively denote the yaw, pitch and roll).

Like in the 2D test cases, the point $\mathbf{y}^i_{B_0}$ must belong to the map, so the constraint $\mathbf{g}$ is now:

$$\begin{cases} \mathbf{y}^i_{B_0} = \mathcal{T}_{B_0 \to B_s}(\mathbf{x}).\mathbf{y}^i_{B_s} \\ \mathbf{y}^i_{B_0} \in \mathbb{M} \end{cases} \tag{37}$$

The localization problem is then formalized as a robust set inversion :

$$\mathbb{X} = \bigcap^{\{q^*\}} \mathcal{T}^{-1}_{B_0 \to B_s}(\mathbb{M}) \tag{38}$$

where Outer-GOMNE can be used to concurrently determine $\mathbf{x}$ and q^*. Based on the generalized notion of integral image in higher dimensional space (see [28]), the image contractor can be used with a voxel grid which represent the map.

6.4 Results

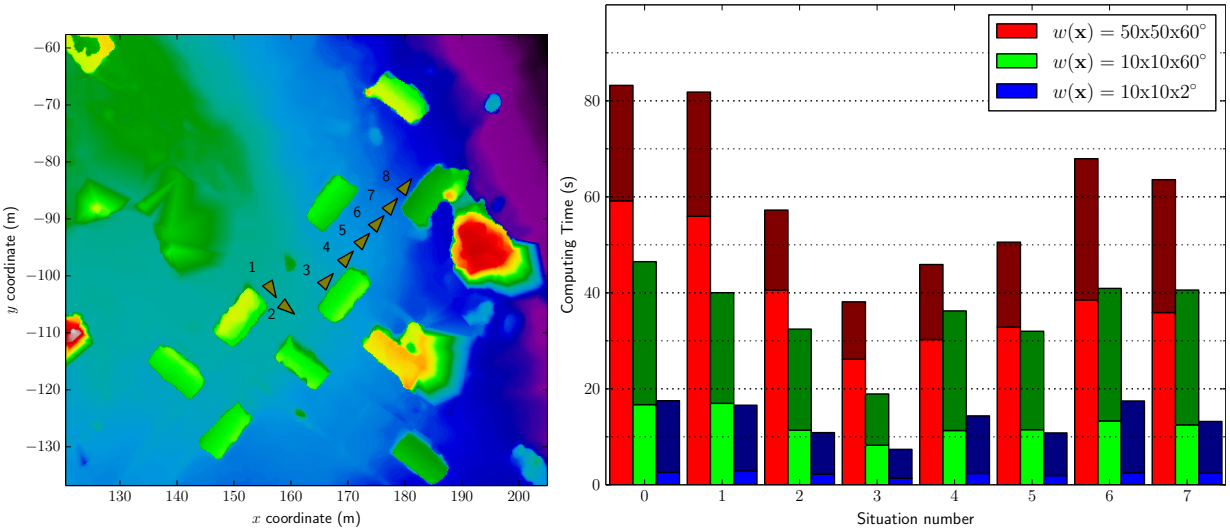
The approach has been evaluated in different configurations using two range data sets acquired in two different days, with ground truth positions provided by a centimeter accuracy RTK-GPS. One data-set has been used to produce the "atLaas" DEM (figure 12), while the second data-set is used to test the localisation algorithm in this DEM and the two "UAV-Vision" and "UAV-Lidar" DEMs derived from UAV data of the site.

Localisation results on the "atLaas" DEM for 8 robot configurations are shown in figure 15a, execution time with different sizes of the initial box are given in figure 15b (measured on a Intel(R) Core(TM) i5-2450M CPU at 2.50GHz), and the characteristics of the resulting sub-paving are given in table 4.

In each case, in accordance with the theory, the integrity of the result is satisfied and the precision of the retrieved position is smaller than one meter. Table 4 also shows the errors err_{xy} and err_θ, which have been computed by comparing the ground truth with the center of the box that bounds the resulting sub-paving.

Computing times are higher than for the 2D test case: on the one hand, the function involved in the set inversion is more complex and introduces more pessimism: hence the contraction is less efficient and more bisections are made. To contraction process can be improved by using more constraints on the pose of the robot. On the other hand, the complexity of the algorithm is linear with respect to the number of measurements used. We use the point cloud given by the segmentation algorithm which contains on average 1000 data points (see table 4 column 2). Using less measurements would speed up the algorithm and especially the optimization part –

yet a way to select relevant measurements has to be defined.



(a) Positions estimated for 8 configurations.

(b) Computation time with different sizes of the initial boxes.

Figure 15: Localisation results using the "atLaas" DEM as the initial map (here shown on the "UAV-Vision" DEM), and execution times as a function of the size of the initial box. Light colored bars correspond to the time needed by the set inversion algorithm.

#	Number of measurements	Number of outliers	$w([x])$ m	$w([y])$ m	$w([\theta])$ $mrad$	err_{xy} m	err_{θ} $mrad$
1	1174	105 (9%)	0.78	0.81	82.3	0.20	9.8
2	1154	34 (2%)	0.41	0.55	50.7	0.20	3.6
3	786	35 (4%)	0.56	0.55	53.9	0.16	18.2
4	738	85 (11%)	0.67	0.79	56.3	0.03	6.3
5	805	73 (9%)	0.80	0.75	67.0	0.09	1.0
6	912	89 (9%)	0.81	0.67	56.3	0.20	6.3
7	1038	84 (8%)	0.76	0.75	80.0	0.10	18.2
8	851	45 (5%)	0.63	0.68	67.0	0.28	24.8

Table 4: Characteristics of the resulting sub-paving (with an initial box of 50m x 50m x 60°) for the 8 position estimates shown figure 15.

Table 5 and figure 16 illustrate the localisation results of two other configurations A and B, with the three initial DEMs considered. In figure 16, the resulting sub-paving is displayed in yellow, the red triangle is the estimated position of the robot, and the range data that match the DEM in green, while the estimated outliers are in red. A straightforward observation is that using the aerial DEMs, Outer-GNOME can deal with the numerous model errors, that generate up to 59% outliers.

Figure 16a and 16b show the final situation when the "atLaas" DEM is used. As for the former trajectory, the robot is very accurately localized: the range data are indeed acquired in conditions close to the acquisition conditions to built the model, and contain less than 10% of outliers.

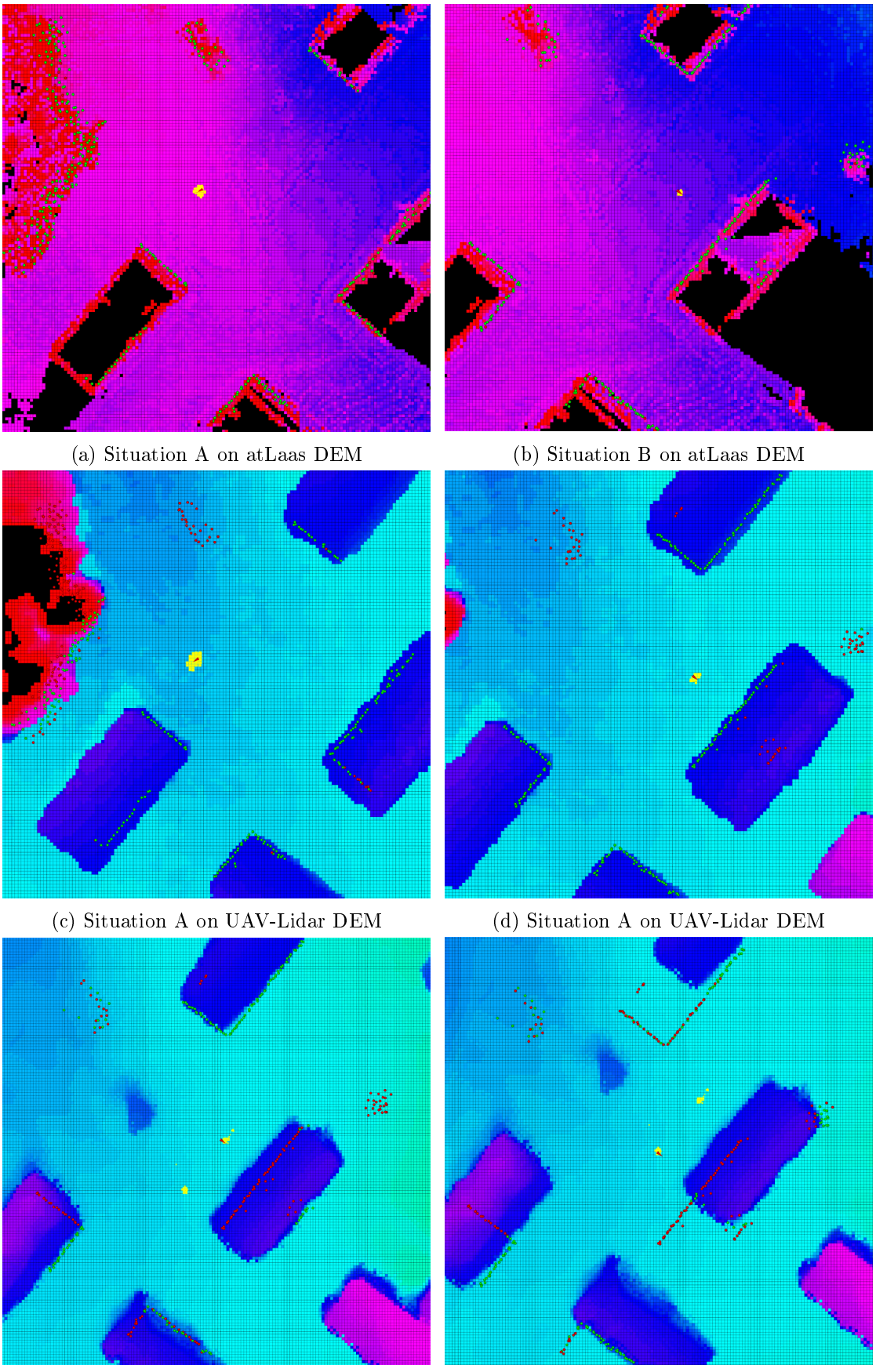
With the "UAV-Lidar" DEM, Outer-GOMNE succeeds to find the pose with a much higher proportion of outliers (figure 16c and 16d).

	situation A 414 measurements			situation B 293 measurements		
	atLaas	UAV-Vision	UAV-Lidar	atLaas	UAV-Vision	UAV-Lidar
$N_{outliers}$	26 (6%)	245 (59%)	222 (53%)	10 (3%)	158 (54%)	102(34%)
time(s)	40.33	122	63.1	7.68	105	8.38
$w([p_x])\ m$	0.72	0.80	1.11	0.36	3.67	0.81
$w([p_y])\ m$	0.89	0.97	1.39	0.37	4.54	0.76
$w([\theta])\ rad$	0.071	0.064	0.200	0.060	0.04	0.035

Table 5: Results of Outer-GOMNE for two positions estimated on three different DEMs.

The “UAV-Vision" DEM contains a scale error and it is nearly impossible for the scan to match two parallel walls in the same time. As a result, the estimated sub-paving is made of three disconnected components, which correspond to different data pairing. Figure 16f show the data points reprojected considering one of this component: they are well aligned with walls of the building on the top, but other points are inside the middle building. Consequently, the number of outliers increases. An other component of the sub-paving, for which the data points are shown in figure 16e, corresponds to an unfeasible pose because the constraint used allows scan to pass through walls.

Outer-GOMNE seems to be robust to gross model errors and able to face complex situation. Of course, the better the initial model is and the shorter the initial box is, the faster and more accurate the localization is.



(a) Situation A on atLaas DEM

(b) Situation B on atLaas DEM

(c) Situation A on UAV-Lidar DEM

(d) Situation A on UAV-Lidar DEM

(e) Situation B on UAV-Vision DEM – first component of the sub-paving

(f) Situation B on UAV-Vision DEM – second component of the sub-paving

Figure 16: Localisation results on the 3 different DEMs considered.

7 Discussion and Hints for Future Work

One of the main issue of map-based localization is the presence of outliers, that mainly come from the dynamic elements of the scene and limitations of the map structure. Being robust with respect to these outliers is of course essential: set membership approaches to this problem have shown to have such a robustness, the GOMNE algorithm being able to estimate the position parameters even in the case of an unknown bound on the outliers.

However GOMNE requires an inner contractor, which can not be provided when the initial model is partial and exhibits non-closed components – which is the case in most operational applications. To cope with this, we proposed Outer-GOMNE, that combines a set inversion algorithm with an optimization method. With experimental results, we have shown that Outer-GOMNE can be applied to the absolute localization problem with Digital Elevation Maps of a 3D environment as initial models, even though the DEM data structure is not well suited to represent verticals and overhangs, and thus causes gross map errors.

Various improvements can be considered at the algorithmic level. In particular, monitoring the bisection process can save considerable time: a too small value of ϵ indeed yields numerous useless bisections in the SIVIA algorithm.

Yet Outer-GOMNE has similar drawbacks as GOMNE, as it and only looks for the solution set which minimizes the number of outliers. Most of the time, this set is the good one, but in scarce cases the true solution may not belong to this set. This is one of the main difference with the Bayesian estimation framework, which, by using infinite density of probability, is less precise but never fails. An arbitrary number of undetected outliers can be used to cope with this issue. But more interestingly, the incremental estimation of the whole trajectory will bring higher robustness.

References

[1] M.A. Abramson, C. Audet, G. Couture, J.E. Dennis, Jr., S. Le Digabel, and C. Tribes. The NOMAD project. Software available at http://www.gerad.ca/nomad.

[2] C. Audet and J. Dennis. A progressive barrier for derivative-free nonlinear programming. *SIAM Journal on Optimization*, 20(1):445–472, 2009.

[3] Charles Audet, Vincent Béchard, and SébastienLe Digabel. Nonsmooth optimization through mesh adaptive direct search and variable neighborhood search. *Journal of Global Optimization*, 41(2):299–318, 2008.

[4] Charles Audet and J. E. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on optimization*, 17:2006, 2004.

[5] A. Bethencourt and L. Jaulin. Solving non-linear constraint satisfation problems involving time-dependant functions. *Mathematics in Computer Science*, 2014.

[6] Clement Carbonnel, Gilles Trombettoni, and Gilles Chabert. Q-intersection Algorithms for Constraint-Based Robust Parameter Estimation. In *Twenty-Eighth Conference on Artificial Intelligence*, 2014.

[7] Patrick J F Carle, Paul T Furgale, and Timothy D Barfoot. Long-Range Rover Localization by Matching LIDAR Scans to Orbital Elevation Maps. *Journal of Field Robotics*, 27(3):344–370, 2010.

[8] G. Chabert and L. Jaulin. Contractor Programming. *Artificial Intelligence*, 173:1079–1100, 2009.

[9] Paul Chew and K. Marzullo. Masking failures of multidimensional sensors. In *Reliable Distributed Systems, 1991. Proceedings., Tenth Symposium on*, pages 32–41, Sep 1991.

[10] F.G. Cozman, E. Krotkov, and C. E. Guestrin. Outdoor visual position estimation for planetary rovers. *Autonomous Robots*, 9:135–150, 2000.

[11] D. Fox, S. Thrun, F. Dellaert, and W. Burgard. Particle filters for mobile robot localization. In A. Doucet, N. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*. Springer Verlag, New York, 2000.

[12] Rémy Guyonneau. *Méthodes ensemblistes pour la localisation en robotique mobile*. PhD thesis, Université d'angers, 2013.

[13] J. Hwangbo, Y. Chen, and R. Li. Integration of orbital and ground image networks for the automation of rover localization. In *ASPRS Annual Conference, San Diego, CA (USA)*, April 2010.

[14] L. Jaulin. Nonlinear bounded-error state estimation of continuous-time systems. *Automatica*, 38:1079–1082, 2002.

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

[16] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis*. Springer, 2001.

[17] L. Jaulin, M. Kieffer, E. Walter, and D. Meizel. Guaranteed robust nonlinear estimation with application to robot localization. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, 32(4):374–381, November 2002.

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

[19] L. Jaulin, Eric Walter, and Olivier Didrit. Guarateed Robust nonlinear parameter Bounding.

[20] Luc Jaulin and Benoît Desrochers. Introduction to the algebra of separators with application to path planning. *Engineering Applications of Artificial Intelligence*, 33(0):141 – 147, 2014.

[21] S. Le Digabel. Algorithm 909: NOMAD: Nonlinear optimization with the MADS algorithm. *ACM Transactions on Mathematical Software*, 37(4):1–15, 2011.

[22] Jesse Levinson, Michael Montemerlo, and Sebastian Thrun. Map-Based Precision Vehicle Localization in Urban Environments. In *Robotics: Science and Systems*, 2007.

[23] Srikumar Ramalingam, Sofien Bouaziz, Peter Sturm, and Matthew Brand. SKYLINE2GPS: Localization in Urban Canyons using Omni-Skylines. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan*, 2010.

[24] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[25] Aashish Sheshadri, Kevin Peterson, Heather Jones, and William (Red) L. Whittaker. Position estimation by registration to planetary terrain. In *International Conference on Multisensor Fusion and Information Integration (MFI)*. IEEE, September 2012.

[26] Jan Sliwka. *Using set membership methods for robust underwater robot localization*. PhD thesis, Ensta Bretagne, 2011.

[27] F. Stein and G. Medioni. Map-based localization using the panoramic horizon. *Robotics and Automation, IEEE Transactions on*, 11(6):892 –896, dec 1995.

[28] Ernesto Tapia. A note on the computation of high-dimensional integral images. *Pattern Recognition Letters*, 32(2):197 – 201, 2011.

[29] Nikolas Trawny, Anastasios I. Mourikis, Stergios I. Roumeliotis, Andrew E. Johnson, and James Montgomery. Vision-aided inertial navigation for pin-point landing using observations for mapped landmarks. *Journal of Fields Robotics*, 5:357 – 378, 2006.

[30] N Vandapel, R R Donamukkala, and M Hebert. Unmanned ground vehicle navigation using aerial ladar data. *The International Journal of Robotics Research*, 25(1):31–51, 2006.