Interval Analysis for Kidnapping Problem using Range Sensors

Rémy GUYONNEAU, Sébastien LAGRANGE
and Laurent HARDOUIN
Laboratoire d'Ingénierie des Systèmes
Automatisés (LISA),
Université d'Angers,
Angers, France.
{remy.guyonneau, sebastien.lagrange,
laurent.hardouin}@univ-angers.fr

***Abstract*— This paper presents a new method to deal with the kidnapping problem of mobile robots. By using a range sensor and a discrete map of the indoor environment, the robot has to determine its pose (position and orientation). The idea is to obtain the smallest set of feasible poses compatible with the mesurements and the map. This method is a set membership approach based on interval analysis and constraint propagation, which allows to get results in a guaranteed way.**

I. INTRODUCTION

Robot localization is an important issue of mobile robotics, it is a key for many applications [1], [2]. The robotics challenge called CAROTTE[1] shows the matter of this problem. The objectives of that challenge is to build and to control a robot that is able to navigate in an unknown indoor environment and draw a map with semantic annotations (e.g. detected objects). When the map is done, the robot is moved somewhere in the environment (kidnapping) and has to find the exit as fast as possible. This paper is focused on the global localization problem.

The localization problem can be divided into two categories: the pose tracking and the global localization [3], [4]. In the pose tracking problem the robot has to find its new pose using the knowledge of its previous pose. In the global localization problem the robot does not have the knowledge of its previous pose.

Most of the proposed solutions to localize a robot are based on probabilistic estimation techniques [5]. The Extended Kalman Filter or one of its improvement is used for pose tracking [6] and more precisely for the SLAM[2] problem [7]. Particle filters [8] with for example the Monte Carlo algorithm [9] and its spin-off [10] are used to deal with global localization.

In this paper a set membership approach will be considered as considered in [2], [11]. The objective is to find a domain that includes the pose of the robot. We assumed that the sensor yields measurements with bounded error. Hence, the results are obtained in a guaranteed way.

The paper is organized as follows. First we present the considered global localization problem in section II. In section III we present several tools that are used in that paper. Section IV presents the grid map intersection and contraction. Next, in section V we present the localization algorithms. Section VI presents the results we had with a simulator in order to show the efficiency of the proposed method. Finally section VII concludes this paper.

[1]CArtographie par RObоT d'un TErritoire (Robot Land Mapping) organized by the french ANR (National Research Agency).

[2]Simultaneous Localization And Mapping

Fig. 1: The robot used in the CAROTTE challenge.

II. THE GLOBAL LOCALIZATION PROBLEM

This section presents the context of that work and the considered robot which moves in an environment (see II-B). Finally subsection II-C presents the objectives of the works.

A. The Robot

We consider a mobile wheeled robot with a laser telemeter, figure 1. The robot's pose $p_r = (x_r, y_r, \theta_r)$ is defined by its location (x_r, y_r) and its orientation θ_r in the environment. The telemeter provides a set of measurements $\mathbb{D} = \{d_i\}$ with $i = 1, \cdots, n$ that correspond to the distances of the closest obstacles in the direction γ_i, figure 2. As we are in a bounded error context we assume an error measurement ε_d and an angle error ε_γ due to the sensor accuracy, i.e. we associate a guaranteed interval to each measurement $[d_i] = [d_i - \varepsilon_d; d_i + \varepsilon_d]$ and angle $[\gamma_i] = [\gamma_i - \varepsilon_\gamma, \gamma_i + \varepsilon_\gamma]$. Moreover, we assumed that the telemeter has a maximal range denoted d_{max}. We also consider that the robot can have an evaluation of its moving using odometry or algorithms like ICP [12].

B. The Environment

The two dimensional environment $\mathbb{E} \subset \mathbb{R}^2$ where the robot is moving is supposed to be known, figure 3. If not it can be the result of a SLAM process. This environment is discretized with a resolution δ_x, δ_y and this lead to a grid

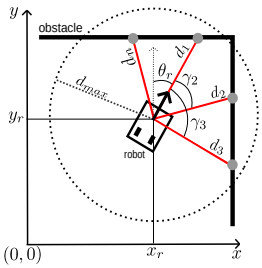

Fig. 2: Sensor measurements $\mathbb{D} = \{d_1, d_2, d_3, d_n\}$.

(a) (b)

Fig. 3: The environment of the CAROTTE challenge.

$\mathbb{G}$ composed of $n \times m$ cells (i, j). At each cell (i, j) is associated $g_{i,j} \in \{0, 1\}$:

$$g_{i,j} = \begin{cases} 1 & \text{if there is an obstacle in the cell } (i,j), \\ 0 & \text{else.} \end{cases}$$

A cell (i, j) represents the following set:

$$\{(x, y) \in \mathbb{E} | i\delta_x \leq x < (i+1)\delta_x, j\delta_y \leq y < (j+1)\delta_y\}.$$

In the sequel $g_{i,j}$ could be abusively denoted $g_{x,y}$ if (x, y) is in the cell (i, j), figure 4.

The figure 5 shows an example of a grid map.

C. The Objective

The aim of that work is to globally localize a robot in a known indoor environment. It is assumed that the robot has been kidnapped somewhere in the environment. Using its telemeter sensor it has to find its pose. This paper proposes a new guaranteed method based on interval analysis to find the smallest $([x_r], [y_r], [\theta_r])$ compatible with the map and sensor measurements.

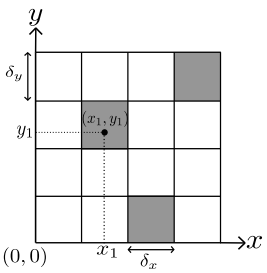

Fig. 4: Illustration of $\mathbb{G}$. The cells such that $g_{i,j} = 1$ are depicted in grey. Since the point (x_1, y_1) is in a grey cell we abusively note $g_{x_1,y_1} = 1$.

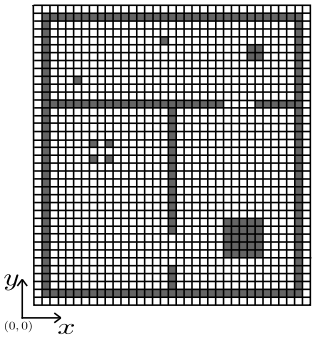

Fig. 5: An example of occupancy grid. A white cell is for a free space ($g_{x,y} = 0$) and a grey cell means there is an obstacle ($g_{x,y} = 1$).

III. Interval Analysis And Constraint Satisfaction Problem

The global localization problem will be viewed as a Constraint Satisfaction Problem (CSP) solved using interval analysis [13], [14], hence in this section we introduce some definitions about interval analysis and CSP.

A. Interval Analysis

An *interval vector*, or a *box* $[\mathbf{p}]$ is defined as a closed subset of $\mathbb{R}^n$:

$$[\mathbf{p}] = ([x], [y], \cdots) = ([\underline{x}, \overline{x}], [\underline{y}, \overline{y}], \cdots) \subset \mathbb{R}^n.$$

Any real number elementary operators such as $+, -, \times, \div$ and functions such as exp, sin, sqr, sqrt, can be easily extended to intervals. For instance:

$$[3, 4.1] \times [0, 1] + [-2, 8] = [0, 4.1] + [-2, 8] = [-2, 12.1],$$
$$\cos([-\tfrac{\pi}{3}, \tfrac{\pi}{4}]) = [\tfrac{1}{2}, \tfrac{\sqrt{2}}{2}].$$

We define the size of a box as $size([\mathbf{p}]) = (\overline{x} - \underline{x}) \times (\overline{y} - \underline{y}) \times \cdots$, i.e. let $[\mathbf{p}] = ([2, 5], [1, 8], [0, 5])$ be a box, $size([\mathbf{p}]) = 105$.

B. Constraint Satisfaction Problem

A CSP is defined by three sets. A set of variables $\mathcal{V}$, a set of domains $\mathcal{D}$ for those variables and a set of constraints $\mathcal{C}$ connecting the variables together:

$$\text{CSP: } \left\{ \begin{array}{c} \mathcal{V} = \{x_1, x_2, \cdots, x_n\} \\ \mathcal{D} = \{[x_1], [x_2], \cdots, [x_n]\} \\ \mathcal{C} = \{c_1, c_2, \cdots, c_m\} \end{array} \right\}. \quad (1)$$

This kind of problem can be solved using *constraint propagation*. Constraint propagation consist in reducing the variable domains by using *contractors* C_{c_i} associated to each constraints c_i.

To illustrate the propagation process, consider the following CSP:

$$\left\{ \begin{array}{c} \mathcal{V} = \{x_1, x_2\} \\ \mathcal{D} = \{[x_1] = [-\infty, \infty], [x_2] = [-\infty, \infty]\} \\ \mathcal{C} = \begin{cases} c_1 : x_2 = {x_1}^2 \\ c_2 : x_1 x_2 = 1 \\ c_3 : x_2 = -2x_1 + 1 \end{cases} \end{array} \right\}. \quad (2)$$

Then we contract the different domains using the constraint's contractors. For example the contractor C_{c_1} of the constraint c_1 is given by Algorithm 1. In this algorithm ∇ corresponds to the stop criterion of the forward-backward process. It can be for example a number of iterations, a contraction threshold or a combination of several parameters. We will use this notation for the other algorithms.

Algorithm 1 C_{c_1}

Require: $[x_1], [x_2]$
while ∇ **do**
  $[x_2] = [x_2] \cap [x_1]^2$
  $[x_1] = [x_1] \cap \sqrt{|[x_2]|}$
end while
$([x_1^*], [x_2^*]) = ([x_1], [x_2])$
return $([x_1^*], [x_2^*])$.

Same kind of contractors can be done for constraints c_2 and c_3. And a solver can be obtained as follows.

Algorithm 2 Solver of CSP (2)

Require: $[x_1], [x_2]$
while ∇ **do**
  $([x_1], [x_2]) = C_{c_1}([x_1], [x_2])$
  $([x_1], [x_2]) = C_{c_2}([x_1], [x_2])$
  $([x_1], [x_2]) = C_{c_3}([x_1], [x_2])$
end while
$([x_1^*], [x_2^*]) = ([x_1], [x_2])$
return $([x_1^*], [x_2^*])$.

And below we give an execution of this algorithm which proves that the solution set of CSP 2 is empty.

$$C_{c_1}([x_1], [x_2]) = ([-\infty, \infty], [0, \infty])$$
$$C_{c_2}([x_1], [x_2]) = ([0, \infty], [0, \infty])$$
$$C_{c_3}([x_1], [x_2]) = ([0, 1/2], [0, 1])$$
$$C_{c_1}([x_1], [x_2]) = ([0, 1/2], [0, 1/4])$$
$$C_{c_2}([x_1], [x_2]) = \emptyset.$$

IV. Map Contractor

This section is focused on the two following problems: Let $\mathbb{G}$ be a discrete map and $([x], [y]) \subset \mathbb{G}$ be a box :

- How to check that the box contains obstacles, i.e. to check that inside $([x], [y])$ it exists $g_{x,y} = 1$.
- How to obtain the smallest box $([x^*], [y^*]) \subset ([x], [y])$ such that $([x^*], [y^*])$ contains the same number of obstacles than $([x], [y])$.

A. Intersection between a box and the grid map

During the localization process the robot will have to check if it exists intersections between the grid map's obstacles and sensor's measurements.

$$\mathbb{G} \cap ([x], [y]) = \begin{cases} \text{true} & \text{if } \exists x \in [x], y \in [y] \\ & \text{such that } g_{x,y} = 1, \\ \text{false} & \text{else.} \end{cases}$$

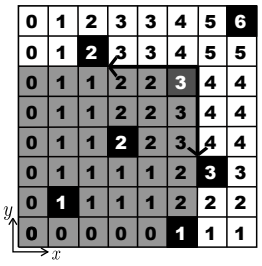

Fig. 6: Output of the pre-calculation algorithm. The black cells represent obstacles ($g_{x,y} = 1$). Integer $\eta_{x,y}$ in each cell represents the number of dark cells in the South-West quarter.

1) Pre-Calculation: In order to allow fast intersection tests we add into the map some pre-computation (see algorithm 3).

The idea is to record the number of obstacles in the South-West quarter of each cell (figure 6).

Algorithm 3 Grid Map's Pre-Calculation

Require: $\mathbb{G} = \{g_{0,0}, \cdots, g_{n,m}\}$
$\eta_{0,0} = g_{0,0}$
for $i = 0$ **to** n **do**
  for $j = 0$ **to** m **do**
    if $i > 0$ **and** $j > 0$ **then**
      $\eta_{i,j} = \eta_{i,j-1} - \eta_{i-1,j-1} + \eta_{i-1,j} + g_{i,j}$
    else
      if $j = 0$ **and** $i > 0$ **then**
        $\eta_{i,j} = \eta_{i-1,j} + g_{i,j}$
      else
        if $i = 0$ **and** $j > 0$ **then**
          $\eta_{i,j} = \eta_{i,j-1} + g_{i,j}$
        end if
      end if
    end if
  end for
end for
return $\{\eta_{i,j}\}$

The output of algorithm 3 is the grid map $\mathbb{G}$ with $\eta_{i,j} (i = 0, \cdots, n$ and $i = 0, \cdots, m)$ representing the number of obstacles in the South-West quarter of the cell (i, j).

2) The Function Φ: Now the pre-calculated coefficient $\eta_{i,j}$ will be used to test intersection. The function Φ returns the number of cells with $g_{i,j} = 1$ that intersect an interval vector $([x], [y])$.

$$\Phi([x], [y]) = \eta_{\overline{x},\overline{y}} + \eta_{\underline{x}-\delta_x,\underline{y}-\delta_y} - \eta_{\underline{x}-\delta_x,\overline{y}} - \eta_{\overline{x},\underline{y}-\delta_y}. \quad (3)$$

Then we have

$$\mathbb{G} \cap ([x], [y]) = \begin{cases} \text{true} & \text{if } \Phi([x], [y]) > 0, \\ \text{false} & \text{else.} \end{cases}.$$

For instance, in the example figure 7:

$\mathbb{G} \cap ([x], [y])$ =true because $\Phi([x], [y]) = 2 (> 0)$.

B. The Map Contractor

The second problem is the following: given an interval domain $([x], [y])$ and a grid $\mathbb{G} = \{g_{i,j}\}$ we define the map contractor $C_{\mathbb{G}}$ such that:

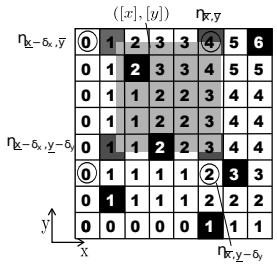

Fig. 7: An example of intersection. The light grey interval vector corresponds to $([x],[y])$, the vector we want to test, $\Phi([x],[y]) = 4+0-0-2 = 2$.

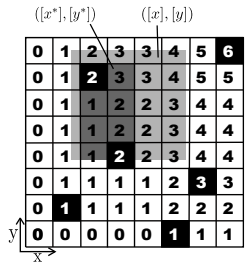

Fig. 8: An example of contraction. The light grey interval vector corresponds to $([x],[y])$, the vector we want to contract and the dark grey interval vector corresponds to $([x^*],[y^*])$ the result of the contraction: $C_{\mathbb{G}}([x],[y]) = ([x^*],[y^*])$.

$C_{\mathbb{G}}([x],[y]) = ([x^*],[y^*])$ is the smallest box that contains the same number of obstacles as $([x],[y])$ (i.e $\Phi([x],[y]) = \Phi([x^*],[y^*])$).

Figure 8 shows an example of map contraction.

Algorithm 4 is the algorithm that contracts $([x],[y])$ over the grid. It performs a binary search in order to find the upper and lower bounds $\underline{x}^*, \overline{x}^*, \underline{y}^*, \overline{y}^*$ that do not changed the value of $\Phi([x],[y])$. It has four identical steps but only the $\overline{y}^*$ step is detailed here.

V. The Localization Algorithm

In this section we present the global localization algorithm. First we describe the different steps of the static localization algorithm, then we detail each step more precisely. We finish by presenting the complete localization algorithm.

A. The Static Localization Algorithm

Let $[\mathbf{p}_r] = ([x_r],[y_r],[\theta_r])$ be an initial domain that enclose the robot pose (x_r, y_r, θ_r) and $d_i, \gamma_i (i = 1, \cdots, n)$ a set of n telemeter measurements.

$\begin{pmatrix} d_i sin(\gamma_i) \\ d_i cos(\gamma_i) \end{pmatrix}$ represents a vector that point out an obstacle (see figure 9).

Hence, the coordinates of an obstacle in the map (w_{i_x}, w_{i_y}) is defined by:

$$w_{i_x} = d_i cos(\theta_r) sin(\gamma_i) + d_i sin(\theta_r) cos(\gamma_i) + x_r, \quad (4)$$

$$w_{i_y} = -d_i sin(\theta_r) sin(\gamma_i) + d_i cos(\theta_r) cos(\gamma_i) + y_r. \quad (5)$$

Algorithm 4 $C_{\mathbb{G}}$

Require: $([x],[y])$ and $\mathbb{G}$
if $\Phi([x],[y]) = 0$ **then**
  $([x^*],[y^*]) = \emptyset$
else
  //contraction of $\overline{y}$
  $y_{mid} = \overline{y}$, $y_{inf} = \underline{y}$, $\overline{y}^* = \overline{y}$
  while $y_{mid} - y_{inf} > 2$ **do**
    $y_{mid} = \lceil (y_{mid} + y_{inf})/2 \rceil$
    if $\Phi([x],[\underline{y}, y_{mid}]) \neq \Phi([x],[y])$ **then**
      $y_{inf} = y_{mid}$, $y_{mid} = \overline{y}^*$
    else
      $\overline{y}^* = y_{mid}$
    end if
  end while
  if $\Phi([x],[\underline{y}, y_{inf}]) = \Phi([x],[y])$ **then**
    $\overline{y}^* = y_{inf}$
  else
    if $\Phi([x],[\underline{y}, y_{inf} + \delta_y]) = \Phi([x],[y])$ **then**
      $\overline{y}^* = y_{inf} + \delta_y$
    else
      $\overline{y}^* = y_{mid}$
    end if
  end if
  //the same for $\underline{y}$, $\underline{x}$ and $\overline{x}$
end if
return $([x^*],[y^*]) = ([\underline{x}^*, \overline{x}^*], [\underline{y}^*, \overline{y}^*])$

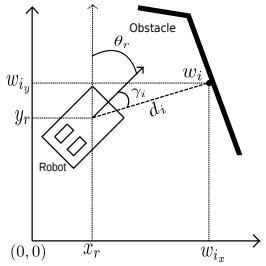

Fig. 9: The robot doing a measurement d_i.

Moreover, note that we have

$$d_i^2 = (x_r - w_{i_x})^2 + (y_r - w_{i_y})^2. \quad (6)$$

Thus, the following CSP can be considered :

$\mathcal{V} = \{x_r, y_r, \theta_r, d_i, \gamma_i, w_{i_x}, w_{i_y}\}$,
$\mathcal{D} = \{$
$[x_r] = [-\infty, +\infty], [y_r] = [-\infty, +\infty], [\theta_r] = [0, 2\pi]$,
$[w_{i_x}] = [-\infty, +\infty], [w_{i_y}] = [-\infty, +\infty]$,
$[d_i]$ =obtained from the sensor,
$[\gamma_i]$ =obtained from the sensor$\}$.

$$\mathcal{C} = \begin{cases} c_{w_{i_x}} : d_i \cos(\theta_r)\sin(\gamma_i) + d_i \sin(\theta_r)\cos(\gamma_i) + x_r \\ c_{w_{i_y}} : -d_i \sin(\theta_r)\sin(\gamma_i) + d_i \cos(\theta_r)\cos(\gamma_i) + y_r \\ c_{d_i} : d_i^2 = (x_r - w_{i_x})^2 + (y_r - w_{i_y})^2 \end{cases}$$

In order to deal with this CSP three contractors are considered : $C_{w_{i_x}}$ Algorithm 5, $C_{w_{i_y}}$ Algorithm 6 and C_{d_i} Algorithm 7. Since variables occur only once in equation 6 the contraction process converges in one step [14] for C_{d_i}.

Furthermore, as we have defined earlier, we have a map contractor $C_{\mathbb{G}}$.

Algorithm 5 $C_{w_{i_x}}$

Require: $[x]$, $[\theta]$, $[w_x]$, $[d_i]$, $[\gamma_i]$
1: //Initialisation
2: $[a] = cos([\theta])$, $[b] = [d_i]sin[\gamma_i] \times [a]$
3: $[c] = sin([\theta])$, $[d] = [c] \times [d_i]cos[\gamma_i]$
4: $[e] = [b] + [d]$, $[f] = [e] + [x]$
5: **while** ∇ **do**
6: //Forward
7: $[a] = [a] \cap [b]/([d_i]sin[\gamma_i])$
8: $[\theta] = [\theta] \cap cos^{-1}([a])$
9: $[c] = [c] \cap [d]/([d_i]cos[\gamma_i])$
10: $[\theta] = [\theta] \cap sin^{-1}([c])$
11: $[f] = [f] \cap [w_x]$
12: $[e] = [e] \cap [f] - [x]$
13: $[x] = [x] \cap [f] - [e]$
14: $[b] = [b] \cap [e] - [d]$
15: $[d] = [d] \cap [e] - [b]$
16: //Backward
17: $[a] = [a] \cap cos([\theta])$
18: $[b] = [b] \cap [d_i]sin[\gamma_i] \times [a]$
19: $[c] = [c] \cap sin([\theta])$
20: $[d] = [d] \cap [c] \times [d_i]cos[\gamma_i]$
21: $[e] = [e] \cap [b] + [d]$
22: $[f] = [f] \cap [e] + [x]$
23: $[w_x] = [w_x] \cap [f]$
24: **end while**
25: $([x^*], [\theta^*], [w_x^*]) = ([x], [\theta], [w_x])$
26: **return** $([x^*], [\theta^*], [w_x^*])$

These contractors are called to solve the CSP as presented in Algorithm 8 of which output is a set of boxes compatible with the map and the measurements. In this algorithm ξ is the stop criteron that corresponds to the precision of the boxes.

B. The Dynamic Algorithm

The static algorithm can be not sufficient to get an accurate estimation of the robot's pose due to the symmetries of the environment (the output of Δ_{static} is a set of poses). To deal with this problem the robot has to move, to get more information about the environment. In this point we assume that the robot has a guaranteed knowledge of its moving. Which means that if at the time t the robot has a estimation of its pose $[\mathbf{p}_t]$, at $t+1$ it can process $[\mathbf{p}_{t+1}]$. If $(x_r, y_r, \theta_r) \notin [\mathbf{p}_{t+1}]$ we consider that the robot has been kidnapped so it has to start a new global localization.

Due to the lack of place, the algorithm is not given but the general idea is the following. First the robot has no knowledge of its pose *i.e.* $[\mathbf{p}]$ is initialized with $([-\infty, +\infty], [-\infty, +\infty], [0, 2\pi])$. The robot gets measurements and processes a first static localization. As a result, it gets a set of feasible poses. Then, the robot moves, and for each feasible poses it calculates (thanks to a bounded odometry[3]) the new pose compatible with its moving. Then, the robot gets new measurements and can process a static localization on each poses using the new data. The robot moves again and so on until it is considered localized or kidnapped.

[3]That means the odometry is known in a bounded way. For instance the robot moves of $m \in [m]$ meters in the direction $\theta_d \in [\theta_d]$. The interval $[m]$ and $[\theta_d]$ are obtained via odometry sensors.

Algorithm 6 $C_{w_{i_y}}$

Require: $[y]$, $[\theta]$, $[w_y]$, $[d_i]$, $[\gamma_i]$
1: (Initialisation)
2: $[a] = sin([\theta])$, $[b] = [d_i]sin[\gamma_i] \times [a]$
3: $[c] = cos([\theta])$, $[d] = [c] \times [d_i]cos[\gamma_i]$
4: $[e] = -[b] + [d]$, $[f] = [e] + [y]$
5: **while** ∇ **do**
6: (Forward)
7: $[\theta] = [\theta] \cap sin^{-1}([a])$
8: $[a] = [a] \cap [b]/([d_i]sin[\gamma_i])$
9: $[c] = [c] \cap [d]/([d_i]cos[\gamma_i])$
10: $[\theta] = [\theta] \cap cos^{-1}([c])$
11: $[e] = [e] \cap [f] - [y]$
12: $[y] = [y] \cap [f] - [e]$
13: $[b] = [b] \cap -[e] + [d]$
14: $[d] = [d] \cap [e] + [b]$
15: $[f] = [f] \cap [w_y]$
16: (Backward)
17: $[a] = [a] \cap sin([\theta])$
18: $[b] = [b] \cap [d_i]sin[\gamma_i] \times [a]$
19: $[c] = [c] \cap cos([\theta])$
20: $[d] = [d] \cap [c] \times [d_i]sin[\gamma_i]$
21: $[e] = [e] \cap -[b] + [d]$
22: $[f] = [f] \cap [e] + [y]$
23: $[w_y] = [w_y] \cap [f]$
24: **end while**
25: $([y^*], [\theta^*], [w_y^*]) = ([y], [\theta], [w_y])$
26: **return** $([y^*], [\theta^*], [w_y^*])$

Algorithm 7 C_{d_i}

Require: $[x_r], [y_r], [w_x], [w_y], [d_i]$
1: (Initialisation)
2: $[a] = [x_r] - [w_x]$, $[b] = [y_r] - [w_y]$
3: $[d_{i_a}] = [d_i]^2 - [b]^2$
4: $[d_{i_b}] = [d_i]^2 - [a]^2$
5: (Contraction in one step)
6: $[a^*] = [a] \cap |\sqrt{[d_{i_a}]}|$
7: $[b^*] = [b] \cap |\sqrt{[d_{i_b}]}|$
8: $[{w_y}^*] = [w_y] \cap [y_r] - [b^*]$
9: $[{y_r}^*] = [y_r] \cap [b^*] + [{w_y}^*]$
10: $[{w_x}^*] = [w_x] \cap [x_r] - [a^*]$
11: $[{x_r}^*] = [x_r] \cap [a^*] + [{w_x}^*]$
12: **return** $[{x_r}^*], [{y_r}^*], [{w_x}^*], [{w_y}^*]$

VI. Results

In this section we will present the simulation and the results obtained.

In order to test this approach, a C++ simulator has been developed. This simulator is available at *http://www.istia.univ-angers.fr/∼lagrange/* as well as a video. The considered map is an occupancy grid with 1000×1000 cells corresponding to a 10×10 meters environment which corresponds to the size of the CAROTTE challenge environment, figure **??**. The precision of the grid is 1×1 centimetre. The sensor of the robot does 32 measurements. Each measurements has a bounded error $\varepsilon = 10$ centimetres, which corresponds to a combination of ε_d and ε_γ. The maximal distance of

Algorithm 8 Δ_{static}, The Static Algorithm

Require: $\mathbb{G}, [x_r], [y_r], [\theta_r], [d_i], [\gamma_i] (i = 1, \cdots, n)$
$\mathcal{L}$.push$[([x_r], [y_r], [\theta_r])]$
while $\mathcal{L}$ is not empty **do**
  $([x], [y], [\theta]) = \mathcal{L}$.pop()
  while ∇ **do**
    for $i = 1$ **to** n **do**
      $([x], [\theta], [w_{i_x}]) = C_{w_{i_x}}([x], [\theta], [w_{i_x}], [d_i], [\gamma_i])$
      $([y], [\theta], [w_{i_y}]) = C_{w_{i_y}}([y], [\theta], [w_{i_y}], [d_i], [\gamma_i])$
      $([w_{i_x}], [w_{i_y}]) = C_{\mathbb{G}}([w_{i_x}], [w_{i_y}])$
      $([x], [y], [w_{i_x}], [w_{i_y}]) = C_{d_i}([x], [y], [w_{i_x}], [w_{i_y}])$
    end for
  end while
  if $size([x], [y], [\theta]) > \xi$ **then**
    bissect$([x], [y], [\theta])$ in two boxes:
    $([x_1], [y_1], [\theta_1])$ and $([x_2], [y_2], [\theta_2])$
    $\mathcal{L}$.push$[([x_1], [y_1], [\theta_1])]$
    $\mathcal{L}$.push$[([x_2], [y_2], [\theta_2])]$
  else
    $\mathcal{O}$.push$[([x], [y], [\theta])]$
  end if
end while
return $\mathcal{O}$

the sensor is $d_{max} = 1$ meter, . The error of the moving estimation (rotation and translation) of the robot is 10% of the moving. To process the static contraction the robot does not uses the measurements that are superior to the maximal range of the sensor (no obstacle on the sensor's ray). We also consider 2 outliers in the data set of the sensor. An outlier is defined as a measure $d_i < d_{max}$ even if there is no corresponding obstacle in the map. To deal with these values, we use the q-relaxed intersection [15].

Figure 10 shows the environment and the simulated position of the robot and the blue boxes show the set of possible locations of the robot in the map after the first contraction. After two moves the robot location is given with only one box with the following size $27cm$ for x_r, $23cm$ for the y_r and 7° for θ_r. The computational time of the global localization is less than two minutes.

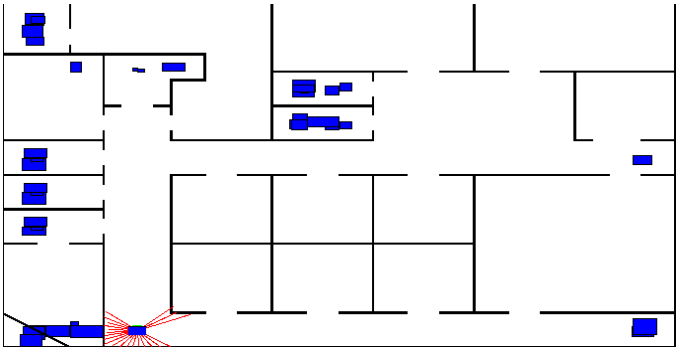

Fig. 10: The environment of the simulation, the red rays represent the sensor's measurements. The blue boxes represent the location obtained after the first call of Δ_{static} algorithm, $i.e.$ the set of compatible boxes. The computational time is 50 seconds.

VII. Conclusion

In this paper we have shown that interval analysis could be used to solve the kidnapping problem. This method uses a discrete map so the time processing depends of the size of the grid. But in fact this size influence only the pre-computation process. The time processing of the localization algorithm depends mainly of the number of measurements and of the accuracy of the range sensor.

Instead of a probabilistic approach, interval analysis yields a robust estimations and allows to deal with non linear estimation.

The simulation results are promising and this method can be efficient in a real context as it will be tested during the next edition of the CAROTTE challenge (June 2011) `http://www.defi-carotte.fr/index.php`.

References

[1] D. Caltabiano, G. Muscato, and F. Russo, "Localization and self-calibration of a robot for volcano exploration," in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, 2004.

[2] L. Jaulin, "A nonlinear set-membership approach for the localization and map building of underwater robots," *IEEE Transactions on Robotics*, vol. 25, pp. 88–98, 2009.

[3] M. Tomono, "Robust robot localization and map building using a global scan matching method," in *International Conference on Intelligent Robots and Systems (IROS)*, 2004.

[4] X. Zezhong, L. Ronghua, and L. Jilin, "Global localization based on corner point," in *Proceedings on the International Computational Intelligence in Robotics and Automation (ICRA)*, vol. 2, 2003, pp. 843–847.

[5] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*, ser. Intelligent robotics and autonomous agents. MIT Press, 2005.

[6] J. Leonard and H. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 376 –382, 1991.

[7] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam: A factored solution to the simultaneous localization and mapping problem," in *In Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2002, pp. 593–598.

[8] G. Cen, N. Matsuhira, J. Hirokawa, H. Ogawa, and I. Hagiwara, "Mobile robot global localization using particle filters," in *International Conference on Control, Automation and Systems (ICCAS)*, 2008, pp. 710–713.

[9] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo localization for mobile robots," *Proceedings 1999 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 128, no. February, pp. 1322–1328, 1999.

[10] L. Zhang, R. Zapata, and P. Lépinay, "Self-adaptive Monte Carlo localization for mobile robots using range sensors," in *Proceedings of the International Conference on Intelligent robots and systems (IROS)*, 2009, pp. 1541–1546.

[11] 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*, vol. 32, no. 4, pp. 374–382, 2002.

[12] C. Lara, L. Romero, and F. Calderón, "A robust iterative closest point algorithm with augmented features," in *Proceedings of the 7th Mexican International Conference on Artificial Intelligence: Advances in Artificial Intelligence*, ser. MICAI '08. Springer-Verlag, 2008, pp. 605–614.

[13] R. E. Moore, *Interval analysis*. Prentice-Hall, 1966.

[14] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, *Applied Interval Analysis*, 1st ed. Springer, Sept. 2001.

[15] L. Jaulin, "Robust set membership state estimation ; application to underwater robotics," *Automatica*, 2008.