Hindawi
Journal of Sensors
Volume 2018, Article ID 3769058, 12 pages
https://doi.org/10.1155/2018/3769058

Research Article

Localization of a Vehicle: A Dynamic Interval Constraint Satisfaction Problem-Based Approach

Kangni Kueviakoe,[1] Zhan Wang,[2] Alain Lambert,[2] Emmanuelle Frenoux,[1] and Philippe Tarroux[1]

[1]*LIMSI, CNRS, Univ. Paris-Sud, Université Paris-Saclay, 91403 Orsay, France*
[2]*LRI, CNRS, Univ. Paris-Sud, Université Paris-Saclay, 91403 Orsay, France*

Correspondence should be addressed to Alain Lambert; alain.lambert@u-psud.fr

Received 13 September 2017; Revised 23 January 2018; Accepted 8 February 2018; Published 10 April 2018

Academic Editor: Oleg Lupan

This paper introduces a new interval constraint propagation (ICP) approach dealing with the real-time vehicle localization problem. Bayesian methods like extended Kalman filter (EKF) are classically used to achieve vehicle localization. ICP is an alternative which provides guaranteed localization results rather than probabilities. Our approach assumes that all models and measurement errors are bounded within known limits without any other hypotheses on the probability distribution. The proposed algorithm uses a low-level consistency algorithm and has been validated with an outdoor vehicle equipped with a GPS receiver, a gyro, and odometers. Results have been compared to EKF and other ICP methods such as hull consistency (HC4) and 3-bound (3B) algorithms. Both consistencies of EKF and our algorithm have been experimentally studied.

1. Introduction

This paper deals with outdoor vehicle ego-localization. Localization is part of many automotive applications where safety is of crucial importance. This estimation problem, which involves fusion of data from dead reckoning and absolute sensors, is generally solved by Bayesian approaches [1, 2]. Interval analysis-based methods are alternative and less known methods. These methods use either set inversion [3, 4] or constraint propagation techniques [5, 6] to deal with the localization problem and provide a set of intervals containing the vehicle pose (position and orientation) rather than probabilities.

Most of the early published papers on constraint programming can be dated back to the 1970s. They have mainly dealt with discrete domains [7]. In the 1980s, Gallaire [8] and Jaffar and Lassez [9] noted that logic programming can be seen as a particular type of constraint programming. Both logic programming and constraint programming imply that the user states what has to be solved instead of how to solve it. Many combinatorial problems such as scheduling and timetabling were then formulated in the form of constraint satisfaction problems (CSP).

Constraint propagation is the most used technique for solving CSPs. It combines consistent techniques and systematic search methods. Arc consistency could be achieved by the AC-3 ("AC" stands for "arc consistency") algorithm [7] or by many other algorithms (like AC-5 [10], AC-6 [11], etc.) using only binary CSPs (each constraint involves at most two variables). However, many real-life problems can be naturally modeled as nonbinary CSPs: the constraint involves more than two variables [12]. Two approaches were developed to deal with nonbinary CSPs. The first one translates a nonbinary CSP into several binary CSPs [13, 14]. The second kind of approach directly deals with nonbinary constraints. For example, GAC-4 [15] is a generalization of AC-4 to nonbinary constraints.

In 1987, Cleary [16] and Davis [17] pioneered the use of constraint propagation for interval analysis. In 1989, a generalized constraint propagation scheme based on interval arithmetic was designed by Hyvönen [18]. Cleary [16] proposed a narrowing algorithm which was improved by Benhamou and

Older [19]. Later, that algorithm was renamed as HC3 [20] because it is very close to the classical AC-3 algorithm. Next, Benhamou et al. proposed that HC4 [20] does not need constraints decomposition but it encounters the well-known dependency problem. To cope with this problem, BC3 [21] was developed. It overcomes the dependency problem at the cost of a huge computing time. BC4 [20] merges both BC3 and HC4 and reduces the computation time. BC5 [22], by using the interval Gauss-Seidel method, further reduces the computing time. All the preceding methods (HC4, BC3, BC4, and BC5) work with one constraint at a time. They compute the variables of a constraint locally without taking into account the whole constraint system. They are then named "local consistency algorithms." By then, they do not reach the optimal contraction for all the domains of variables of the problem. This leads to a problem called "locality problem." To take into account all the constraints of a given problem at a time, strong consistency algorithms such as the 3B algorithm [23] have been proposed. Strong consistency algorithms seek the strength of the domain contractions rather than the shortness of computation time.

Interval constraint propagation (ICP) was used to solve the robotic localization problems by Gning and Bonnifait [6, 24] more than ten years ago. It was mainly used for outdoor vehicle localization [25, 26] and underwater robot localization [3, 5]. Those works use a forward-backward propagation technique based on primitive constraints (following the principle of the Waltz algorithm [27]). Drevelle and Bonnifait [28] use a relaxed constraint propagation approach to deal with erroneous GPS measurements. Lassoued and Bonnifait [29] combine constraint propagation and set inversion techniques and present a cooperative localization method with significant enhancement in terms of accuracy and confidence domains. Some other researchers propose specific contractors [30] or separators [31] to solve the localization problem. Tubes are used in scenarios where the system is described by a continuous time state equation. This tool allows to both localize the robot and correct its clock. The readers might refer to [32, 33] to see more details about the application of tubes in the robotic area. The main advantage of ICP over Bayesian algorithms is that it guarantees that the position of the robot is contained in a box. Bayesian algorithms [1] can only associate a probability to such a box. Consequently, safety cannot be guaranteed by Bayesian algorithms.

A previous work [34] demonstrated that classical forward-backward ICP do not reduce the heading imprecision for a vehicle equipped with a GPS receiver, a gyro, and odometers. Another study [35] concluded that various ICP algorithms ([16, 20, 21, 23]) have the same drawback. Only the 3B consistency algorithm [23] (based on a strong consistency technique) can reduce the heading imprecision of the vehicle during its displacement. The price to pay is an increase in the computing time.

We improved previous works by proposing a new ICP method for vehicle localization with heading correction and real-time capabilities. The proposed algorithm employs HC4 as a low-level algorithm. It is easier to be used than previous ones as it does not need to decompose all the constraints into elementary/binary constraints. Our algorithm has been validated with an outdoor vehicle equipped with a GPS receiver, a gyro, and odometers. The results obtained by our method are compared with three algorithms by using a very accurate positioning system as ground truth. This paper includes and extends results previously presented in [36].

Section 2 introduces constraint propagation and interval analysis. Two ICP algorithms (HC4 and 3B) are discussed in Section 3. A new localization problem formalization and its solving are presented in Section 4 while Section 5 shows our experimental results.

2. Overview of Interval Analysis and Constraint Propagation

Interval analysis and constraint propagation are the two main mathematical concepts on which our work is based. A brief overview of these methods is given in this section.

2.1. Interval Analysis. Interval analysis was introduced in the 1960s by Moore [37–39] in order to deal with approximation problems encountered during calculation. The main idea is to represent numbers by intervals which include their values. An interval $[x]$ is a connected subset of $\mathbb{R}$, defined by its lower bound $\underline{x}$ and upper bound $\bar{x}$. The following notations are used as shorthands to denote closed/open interval: $[\underline{x}, \bar{x}] = \{x \in \mathbb{R}|_{\underline{x}} \leq x \leq \bar{x}\}$ and $[\underline{x}, \bar{x}) = \{x \in \mathbb{R}|_{\underline{x}} \leq x < \bar{x}\}$. When no confusion may arise, $[x]$ is used to represent an interval whether it is closed or not. The width of a nonempty interval $[x]$ is defined by $w([x]) = \bar{x} - \underline{x}$.

Rules have been defined to perform every usual arithmetical operation on intervals [38, 39]. Considering two intervals $[x]$ and $[y]$ and a binary operator $\diamond \in \{+, -, \times, \div\}$, the smallest interval which contains all feasible values for $[x] \diamond [y]$ is defined as

$$[x] \diamond [y] = \left\{ x \diamond y|_{x \in [x], y \in [y]} \right\}. \quad (1)$$

The set-theoretic operations can be applied to intervals. The *intersection* of two intervals is defined by

$$[x] \cap [y] = \left\{ z \in \mathbb{R}|_{z \in [x], z \in [y]} \right\}, \quad (2)$$

which is always an interval. This is not the case for their *union*

$$[x] \cup [y] = \left\{ z \in \mathbb{R}|_{z \in [x] \text{ or } z \in [y]} \right\}. \quad (3)$$

Consequently, an *interval hull* is defined as the smallest interval that contains a subset $\mathbb{X}$ of $\mathbb{R}$. For example, the interval hull of $[2, 4] \cup [5, 8]$ is the interval $[2, 8]$. The *interval hull* of the *union* of $[x]$ and $[y]$ is denoted by $[x] \sqcup [y]$.

2.2. Interval Extension. We often need to compute the image of a number by a function. One purpose of interval analysis is to provide, for a large variety of functions f, interval extensions $[f]$ (also called "inclusion function") that can be evaluated reasonably quickly and such that $[f]([x])$ is not too large.

For any function f defined with arithmetical operators and elementary functions,

$$f([x]) \subset [f]([x]), \quad \forall [x] \in \mathbb{R}. \tag{4}$$

To create an inclusion function, it is possible to replace all the variables by their intervals and then all the included operators by their interval equivalents. The resulting interval function is called a "natural inclusion function." For instance,

$$f(x) = 4x^4 - 7\ln(x) - 3, \quad \forall x \in \mathbb{R} \tag{5}$$

has the natural inclusion function

$$[f]([x]) = 4[x]^4 - 7\ln([x]) - 3, \quad \forall [x] \in \mathbb{R}. \tag{6}$$

An interval extension can be used to represent an equation based on interval variables. Such equations, also called constraints, are the core of an interval-based constraint satisfaction problem which is covered by the next subsection.

2.3. Contractor. The concept of the contractor is directly inspired by the ubiquitous concept of filtering algorithms in constraint programming. Given a constraint c relating a set of variable x, an algorithm $\mathcal{C}$ is called a contractor (or filtering algorithm) if $\mathcal{C}$ returns a subdomain of the input domain $[x]$ and the resulting subdomain $\mathcal{C}([x])$ contains all the feasible points with respect to the constraint c:

$$\begin{gathered} \mathcal{C}([x]) \subseteq [x], \\ c(x) \Rightarrow x \in \mathcal{C}([x]), \quad \forall x \in [x]. \end{gathered} \tag{7}$$

2.4. Interval Constraint Satisfaction Problem (ICSP). The concept of interval constraint satisfaction problem (ICSP) was introduced by Hyvönen [40] in 1992. An interval constraint satisfaction problem (ICSP) or interval CSP is a mathematical problem solved by finding states satisfying the constraints. An ICSP is defined by

(i) a set V of variables $\{v_1, v_2, \ldots, v_n\}$,

(ii) a set D of domains $\{D_1, D_2, \ldots, D_n\}$, such as for each variable v_i, a domain D_i with the possible values for that variable is given and D_i is an interval or union of intervals;

(iii) a set C of p constraints $\{c_1, c_2, \ldots, c_p\}$, defining the relations between a subset of variables, restricting the possible solutions.

Hyvönen et al. proposed, in 1993, "INC++" [41], a basic interval processing tool which allows evaluating interval functions and aims at solving ICSP. Many other interval solvers such as Numerica [42], RealPaver [43], ALIAS [44], ICOS [45], GloptLab [46], and IBEX (IBEX is available at http://www.ibex-lib.org/) [47, 48] have been designed so far.

Interval constraint propagation (ICP) consists in iterating domain reductions, by using the set of p constraints, until no domain can be contracted. Many kinds of CSP have been introduced in order to express specific situations. The concept of "dynamic CSP" (DCSP) was introduced by Dechter [49]. It represents a CSP that is evolving or changing with the time. This means that at any time, new constraints can be added to the constraint network and other constraints can be removed from the CSP. The addition (resp. deletion) operation is also known as "restriction" (resp. "relaxation"). An interval CSP which is dynamic is consequently a "dynamic ICSP" or DICSP for short.

2.5. Interval Vector. Interval vector is a generalization of the interval's concept: an interval vector, also called "a box," has intervals as components. It is a Cartesian product of n intervals. The width of an interval vector $w([\mathbf{x}]) = \max_{i=1,\ldots,n} w([x_i])$ can be computed in order to estimate its size.

A vehicle pose is represented by 3 parameters: x-axis and y-axis values and the heading information θ. Consequently, for vehicle localization, the solution is a three-dimensional interval vector. Interval constraint propagation (ICP) algorithms process interval vectors as the variables of an interval constraint satisfaction problem. We use these ICP algorithms to solve the vehicle localization problem which we stated as an ICSP. The next section will present two ICP algorithms.

3. ICP Algorithms

In this section, we introduce two interval constraint propagation (ICP) algorithms. Previous studies [35] have shown that HC4 is the fastest one and 3B is the one that gives the best contraction of domains. We do not introduce BC5 which is slower than HC4 and unnecessary because of no dependency inside our constraints.

3.1. The HC4 Algorithm. HC4 [20] enforces hull consistency over complex constraints without decomposing them into primitive ones. Hull consistency [20] is ensured, for a constraint c, with regard to an interval vector $\mathbf{B}$, if and only if

$$\mathbf{B} = \text{hull}_{\square}(\mathbf{B} \cap \rho_c), \tag{8}$$

where $\text{hull}_{\square}$ is a function computing an approximation of the smallest interval vector containing its argument and ρ_c denotes the underlying relation of the given constraint c. If P is an ICSP involving an interval vector $\mathbf{B}$, P is hull consistent if each constraint c of P is hull consistent. Consequently, $\mathbf{B}$ cannot be reduced anymore by applying c. Hull consistency is also called 2B consistency [50]. The interval vector $\mathbf{B}$ is the Cartesian product of n intervals: $\mathbf{B} = I_1 \times \cdots \times I_n$. HC4 follows a loop propagation and processes constraints individually using the HC4 revise function. HC4 revise reduces the domains by removing inconsistent values and returns the new narrowed domains to HC4.

HC4 revise uses a binary tree representation of constraints, where leaves are constants or variables and nodes represent elementary operation symbols such as /, $\times$, and $\log()$. HC4 revise works in two phases:

(i) The "forward evaluation phase" goes through the tree, from the leaves to the root. It evaluates recursively the interval of subexpression represented by the current node using the natural extensions of the underlying functions.

(ii) The "backward propagation phase" traverses the tree from the root to the leaves applying on each node a contraction operator (a projection). The contraction operator narrows the interval of the current node by removing inconsistent values.

The main limitation of the hull consistency algorithms (HC4 and HC3) is their sensitivity to multiple occurrences of variables [21]. This is referred as the dependency problem. When considering, for instance, the constraint $x \times x + y^2 = 2$, with $(x, y) \in [-2, 4] \times [-1, 1]$, hull consistency algorithms would not reduce the initial box, but would perfectly reduce it when the constraint is stated like $x^2 + y^2 = 2$.

The 3-bound consistency algorithm overcomes the dependency problem and the locality problem as described in Section 1.

3.2. The 3B Algorithm. 3B algorithm [23] also referred as the "strong consistency algorithm" computes the projection of the sets of constraints over the variables: it combines constraints in order to improve the precision of domain narrowing. The principle of 3B is that a variable is instantiated to one of its interval bounds and the remaining problem is solved in a hull or box consistency way: 3B uses a low level algorithm (for instance BC4) in order to contract the intervals. Let P be an ICSP, x a variable of P, and $D_x = [\underline{x}, \overline{x}]$ the domain of x. D_x is 3B consistent if and only if P' and P'' are both nonempty where $P' = \Phi(P \cup \{x = \underline{x}\})$ and $P'' = \Phi(P \cup \{x = \overline{x}\})$ with $\Phi(P)$ denoting the enclosure of P. P is 3B consistent if all the domains of variables are 3B consistent.

Example 1. Let us consider that the constraints of P is given as follows:

$$\begin{aligned} x + y &= 2, \\ y &\leq x + 1, \\ y &\geq x, \end{aligned} \tag{9}$$

with $D_x = [0.5, 1]$ and $D_y = [1, 1.5]$. For each instantiation of x, we have a 2B-consistent subproblem to solve (see Table 1). For each line of Table 1, the solution domain is not empty, which indicates that the domain of variable x is 3B consistent. The same method is used for the variable y and ultimately proves that the ICSP is 3B consistent.
For further information about 3B consistency methods, the reader might refer to [23].

After presenting the mathematical tools required to understand our approach, let us now deal with the localization problem itself.

4. The Localization Problem

4.1. Problem Statement. The localization problem consists of estimating in a continuous way the position and heading information of an autonomous mobile system. Figure 1 depicts a localization situation. The goal is to determine the current pose x_k, y_k, and θ_k knowing the previous pose x_{k-1}, y_{k-1}, and θ_{k-1}. It is a state estimation of a nonlinear dynamic system. Our vehicle is equipped with two proprioceptive sensors (gyro and odometers) and an exteroceptive sensor (GPS receiver). The elementary displacement and rotation ($[\delta s_k]$ and $[\delta\theta_k]$) are given by a static fusion process which involves (see [24, 26] for more details) (10), (11), and (12). Then, they can be directly used in a dynamic fusion process (Algorithm 1) to estimate the vehicle location when a GPS measurement is available.

Table 1: Establishing 3B consistency.

Instantiation	Constraint	Solution	Empty
$P' = \Phi(P \cup \{x = 0.5\})$	$x + y = 2$ $y \leq x + 1$ $y \geq x$ $x = 0.5$	$x = 0.5$ $y = 1.5$	Non
$P'' = \Phi(P \cup \{x = 1\})$	$x + y = 2$ $y \leq x + 1$ $y \geq x$ $x = 0.5$	$x = 1$ $y = 1$	Non

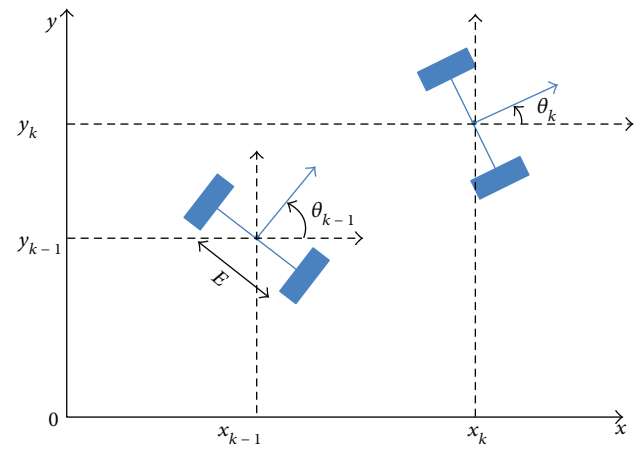

Figure 1: Vehicle localization problem.

4.1.1. Gyro. A gyro is a heading sensor which measures the rotational speed in an inertial reference system. It uses the Coriolis force to output a yaw rate from which we deduce the elementary rotation $\delta\theta_{\text{gyro}}$ and its associated interval $[\delta\theta]$:

$$[\delta\theta] = \left[\delta\theta_{\text{gyro}} - \varepsilon_{\text{gyro}}, \delta\theta_{\text{gyro}} + \varepsilon_{\text{gyro}}\right], \tag{10}$$

where $\varepsilon_{\text{gyro}}$ is the error on $\delta\theta$.

4.1.2. Odometers. Odometers are set on both rear wheels of our vehicle. They give the distance traveled by each wheel independently. The accuracy of an odometer relies on its number of steps and its maximum error. In our case, the maximum error is known to be one step. Thus the value of the displacement of a nonsliding wheel can be bounded by $[\delta p] = [\delta p_{\text{odo}} - 1, \delta p_{\text{odo}} + 1]$ where δp_{odo} stands for the number of measured steps. Then, the movement of a sliding wheel can be deduced from a nonsliding one by adding a

Ensure: $DICSP_k$
(1) $([\theta^1_{k-\Xi}], \dots, [\theta^v_{k-\Xi}]) \leftarrow$ **Split**$([\theta_{k-\Xi}], v)$
(2) **for** $i \leftarrow 1 : v$ **do**
(3) $\quad \text{DICSP}^i_k \leftarrow$ **Build**$(DICSP_k, [\theta^1_{k-\Xi}])$
(4) $\quad [X^i_k] \leftarrow$ **Solve**(DICSP^i_k)
(5) **end for**
(6) $[X_{\text{res}}] \leftarrow$ **Hull**$([X^1_k], \dots, [X^v_k])$
(7) **return** $[X_{\text{res}}]$

Algorithm 1: DICPS.

sliding noise $[\varepsilon_{\text{odo}}]$. Consequently, the displacement considering a sliding wheel is defined by

$$[\delta p] = [\delta p_{\text{odo}} - 1 - \varepsilon_{\text{odo}}, \delta p_{\text{odo}} + 1 + \varepsilon_{\text{odo}}]. \quad (11)$$

The elementary displacement $[\delta s_k]$ of the vehicle is then obtained by

$$[\delta s_k] = \frac{[\pi]([w_{\text{l}}][\delta p_l] + [w_{\text{r}}][\delta p_r])}{P}, \quad (12)$$

where w_{r} and w_{l}, respectively, stand for the radius of the right and left wheel and P represents the odometer resolution.

4.1.3. GPS Receiver. The Global Positioning System (GPS) is a global navigation satellite system that provides geolocation and time information to a GPS receiver anywhere on or near the Earth. When a GPS receiver reads the transmission of three or more satellites, it calculates the arrival time differences and its relative distance to each satellite. Our GPS receiver performs the necessary calculation and returns latitude and longitude coordinates which are converted as a position $[\mathbf{y}] = ([x], [y])^T$ in a Cartesian local frame. Furthermore, the GPS receiver computes the measurement imprecision ε_{gps} on-line and sends it into the GST NMEA frame.

$$[\mathbf{y}] = \begin{pmatrix} [x_{\text{gps}} - \varepsilon_{x,\text{gps}}, x_{\text{gps}} + \varepsilon_{x,\text{gps}}] \\ [y_{\text{gps}} - \varepsilon_{y,\text{gps}}, y_{\text{gps}} + \varepsilon_{y,\text{gps}}] \end{pmatrix}. \quad (13)$$

For further information about sensors, the reader might refer to [51].

Once we have constituted the intervals from sensor data, we can define a nonlinear dynamic-bounded error model to build constraints of a DICSP, as presented in the next section.

4.2. A Dynamic Interval Constraint Satisfaction Problem (DICSP). We propose to define the localization problem as a dynamic interval constraint satisfaction problem: at time k, a new ICSP_k (V_k, D_k, C_k) is generated and defined as follows:

(i) a set of variables, V_k,

$$V_k = \{x_k, y_k, \theta_k, x_{k-1}, y_{k-1}, \theta_{k-1}, \delta s_k, \delta\theta_k\}, \quad (14)$$

(ii) a set of domains, D_k,

$$D_k = \{[x_k, y_k, \theta_k, x_{k-1}, y_{k-1}, \theta_{k-1}, \delta s_k, \delta\theta_k]\}, \quad (15)$$

(iii) a set of constraints, C_k,

$$C_k = \left(C^1_k, C^2_k, C^3_k\right)^T, \quad (16)$$

$$\begin{aligned} C^1_k &\Leftrightarrow [x_k] = [x_{k-1}] + [\delta s_k]\cos\left([\theta_{k-1}] + \frac{[\delta\theta_k]}{2}\right), \\ C^2_k &\Leftrightarrow [y_k] = [y_{k-1}] + [\delta s_k]\sin\left([\theta_{k-1}] + \frac{[\delta\theta_k]}{2}\right), \\ C^3_k &\Leftrightarrow [\theta_k] = [\theta_{k-1}] + [\delta\theta_k], \end{aligned} \quad (17)$$

where C_k represents the set of constraints deduced from the vehicle displacement model proposed by Seignez et al. [52]. The domains of variables are defined as follows:

(i) $[\delta s_k]$ and $[\delta\theta_k]$ are given by the odometer and gyro measurements thanks to (12) and (10).

(ii) $[x_k]$ and $[y_k]$ are initialized with GPS measurements (see (13)).

(iii) $[\theta_k]$ is initialized to $-\pi, \pi]$.

(iv) $[x_{k-1}]$, $[y_{k-1}]$, and $[\theta_{k-1}]$ are obtained from the solution of the previous CSP (at time $k-1$).

We consider all the state equations from time $k - \Xi + 1$ to time k (where k is the current time and Ξ is the window size): the contractions are performed in a sliding window of length Ξ. In other words, we use Ξ sets of equations (current and past information) in our interval constraint satisfaction problem in order to localize the vehicle. At each time k, we add a new constraint set C_k to the existing constraint sets. If $k > \Xi$, we delete the oldest constraint set $C_{k-\Xi}$ from it. The addition (resp. deletion) operation is also known as "restriction" (resp. "relaxation").

The localization DICSP (dynamic ICSP) keeps the same number of constraints $(3 \times \Xi)$ but changes temporally. The localization DICSP is defined at time k by

$$\text{DICSP}_k = \begin{pmatrix} \text{ICSP}_k \\ \text{ICSP}_{k-1} \\ \text{ICSP}_{k-2} \\ \dots \\ \text{ICSP}_{k-\Xi+1} \end{pmatrix}. \quad (18)$$

ICP algorithms allow us to solve DICSP_k. For our experiment, some local consistency algorithms including HC4 and the strong consistency 3B algorithm have been used.

We compared many ICP algorithms for solving our DICSPs. The best results are obtained by the 3B algorithm [35]. Unfortunately, 3B is too slow to work in a real-time environment. And consequently, it cannot be used to solve our real-time problem. We noticed that apart from 3B, no

algorithms can contract the θ parameter [36], as explained detailedly in the next section.

4.3. Uncorrected Yaw Problem. Vincke and Lambert [34] had shown that classical interval propagation algorithms can not reduce the heading uncertainty of a vehicle equipped with a GPS receiver, a gyro, and odometers. Let us consider a simple forward-backward propagation algorithm, the contraction of the heading uncertainty is performed by (19), which is deduced from (17) in a backward way.

$$\begin{aligned}[\theta_{k-1}] &= [\theta_{k-1}] \bigcap \left(\arccos\left(\frac{[x_k]-[x_{k-1}]}{[\delta s_k]}\right) - \frac{[\delta\theta_k]}{2}\right),\\ [\theta_{k-1}] &= [\theta_{k-1}] \bigcap \left(\arcsin\left(\frac{[y_k]-[y_{k-1}]}{[\delta s_k]}\right) - \frac{[\delta\theta_k]}{2}\right).\end{aligned} \tag{19}$$

Example 2. Considering the case study with a straight displacement along the x-axis, the orientation variation is null ($-\delta\theta_k/2 = 0$). The correction on θ is realized by $\arcsin(([y_k] - [y_{k-1}])/[\delta s_k])$ and $\arccos(([x_k] - [x_{k-1}])/[\delta s_k])$.
The width of $[y_{k-1}]$ is around 10 m in most of our experiments: $[y_{k-1}] = [-5, 5]$. Thus, we can get

$$[y_k] - [y_{k-1}] = [-5 + \epsilon_y, 5 - \epsilon_y] - [-5, 5] = [-10 + \epsilon_y, 10 - \epsilon_y], \tag{20}$$

where ϵ_y is a small reduction of the imprecision due to a new GPS measurement. $[\delta s_k]$ is around 1 m:

$$[\delta s_k] = [1 - \epsilon_s, 1 + \epsilon_s]. \tag{21}$$

Finally,

$$\arcsin\left(\frac{[y_k]-[y_{k-1}]}{[\delta s_k]}\right) = \arcsin\left(\frac{[-10+\epsilon_y, 10-\epsilon_y]}{[1-\epsilon_s, 1+\epsilon_s]}\right) = [-\pi, \pi]. \tag{22}$$

The value inside the arcsin function is far beyond $[-1, 1]$, leading to a full ignorance on the orientation. The same reasoning leads to the same result for the arccos function. Thus, (19) is unable to correct the orientation.

During the vehicle's displacement, the interval representing the heading uncertainty will grow continuously although other parameters of the pose are contracted. It is a big issue because the heading information is important for a good estimation of the vehicle's position on the x-axis and y-axis. Reference [34] explained the problem but gives no solution. Let us notice that their approach needed to decompose all the constraints of the problem into elementary/binary constraints.

The uncorrected yaw problem is observed only with local consistency algorithms which solve constraints one by one. They are subject to the "locality problem": the solution is always local with regard to one constraint at a time. Only a strong consistency algorithm (3B for instance) can overcome this problem and correct the yaw. Unfortunately, strong consistency algorithms have a huge computing time and are not suited for real-time issue [53]. We propose to solve this problem by introducing a new algorithm which splits the θ parameter and creates multiple DICSP (called νDICSP). The next section details our method.

4.4. A Multiple Dynamic Interval Constraint Satisfaction Problem. In order to correct the yaw, we propose to split the variable $\theta_{k-\Xi}$ into ν variables $\theta^i_{k-\Xi}$, $i \in \{1, 2, \dots, \nu\}$ where ν stands for the "splitting factor."

Each domain $[\theta^i_{k-\Xi}]$ is then defined by

$$\left[\theta^i_{k-\Xi}\right] = \left[\underline{\theta_{k-\Xi}} + (i-1)\underline{\mathbf{S}}, \underline{\theta_{k-\Xi}} + i\bar{\mathbf{S}}\right], \tag{23}$$

where $k \geq \Xi$, $\mathbf{S} = w([\theta_{k-\Xi}])/\nu$ is the width of each $[\theta^i_{k-\Xi}]$, and $w([\theta_{k-\Xi}])$ represents the width of the interval $[\theta_{k-\Xi}]$. $\underline{\mathbf{S}}$ and $\bar{\mathbf{S}}$ are, respectively, the round down and the round up of the number $\mathbf{S}$.

Example 3. To illustrate, let us consider $[\theta_{k-\Xi}] = [1, 2]$. If $\nu = 10$ is the "splitting factor," then, the following intervals will be generated: $[\theta^1_{(k-\Xi)}] = [\underline{1}, \overline{1.1}]$, $[\theta^2_{(k-\Xi)}] = [\underline{1.1}, \overline{1.2}]$, $[\theta^3_{(k-\Xi)}] = [\underline{1.2}, \overline{1.3}]$, $\dots$, $[\theta^{10}_{(k-\Xi)}] = [\underline{1.9}, \bar{2}]$.

In the first step of the algorithm, $k < \Xi$ and $[\theta_{k-\Xi}]$ is undefined: the splitting is realized on $[\theta_0]$. $[\theta_{k-\Xi}]$ is the union of all the $[\theta^i_{k-\Xi}]$:

$$[\theta_{k-\Xi}] = \bigcup_{i=1}^{\nu} \left[\theta^i_{k-\Xi}\right]. \tag{24}$$

Other domains remain unchanged (unsplitted). The proposed splitting leads to a νDICSP:

$$\nu\text{DICSP}_k = \left(\text{DICSP}^1_k, \text{DICSP}^2_k, \dots, \text{DICSP}^\nu_k\right), \tag{25}$$

with

$$\text{DICSP}^i_k = \left(\text{ICSP}^i_k, \text{ICSP}^i_{k-1}, \dots, \text{ICSP}^i_{k-\Xi+1}\right)^T. \tag{26}$$

The DICSPs (columns of the νDICSP) are separately solved by a local consistency algorithm. The solution of the localization problem is given by applying the interval hull function on the results obtained from the first row of the ν DICSP_k. It defines the current position and orientation of the vehicle as an interval vector ($[x_k, y_k, \theta_k]$). When new measurements are available, the next localization process starts by applying the interval hull function on the results of the last row of the νDICSP_k (first step before splitting).

Sometimes, one (or more) DICSP does not get a solution: one of the domains is empty. This is an expected behavior that enables fast correction by eliminating a whole set of erroneous pose.

The building and the solving of the multiple dynamic interval constraint satisfaction problem are realized by our proposed dynamic interval constraint propagation with the splitting algorithm (DICSP, see Algorithm 1), explained hereafter:

(i) Line 1: the oldest variable $[\theta_{k-\Xi}]$, representing the orientation in the initial DICSP, is split into ν intervals.

(ii) Line 3: for each $[\theta^i_{k-\Xi}]$, a DICSP is generated.

(iii) Line 4: each DICSP is submitted to a solving process (by an ICP algorithm).

(iv) Line 6: the new domains, resulting from the process of the preview step, are grouped with an interval *hull* function.

(v) Line 7: the interval hull constitutes an interval vector which is returned as the vehicle position.

This algorithm has been implemented and tested on a vehicle running on a test track as explained in the next section.

4.5. About Completeness. "The two most appealing features of interval analysis and numerical constraint propagation, when used to solve numerical problems, are completeness and rigor. Completeness means the ability to find all solutions, whereas rigor is the ability to control the rounding errors due to floating-point computation" [54]. The rigor property is kept by our algorithm as we only used interval operators. The completeness is discussed because our algorithm splits the variables before applying classical constraint propagation techniques. We propose a new ICP algorithm with splitting: it splits an interval vector $[\mathbf{x}]$ into boxes $[x_1, x_2], \ldots, [x_m]$, so that

$$[\mathbf{x}] = \bigcup_{i=1}^{m} [x_i]. \tag{27}$$

Then, the range of a function $f(\cdot)$ over $[\mathbf{x}]$, possibly defined by (17), can be evaluated by

$$f([\mathbf{x}]) = f\left(\bigcup_{i=1}^{m} [x_i]\right). \tag{28}$$

The bisection usually leads to an improved estimate. Our algorithm takes advantage of this property and combine it with DICSP to solve the uncorrected yaw problem by splitting the parameter θ. According to the interval extension definition given in Section 2.2, we have

$$f([\mathbf{x}]) \subseteq \bigcup_{i=1}^{m} [f]([x_i]). \tag{29}$$

And thanks to the definition of interval hull (see Section 2.2), we obtain

$$f([\mathbf{x}]) \subseteq \bigsqcup_{i=1}^{m} [f]([x_i]). \tag{30}$$

Since $\sqcup_{i=1}^{m} [f]([x_i])$ is a genuine superset of the left-hand side, all correct answers are included in it. Furthermore, each term of the interval hull is solved by a local consistency algorithm which ensures that none of the solutions are removed (thus achieving the completeness of our method).

5. Results

This section introduces the tools and equipment (vehicle prototype with its sensors) that we used to validate our method as well as the test track where data have been collected. Our algorithm has been compared with 2 interval algorithms and a Bayesian one.

5.1. Experimental Setup. The proposed algorithm has been implemented in C language on a computer equipped with an Intel Core i7 CPU 960 @ 3.20 GHz running under Ubuntu 14.04 LTS. Solving has been realized with a set of time-stamped data that has been collected on the Satory test track (Figure 2(c)) with LIVIC's prototype (Figures 2(a) and 2(b)) running at an average speed of 50 km/h. A solid-state vertical gyro VG400CC provides the yaw rate data. The global localization is performed by an AgGPS132 (datasheet: http://dragadoshidraulicos.com/batimetria/GPS-Trimble-AGP132.pdf). GPS measurements and gyro/odometer data acquisitions are realized at a 5 Hz frequency. The centimeter reference used for the evaluation of the positioning method is provided by a RTK GPS (datasheet: http://www.ctsystems.eu/Resources/Sagitta.pdf).

5.2. Comparison between DICPS, HC4, and 3B Algorithms. A full lap run last 400 seconds while 2000 localization DICSPs have been solved under different window sizes ($\Xi = 10, 20, 30, 40, 50, 100$). Figure 3(a) shows how the value of the splitting factor ν influences the correction of heading imprecision in DICPS. The efficiency of the splitting decreases as a hyperbolic function, indicating that the value of the splitting factor ν can be bounded around 20. Nevertheless, the higher both variables are (Ξ and ν), the lower is the localization error.

Computing time has been measured for HC4, 3B, and our DICPS algorithm. Table 2 shows that HC4 is the fastest, but Figure 3(b) indicates its inability to correct the heading imprecision: the heading imprecision increases linearly as the time goes by.

This is a common result obtained by local consistency algorithms such as HC3 [19], BC3 [21], BC4 [20], and BC5 [22]. DICPS results are observed with three window sizes Ξ (20, 40, and 100) depicted in Figure 3(b). The heading imprecision is stabilized between 10 and 35 degrees depending on the parameters. It shows that the window size is an important parameter that improves the correction of heading imprecision.

3B consistency considers the bounds of each variable's domain. It tends to calculate the solution of the localization DICSP for each bound (see Section 3.2). If no solution is found, the bound is eliminated from the considered domain and so on. We use 3B with the same window size than DICPS: $\Xi = 20$. The observed results are the same while 3B computing time is much larger than the DICPS one: 5257 ms for 3B versus 13.32 ms for DICPS (see Table 2). Different window sizes give the same localization results but with a 3B prohibitive computing time.

If the window size Ξ is increased to 40 while keeping the same splitting ($\nu = 20$), then the computing time of DICPS is 172 ms. Thus, it is still real-time while ensuring a 16° (±8°)

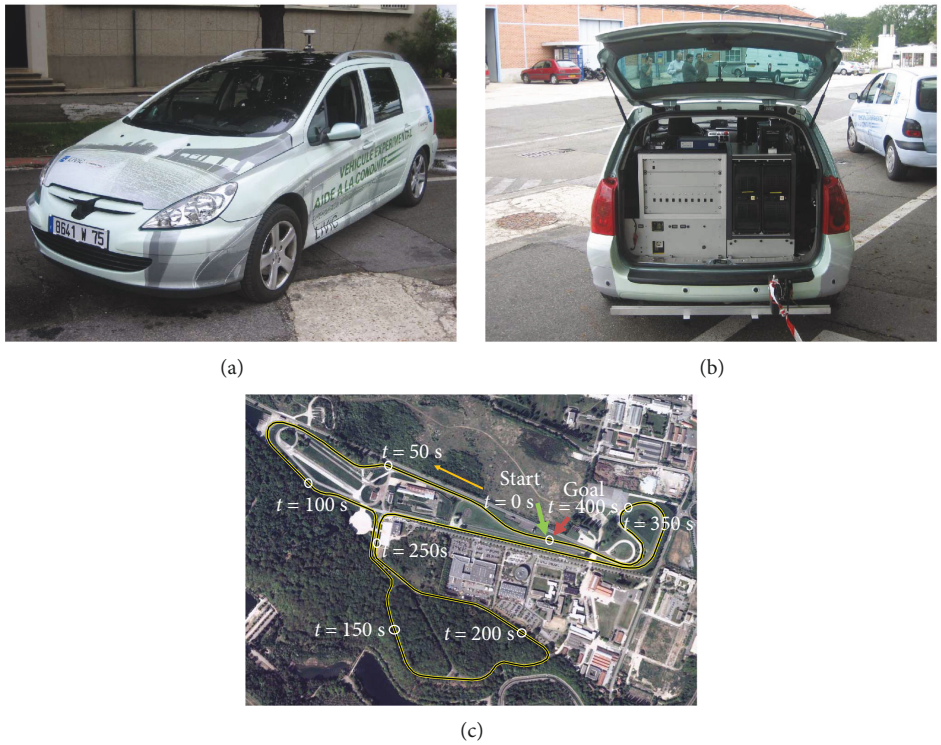

Figure 2: Experimental platform. (a) CARLLa platform. (b) Embedded hardware architecture. (c) The Satory track with timestamps.

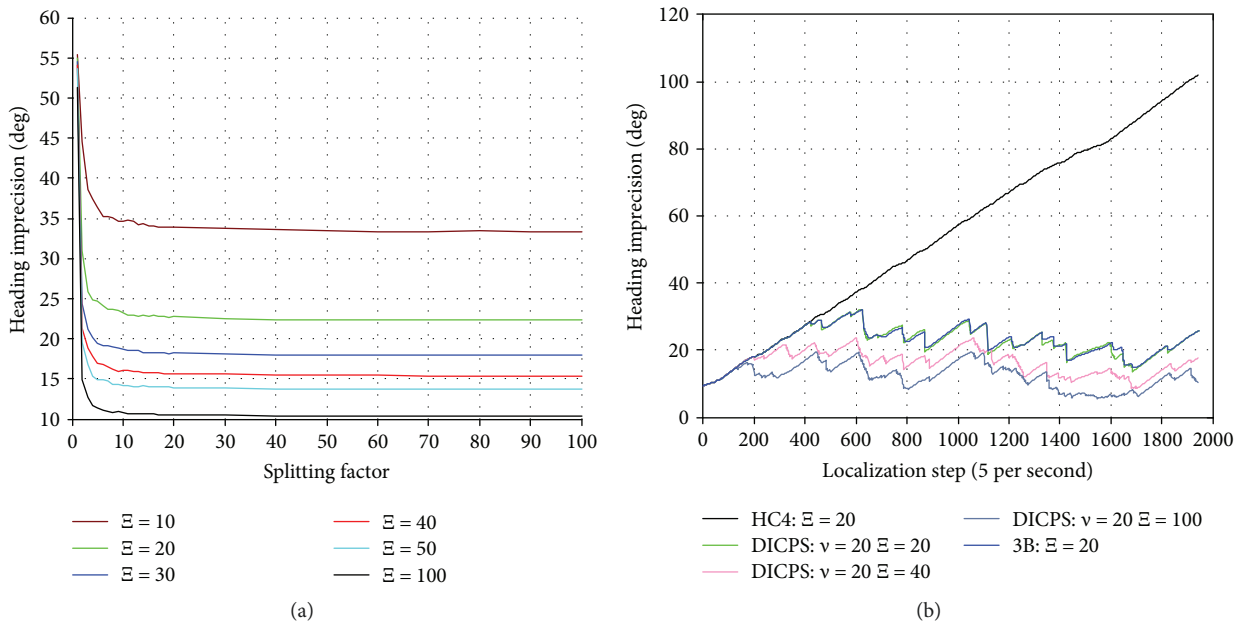

Figure 3: Heading imprecision. (a) Average heading imprecision for different splitting factors. (b) Heading imprecision at each time step for different ICP algorithms.

Table 2: Mean duration of the algorithm for a localization step.

Algorithms parameters	DICPS $\Xi = 20$; $\nu = 20$	HC4 $\Xi = 20$	3B $\Xi = 20$
Duration (ms)	13.32	0.542	5257

imprecision in average. This imprecision is below the value provided by other interval approaches (including CP and BESE [55]) in the same environment.

5.3. Comparison between DICPS and EKF. The previous section shows that DICPS outdoes the other methods based on

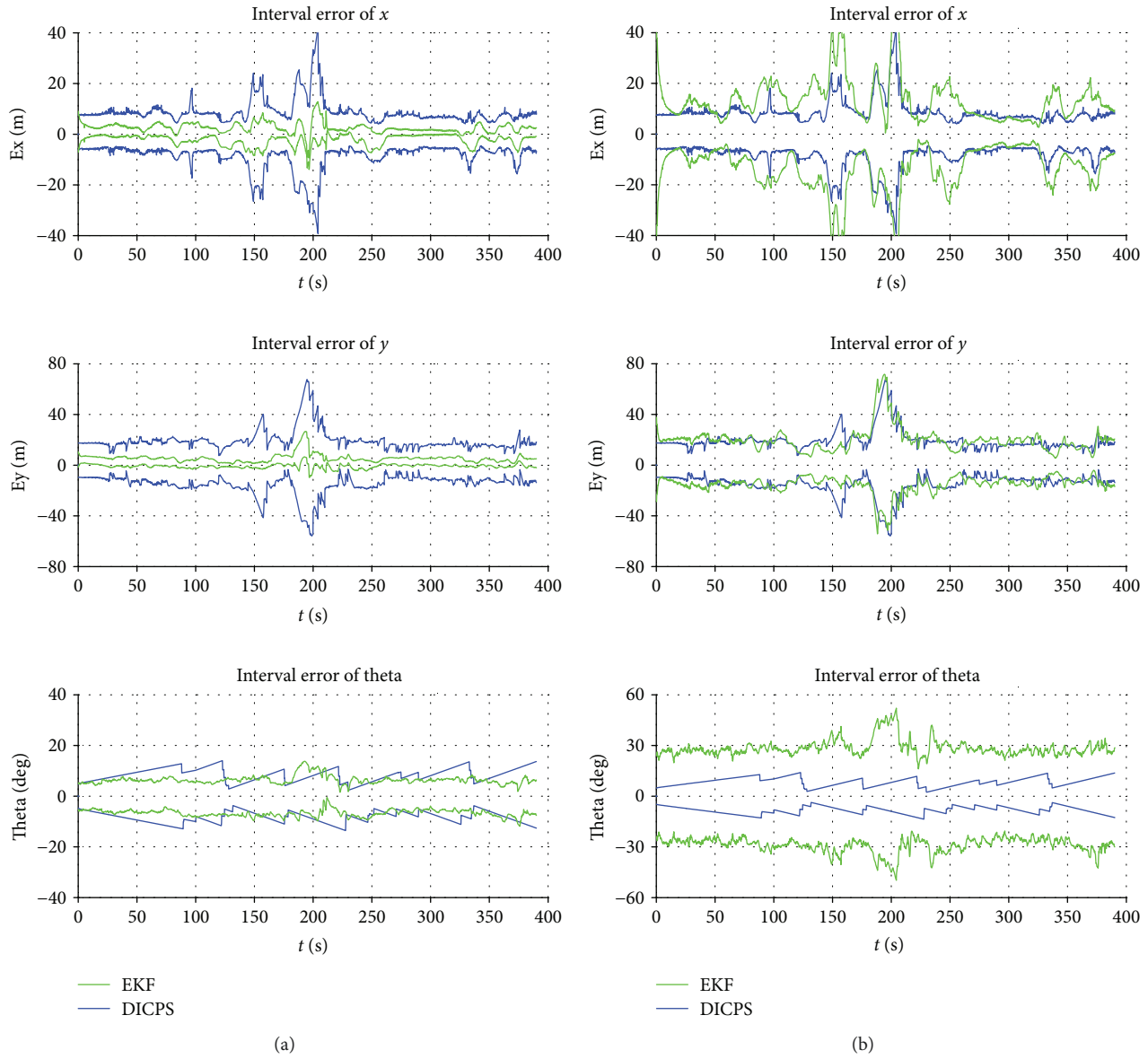

Figure 4: Comparison between DICPS and EKF. (a) DICPS (3σ) and inconsistent EKF with 3σ error bounds. (b) DICPS (3σ) and consistent EKF with 17σ error bounds.

interval analysis. This section aims to compare DICPS with a wide-spread Bayesian filter: the extended Kalman filter (EKF).

5.3.1. Theoretical Comparison. DICPS considers bounded errors (with no assumption on the error inside the bound) while EKF deals with Gaussian errors. DICPS returns interval vectors which contain the vehicle's pose (it searches all solutions that are consistent with the model and data of the given problem). EKF gives a vector (that has the highest probability to be the solution) and its associated Gaussian uncertainty, represented by a matrix.

5.3.2. Experimental Comparison. We set up DICPS with parameters $\nu = 20$ and $\Xi = 40$. In order to ensure fairness, EKF and DICPS use the same data set.

The EKF is fed with the standard deviation σ of the sensors whereas DICPS uses a 3σ error bound which corresponds to a 0.99 confidence level. Consequently, where we should consider finding a data with a 0.99 probability, the DICPS assumes to find it with a 1 probability.

Figure 4(a) shows the interval error of the DICPS and the EKF. The interval error is defined by the upper and lower bounds of the estimated state minus the corresponding bound reference state. A filter exhibits good results if its interval error is thin and always includes the reference value zero. The imprecision area of DICPS (using a 3σ error bound) has directly been used to compute the interval error. For the EKF, the imprecision (standard deviation) has been magnified 3 times.

The EKF is more confident (has a thinner corridor) than DICPS, but the EKF is sometimes inconsistent (does not

include the zero value) whereas DICPS is always consistent. The large peaks (around $t = 100$, 150, 200, 340, and 370) are caused by GPS measurements with large imprecision. The top peaks define steps when the imprecision of the robot state has increased before being reduced thanks to GPS measurement. The great inconsistency of the EKF around $t = 200$ s is due to a large number of repeated biased GPS measurements. The EKF greatly underestimates its covariance matrices when dealing with repeated biased measurements. DICPS does not suffer from those biased measurements because it has no assumption on the distribution of the noise (contrarily to the EKF).

We could get a consistent EKF by tuning carefully (enlarging) the noise value for each simulation (we can do it a posteriori by using the reference). However, such an EKF tuning does not seem feasible for the localization problem in the real world because we do not have such a reference (if we have such a reference then computing the localization will be useless). Nevertheless, as part of this paper, we have such a reference and we have computed the x value for the EKF to be consistent. We found 17 and drew the 17σ interval error in Figure 4(b). Thus, we can compare both consistent filters. The resulting EKF corridors are larger than those of the consistent DICPS. Consequently, even if it was possible to compute the x value during the localization, the EKF results would not be better than the results of DICPS.

6. Conclusion

This paper deals with the vehicle localization problem using interval analysis. Different algorithms have been compared on a vehicle equipped with a GPS receiver, a gyro, and odometers. Every algorithm can correct the position. But under real-time constraints, no ICP algorithms were able to correct the vehicle's orientation. Only 3B was capable of orientation correction but at the cost of a huge computing time.

Our algorithm (DICPS) is able to correct the orientation and is real-time executable. As the computation time increases with the splitting factor, we recommend to conduct DICPS with parameters $\Xi = 20$ and $\nu = 40$ in order to keep a reasonable calculation time (172 ms for each localization step). DICPS has been shown to reduce the orientation imprecision around ±8. This method is robust and provides consistent results.

DICPS has been compared to the well-known extended Kalman filter. We point out that the EKF can locally converge towards a wrong solution due to bias measurements which lead to a huge local inconsistency. We claim that consistency is the biggest advantage of DICPS over Bayesian approaches.

Future works will deal with improvements to further reduce the imprecision and minimizing the CPU load. It can be achieved by dynamically defining the involved parameters (window size and splitting factor). In addition, no change or partial change on variables will allow us to simplify the localization problem by removing constraints. Finally, small changes could be stacked up, waiting for a significant amount of change to start the constraint propagation process.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

Kangni Kueviakoe and Zhan Wang contributed equally to this work.

References

[1] D. Gruyer, A. Lambert, M. Perrollaz, and D. Gingras, "Experimental comparison of Bayesian vehicle positioning methods based on multi-sensor data fusion," *International Journal of Vehicle Autonomous Systems*, vol. 12, no. 1, pp. 24–43, 2013.

[2] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents Series)*, The MIT Press, Cambridge, MA, USA, 2005.

[3] L. Jaulin, "Robust set-membership state estimation; application to underwater robotics," *Automatica*, vol. 45, no. 1, pp. 202–206, 2009.

[4] E. Colle and S. Galerne, "A multihypothesis set approach for mobile robot localization using heterogeneous measurements provided by the Internet of Things," *Robotics and Autonomous Systems*, vol. 96, pp. 102–113, 2017.

[5] L. Jaulin, "Localization of an underwater robot using interval constraint propagation," in *International Conference on Principles and Practice of Constraint Programming*, pp. 244–255, Nantes, France, 2006.

[6] A. Gning and P. Bonnifait, "Constraints propagation techniques on intervals for a guaranteed localization using redundant data," *Automatica*, vol. 42, no. 7, pp. 1167–1175, 2006.

[7] A. K. Mackworth, "Consistency in networks of relations," *Artificial Intelligence*, vol. 8, no. 1, pp. 99–118, 1977.

[8] H. Gallaire, "Logic programming: further developments," in *IEEE Symposium on Logic Programming*, pp. 88–99, Boston, MA, USA, 1985.

[9] J. Jaffar and J. L. Lassez, "Constraint logic programming," in *Proceedings of the 14th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages - POPL '87*, pp. 111–119, Munich, Germany, 1987.

[10] P. van Hentenryck, Y. Deville, and C. M. Teng, "A generic arc-consistency algorithm and its specializations," *Artificial Intelligence*, vol. 57, no. 2-3, pp. 291–321, 1992.

[11] C. Bessiere, "Arc-consistency and arc-consistency again," *Artificial Intelligence*, vol. 65, no. 1, pp. 179–190, 1994.

[12] Z. Yuanlin and R. Yap, "Arc consistency on n-ary monotonic and linear constraints," in *International Conference on Principles and Practice of Constraint Programming*, pp. 470–483, Singapore, 2000.

[13] R. Dechter and J. Pearl, "Tree clustering for constraint networks," *Artificial Intelligence*, vol. 38, no. 3, pp. 353–366, 1989.

[14] F. Rossi, C. Petrie, and V. Dhar, "On the equivalence of constraint satisfaction problems," in *Proceedings of the 9th European Conference on Artificial Intelligence (ECAI'90)*, pp. 550–556, Stockholm, Sweden, 1990.

[15] R. Mohr and G. Masini, "Good old discrete relaxation," in *Proceedings of the 8th European Conference on Artificial Intelligence (ECAI'88)*, pp. 651–656, Munich, Germany, 1988.

[16] J. Cleary, "Logical arithmetic," *Future Computing Systems*, vol. 2, pp. 125–149, 1987.

[17] E. Davis, "Constraint propagation with interval labels," *Artificial Intelligence*, vol. 32, no. 3, pp. 281–331, 1987.

[18] E. Hyvönen, "Constraint reasoning based on interval arithmetic," in *Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI'89)*, vol. 2, pp. 1193–1198, Detroit, MI, USA, 1989.

[19] F. Benhamou and W. J. Older, "Applying interval arithmetic to real, integer, and Boolean constraints," *The Journal of Logic Programming*, vol. 32, no. 1, pp. 1–24, 1997.

[20] F. Benhamou, F. Goualard, L. Granvilliers, and J. F. Puget, "Revising hull and box consistency," in *Proceedings of the 1999 International Conference on Logic Programming*, pp. 230–244, Las Cruces, NM, USA, 1999.

[21] F. Benhamou, D. McAllester, and P. V. Hentenryck, "CLP(intervals) revisited," in *Proceedings of the 1994 International Symposium on Logic Programming*, pp. 124–138, Melbourne, VIC, Australia, 1994.

[22] L. Granvilliers, "Towards cooperative interval narrowing," in *International Symposium on Frontiers of Combining Systems*, pp. 18–31, Nancy, France, 2000.

[23] O. Lhomme, "Consistency techniques for numeric CSPs," in *Proceedings of the 13th International Joint Conference on Artifical Intelligence (IJCAI'93)*, vol. 1, pp. 232–238, Chambery, France, 1993.

[24] A. Gning and P. Bonnifait, "Guaranteed dynamic localization using constraints propagation techniques on real intervals," in *IEEE International Conference on Robotics and Automation, 2004*, pp. 1499–1504, New Orleans, LA, USA, 2004.

[25] B. Vincke, A. Lambert, D. Gruyer, A. Elouardi, and E. Seignez, "Static and dynamic fusion for outdoor vehicle localization," in *2010 11th International Conference on Control Automation Robotics & Vision*, pp. 437–442, Singapore, 2010.

[26] B. Vincke and A. Lambert, "Enhanced constraints propagation for guaranteed localization predictive step," in *IEEE/RSJ IROS, Workshop on Planning, Perception and Navigation for Intelligent Vehicles*, pp. 30–36, St Louis, MO, USA, 2009.

[27] D. L. Waltz, *Generating Semantic Descriptions from Drawings of Scenes with Shadows, [Ph.D. thesis]*, AI Lab, MIT, Cambridge, MA, USA, 1972.

[28] V. Drevelle and P. Bonnifait, "Robust positioning using relaxed constraint-propagation," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4843–4848, Taipei, Taiwan, 2010.

[29] K. Lassoued, P. Bonnifait, and I. Fantoni, "Cooperative localization with reliable confidence domains between vehicles sharing GNSS pseudoranges errors with no base station," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 1, pp. 22–34, 2017.

[30] B. Desrochers and L. Jaulin, "A minimal contractor for the polar equation: application to robot localization," *Engineering Applications of Artificial Intelligence*, vol. 55, pp. 83–92, 2016.

[31] L. Jaulin, A. Stancu, and B. Desrochers, "Inner and outer approximations of probabilistic sets," in *Proceedings of the American Society of Civil Engineers (ASCE) Second International Conference on Vulnerability and Risk Analysis and Management and Sixth International Symposium on Uncertainty Modelling and Analysis*, pp. 13–16, Liverpool, UK, 2014.

[32] C. Aubry, R. Desmare, and L. Jaulin, "Loop detection of mobile robots using interval analysis," *Automatica*, vol. 49, no. 2, pp. 463–470, 2013.

[33] F. Le Bars, J. Sliwka, L. Jaulin, and O. Reynet, "Set-membership state estimation with fleeting data," *Automatica*, vol. 48, no. 2, pp. 381–387, 2012.

[34] B. Vincke and A. Lambert, "Experimental comparison of bounded-error state estimation and constraints propagation," in *2011 IEEE International Conference on Robotics and Automation*, pp. 4724–4729, Shanghai, China, 2011.

[35] I. K. Kueviakoe, A. Lambert, and P. Tarroux, "Comparison of interval constraint propagation algorithms for vehicle localization," *Journal of Software Engineering and Applications*, vol. 05, no. 12, pp. 157–162, 2012.

[36] I. K. Kueviakoe, A. Lambert, and P. Tarroux, "Vehicle localization based on interval constraint propagation algorithms," in *2013 IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems*, pp. 179–184, Nanjing, China, 2013.

[37] R. E. Moore, *Interval Analysis*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1966.

[38] R. E. Moore and F. Bierbaum, *Methods and Applications of Interval Analysis*, vol. 2, Society for Industrial Mathematics, Philadelphia, PA, USA, 1979.

[39] R. E. Moore, R. B. Kearfott, and M. J. Cloud, *Introduction to Interval Analysis*, Society for Industrial Mathematics, Philadelphia, PA, USA, 2009.

[40] E. Hyvönen, "Constraint reasoning based on interval arithmetic: the tolerance propagation approach," *Artificial Intelligence*, vol. 58, no. 1-3, pp. 71–112, 1992.

[41] E. Hyvönen, S. D. Pascale, and A. Lehtola, "Interval constraint satisfaction tool INC++," in *Proceedings of 1993 IEEE Conference on Tools with AI (TAI-93)*, pp. 298–305, Boston, MA, USA, 1993.

[42] P. V. Hentenryck, *Numerica: A Modeling Language for Global Optimization*, MIT Press, Cambridge, MA, USA, 1997.

[43] L. Granvilliers and F. Benhamou, "Algorithm 852: RealPaver: an interval solver using constraint satisfaction techniques," *ACM Transactions on Mathematical Software*, vol. 32, no. 1, pp. 138–156, 2006.

[44] J. P. Merlet, "ALIAS: an interval analysis based library for solving and analyzing system of equations," in *Séminaire Systèmes et Equations Algébriques*, pp. 1964–1969, INRIA Sophia Antipolis, Valbonne, France, 2000.

[45] Y. Lebbah, "ICOS: a branch and bound based solver for rigorous global optimization," *Optimization Methods and Software*, vol. 24, no. 4-5, pp. 709–726, 2009.

[46] F. Domes, "GLOPTLAB: a configurable framework for the rigorous global solution of quadratic constraint satisfaction problems," *Optimization Methods and Software*, vol. 24, no. 4-5, pp. 727–747, 2009.

[47] G. Chabert, "IBEX, an interval-based explorer," 2007, http://www.ibex-lib.org/.

[48] G. Chabert and L. Jaulin, "Contractor programming," *Artificial Intelligence*, vol. 173, no. 11, pp. 1079–1100, 2009.

[49] R. Dechter and A. Dechter, "Belief maintenance in dynamic constraint networks," in *Proceedings of the Seventh AAAI National Conference on Artificial Intelligence (AAAI'88)*, pp. 37–42, Saint Paul, MN, USA, 1988.

[50] H. Collavizza, F. Delobel, and M. Rueher, "A note on partial consistencies over continuous domains," in *International Conference on Principles and Practice of Constraint Programming*, pp. 147–161, Pisa, Italy, 1998.

[51] H. R. Everett, *Sensors for Mobile Robots: Theory and Application*, AK Peters, Natick, MA, USA, 1995.

[52] E. Seignez, M. Kieffer, A. Lambert, E. Walter, and T. Maurin, "Experimental vehicle localization by bounded-error state estimation using interval analysis," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1084–1089, Edmonton, AB, Canada, 2005.

[53] I. K. Kueviakoe, A. Lambert, and P. Tarroux, "A real-time interval constraint propagation method for vehicle localization," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pp. 1707–1712, The Hague, Netherlands, 2013.

[54] C. Jermann, Y. Lebbah, and D. Sam-Haroud, "Interval analysis, constraint propagation and applications," in *Trends in Constraint Programming*, F. Benhamou, N. Jussien, and B. A. O'Sullivan, Eds., John Wiley & Sons, Hoboken, NJ, USA, 2007, Chapter 4.

[55] A. Lambert, D. Gruyer, B. Vincke, and E. Seignez, "Consistent outdoor vehicle localization by bounded-error state estimation," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1211–1216, St. Louis, MO, USA, 2009.