A Journal of Software Engineering and Applications, 2012, 5, 157-162
doi:10.4236/jsea.2012.512b030 Published Online December 2012 (http://www.scirp.org/journal/jsea)

Comparison of Interval Constraint Propagation Algorithms for Vehicle Localization

I. K. Kueviakoe[1,2] , A. Lambert[3] , P. Tarroux[1,4]

[1] LIMSI, CNRS, 91400 Orsay, France; [2] Univ. Paris Sud, IEF, 91400 Orsay, France; [3] LIVIC, IFSTTAR, 14 route de la minière, 78000 Versailles,France; [4] ENS, 45 rue d'ULM, 75230 Paris, France
Email: kangni.kueviakoe@limsi.fr

Received 2012

ABSTRACT

Interval constraint propagation (ICP) algorithms allow to solve problems described as constraint satisfaction problems (CSP). ICP has been successfully applied to vehicle localization in the last few years. Once the localization problem has been stated, a large class of ICP solvers can be used. This paper compares a few ICP algorithms, using the same experimental data, in order to rank their performances in terms of accuracy and computing time.

Keywords: Interval Analysis; Constraint Propagation; Data Fusion; Vehicle Positioning, GPS.

1. Introduction

Most of the early published papers on constraint programming date back to the seventies and were defined for discrete domains [1-3]. In the eighties, Gallaire [4], Jaffar and Lassez [5] noted that logic programming was just a particular kind of constraint programming. Logic programming as well as constraint programming implies that the user states what has to be solved instead of how to solve it.

Among the techniques used for solving a CSP (Constraint Satisfaction Problem), Constraint Propagation is the most used. It is based on combining systematic search methods and consistency techniques. Mainly three kinds of consistency concepts such as node consistency, arc consistency and path consistency are known. The most popular is arc consistency, achieved by the AC-3 algorithm [3] and by many other algorithms (such as AC-5 [6], AC-6 [7], etc.).

Those works only use binary CSPs (each constraint involves at most two variables). However, many real-life problems are naturally modeled as non-binary CSPs: the constraint involves more than two variables [8].

Two approaches were developed to deal with non binary CSPs. The first approach translates a non binary CSP into different binary CSPs [9,10]. After this so called "binarization" step, the classical techniques in binary CSP can be used to solve the transformed CSP. The second approach directly deals with non binary constraints; for instance, GAC-4 [11] is a generalization of AC-4 to non binary constraints. In 1987, Cleary [12] and Davis [13] were the first to deal jointly with constraint propagation on interval analysis. In 1989, Hyvonen [14] designed a generalized constraint propagation scheme based on interval arithmetic. A narrowing algorithm was proposed by Cleary [12] and improved by Benhamou [15]. Later, that algorithm was renamed HC3 [16] because it is very close to the classical AC-3. Next, Benhamou proposed HC4 [16] that does not need the decomposition of constraints. However HC4 suffers from the dependency problem (see section 2.3). To cope with this problem, BC3 [17] was developed but suffers of slowness. BC4 [16] merges both BC3 and HC4 and reduces the computation time. BC5 [18], by using the interval Gauss-Seidel method, further reduce the computing time. 3B algorithm [19] was led by the search of the strongest contractions rather than the smaller computation time. Interval Constraint Propagation (ICP) was introduced in the mobile robotic area ten years ago. It was mainly used for outdoor vehicle localization [20-22] and underwater robot localization [23]. Those works use a Forward-Backward propagation technique based on primitive constraints (following the Waltz algorithm [24] principle). The main advantage of ICP over Bayesian algorithms is that ICP guarantees that the actual position of the robot is contained in a box. Bayesian algorithms [25] can only associate a probability to such a box. Consequently, safety cannot be guaranteed by Bayesian algorithms.

All those techniques decompose their constraints into primitive ones (binary constraints). Decomposing constraints into primitive constraints implies creating many auxiliary variables that should be also narrowed and can interfere the narrowing of the primary variables. Some comparisons [18, 16, 26] have been realized between ICP algorithm applied to theoretical problems taken from areas like chemistry, astrophysics, etc. For every problem, one CSP is formalized and solved. The computation time and precision are then analyzed. We will extend this work to the vehicle localization problem where the CSP evolve with a sliding time window. Our goal is to compare the most achieved ICP algorithms found in the state of the art, in order to show which one of those algorithms is the more suitable for the vehicle localization (in terms of the processing time and the precision of the results).

Section 2 introduces interval analysis. ICP (Interval Constraint Propagation) algorithms are discussed in Section 3. The localization process including models and sensors is presented in Section 4. Section 5 shows our experimental results before concluding in Section 6.

2. Basics of Interval analysis and Constraint Propagation

2.1. Overview of Interval analysis

Interval analysis [27] was introduced in the sixties in order to solve the problem of approximations made during calculations. The key idea of interval analysis is to represent numbers by intervals which included the real values. An interval is represented using [ ]. For example: $[x]=[\underline{x},\overline{x}]=\{x\in\mathbb{R}\mid \underline{x}\le x\le\overline{x}\}\in\mathbb{I}(\mathbb{R})$ is the interval containing x. A set of rules have been defined to perform all the usual operations on intervals. The vehicle configuration is represented by several intervals. To estimate and handle the sets implied in our problem, we use the concept of boxes as bounded configurations: a box $[\mathbf{x}]\in\mathbb{I}(\mathbb{R}^3)$ is a Cartesian product of interval domains.

2.2. Inclusion Function

For any function f defined as combinations of arithmetical operators and elementary functions, interval analysis defines the inclusion function (also called interval extension) $f_{[\,]}$ as:

$$\forall x\in D,\ f_{[]}([x])\subset f([x]) \qquad (1)$$

The easiest way to obtain the inclusion function is to replace all the variables by their intervals. For instance,

$$\forall x\in D,\ f(x)=x^2+2x+4 \qquad (2)$$

has the following inclusion function:

$$\forall x\in D,\ f_{[]}([x])=[x]^2+2[x]+4 \qquad (3)$$

2.3. The dependency problem

Interval arithmetic handles multiple occurrences of a same variable as many different variables. For instance, another inclusion function of Equation (2) is

$$\forall x\in D,\ f_{[],2}([x])=([x]+2)^2 \qquad (4)$$

$f_{[]}$ has two occurrences of the variable [x] whereas $f_{[],2}$ has only one. The images of the interval I = [−3, 4] are $f_{[]}(I)$ = [−8, 36] and $f_{[],2}$ (I) = [0, 36]. The interval image computed by f[],2 is sharper than the one produced by $f_{[]}$. It is well known that the problem of finding the optimal interval image is complicated by the multiple occurrences of a variable: it is the dependency problem [28].

2.4. Constraint Satisfaction Problem

Constraint satisfaction problems (CSP) are mathematical problems that are solved by finding states satisfying the constraints. A constraint restricts the possible solutions. A Constraint Satisfaction Problem is defined by:

• a set of variables $\{x_1, x_2, .., x_n\}$,

• a set of domains $\{D_1, D_2, ..,D_n\}$, such as for each variable x_i , a domain D_i with the possible values for that variable are given,

• a set of constraints $\{C_1, C_2, .., C_m\}$, which are relations between the variables. Constraint propagation consists of iterating domain reductions, by using the set of m constraints, until no domain can be contracted. Interval Constraint Satisfaction Problem defines each domain as an interval. The Cartesian product of the contracted interval domains is our solution box which is guaranteed to contain the real vehicle localization. Such contractions can be achieved by various algorithms that are described in the next section.

3. Constraint Propagation Algorithms

3.1. HC3

HC3 [12] enforces consistency over simple (primitive) constraints. Let's consider the constraint $z = cos(2x − y)$ with [x], [y], [z] the intervals to contract. This constraint is not a primitive one. It is called "user constraint" or "complex constraint". A primitive constraint involves only one arithmetic operator or one of the usual functions like *sin()*, *cos()*, *exp()*, etc. The complex constraint is first decomposed into primitive operations:

$$\begin{cases} a &= 2\times x\\ b &= a-y\\ z &= cos(b)\end{cases} \qquad (5)$$

Then two phases are realized to contract the intervals:

- Forward propagation allows to narrow the left terms (a, b and z) of Equation (4)

$$\begin{cases} [a] &= [a]\cap(2\times[x]) \\ [b] &= [b]\cap([a]-[y]) \\ [z] &= [z]\cap cos([b]) \end{cases} \quad (6)$$

- Backward propagation which reduces the right terms (*x*, *a*, *y* and *b*) of Equation (4)

$$\begin{cases} [x] &= [x]\cap \quad ([a]/2\ ) \\ [a] &= [a]\cap \quad ([b]+[y]) \\ [y] &= [y]\cap \quad ([a]-[b]) \\ [b] &= [b]\cap \quad arccos([z]) \end{cases}$$

(7)

A well known drawback of this method is that the decomposition into primitive constraints introduces new variables in the CSP. This hinders efficient domain tightening. Previous works on vehicle localization use the same Forward/Backward propagation as HC3.

3.2. HC4

HC4 [16] does not decompose complex constraints into primitive ones. It follows a loop propagation and process constraints individually using HC4-revise function. HC4-revise reduced domains by removing inconsistent values across them 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 +, −, *sin*(), etc. HC4-revise works in two phases:

• The first phase called "forward evaluation phase" goes through the graph from the leaves to the root of the tree. It evaluates recursively the interval of sub-expression represented by a current node by using a natural extension of the underlying functions.

• The second phase is called "backward propagation phase". It traverses the tree from the root to the leaves. It applies on each node a contraction operator (a projection). The contraction operator narrows the interval of the current node by removing inconsistent values w.r.t the basis operator of its ascendant node.

The main limitation of HC4 is its sensitivity to the multiple occurrences of variables[17]. For instance, when considering a constraint like: $x\times x+y^2=2$, with $(x,y)\in[-2,4]\times[-1,1]$ HC4 (and HC3) would not reduce the initial box. But they would reduce perfectly the initial box when the constraint is stated like $x^2+y^2=2$.

3.3. BC3

BC3 [17] tightens the domains with a search procedure based on bisection over natural interval extension of constraints. The following example shows the algorithm principle. Let $y-x=0$ be a constraint and $(x,y)\in[0,1]\times[0,8]$. The left bound of I*y* is box consistent since. $0 \in 0-2\times I_x = [-2,0]$. And on the other hand, the right bound of I*y* is not box consistent because $0 \in 8-2\times I_x = [6, 8]$. The domain of *y* is divided to find the most consistent value, in the way as follows:

[0, 8] is divided into [0, 4] and [4, 8]

[0, 4] − 2 × I_x = [-2, 4]

[4, 8] − 2 × I_x = [2, 8] ⇒ [4, 8] is eliminated

[0, 4] is divided into [0, 2] and [2, 4] because the bound 4 is not box consistent

[0, 2] − 2 × I_x = [-2, 2]

[2, 4] − 2 × I_x = [0, 4]

[2, 4] is divided into [2, 3] and [3, 4]

[2, 3] − 2 × I_x = [0, 3]

[3, 4] − 2 × I_x = [1, 4] ⇒ [3, 4] is eliminated

...

The final domain of y is I*y* = [0, 2]. The same technique is applied to determine I_x = [0, 1]. This algorithm is more time consuming than HC3 and HC4 but it does not suffer from the dependency problem.

3.4. BC4

BC4 [16] combines BC3 and HC4. It fights the drawbacks of HC4 (multiple occurrences) and BC3 (slowness). It adapts its computation technique to the number of occurrences of each variable in a constraint. HC4 reduces the domain of variables that occur only once in a constraint. Variables occurring more than once are narrowed by searching "extreme quasi-zeros" using BC3 combined with an interval Newton method described in [17].

3.5. 3B

3B algorithm [19] also referred as strong consistency algorithm, computes the projection of sets of constraints over the variables. It combines constraints in order to improve the precision of domains narrowing. Instead of projecting constraint one by one, it projects the whole set of constraints of the CSP. For example, let's consider two variables x_k and y_k such as $(x_k,y_k)\in[-2,2]\times[-2,2]$ with the following constraints:

$$\begin{cases} x_k+y_k &= 0 \\ x_k-y_k &= 0 \end{cases} \quad (8)$$

The domains [−2, 2] are consistent solutions which cannot be further reduced if we take constraints one by one. However, we can notice that for $x=-2$ the first constraint gives $y = 2$ and the second $y=-2$ which are not rigorously correct. 3B takes into account both constraints and leads then to the solution [0, 0] which is mathematically correct. 3B uses a low level algorithm in order to contract the intervals; we have chosen to use BC4 which combines the advantages of BC3 and HC4.

4. Localization process

4.1. Sensors

4.1.1. Odometers

Odometers are set on the two rear wheels of our vehicle. They give the distance traveled by each wheel independently. The accuracy of an odometer depends on its number of steps and its maximum error is known to be one step. Consequently the real value of the displacement of a non sliding wheel can be bounded by $[\delta p]=[\delta p_{odo}-1, \delta p_{odo}+1]$ with δp_{odo} the number of measured steps. When considering sliding, the movement of a sliding wheel can be deduced from a non-sliding one by adding a sliding noise $[\varepsilon_{odo}]$. The displacement with a sliding wheel is defined by:

$$[\delta p]=[\delta p_{odo}-1-\varepsilon_{odo}, \delta p_{odo}+1+\varepsilon_{odo}] \tag{9}$$

4.1.2. Gyro

A gyro is a heading sensor which measures the rotational speed in an inertial reference system. It is are based on a technique that consists in vibrating silicon structures that use the Coriolis force to output angular rate independently of acceleration. Our gyro measures the yaw rate from which we deduce the elementary rotation $[\delta\theta]$.

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

4.1.3 Global Positioning System receiver

The GPS satellites orbit at a height of 20.190 km and synchronize their transmissions so that their signals are sent at the same time. 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.

$$[\mathbf{y}] = \begin{pmatrix} [x_{gps}-\varepsilon_{x,gps}, x_{gps}+\varepsilon_{x,gps}] \\ [y_{gps}-\varepsilon_{y,gps}, y_{gps}+\varepsilon_{y,gps}] \end{pmatrix} \tag{11}$$

4.2. The bounded displacement model

The initial imprecise configuration of the vehicle is represented by a box $[\mathbf{x}] = ([x][y][\theta])^T$. (x, y) are the coordinates of the rear axle center and θ is the orientation of a local frame attached to the vehicle. The prediction step consists in moving a box between the steps $k-1$ and k:

$$\left[x_k^{pred}\right] = \begin{Bmatrix} [x_{k-1}]+[\delta s_k]\cos([\theta_{k-1}]+\frac{[\delta\theta_k]}{2}) \\ [y_{k-1}]+[\delta s_k]\sin([\theta_{k-1}]+\frac{[\delta\theta_k]}{2}) \\ [\theta_{k-1}]+[\delta\theta_k] \end{Bmatrix} \tag{12}$$

where [ ] are intervals including the real values. $[\delta s_k]$ is obtained from the odometers measurement:

$$[\delta s_k] = \frac{[\pi]([w_l][\delta p_l]+[w_r][\delta p_r])}{P} \tag{13}$$

where w_r stands for the radius of the right wheel, w_l is the radius of the left wheel and P represents the odometer's resolution.

4.3. The CSP

We consider from time k-w+1 to time k (the current time) all the state equations: the contractions are done in a window of length w and the CSP is defined by 3×w constraints. The constraints at time k are given by:

$$\begin{cases} [x_k] = [x_{k-1}]+[\delta s_k]\cos([\theta_{k-1}]+\frac{[\delta\theta_k]}{2}) \\ [y_k] = [y_{k-1}]+[\delta s_k]\sin([\theta_{k-1}]+\frac{[\delta\theta_k]}{2}) \\ [\theta_k] = [\theta_{k-1}]+[\delta\theta_k] \end{cases} \tag{14}$$

where

- $[\delta s_k]$, $[\delta\theta_k]$ are given by the odometers and gyro measurements thanks to Equations (13) and (10).
- $[x_k]$ and $[y_k]$ are initialized with GPS measurements (see Equation (11)).
- $[\theta_k]$ is initialized to $]-\pi, \pi]$.
- $[x_{k-1}]$, $[y_{k-1}]$ and $[\theta_{k-1}]$ are obtained from the resolution of the previous CSP (at time $k-1$).

5. Results

HC4, BC3, BC4 and 3B-BC4 have been used in order to solve the CSP (14). We had used C source code from the RealPaver Solver [29] on a PC equipped with an Intel Core i7 CPU 960 @ 3.20GHz running under Ubuntu 12.04 LTS. The solving has been realized off-line, for each algorithm, with a set of real data that have been collected on the Satory test track (Fig. 1) with LIVIC's prototype running at an average speed of 50km/h. A solid-state vertical gyro VG400CC provides the yaw rate data. Thanks to an Ag GPS132, the global localization is performed. GPS measurements and gyro/odometer data acquisitions are realized at a 5Hz frequency; timestamped data are used off-line. The centimeter reference used for the evaluation of the positioning method is a RTK GPS. In order to determine the best suited windows size *w*, *w* has been varied between 1 to 200 (see Fig. 2). We had computed the average area size of the localization box for one lap's track. The graph of Fig. 2 corresponds to an hyperbolic function. The higher is the window size, the lower is the localization error. The area decreases of 15% (from 760 to 650 m^2) for *w=20*. Enlarging the *w* value only improves of 4% the size of the area.

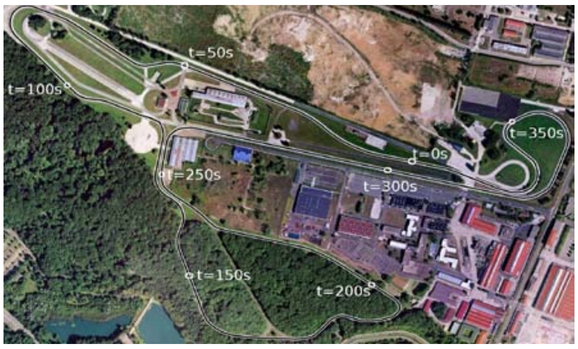

Figure 1: Satory's track

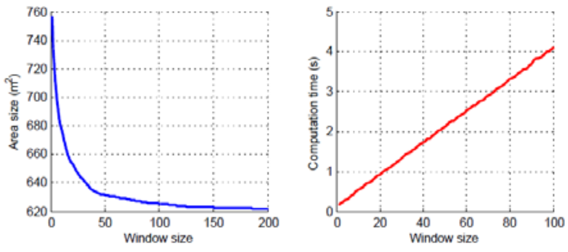

Figure 2: Average localization area and computing time

Table 1. Duration of the algorithm for a CSP.

Algorithms	*HC4*	*BC3*	*BC4*	*3B-BC4*
Durations (ms)	0.52	1.02	0.58	1400

Fig. 2 shows that the computing time increases linearly with *w*. Computing time has been measured for a full lap: 2000 CSPs have been solved. Similar results has been obtained for BC3 and BC4. Table 1 shows the average time taken by the algorithms for solving the CSP at every time step. Each CSP has *3w* equations (we had chosen *w = 20*). HC4 uses forward/backward and is the quickest algorithm (each CSP is solved in 0.52 ms). BC3 is more time consuming than HC4: bisection is less time efficient than forward/backward. BC4 fights the slowness of BC3 by using HC4 when a variable occurs only once. Equation (14) shows that each variable occurs only once on each line of the CSP. Consequently BC3 uses only HC4 and the computing times are similar. The little difference is due to the BC4 embedded test which chooses between HC4 and BC3. 3B-BC4 should be more efficient for the tightening of the variables; unfortunately, it suffers from a prohibitive computing time and cannot be real time (it takes 9.4 s for a 200 ms experiment). The interval error of an estimated state a is defined by $[\underline{a}\text{-}a_{ref}\ ,\overline{a}\ \text{-}a_{ref}]$ where a_{ref} is the reference state. Consequently, a localization algorithm exhibits good results if its corridor is thin and if it always includes the zero value (it means that the algorithm imprecision embraces the reference).

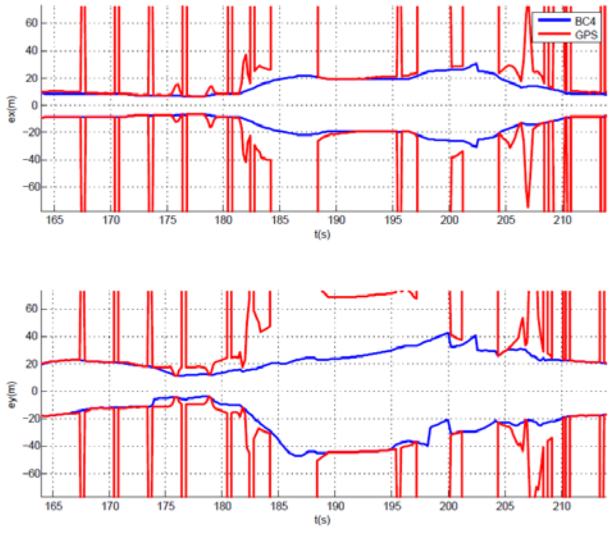

Figure 3: Interval error

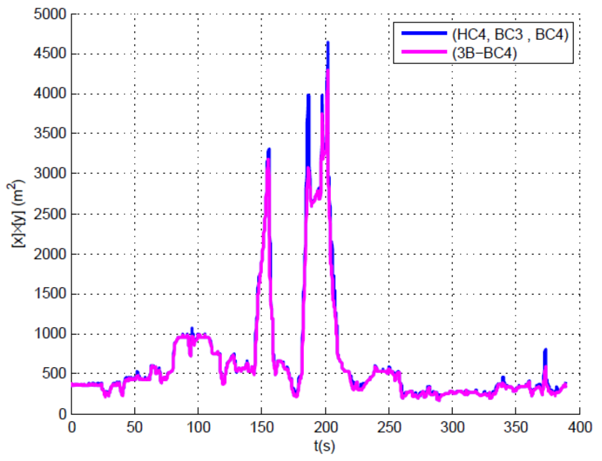

Figure 4: Size of the localization area

The GPS sensor provides consistent measurements (measurements which embrace the reference) with important variations (see Fig. 3). BC4 smoothes in the GPS measurements and acts like a classical Bayesian filter (for instance, an Extended Kalman Filter). HC4, BC3 and BC4 have similar localization results (see Fig. 4). 3B-BC4 provides slightly better localization results than the other algorithms. All the compared algorithms provide consistent results contrarily to Bayesian filters that can lose their consistency [30].

6. Conclusion

Constraint propagation algorithms are often used to solve once an unique system. We have used them on evolving CSPs having a time window in order to solve a localization problem. We aim at localizing a vehicle equipped with odometers, gyro and GPS. At every time step, the imprecise displacement of the vehicle is measured and the position of the vehicle is corrected with a GPS measurement. When a GPS measurement is available, we generate a new small CSP (3 equations) which is added to the current CSP where the three oldest lines are deleted. We have compared 4 constraint propagation algorithms (HC4, BC3, BC4 and 3B-BC4) on the same experimental data. Contrarily to previous works, we shows that it is not necessary to break the constraints into elementary ones [21]. Furthermore, avoiding the use of elementary constraints do not increase the computing time (with ref. to [21]). The 4 tested algorithms provided similar results, although 3B-BC4 provides slightly better contractions than other ones. HC4 and BC4 have similar computing time. BC3 is twice slower than HC4 and BC4. 3B-BC4 is too slow and is not suited for real time issue. Consequently, we recommend the use of HC4 or BC4 for vehicle localization. Further work will deal with an automatic adjustment of the CSP window in order to obtain the best result according to the CPU and the data flow.

REFERENCES

[1] M. Clowes, "On seeing things," Artificial intelligence, vol. 2,no. 1, pp. 79–116, 1971.

[2] D. Waltz, "Understanding line drawings of scenes with shadows," Psychology of Computer Vision, McGraw-Hill,New York, 1975.

[3] A. Mackworth, "Consistency in networks of relations," Artificial intelligence, vol. 8, no. 1, pp. 99–118, 1977.

[4] H. Gallaire, "Logic programming: Further developments," in IEEE Symposium on Logic Programming, pp. 88–99, 1985.

[5] J. Jaffar and J. Lassez, "Constraint logic programming," in ACM SIGACT-SIGPLAN symposium on Principles of programming languages, pp. 111–119, 1987.

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

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

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

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

[10] F. Rossi, C. Petrie, and V. Dhar, "On the equivalence of constraint satisfaction problems," in European Conference on Artificial Intelligence, pp. 550–556, 1990.

[11] R. Mohr, G. Masini, et al., "Good old discrete relaxation," in European Conference on Artificial Intelligence, pp. 651–656, 1988.

[12] J. Cleary, "Logical arithmetic," Future computing systems, vol. 2, no. 2, pp. 125–149, 1987.

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

[14] E. Hyvonen, "Constraint reasoning based on interval arithmetic," in International Joint Conference on Artificial Intelligence, pp. 1193–1198, 1989.

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

[16] F. Benhamou, F. Goualard, L. Granvilliers, and J. Puget, "Revising hull and box consistency," in Int. Conf. on Logic Programming, Citeseer, 1999.

[17] F. Benhamon, D. McAllester, and P. Van Hentenryck, "Clp (intervals) revisited," tech. rep., Citeseer, 1994.

[18] L. Granvilliers, "Towards cooperative interval narrowing," Frontiers of Combining Systems, pp. 18–31, 2000.

[19] O. Lhomme, "Consistency techniques for numeric csps," in International Joint Conference on Artificial Intelligence, pp. 232–232,1993.

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

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

[22] 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, 2009.

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

[24] D. L. Waltz, "Generating semantic descriptions from drawings of scenes with shadows," PhD thesis, AI Lab, MIT, 1972.

[25] 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. 10, no. 3-4, 2012.

[26] G. Chabert and L. Jaulin, "Hull consistency under monotonicity," International Conference on Principles and Practice of Constraint Programming, pp. 188–195, 2009.

[27] R. Moore, Interval analysis. Prentice-Hall Englewood Cliffs, NJ, 1966.

[28] V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, Computational complexity and feasibility of data processing and interval computations. Kluwer Academic Publishers Netherlands, 1998.

[29] 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.

[30] A. Lambert, D. Gruyer, B. Vincke, and E. Seignez, "Consistent outdoor vehicle localization by bounded-error state estimation,"

in IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1211–1216, 2009. Best Paper Award Finalist.