The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Consistent Outdoor Vehicle Localization by Bounded-Error State Estimation

Alain Lambert, Dominique Gruyer, Bastien Vincke, Emmanuel Seignez

Abstract— **Localization is a part of many automotive applications where safety is of crucial importance. We think that the best way to guarantee the safety in these applications is to guarantee the results of their embedded localization algorithms. Unfortunately localization of vehicles is mostly solved by Bayesian methods which (due to their structure) can only guarantee their results in a probabilistic way. Interval analysis allows an alternative approach with bounded-error state estimation. Such an approach provides a bounded set of configurations that is guaranteed to surround the actual vehicle configuration. We have validated the bounded-error state estimation with an outdoor vehicle equipped with odometers, a GPS receiver and a gyro. With the experimental results we compare the bounded-error state estimation with the particle filter localization in terms of consistency and computation time.**

I. INTRODUCTION

Most of the works dealing with the localization problem use both proprioceptive and exteroceptive sensors [3]. Using proprioceptive sensors to calculate the vehicle trajectory generates a cumulative error as the vehicles moves. Consequently, only using proprioceptive sensors does not give a positioning which is satisfactory enough to be used in higher level task like path following or path planning. To cope with this problem, all localization processes also use exteroceptive sensors to improve the predicted configurations of the vehicle. Thus, localization processes are classically broken down into two steps called iteratively. The resulting algorithms alternate a prediction stage using proprioceptive sensors and a correction stage using exteroceptive sensors.

The most commonly used methods are based on Kalman filtering [17], [9], [22], [8] and Markov localization, using either a probability grid [5] or particle filtering [2], [28].
Kalman filtering [7] (in its basic implementation) does neither require a great computation power nor great memory capacity. In return, it cannot perform a global localization and it can only track one vehicle configuration. Moreover, with its version dedicated to the non linear model (the Extended Kalman Filter), it can diverge very swiftly. Often, this issue persists even using methods developed to overcome some of its limitations, as reported in [14].
The Markov localization methods require more computing power but provide more reliable estimated configurations (especially in complex, dynamic or badly mapped environments [6]). They have been reigning for the last few years and the research to improve them is still going on, especially the Monte Carlo localization [29]. These methods evaluate the probability for a vehicle to be positioned in a given region but nothing ensures that the vehicle is indeed in the configuration with the highest probability.

Bounded-error state estimation [15] is an alternative and less known method which has been proposed for initial localization [23] and tracking [18] applications. This method is based on the SIVIA algorithm [16] and takes its roots on R. E. Moore's work [24] on interval analysis. [24] propose to represent a solution to a problem by an interval in which the real solution is guaranteed to be. Thus, whereas most of localization methods provide probabilistic results, bounded-error state estimation gives a set of bounded configurations in which the vehicle is guaranteed to be.
In bounded-error state estimation, all model and measurement errors are assumed to be bounded, with known bounds. At each time instant, the bounded-error recursive state estimation provides a set containing all possible vehicle configurations given the measurements and the noise bounds. The methodology has proved its feasibility in simulations [18]. Experiments have demonstrated that this method can be made operational on a mobile robot navigating in an indoor environment [26], [27]. This paper extends such a work by providing results for an outdoor vehicle (navigating at higher speeds than an indoor mobile robot and using different sensors). Similar works [4] do not give computation times. Other publications [10] claim that the predictor/estimator used in [18], [4] is very slow and thus is not adapted to a real time implementation. The goal of this paper is twofold: we show with our experimental data that the method achieves the real time constraints and we provide a comparison with the popular particle filter approach in term of computation time and consistency.

Section II describes the necessary mathematical tools based on interval analysis. Then, section III presents the subpavings which are used in order to represent the solution set. Section IV describes the localization process. Finally, Section V shows the experimental results of bounded-error state estimation and compares it with a particle filter based localization.

II. INTERVAL ANALYSIS

A. Overview

Bounded-error state estimation is based on interval analysis. Interval analysis was introduced in the sixties in order to avoid approximations in calculations. R.E. Moore [24] proposes to represent a solution of a problem by an interval

A. Lambert and B. Vincke are with IEF, UMR CNRS 8622, University of Paris Sud-XI, Bat. 220, Centre d'Orsay, 91405 Orsay cedex - France Alain.Lambert@u-psud.fr

D. Gruyer is with LIVIC, INRETS/LCPC, Bat 824, 14 route de la minière, 78000 Versailles Satory - France

E. Seignez is with the Ecole Supérieure d'Ingénieurs en Electronique et Electrotechnique, 14 quai de la Somme, 80000 Amiens - France

in which the real solution is guaranteed to be. Interval analysis provides a set of rules to calculate with intervals $[x] = [\underline{x}, \overline{x}] \subset \mathbb{R}$ where $\underline{x}$ and $\overline{x}$ are respectively the minimum and the maximum of $[x]$. The width of an interval is $w[x] = \overline{x} - \underline{x}$. Arithmetical operations ($+$, $-$, * and $/$) and standard mathematical functions readily extend to intervals. For example,

$$\begin{aligned} [1,2] + [3,4] &= [4,6] \\ \ln([1,e]) &= [0,1]. \end{aligned} \quad (1)$$

B. A free library

The computation with interval analysis is simplified by the use of PROFIL/BIAS (Programmer's Runtime Optimized Fast Interval Library / Basic Interval Arithmetic Subroutines) [20], [19], [1], a C++ class library supporting the most commonly needed interval and real operations in a user friendly way. This library allows manipulating intervals as numbers. All basic mathematical functions are implemented to accept numbers as well as intervals.

C. Inclusion function

The notion of *inclusion function* is one of the most important tools provided by interval analysis [15]. For any function $f : \mathcal{D} \subset \mathbb{R} \rightarrow \mathbb{R}$ defined as combinations of arithmetical operators and elementary functions, interval analysis makes it possible to build inclusion functions $f_{[]}$ satisfying

$$\forall [x] \subset \mathcal{D}, f([x]) \subset f_{[]}([x]), \quad (2)$$

where $f([x])$ denotes the set of all values taken by $f(.)$ over $[x]$.

The simpliest way to obtain an inclusion function is to replace all real variables by interval ones and all real-valued operators or elementary functions by their interval counterparts. The *natural inclusion function* is then obtained. For example, the natural inclusion function for

$$f(x) = x^2 - x + 1 \quad (3)$$

is

$$f_{[]}([x]) = [x]^2 - [x] + 1. \quad (4)$$

It is then possible to enclose the set of all values taken by a function over a given interval into a computable image interval.

For *convergent* functions (all functions considered in this work are convergent), the width of the image interval tends to zero when the width of the corresponding argument interval tends to zero. As a consequence, cutting the interval into smaller intervals improves the result of the inclusion function (see [25], [15]). For instance using Eq. (3), $f_{[]}([0,2])$ gives a result of $[-1,5]$. Whereas a calculation of $f_{[]}([0,1]) \cup f([1,2])$ gives an interval of $[0,4]$ better than $f_{[]}([0,2])$.

Outer-approximation of sets may be achieved by a union of non-overlapping boxes or *subpaving*. Subpaving combined with direct image evaluation and inverse image evaluation algorithms are the building stones of the bounded-error state estimation algorithm.

III. Dealing with subpavings

A. Introduction

The vehicle configuration in the global frame is denoted by $\mathbf{x} = (x, y, \theta)^{\mathrm{T}}$ where (x, y) are the coordinates of the rear axle center and θ specifies the orientation of a local frame attached to the vehicle with respect to the global frame.

To estimate and handle the sets implied in our problem, we use the concept of boxes as bounded configurations: a box $[\mathbf{x}] \in \mathbb{R}^3$ is composed of 3 intervals $[\mathbf{x}] = [x] \times [y] \times [\theta]$. Then, subpavings are a union of non-overlapping bounded configurations.

B. Use of a subpaving in the localization process

The principle of localization consists of testing the binary relevance of each box according to the data returned by the exteroceptive sensors. We verify if a box (or a part of it) is compatible with the measurement. If the answer is positive then the box is kept, if not the box is discarded. If only a part of the box is compatible with the measurement, then the box is cut into smaller boxes making it possible to improve the description of the solution. The set of vehicle configurations is either represented as a list in which overlapping boxes are present (a temporary representation during the prediction step), or as a binary tree which avoids overlapping boxes. For the tree representation, each node has the description of a box and the box located at the root of the tree contains all the boxes in the tree.

C. Tree organization

The binary tree is built by dichotomy: a box is cut into two children boxes by interval bisection. One dimension of the box is cut at the middle to give two boxes. These boxes have equal subinterval lengths in one dimension and unchanged lenghts in the other dimensions. We chose to cut the box on the largest length. For instance, cutting the root box $[\mathbf{x}]$ among y dimension leads to two boxes named $L[\mathbf{x}]$ and $R[\mathbf{x}]$ (corresponding respectively to the left and the right children):

$$\begin{aligned} L[\mathbf{x}] &= [\underline{x}, \overline{x}] \times [\underline{y}, \tfrac{\underline{y}+\overline{y}}{2}] \times [\underline{\theta}, \overline{\theta}], \\ R[\mathbf{x}] &= [\underline{x}, \overline{x}] \times [\tfrac{\underline{y}+\overline{y}}{2}, \overline{y}] \times [\underline{\theta}, \overline{\theta}]. \end{aligned} \quad (5)$$

Consequently, the root of the tree is the largest box and going down the tree leads to smaller boxes. This property is used by the correction step of the localization algorithm to quickly discard unsuited areas.

IV. Localization process

The previously described mathematical tools and the notions of boxes/subpaving are now applied to the localization process divided in a prediction/correction scheme.

A. Prediction step

The prediction step uses the measurements of two proprioceptive sensors of our experimental platform: the speed of the rear wheels and the yaw rate.

1) Proprioceptive data integration: A non linear discrete-time state-space model is considered to describe the evolution of the configuration $\mathbf{x}_k$ of the vehicle

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{v}_k), \tag{6}$$

where $\mathbf{u}_k$ is a known two-dimensional control vector, assumed constant between the times indexed by $k-1$ and k and $\mathbf{v}_k$ is an unknown state perturbation vector that accounts for the model uncertainties. The classical evolution model, described in [21], is considered:

$$\mathbf{f}(\mathbf{x}_{k-1}) = \begin{pmatrix} x_{k-1} + \delta s. \cos\left(\theta_{k-1} + \frac{\delta\theta}{2}\right) \\ y_{k-1} + \delta s. \sin\left(\theta_{k-1} + \frac{\delta\theta}{2}\right) \\ \theta_{k-1} + \delta\theta. \end{pmatrix}, \tag{7}$$

where δs is the longitudinal motion and $\delta\theta$ is the rotational motion.

$\mathcal{X}$ is a subpaving of $\mathbb{R}^3$ in which the vehicle is guaranteed to be. The integration of the proprioceptive data is done by founding the subpaving that includes $\mathcal{Y} = \mathbf{f}(\mathcal{X})$ where $\mathbf{f}$ is a known non linear function. $\mathcal{X}_{k|k-1}$, the set that is consistent with sensor data provided at time k knowing $k-1$, is computed using the following equation:

$$\begin{aligned} \mathcal{X}_{k|k-1} &= \mathbf{f}\left(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{v}_k\right) \\ &\quad \text{with} \left\{ \begin{array}{c} \mathbf{x}_{k-1} \in \mathcal{X}_{k-1|k-1}, \\ \mathbf{u}_k + \mathbf{v}_k \in [\mathbf{u}_k + \mathbf{v}_k] \end{array} \right\} \\ &= \mathbf{f}\left(\mathcal{X}_{k-1|k-1}, [\mathbf{u}_k + \mathbf{v}_k]\right). \end{aligned} \tag{8}$$

with $[\mathbf{u}_k + \mathbf{v}_k]$ the state vector $\mathbf{u}_k$ added with the bounded-error $\mathbf{v}_k$ on the measurement.

Applying bounded parameters on Eq. (7) for each box $[\mathbf{x}_{k-1}] \in \mathcal{X}_{k-1|k-1}$ the bounded evolution of the vehicle is described by the following inclusion function:

$$\mathbf{f}_{[]}([\mathbf{x}_{k-1}]) = \begin{pmatrix} [x_{k-1}] + [\delta s] \times \cos\left([\theta_{k-1}] + \frac{[\delta\theta]}{2}\right) \\ [y_{k-1}] + [\delta s] \times \sin\left([\theta_{k-1}] + \frac{[\delta\theta]}{2}\right) \\ [\theta_{k-1}] + [\delta\theta] \end{pmatrix}. \tag{9}$$

$[\mathbf{u}_k + \mathbf{v}_k] = ([\delta s] \;\; [\delta\theta])^{\mathrm{T}}$ is deduced in a bounded way from the longitudinal speed $\dot{s}$ and yaw rate $\dot{\theta}$ by taking into account the elapsed time δt: $[\delta s] = [\dot{s}]\,[\delta t]$ and $[\delta\theta] = \left[\dot{\theta}\right]\,[\delta t]$. As the set of boxes $\mathcal{X}_{k-1|k-1}$ is described by a subpaving and an inclusion function for the prediction function $\mathbf{f}$ is available, an outer-approximation $\mathcal{X}_{k|k-1}$ can be evaluated using the IMAGESP (IMAGE SubPaving) algorithm [15].

2) IMAGESP*:* The IMAGESP algorithm can be decomposed into three steps:

a) Mincing the subpaving: First, the boxes of the subpaving are minced, *i.e.*, all the boxes are bisected until their width is lower than a specified precision parameter ε. Previous works [27] do not clearly state that it is not necessary to bisect the boxes in all their dimensions: only the width θ should be bisected. When a variable occurs only once in an expression then the natural inclusion function is minimal for this variable [24], [25]. This property allows us to significantly decrease the computation time compared with the brute approach of bisecting all widths.

b) Computation of the inclusion function: The images of the resulting boxes are then evaluated using the inclusion function for $\mathbf{f}$ and the PROFIL/BIAS library. According to Eq. (2), the union of these images is guaranteed to contain $\mathbf{f}\left(\mathcal{X}_{k-1|k-1}, [\mathbf{u}_k + \mathbf{v}_k]\right)$. When ε decreases, the image subpaving $\mathcal{X}_{k|k-1}$ gets closer to the optimal subpaving. The cost of a such decreasing is an increasing of the prediction computation time.

c) Merging the boxes: Finally, these images are merged in order to get a subpaving that is guaranteed to contain the configuration of the vehicle. This last step builds a subpaving (a set of non-overlapping boxes) represented as a binary tree. It reduces the number of boxes that are used in the further steps.

B. Correction step

The correction step integrates the data provided by the GPS receiver of the experimental platform.

1) Measurement equation and estimation principle: The measurement equation at time k for the GPS receiver can be described as

$$[\mathbf{y}_k] = \mathbf{h}\left(\mathbf{x}_k\right) + [\mathbf{w}_k], \tag{10}$$

with $\mathbf{h}\left(\mathbf{x}_k\right) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{x}_k = \mathbf{x}_k^{2\times 2}$ and $[\mathbf{w}_k]$ the measurement noise matrix. In short, $[\mathbf{y}_k]$ is returned by the GPS receiver as a point (x, y) with an associated oriented ellipse. In fact the GPS does not directly return a cartesian point but a latitude and longitude coordinates which are converted in a Cartesian local frame. $[\mathbf{w}_k]$ is obtained thanks to the GST NMEA frame.

The output $[\mathbf{y}_k]$, available at time k, can be taken into account by updating $\mathcal{X}_{k|k-1}$ (the previous localization set obtained during the prediction step) into an outer approximation $\mathcal{X}_{k|k}$

$$\mathcal{X}_{k|k} = \mathbf{h}^{-1}\left([\mathbf{y}_k]\right) \cap \mathcal{X}_{k|k-1}. \tag{11}$$

We have described $\mathcal{X}_{k|k-1}$ as a union of boxes. In the same way $\mathcal{X}_{k|k}$ may be obtained as a union of boxes by the use of the SIVIA (Set Inversion Via Interval Analysis) algorithm [15].

2) SIVIA*:* The goal of the Set Inversion Via Interval Analysis is to compute the reciprocal image of a set by a function and to approximate it by an external subpaving. The key idea is to test if the image of the boxes located in a search domain are in the image set. If the images of the boxes lay within or intersect with the image set then they are part of the reciprocal image.

More specifically, starting from the root of the binary tree provided by the prediction step, each box of the tree is evaluated as follows:

- If $\mathbf{h}_{[]}([\mathbf{x}_k]) \subset [\mathbf{y}_k]$, then any $\mathbf{x}_k \in [\mathbf{x}_k]$ is consistent with the measurements and noise bounds and $[\mathbf{x}_k]$ is proved to be in $\mathcal{X}_{k|k}$. $[\mathbf{x}_k]$ is kept in the solution list.

Fig. 1. CARLLa platform

Fig. 2. Embedded hardware architecture

- If $\mathbf{h}_{[]}([\mathbf{x}_k]) = \emptyset$, then there is no $\mathbf{x}_k$ in $[\mathbf{x}_k]$ consistent with the measurements and noise bounds and $[\mathbf{x}_k]$ does not belong to $\mathcal{X}_{k|k}$. $[\mathbf{x}_k]$ is eliminated.
- If $\mathbf{h}_{[]}([\mathbf{x}_k]) \cap [\mathbf{y}_k] \neq \emptyset$ and if $[\mathbf{x}_k] < \varepsilon$, then the box is too little to be cut. At least one configuration in $[\mathbf{x}_k]$ is consistent with the measurements and noise bounds and due to its small dimension, $[\mathbf{x}_k]$ is kept.
- If $\mathbf{h}_{[]}([\mathbf{x}_k]) \cap [\mathbf{y}_k] \neq \emptyset$ and if $[\mathbf{x}_k] > \varepsilon$, the same tests are applied to the right and left children (if they do not exist they are computed).

As this work considers GPS measurements, $\mathbf{h}_{[]}([\mathbf{x}_k]) = [\mathbf{x}_k]^{2\times 2}$ and $[\mathbf{y}_k]$ is the surface of an ellipse. Consequently SIVIA does some simple tests between rectangles and an ellipse.

V. Experimental results

A. The vehicle and its embedded architecture

In order to test and to validate the bounded-error state estimation approach for the positioning of a vehicle on road, a set of real data have been collected with LIVIC's prototype (CARLLa). On this platform (Fig. 1 and 2), both the hardware and the software architectures have been embedded. The hardware architecture includes 2 computers, the first one is dedicated to the proprioceptive sensors and the GPS receivers management. The control/command algorithms and the localization algorithms are also executed on this computer. The second computer is used for the other exteroceptive sensors and for the perception algorithms

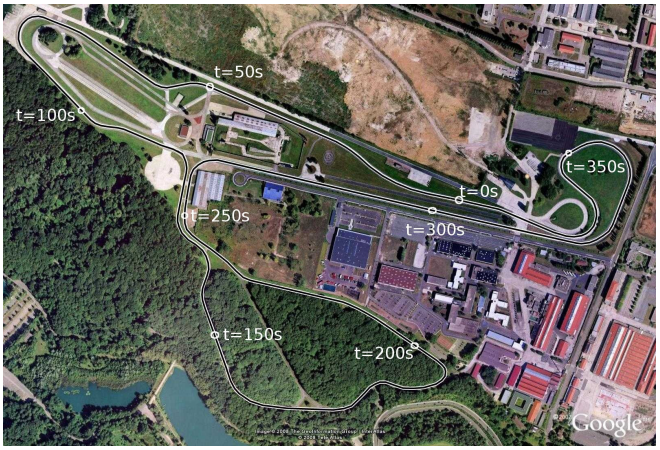

Fig. 3. Satory's tracks

(including image processing algorithms). In this study, we only use the first computer: it collects data (coming from the gyro, the odometers and the GPS receiver) and executes the localization algorithms.
A solid-state vertical gyro VG400CC provides the yaw rate data. The odometry is rebuilt from the data coming from the CAN bus (only the rear wheel speeds are used). The GPS sensor (AgGPS 132) returns a position vector (latitude, longitude) with a covariance matrix. The localization is done in a local tangential frame to surface earth (French Lambert I conic projection) which is the projection of the WGS84 terrestrial global reference frame. In order to collect the data, the RTMaps platform has been used. This platform provides powerful tools in order to collect data in real time with an accurate time stamping.
The reference used for the evaluation of the positioning method is a RTK GPS. This sensor provides a centimeter accuracy of the ego-vehicle. In addition to the RTK reference, we have a very accurate map of these tracks (centimeter accuracy) with its left, right and center road markings. With a camera we have confirmed the lateral accuracy given by the RTK GPS [12], [13].

B. The test tracks

In order to validate the localization algorithms, the data have been collected on the Satory test tracks. The first track is similar to a peri-urban road (on the top of the Fig. 3) and the second one to a rural road (on the bottom of the Fig. 3). The second track is a very interesting area because of the presence of a forest. In this area, major troubles with the AgGPS 132 have been observed due to important signal losses. These troubles have a particularly long duration during our experiments and allows us to heavily test the robustness of our localization algorithms.

C. The localization results

The collected data have been used to localize our vehicle with two algorithms: the Bounded-Error State Estimation (BESE) previously describes and a classical Particle Filter (PF). The implementation of the PF is not described in this

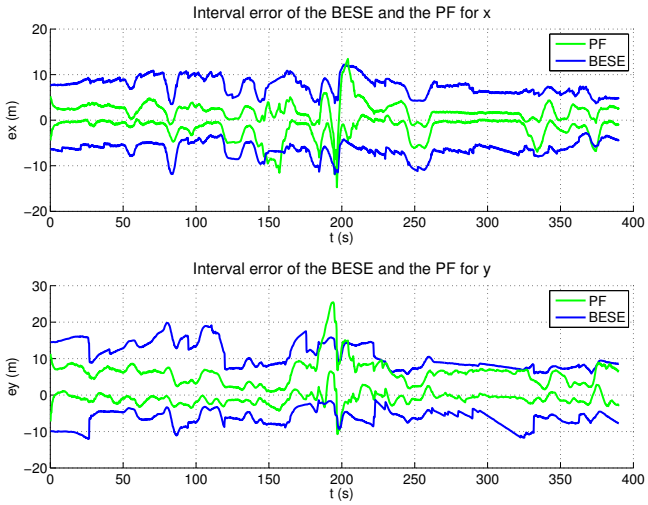

Fig. 4. Comparison between PF and BESE for 3σ error bounds

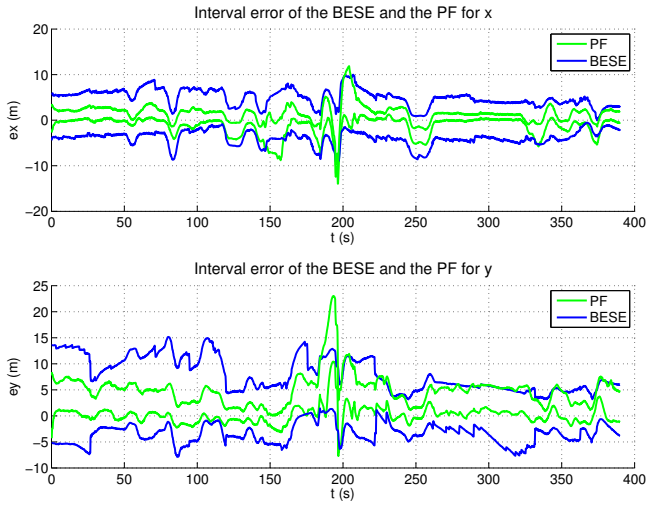

Fig. 6. Comparison between PF and BESE for 2σ error bounds

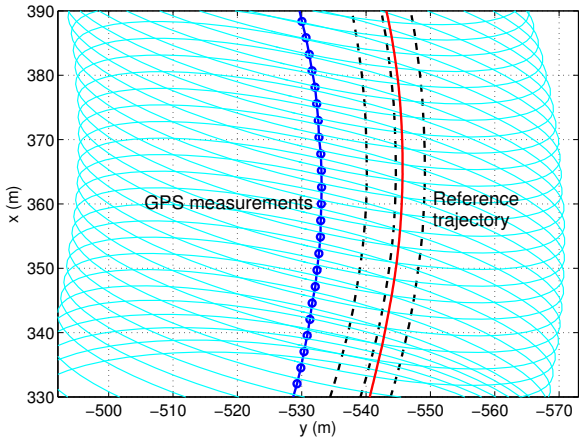

Fig. 5. Biased GPS measurements

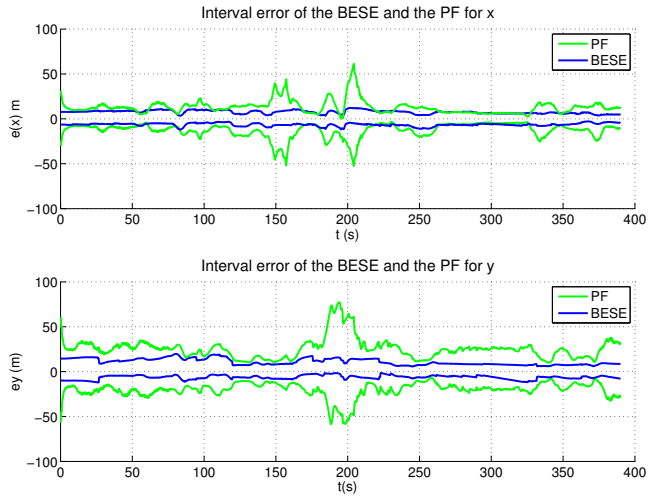

Fig. 7. Comparison between BESE (3σ) and consistent PF with 19σ error bounds

paper. The reader who is not familiar with the subject could read the algorithmic principles in [2] and an application to outdoor vehicles in [11]. Both the BESE and the PF use the same sensor data. But the PF is fed with the standard deviation σ of the sensors whereas the BESE uses a $x\sigma$ error bound where x is defined afterwards. Two main values of x have been used in this paper: $x = 2$ which defines a 0.95 confidence level and $x = 3$ which corresponds to a 0.99 confidence level. Consequently, where we should considers to find a data with a 0.95 or a 0.99 probability, the BESE assumes to find it with a 1 probability. Practical experiments show us that the 3σ bound is definitely safe enough (the true value always lies in the 3σ bound during our experiment).

The vehicle has followed the 7 minutes path drawn on Fig. 3 at the mean speed of 60 kph. Fig. 4 shows the interval error of the BESE and the PF. The imprecision area of the BESE (using a 3σ bound errors) has directly been used to compute the interval error. For the PF the imprecision (standard deviation) has been magnified 3 times. A filter exhibits good results if its corridor is thin and if it always includes the zero value (it means that the filter imprecision embraces the reference value). The PF is more confident than the BESE but the PF is sometimes inconsistent whereas the BESE is always consistent. The great inconsistency of the PF before t= 200s is due to a large number of repeated biased measurements (see Fig. 5). The BESE does not suffer from those biased measurement as the data meet the bounded-error constraints.

The total computation time of both prediction and correction (for the 400s of the experiments) is 68 seconds for the BESE and 24 seconds for the PF (with 1000 particles). The single prediction and correction time is 34ms for the BESE and 12ms for the PF. A higher number of particles for the PF shows higher computation times but similar results. Those computation times emphasize the real time

capability of the BESE for dealing with the outdoor vehicle localization on standard computers.

In order to drive the BESE to the wall, the Figure 6 uses a 2σ error bounds. Not surprisingly the PF is more inconsistent than with the 3σ error bound. We are much more surprised by the results of the BESE which stays consistent most of the time. An analysis of the interval error shows that the BESE is (lightly) inconsistent for t$\in [170, 200]$s and around t$= 225$s. This inconsistency is related to some inconsistent GPS measurements occurring at these times. The most important point is that the BESE quickly recovers from those inconsistency.

Finally we have computed the x value for the PF to be consistent. We found 19 and drew the 19σ interval error on Fig. 7. Thus we can compare both consistent filters. The PF is greatly penalized by its inappropriate great confidence (a thin corridor which is far from the 0 value, see Fig. 4) around t=195s which leads to a huge 19σ interval error to recover from this error. The resulting corridor is up to 100 meters width which is much larger than the consistent BESE.

VI. CONCLUSION

This paper presents a bounded-error state estimation to the localization problem of an outdoor vehicle. Experiments show the ability of the method to solve the localization problem in real time with better consitency then PF. Furthermore we point out that the PF can locally converge towards a wrong solution due to bias measurements which lead to a huge local inconsistency. Similar experiments with an Extend Kalman Filter (EKF) could show the same phenomenon. EKF strongly underestimates its covariance matrix in presence of repeated bias measurements. We claim that this is the biggest advantage of the BESE approach over Bayesian approach. Unfortunately, the price to pay for such advantage is the difficulty of BESE to deal with inconsistent measurements (outliers). This difficulty is not verified in our experiments: BESE quickly recovers from outliers with only small local inconsistency. Furthermore a simple test could allow BESE to detect (and signal to the end user) huge inconsistency in measurement thus avoiding most of the possible failures of the BESE. Nevertheless we cannot guarantee that such failure will not appear. We think that further works should deal with this subject.

REFERENCES

[1] http://www.ti3.tu-harburg.de/software/profilenglisch.html.

[2] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions Signal Processing*, 50(2):174–188, 2002.

[3] J. Borenstein, B. Everett, and L. Feng. *Navigating Mobile Robots: Systems and Techniques*. A. K. Peters Ltd., Wellesley, MA, 1996.

[4] P. Bouron, D. Meizel, and P. Bonnifait. Set-membership non-linear observers with application to vehicle localisation. In *6th European Control Conference. Porto*, pages 1255–1260, 2001.

[5] W. Burgard, D. Fox, D. Hennig, and T. Schmidt. Estimating the absolute position of a mobile robot using position probability grids. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, volume 2, pages 896–901, Portland, OR, 1996.

[6] W. Burgard, D. Fox, and S. Thrun. Active mobile robot localization. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1997.

[7] C.K. Chui and G. Chen. *Kalman filtering with real-time applications*. Springer Verlag, 1990.

[8] I. Cox and editors G. Wilfong. *Autonomous Robot Vehicles*. Springer-Verlag, 1990.

[9] J.L. Crowley. World modeling and position estimation for a mobile robot using ultrasonic ranging. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 674–680, Scottsdale, AZ, 1989.

[10] A. Gning and P. Bonnifait. Constraints propagation techniques on intervals for a guaranteed localization using redundant data. *Automatica*, 42(7):1167–1175, 2006.

[11] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.J. Nordlund. Particle filters for positioning, navigation, and tracking. *IEEE Transactions on signal processing*, 50(2):425–437, 2002.

[12] S. S. Ieng and D. Gruyer. Merging lateral cameras information with proprioceptive sensors in vehicle location gives centimetric precision. In *Enhanced Safety of Vehicles (ESV'03)*, 2003.

[13] S. S. Ieng, J. Vrignon, and D. Gruyer. A new multi-lanes detection using multi-camera for robust vehicle location. In *Intelligent Vehicles Symposium (IV'05)*, pages 700–705, June 6-8, 2005 Las Vegas, Nevada, USA, December 5–9, 2005.

[14] K. Ito and K. Xiong. Gaussian filters for nonlinear filtering problems. In *IEEE Transactions on Automatic Control*, volume 45, pages 910–927, May 2000.

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

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

[17] R.E. Kalman. A new approach to linear filtering and prediction problems. *Transaction of the ASME, Journal of Basic Engineering*, 82:34–45, 1960.

[18] M. Kieffer, L. Jaulin, and E. Walter. Guaranteed recursive non-linear state bounding using interval analysis. *International Journal of Adaptive Control and Signal Processing*, 16(3), 2002.

[19] O. Knuppel. Bias-basic interval arithmetic subroutines. Technical report, 93.3, Harburg-Hamburg, Germany, 1993.

[20] O. Knuppel. Profil-programmers runtime optimized fast interval library. Technical report, 93.4, Harburg-Hamburg, Germany, 1993.

[21] A. Lambert and N. Le Fort-Piat. Safe task planning integrating uncertainties and local maps federation. *International Journal of Robotics Research*, 19(6):597–611, 2000.

[22] J.J. Leonard and H.F. Durant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(3):376–382, 1991.

[23] D. Meizel, O. Leveque, L. Jaulin, and E. Walter. Initial localization by set inversion. *IEEE Transactions on Robotics and Automation*, 18(6):966–971, December 2002.

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

[25] R.E. Moore. *Methods and Applications of Interval Analysis*. SIAM, Philadelphia, PA, 1979.

[26] E. Seignez, M. Kieffer, A. Lambert, E. Walter, and T. Maurin. Experimental vehicle localization by bounded-error state estimation using interval analysis. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 1277–1282, Canada, 2005.

[27] E. Seignez, M. Kieffer, A. Lambert, E. Walter, and T. Maurin. Real-time bounded-error state estimation for vehicle tracking. *International Journal of Robotics Research*, 28(1):34–48, 2009.

[28] S. Thrun, M. Bennewitz, W. Burgard, A. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. Minerva: A second generation mobile tour-guide robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 5, pages 1999–2005, 1999.

[29] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence Journal*, 128(1-2):99–141, 2001.