Development of an Particle Swarm Algorithm for Vehicle Localization

J. Godoy, D. Gruyer, A. Lambert, J. Villagrá

***Abstract*—This paper describes the development of a filter algorithm based on the behaviour of biological swarms. The main goal of the algorithm is to perform vehicle localization by combining the data from different sensors – GPS, IMU, speedometers, etc. – and digital maps. In this sense, the algorithm considers several solutions at the same time like Particles Filters. The algorithm has been developed off-line using real data captured from an instrumented vehicle at LIVIC. Performance of the algorithm has been validated and compared with and EKF with encouraging results.**

I. Introduction

Vehicles localization is one of the most essential task for Intelligent Transport Systems (ITS) applications development, but at the same time it is one of the most complex due to the common both high robustness and high accuracy requirements. Nowadays, Global Positioning Systems (GPS) are considered as the default solution to this problem. Nevertheless, it is also well know that GPS measurements are commonly affected by interference, occultation and reflection in satellites signal reception – e.g. trees, buildings, meteorological conditions, etc – reducing considerably its accuracy [1]. For most of the cases, this issue can be corrected by implementing differential systems - DGPS; however, these solutions are quite expensive and could reduce the system autonomy. For these reasons, architectures combining both proprioceptive and exterioceptive data from different sensors – e.g. GPS, INS; LIDAR, cameras – are increasingly used.

Up to now, different filter algorithms have been presented as solutions for merging data, being the Kalman Filter (KF) the most popular method [2]. Extended (EKF) and Unscented (UKF) [3] approaches allow removing some KF limitations, as model linearity, improving filtering performance and applicability. Both EKF and UKF have been widely applied to vehicle localization in different works [4], [5], [6]. Despite its great acceptance, Gaussian noise assumption by KF evolution and measure models could be a limiting factor for some applications as vehicle localizations. This has resulted in the application of other filtering methods as Particle Filter (PF).

Particles filters were introduced at the beginning of 90's to solve several filtering problems [7]. Unlike KF, particle filter represent several possible states of the system at the same time. Each one of these states – called particles – is associated to a weight representing its degree of confidence of being the real state of the system. Nowadays several versions of PF can be found in the literature, being the best known the Sampling Importance Resampling filter (SIR). The importance of this last algorithm lies in the resampling process, which solve the degeneration of the particles – after several iterations all but one particle have negligible weight – by propagating particles with high weights and eliminating those with the smallest ones [8]. [9], [10] are examples of applications that apply PF for localization.

Another solution used to improve PF performance is the Particle Swarm Optimization (PSO). PSO is an optimization method based on the behaviour of biological swarms – e.g. flock of birds or schooling fishes – [11]. In this method a set of potential solutions – called particles – are initialized inside the search space. Each particle moves through the search space seeking the optimal solution, considering both their own experience and the best experience of the swarm. In [12] the idea of a Swarm Optimized Particle Filter was introduced. In this approach the main goal was to attract particles to a high likelihood region when the best global value and previous value for each particle were both very small. To this end, PSO was embedded as a continuous attraction loop in the sampling stage of PF that, once activated, updated directly the position and velocities for each particle until the best goal value reached a defined threshold.

In this paper, the development of a novel localization algorithm based on PSO is presented. This algorithm combines data from several sensors and digital maps to perform vehicle localization. The algorithm has been developed off-line using real data captured from a sensorised vehicle during a human-drive lap. The paper is organized as follows: section II describes the general idea and main goals of the algorithm. Section III describes step by step the development of the algorithm. In section IV the results obtained are shown and a comparison with the EKF is presented. Finally, conclusions and future work are presented in section V.

II. Goals and general idea

As introduced before, the main goal for this work is to develop a new localization algorithm based on emulation of biological swarm behaviour. As other localization algorithms, this approach – henceforth called Particle Swarm Localization algorithm (PSL) – will merge the information obtained through different sensors in order to estimate the vehicle state – position and orientation – in real time. For this, a set of N individuals or particles – from now called Localization Swarm

J. Godoy and J. VIllagrá are with Programa AUTOPÍA at Centro de Automática y Robótica, Consejo Superior de Investigaciones Científicas, 28500 Madrid, Spain {jorge.godoy, jorge.villagra}@csic.es

D. Gruyer and A. Lambert are with Laboratoire sur les Interactions Véhicules-Infrastructure-Conducteurs (LIVIC), 78000 Versailles, France. {dominique.gruyer, alain.lambert}@ifsttar.fr

(LS) – representing possible solutions to the vehicle state will be defined and evolved step by step according to the data received from sensors.

As secondary goal, it is desired to reduce the need of high accuracy GPS measurements as a way to reduce the final cost of application by implementing low-cost equipment. Additionally, this will increase the robustness of the algorithm to errors in GPS measurements or unavailability of them.

From position point of view, final algorithm should work as follow: at first step, LS's particles must be initialized inside an uncertainty area where the vehicle is supposed to be. As different measurements are obtained, individuals state are evolved according to their own estimated but also following the swarm estimation. When a particle evolves, also does its probability of being a solution for the vehicle state. If this probability is considered as an attraction weight by swarm estimation, this means that heavier particles would attract the lighter ones to them, decreasing the uncertainty area after several steps. Likewise, the vehicle heading would be adjusted.

III. Algorithm development

This section describes the development of PSL algorithm. In order to be able to develop the algorithm off-line, data from different sensors equipped on a LIVIC test vehicle was recorded during a human-drive lap over the Satory's tracks. Fig 1 shows an image of both the vehicle and the tracks. The sensors implemented in the vehicle are a GPS Trimble AG132, an INS Crossbow VG400, the vehicle odometer and steering wheel angle. Sensor data from all the sensors have been synchronously captured at GPS rate. Additionally, data from a RTK GPS and a high accuracy INS have been captured at the same time to provide the reference trajectory of the test lap and compare the algorithm results.

A. Modelling and state representation

As for other algorithms applied to vehicle localization, it is necessary to implement an evolution model that defines the state transition from current step to the next one. In other approaches as KF, this model is implemented in the prediction stage. For PSL, vehicle state – and hence particle state – will be defined as the vector state $[x_k \;\; y_k \;\; \psi_k]^T$, where x_k and y_k represent vehicle position at step k in the reference frame $[X \;\; Y \;\; Z]$ and ψ_k its orientation or yaw angle. The evolution between two sequential steps will be described by the kinematic bicycle model:

$$\begin{cases} x_k = x_{k-1} + S_k \cdot \Delta t \cdot \cos\left(\psi_{k-1} + \frac{\Delta t \cdot \dot{\psi}_k}{2}\right) \cdot \cos(\delta_k) \\ y_k = y_{k-1} + S_k \cdot \Delta t \cdot \sin\left(\psi_{k-1} + \frac{\Delta t \cdot \dot{\psi}_k}{2}\right) \cdot \cos(\delta_k) \\ \psi_k = \psi_{k-1} + \Delta t \cdot \dot{\psi}_k \end{cases} \quad (1)$$

where Δt is the time period, δ_k is the steering front wheel angle, S_k is the vehicle speed and $\dot{\psi}_k$ is the yaw rate. Inclusion of the steering front wheel angle allows taking into account the kinematic constraints of vehicle.

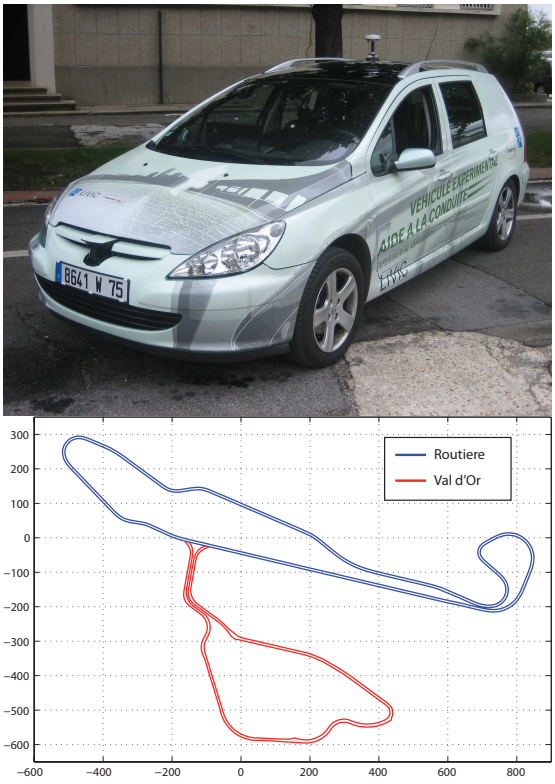

Fig. 1. Top: LIVIC Vehicle. Bottom: Satory's test tracks.

B. Swarm attraction

Having defined the state representation and evolution model for PSL, the next step is the definition of the attraction rule for swarm particles. As was mentioned in section II, this algorithm considers particles probability through weight definition, similar to PF.

As first approach, it was intended to apply swarm attraction directly over the *a priori* estimated states. However, since the swarm estimation is represented by a particle, this would mean that after several iterations all the swarm particles would converge to one single state, losing all the swarm variance. Other solutions, as attraction radius were not considered since they would just solve the problem partially, reducing algorithm swarm functionality. From these statements, a "speed" attraction was proposed and implemented.

Consider that at step $k-1$ we have a swarm consisting of 4 particles A, B, C, D which states are X^A_{k-1}, X^B_{k-1}, X^C_{k-1}, X^D_{k-1}; and probability weights W^A_{k-1}, W^B_{k-1}, W^C_{k-1}, W^D_{k-1} respectively – see Fig. 2a. Likewise, we would have a swarm estimation X^{sw}_{k-1} obtained from particles weighted addition with its own probability weigth W^{sw}_{k-1}. All weights have been updated from sensor information. At step k, all particles, including swarm estimation, evolve according to the process model, getting a set of *a priori* estimated particles: $\hat{X}^A_k$, $\hat{X}^B_k$, $\hat{X}^C_k$, $\hat{X}^D_k$ and $\hat{X}^{sw}_k$ – see Fig. 2b. An *a priori* state transition between steps *k-1* and *k* is then described by an *a priori* velocity vector for each particle: $\hat{V}^A$, $\hat{V}^B$, $\hat{V}^C$, $\hat{V}^D$, and $\hat{V}^{sw}$; being the last one the swarm movement – Fig. 2c.

In order to obtain *a posteriori* estimations, all *a priori* velocity vectors must be modified according to their weight

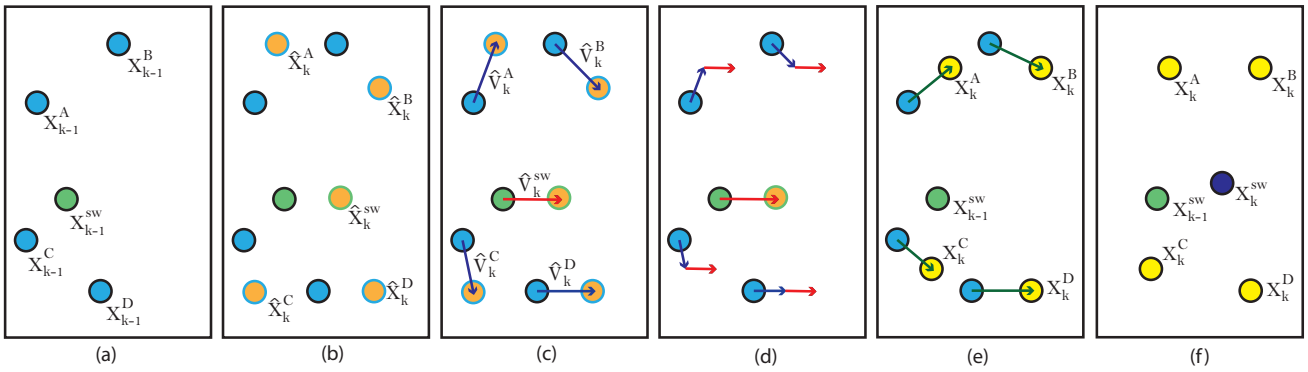

Fig. 2. Particles' evolution for PSL algorithm.

relationship with the swarm state estimation $\hat{X}_k^{sw}$. In this way, vectors of lighter particles would be more attracted than those of the heavier ones – Fig. 2d. Finally, *a posteriori* particles X_k^A, X_k^B, X_k^C and X_k^D can be generated by adding their corresponding *a posteriori* velocity vector – Fig. 2e. As final step, the new swarm estimation X_k^{sw} is calculated from the new particles states and weights are updated from sensor information – Fig. 2f.

By implementing this kind of attraction it is guaranteed that all particles follow swarm behaviour without converging to a single state after a few iterations. Additionally, definition of an inertia weight $W^{Inertia}$ for each particle helps low-weight particles not being attracted to swarm estimation in one iteration. Final equations are showed below:

$$\hat{V}_k^i = \hat{X}_k^i - X_{k-1}^i \; ; \; \hat{V}_k^{sw} = \hat{X}_k^{sw} - X_{k-1}^{sw}$$

$$V_k^i = \frac{\left(W_k^i + W^{Inertia}\right) \cdot \hat{V}_k^i + W_k^{sw} \cdot V_k^{sw}}{W_k^i + W^{Inertia} + W_k^{sw}}$$

$$X_k^i = X_{k-1}^i + V_k^i$$

C. Weight definition

As was mentioned in the previous section, particle likelihood of being a plausible solution to vehicle state would be represented by a particle weight by PSL algorithm. At this point, data from odometer and steering wheel sensor is already used for model evolution, meaning that GPS measurements would be used for weighting process. However, since one of the goals is to reduce the GPS accuracy dependency, data from simple digital maps would be implemented for particle weighting. The final particle weight is then defined by a combination of different auxiliary weights.

1) GPS Weight: Each GPS measurement has an uncertainty associated. This information can be easily obtained in most of the commercial equipments from NMEA message GPGST or by using typical values for each GPS mode – i.e. RTK, DGPS or standalone. If message is provided, three additional fields can be associated to each measurement: semi-major and semi minor standard deviation, and semi-major orientation. By using this data, an uncertainty ellipsoid can be defined around the measurement. In order to maximize the probability of the real position of being inside the ellipsoid up to 99%, both semi-major and semi-minor axes must be multiply by 3.

In this algorithm, a temporal binary weight $Tgps$ is assigned to each particle as soon as a GPS measurement is processed. This weight indicates which particle is inside the uncertainty ellipsoid and which is not. After $Tgps$ assignation, GPS weight is updated considering an inertial relationship $Rgps$ with respect to the last value – see Eq. (2).

$$Wgps_k^i = Rgps_k \cdot Wgps_{k-1}^i + (1 - Rgps_k) \cdot Tgps^i \quad (2)$$

2) Map weight: Since GPS measurements have been implemented as an uncertainty area instead of a single point, it was necessary to merge additional information in the algorithm for particles weighting and to reduce the uncertainty level of the algorithm solution. To achieve such task, data from digital maps of the LIVIC test tracks have been used during the algorithm development. Each map is composed of two sets of data – one for each path edge – being the separation between samples around 1.5 metres for curve sections and 45 metres for straight ones.

For map weighting a constraint has been added to the algorithm: wherever the vehicle is located, it must be over the road and never outside it. From this statement, a temporal binary weight $Tmap$ is defined for each particle every step to indicate whether it is inside the map or not. As for GPS weighting, after $Tmap$ assignation the map weight $Wmap$ is then updated considering an inertial relationship $Rmap$ with the last value – see Eq. (3).

$$Wmap_k^i = Rmap_k \cdot Wmap_{k-1}^i + (1 - Rmap_k) \cdot Tmap^i \quad (3)$$

3) Particle final weight: Having defined auxiliary weights based on GPS and map information, the next step is final weight definition. From the algorithm assumption that the vehicle is always over the road, it is obvious that particles satisfying both conditions – over the road and inside GPS ellipsoid – should have a higher probability of being a plausible solution than those that only satisfy one of them. Thus, final weight should be defined as the result of GPS and map weights multiplication. However, if only this product is considered, particles satisfying one condition would have the same weight that those satisfying none of them. From these statements,

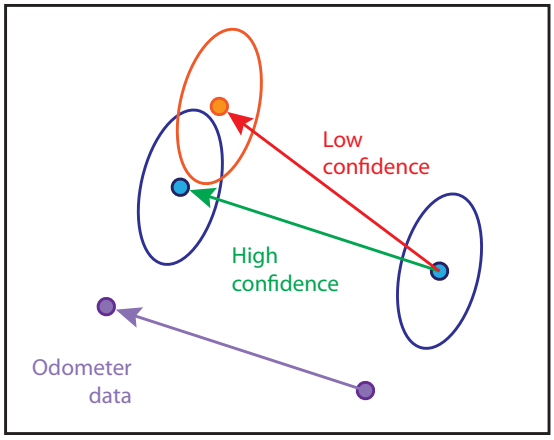

Fig. 3. Estimation of GPS confidence

the combination of both individual an product weights is considered for the final weight estimation. Eq. (4) shows the relation proposed as first approach, being G_{gps}, G_{map} and G_{gam} the relationship gains for each condition: inside GPS ellipsoid, over the map and both respectively.

$$W_k^i = G_{gps} \cdot W gps_k^i + G_{map} \cdot W map_k^i + G_{gam} \cdot W gam_k^i \quad (4)$$

After final weights of all particles are calculated, they are normalized by the maximum value obtained in the swarm – see Eq. (5). Particle inertia has not been considered for final weight because inertial relations were already included in auxiliary weights assignation.

$$W_k^i = \frac{W_k^i}{\max\left(W_k^1, W_k^2, \ldots, W_N^i\right)} \quad (5)$$

D. GPS confidence

It is well known that interference in received signal from satellites can generate temporal errors in GPS measurements. In some scenarios, when interferences are short or localized in a small area, even the implementation of augmentation systems – as EGNOS or WAAS – cannot correct them. This means that GPS could provide a displaced measurement with low uncertainty. In order to provide a way to avoid estimation errors due to those temporal displacements, a GPS confidence factor was implemented.

The main goal is that in case of a low value for GPS confidence, GPS measurements have a small influence in weight updating and therefore in the swarm estimation. For this, a displacement vector between two consecutive GPS measurement is compared to accumulated displacement measured from odometer – see Fig. 3. If the difference between both is larger than a defined threshold, the value of $Rgps$ should be then set as 1: keep your GPS weight and dismiss new measurements. This value will be kept during several wait steps until the GPS confidence has been stabilized and then, it will be progressively restore to its default value.

E. Lost particles

When some particles do not follow the swarm movement properly, they lose both GPS and Map weights due to their movement does not correspond to vehicle behaviour. This situation could make that only a few particles remain as possible candidates to vehicle state after several iterations – similar to degeneracy phenomenon occurring in PFs without re-sampling process [13].

To solve this issue, two different thresholds $Ugps$ and $Umap$ were defined for GPS and Map weights respectively. The main goal of thresholds implementation is indicating PSL which particles must be considered as "lost" and consequently be re-sampled. The resampling process has been implemented through two additional swarm attraction rules. In this case, the attraction is applied directly over the particle state and no over its transition vector. As can be seen in Eq. (6), in case of low GPS weight, lost particles are attracted towards GPS measurement $[xgps_k\ ygps_k]$, keeping their current yaw angle. Likewise, in case of low map weight, particles are attracted towards current swarm estimation – see Eq. (7). It can be also seen that for both rules a random noise has been added at the end of the equations. This factor disperses the attracted particles in order to reduce particle concentration around one single state. Constants D_x, D_y and D_ψ specify the maximum dispersion ratio for X, Y and ψ. $[x_{rand}, y_{rand}, z_{rand}]$ is a vector of random values between -1 and 1 calculated for each particle. Both rules are implemented after its corresponding weighting.

F. Final estimation

A priori swarm estimation – obtained from last step *a posteriori* estimation through model evolution – is only considered as a way to represent the swarm movement according to last step state. *A posteriori* value – and final vehicle state – is calculated directly from the updated states of swarm particles after the evolution process. As is shown in Eq. (8), this estimation is made by the weighted addition of all particles states.

$$X_k^{sw} = \frac{\sum_{i=1}^{N} W_k^i \cdot X_k^i}{\sum_{i=1}^{N} W_k^i} \quad (8)$$

IV. Experimental results

In this section, the results for two different tests are shown. In the first of them, the performance of the algorithm is analysed for different swarm lengths in order to determinate the most efficient number of particles. At second one, a comparison between PSL and EKF results is presented.

A. Swarm length

In order to determinate the most efficient swarm length for this application, the performance of PSL was observed for different sizes between 50 and 1000 – increased by steps of 50 particles. Due to the randomness of the algorithm, 20 runs of each size were made and the average values were taken. As a way to compare the results, the mean square error (MSE)

$$\begin{bmatrix} x_k^i \\ y_k^i \\ \psi_k^i \end{bmatrix} = \frac{(Ugps + Wgps_k^i)\begin{bmatrix} x_k^i \\ y_k^i \\ \psi_k^i \end{bmatrix} + (Ugps - Wgps_k^i)\begin{bmatrix} x_k^{gps} \\ y_k^{gps} \\ \psi_k^i \end{bmatrix}}{2 \cdot Ugps} + \begin{bmatrix} D_x \cdot x_{rand} \\ D_y \cdot y_{rand} \\ D_\psi \cdot \psi_{rand} \end{bmatrix} \quad (6)$$

$$\begin{bmatrix} x_k^i \\ y_k^i \\ \psi_k^i \end{bmatrix} = \frac{(Umap + Wmap_k^i)\begin{bmatrix} x_k^i \\ y_k^i \\ \psi_k^i \end{bmatrix} + (Umap - Wmap_k^i)\begin{bmatrix} x_k^{sw} \\ y_k^{sw} \\ \psi_k^{sw} \end{bmatrix}}{2 \cdot Umap} + \begin{bmatrix} D_x \cdot x_{rand} \\ D_y \cdot y_{rand} \\ D_\psi \cdot \psi_{rand} \end{bmatrix} \quad (7)$$

$Rgps$	$Rmap$	$Ugps$	$Umap$	G_{gps}	G_{map}	G_{gam}
0.7	0.5	0.3	0.15	1	1	5

TABLE I
VALUES SET FOR RELATIONSHIP GAINS AND THRESHOLDS.

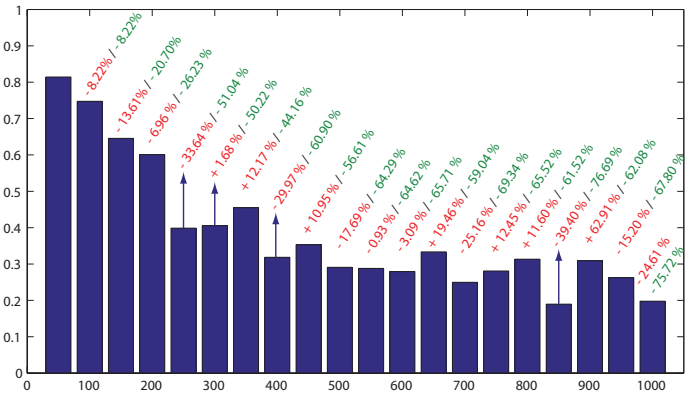

Fig. 4. Standard deviation of MSE.

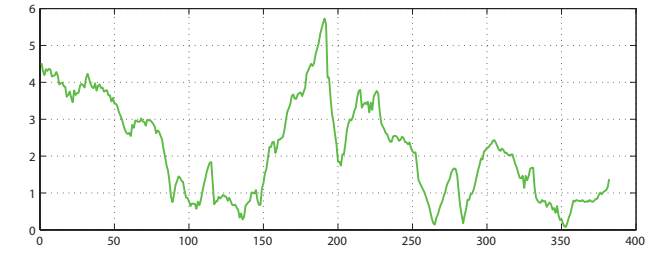

Fig. 5. Error evolution for 250-PSL.

and average time-per-cycle were calculated for each run. Table I shows the values used for the different relationship gains and thresholds.

Results obtained are shown in table II. It can be seen that the average of the MSE is very similar for all the swarm sizes. However, as shown in Fig. 4, the standard deviation of the MSE values decreases as the swarm length is increased. This indicates that the algorithm becomes more stable with a larger swarm, being less dependent of particles initialization. The values in red indicates the standard deviation reduction with respect to the previous size and the green one the reduction with respect to the 50 particles swarm. Likewise, results show that the execution time increases proportionally to the swarm size, meaning that the selection of the swarm size is a compromise between both factors.

Considering the obtained results and the trade-off between execution time and MSE, it has been decided that the best swarm size is between 250 and 300 particles. In this range the standard deviation of MSE is less than 50% of the value obtained for a swarm of only fifty particles and the average cycle execution time is not larger that 50ms. This last value is very important since normally the GPS eqquipment are set at 5 or 10 Hz for vehicle localization.

Fig. 5 shows the worst scenario obtained with a 250-PSL. It can be appreciated that the error decreases almost continuously from the start to the 100th sample, remaining lower than 2 metres just to 150th sample where a notorious increment is observed. This increase corresponds to the area surrounded by trees of the Val d'Or track, where the reference line has been estimated from high accuracy INS due to GPS unavailability. Around sample 290, vehicle returns to Routiere track and error remains – after an increment – lower than 2 metres.

B. Performance comparison

Having analysed how swarm length affects the algorithm performance and selecting an adequate size for the swarms, PSL have been compared to an EKF developed in [14] for the same scenario. This version of the EKF also implements the kinematic bicycle model as the evolution model of the process but does not use any data from digital maps.

Results are shown in Fig. 6. After algorithms initialization, it can be appreciated that the 250-PSL performance is always better than EKF. Both algorithms present an error increment after 150th sample – in Val d'Or track – where the GPS measurements have a high error. However, due to the inclusion of maps information in PSL, it was possible to limit the maximum error in PSL to 6 metres, while EKF's increases up to 16 metres. At first impression, it is possible to think that comparison is unfair due to the inclusion of maps data in PSL algorithm. Nevertheless, it is necessary to remark that, unlike the EKF, PSL implements GPS measurements as uncertainty areas.

V. CONCLUSIONS AND FUTURE WORK

Having as reference the behaviour of biological swarms – where all the individual move together but not in the same

Size	50	100	150	200	250	300	350	400	450	500
MSE	5.531	5.619	5.140	5.123	5.491	5.345	5.452	4.897	5.439	5.422
time-per-cycle [ms]	14.01	20.82	27.81	35.36	41.97	49.60	57.16	64.01	71.74	78.94
Size	550	600	650	700	750	800	850	900	950	1000
MSE	5.576	5.377	5.159	5.416	5.509	5.627	5.477	5.230	5.091	5.424
time-per-cycle [ms]	86.53	93.70	100.09	107.26	114.22	122.12	128.85	136.81	142.87	151.01

TABLE II
AVERAGE MSE AND EXECUTION TIME ACCORDING TO SWARM SIZE.

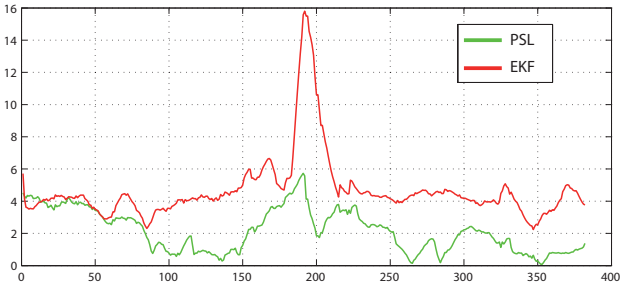

Fig. 6. Error comparison for EKF and 250-PSL.

way – a novel algorithm has been developed for vehicles localization. By using a set of particles, this algorithm is able to merge data from different sensors as GPS, INS, odometer and digital maps in order to estimate the most plausible solution to vehicle state. The algorithm has been developed off-line using real data captured from an instrumented vehicle at LIVIC.

Performance of the algorithm has been validated through two different tests. In first hand, the influence of the swarm length over MSE and average execution time per cycle was analysed. It was determined from results that the best swarm size for this application is around 250 particles. Finally, the comparison with EKF clearly showed that PSL has a better result performing vehicle localization.

As future work it is planned to integrate the algorithm in real vehicles, analysing its performance in real time under different driving conditions as constant speed, accelerations and turns. Furthermore, the inclusion of other sensors as lane detectors or visual odometers could help to improve the performance of the algorithm. In this sense PSL provides an easy way to implement sensors information as additional particles weights. Finally, a deep study of the influence of the different relationship gains in algorithm estimation must be done.

ACKNOWLEDGMENT

J. Godoy wants to specially thank to the JAE program (Consejo Superior de Investigaciones Científicas) for its support in the development of this work.

REFERENCES

[1] E. D. Kaplan, *Understanding GPS: Principles and applications*, E. D. Kaplan, Ed. Artech House, 1996.
[2] G. Welch and G. Bishop, "An introduction to the kalman filter," University of North Carolina, Department of Computer Science., Tech. Rep. TR 95-041, 1995.
[3] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.
[4] P. Zhang, J. Gu, E. E. Milios, and P. Huynh, "Navigation with imu/gps/digital compass with unscented kalman filter," in *Proc. IEEE Int Mechatronics and Automation Conf*, vol. 3, 2005, pp. 1497–1502.
[5] V. Milanés, J. Naranjo, C. González, J. Alonso, R. GArcía, and T. de Pedro, "Autonomous vehicle based in cooperative gps and inertial systems," *ROBOTICA*, vol. 26, pp. 627–633, 2008.
[6] W. Li and H. Leung, "Constrained unscented kalman filter based fusion of gps/ins/digital map for vehicle localization," in *Proc. IEEE Intelligent Transportation Systems*, vol. 2, 2003, pp. 1362–1367.
[7] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," *Radar and Signal Processing, IEE Proceedings F*, vol. 140, no. 2, pp. 107 –113, apr 1993.
[8] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.
[9] C. Urmson, J. Anhalt, H. Bae, J. A. D. Bagnell, C. R. Baker , R. E. Bittner, T. Brown, M. N. Clark, M. Darms, D. Demitrish, J. M. Dolan, D. Duggins, D. Ferguson, T. Galatali, C. M. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. Howard, S. Kolski, M. Likhachev, B. Litkouhi, A. Kelly, M. McNaughton, N. Miller, J. Nickolaou, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, V. Sadekar, B. Salesky, Y.-W. Seo, S. Singh, J. M. Snider, J. C. Struble, A. T. Stentz, M. Taylor , W. R. L. Whittaker, Z. Wolkowicki, W. Zhang, and J. Ziglar, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics Special Issue on the 2007 DARPA Urban Challenge, Part I*, vol. 25, no. 1, pp. 425–466, June 2008.
[10] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, "Minerva: a second-generation museum tour-guide robot," in *Proc. IEEE Int Robotics and Automation Conf*, vol. 3, 1999, pp. 1999–2005.
[11] N. . Nedjah and L. de Macedo, *Swarm Intelligent Systems*, N. . Nedjah and L. de Macedo, Eds. Springer, 2006.
[12] G. Tong, Z. Fang, and X. Xu, "A particle swarm optimized particle filter for nonlinear system state estimation," in *Proc. IEEE Congress Evolutionary Computation CEC 2006*, 2006, pp. 438–442.
[13] A. Doucet, S. Godsill, and C. Andrieu, "On sequential monte carlo sampling methods for bayesian filtering," *Statistics and Computing*, vol. 10, pp. 197–208, 2000.
[14] A. Ndjeng Ndjeng, A. Lambert, D. Gruyer, and S. Glaser, "Experimental comparison of kalman filters for vehicle localization," in *Proc. IEEE Intelligent Vehicles Symp*, 2009, pp. 441–446.