Map-Aided Dead-Reckoning With Lane-Level Maps and Integrity Monitoring

Franck Li, Philippe Bonnifait, and Javier Ibañez-Guzmán, *Member, IEEE*

***Abstract*—Navigation maps provide critical information for advanced driving assistance systems and autonomous vehicles. When these maps are refined to lane-level, ambiguities may occur during the map-matching process, particularly when positioning estimates are inaccurate. This paper presents a dead-reckoning method implementing a particle filter to estimate a set of likely map-matched hypotheses containing the correct solution with a high probability. Our method uses lane-level maps that feature dedicated attributes such as connectedness and adjacency. The vehicle position is essentially estimated by dead-reckoning sensors and lane detection using an intelligent camera. We also describe an integrity monitoring method for assessing the coherence of the set of hypotheses, using the fix of a global navigation satellite system receiver. The method provides in real-time a "Use/Don't Use" characterization of the vehicle positioning information that is transmitted to safety functions, where integrity is fundamental. The performance of the proposed map-aided dead-reckoning method with integrity monitoring is evaluated using data acquired by an experimental car on suburban public roads. The results obtained validate the approach.**

***Index Terms*—Map-matching, lane-level maps, GNSS, particle filters, camera lane detection, integrity.**

I. Introduction

INTELLIGENT vehicles need a digital description of the world to navigate autonomously. For this, prior information on road network features is crucial. The information is usually stored in georeferenced road maps providing geometric and contextual information such as lane markings, traffic signs, etc. Map suppliers are working intensively to produce maps that meet the stringent requirements of intelligent vehicles in terms of content and accuracy.

Accessing relevant information requires that a vehicle is accurately localized with respect to these maps. For this purpose, Global Navigation Satellite Systems (GNSS) receivers provide an affordable means of acquiring the vehicle's absolute position on Earth. The process of associating these position estimates to roads on a map is known as *Map-Matching* (MM). Given that GNSS measurements are often subject to errors of as much as several meters [1], matching the true position on a lane-level map remains challenging, and GNSS technology is often supplemented using information from the car's sensors. Most solutions use wheel speed sensors and low cost gyros to improve accuracy. This is called dead-reckoning (DR). Exteroceptive information acquired by on-board perception sensors (e.g., cameras and lidars) can also be processed to improve positioning estimates.

Combining the various information sources is not always enough to remove all ambiguities on the matched positions. To guarantee the reliability of the information provided by the MM, the algorithm has to be able to track multiple hypotheses [2]. But this is only a first step in creating a positioning system whose integrity may be relied on. To be reliable, the positioning information needs to be unambiguous and confirmed by a redundant information source. To comply with this, a fault detection test can be performed on the hypotheses tracked by the system in order to reduce the set of hypotheses to a minimum, and ideally to a single solution usable by client applications for which safety is particularly important.

This paper presents a method that solves lane-level MM using a Particle Filter (PF) and provides integrity monitoring of the result using GNSS information. Section II reports related work. Section III presents the lane-level map used in this research. Section IV provides details about the PF MM developed along with a multi-hypotheses integrity consideration. Section V then develops an integrity monitoring method to check the coherence of the positioning information. A "Use/Don't Use" decision strategy is defined. Finally, results using real road data are presented in Section VI.

II. Related Work

Previous works have investigated PF for MM purposes: Oh *et al.* [3] included map priors with GPS positioning in a PF-based pedestrian localization algorithm. The map data represents a probability distribution over the test area, defining the likelihood of the location. Positioning enhancement is the main contribution, demonstrating the use of map data in the filter. The particles were used only to represent the position posterior density, and not to manage multiple hypotheses.

The same principle was applied to vehicles by Peker *et al.* [4], who in addition merged odometry information. Topology is added to the map, providing information on the probability of a

Manuscript received September 12, 2017; revised November 21, 2017 and December 21, 2017; accepted December 23, 2017. Date of publication January 16, 2018; date of current version March 19, 2018. *(Corresponding author: Franck Li.)*

F. Li is with the CNRS, Heudiasyc UMR 7253, CS 60319, Sorbonne Université, Université de Technologie de Compiègne, Compiègne 60203 cedex, France, and also with the Renault s.a.s., Guyancourt 78280, France (e-mail: franck.li@hds.utc.fr).

P. Bonnifait is with the CNRS, Heudiasyc UMR 7253, CS 60319, Sorbonne Université, Université de Technologie de Compiègne, Compiègne 60203 cedex, France (e-mail: philippe.bonnifait@hds.utc.fr).

J. Ibañez-Guzmán is with the Renault s.a.s., Guyancourt 78280, France (e-mail: javier.ibanez-guzman@renault.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIV.2018.2792843

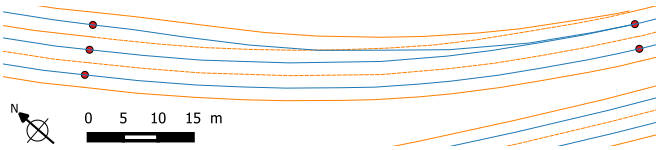

Fig. 1. Detail of the map at a lane forking in Compigne, France. The centerlines of the lanes are drawn in blue and the lane markings in brown. The upper road is drivable from east to west. The upper blue polyline of the right lane separates into two lanes.

given trajectory. The results focus not only on raw positioning but on the correctness of the MM. Once again, even though the algorithm uses the particles to consider multiple possible matchings, the output is still mono-hypothesis. Choosing the hypothesis with the highest weight can lead to errors.

Filters can also process information from exteroceptive sensors, as done by Gu *et al.* [5], where lane marking detections from a camera were used. Focusing on dense urban environments prone to multipath problems, the authors processed the GNSS information using 3D maps of buildings to detect this kind of error. This provides a cleaner GNSS position to be integrated into the positioning system. It is then fused with lane markings using a PF and map information.

The most important difference between the present work and the previously cited papers is that we retain the multi-hypothesis ability of the PF all the way up to the output of the system: where classic algorithms would make a choice, our filter outputs all likely hypotheses so as to avoid having to choose arbitrarily a hypothesis that is potentially wrong. If the filter has enough non-ambiguous information to provide a single hypothesis, then this hypothesis will be very reliable. The other significant difference is that the filter is designed to assess the coherence of the positioning information using redundancy. GNSS positioning information is therefore used as little as possible in the filter. Where it is used, this is mainly to evaluate its consistency with map-aided dead-reckoning.

III. High-Definition Map

Accurate road maps are useful in many intelligent vehicle applications as providers of contextual information. Multiple representations of the road network have been proposed, such as clothoidal models [6], Lanelets [7]. This section gives a description of the map structure used in this paper.

A. Lane-Level Road Maps

The road map used in this research is a mesoscale lane-level road map [8] (see Fig. 1). Mesoscale, situated between macroscale (e.g., a road guidance map) and microscale (e.g., a dense point cloud from perception sensors), is the most suitable scale for intelligent vehicles, as it carries accurate information without being too dense for easy use [9]. The prototype map used in this paper, produced from field survey data acquired by a mapmaker, covers 4 km of public roads in Compigne with an absolute accuracy of 2 cm. The map is stored as an SQLite database containing the following relevant information:

1) *Road Geometric Information:* common to all road maps (blue lines in Fig 1). Polylines describe the geometry of the driving lanes by their center line, using a local Cartesian frame. The road network is split into *Links* representing a segment of road on which the properties stay constant (e.g., width, lane markings, etc).
2) *Lane Markings:* every drivable lane has information on the nature of its delimiting borders (orange lines in Fig 1). A geometric description is given and additional information is included, such as the type of marking (e.g., a solid line). Note that not only painted markings are referenced: for example, a pavement can be indicated if no other delimitation is present.
3) *Road Connectedness:* to navigate in the *Links* network, information about connected links (i.e. previous and next accessible links) is given. This gives a more efficient way to determine which link is reachable along the track of the current position (respecting traffic rules) without costly distance calculation to determine nearby links. This describes intersections, but also lane merging and forking (a situation shown in Fig. 1).
4) *Lane Adjacency:* similarly to Road Connectedness (above), this gives information on lanes adjacent to the current lane that may be available for cross-track evolution, i.e. lane changing.

All this information is held in a relational database and therefore available easily and at low computational cost for any *Link* in the map. The filter described below relies on this to perform efficiently.

B. Semantic Information

A digital road map can be viewed as a Geographic Information Database, containing the coordinates of the different road structures. This is the natural approach when dealing with a digital map. But the most recent digital maps, such as the one used in this article, contain richer information that allows advanced MM processing. For instance, every link accessible from a given matched position is easily accessible from the database, without the need for costly computation. Using a traditional map, when the 2D position of the car changes, a new matching has to be performed. However, where this additional information is present, the evolution of the map-matching can be directly estimated: a 2D evolution can be projected into a 1D evolution on the map links. The map thus provides a framework relatively independent of the 2D-plane geometry: the evolution of the MM takes place in the 1D map space instead of the 2D plane space, providing a more efficient evolution.

C. Adjacent Links

Another important feature of our research map is adjacency information: every link is aware of the links on either side. This provides a useful way of checking for matching ambiguities, given that adjacent links present the biggest challenge for lane-level MM. Where this information is available directly from the map, no costly computations are required, and the exploration of

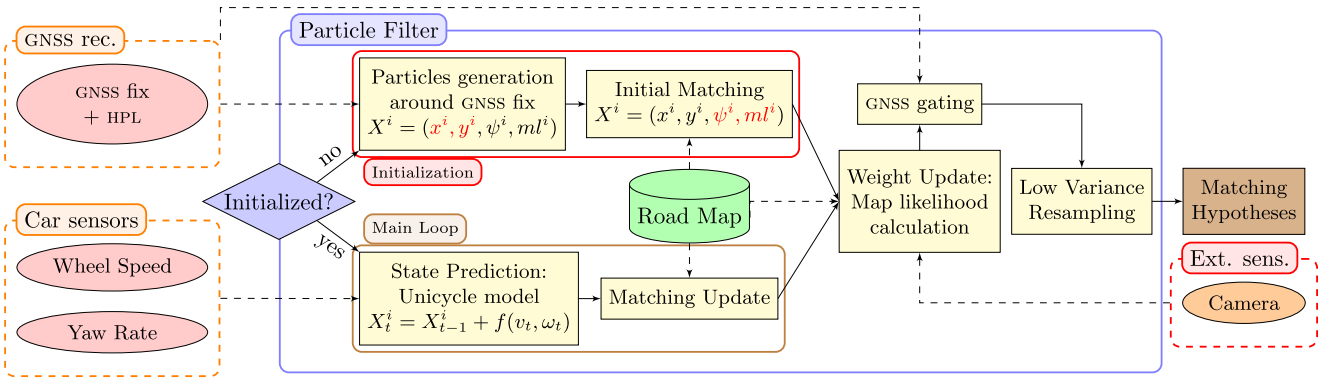

Fig. 2. Flowchart of the map-matching process. Based on a particle filter, the algorithm takes as input a GNSS fix with HPL and the car's dead-reckoning. The process is separated into two steps, namely a computationally intensive initialization followed by a highly efficient main loop. Exteroceptive information is used in the computation of likelihood.

hypotheses is greatly improved. This adds an interesting lateral capability for map exploration in addition to the longitudinal capability that is provided by the information from successive links. The matching algorithm thus has a complete framework for making efficient use of the map, removing some of the heavy computation and taking advantage of the efficient map design.

IV. Sequential Map-Matching Using Particle Filtering

A. Multiple Hypotheses Approach

The goal of the method described here differs from the most common MM methods. For instance, in classical automotive systems used for turn-by-turn navigation, MM should give the user a *single* position estimate as a result. This corresponds to the usual requirement of these systems: a single position must be used to calculate a route and, in most cases, if this position is erroneous, the user is able to notice the error, disregard the given information and wait for a correct matching.

MM for autonomous navigation systems, on the other hand, must not be over-confident about its results: the worst-case scenario would be to provide an erroneous single position, since there may be no human to detect the error and correct it. If the matching is ambiguous, then the algorithm should retain the ambiguity and not make a decision. In the past, reference algorithms [10] often addressed map-matching only via a roadway level map (i.e. where roads are described by a single link for each direction). More recent research tends to be oriented toward lane-level maps [9], [11], which present a lot of ambiguities that must be taken into consideration.

This illustrates the notion of *MM integrity* [12], which will be developed later in this paper. In this context, the ability to track multiple MM hypotheses is highly relevant, hence our choice of a PF [13]. PF is a commonly used method for map-matching [14]–[17]. Its ability to manage multiple hypotheses is interesting in this context for providing a certain level of integrity, especially in ambiguous situations, where single hypothesis methods could lose track of the correct solution. The PF we describe implements measures to avoid this situation by exploring and retaining a number of likely hypotheses, which is made possible by an efficient use of the map. This section describes the different steps of the MM algorithm developed in this study (see Fig. 2).

B. Particle Definition

The PF is used to estimate the posterior distribution of the car's position, given its sensor inputs and the map. The particles model the car's 2D pose X_p^i (composed of the Cartesian 2D coordinates (x^i, y^i) and heading ψ^i), plus (ml^i), the ID of the map link that it is matched to, as described by (1). This is the state X^i that each particle possesses, representing a single-position hypothesis together with its map-matched solution.

$$X^i = (X_p^i, ml^i) = (x^i, y^i, \psi^i, ml^i) \tag{1}$$

Additionally, each particle possesses a weight w^i characterizing its likelihood as a positioning solution. The complete structure of a particle is then given by (2):

$$\text{Part}^i = (X^i, w^i) \tag{2}$$

The particle's 2D pose follows a unicycle evolution model, using as inputs the speed and yaw rate obtained from the car's proprioceptive sensors:

$$\begin{cases} x_t^i &= x_{t-1}^i + v_t^i \cdot \Delta t \cdot \cos \psi_{t-1}^i \\ y_t^i &= y_{t-1}^i + v_t^i \cdot \Delta t \cdot \sin \psi_{t-1}^i \\ \psi_t^i &= \psi_{t-1}^i + \omega_t^i \cdot \Delta t \end{cases} \tag{3}$$

$U_t^i = [v_t^i, \omega_t^i]^T$ is the input vector of the ith particle, with $v_t^i \sim \mathcal{N}(v_{\text{raw}}, \sigma_v^2)$ and $\omega_t^i \sim \mathcal{N}(\omega_{\text{raw}}, \sigma_\omega^2)$, based on raw measurements of the vehicle speed and yaw rate $(v_{\text{raw}}, \omega_{\text{raw}})$ with an added random noise. The noise, added independently for each particle, allows them to spread as they evolve.

The matched ID ml^i, being a discrete value, evolves differently, without a dynamic model *per se*. Two approaches are used consecutively: first, particles are map-matched once in the filter initialization step (see Section IV-C) and, second, in the main loop the links are followed logically using attributes stored in the lane-level map describing connectedness and adjacency (as described in Section IV-D). This separation into two steps

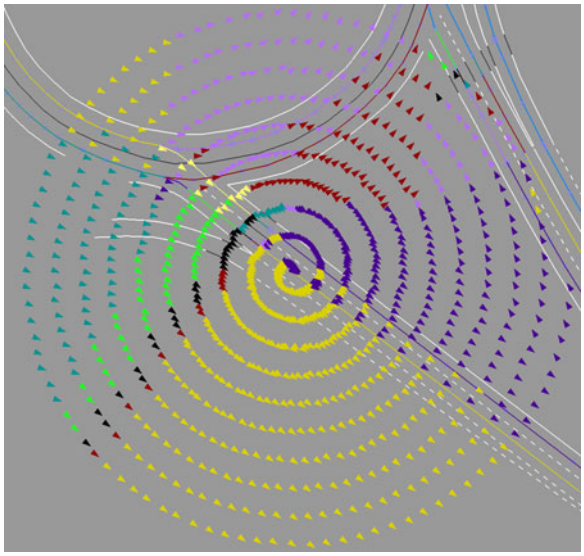

Fig. 3. Particle Initialization: colors denote the matched link. The initial heading corresponds to the matched link. Some particles are far from the links due to the 50 m HPL (high value chosen to be conservative), but will potentially be quickly eliminated during a resampling step.

provides an optimization in terms of real-time computation, as described in detail below.

C. Initialization Step

The first step is the map handling. SQL-based map formats are appropriate when using large map coverage and database size, because of the possibility they offer to make spatial queries to a specific position. Although queries generally execute rapidly, query formatting and returned data parsing cause significant CPU load and consequently a long response time. SQL should therefore be limited to one-off queries, and the process can be made more efficient by incorporating caching methods, as implemented by Bonnifait *et al.* [18] where map information of the local area is loaded into memory as the car moves from one position to another. The approach presented in this paper loads the whole map directly when initializing, as its size is relatively limited.

After loading the SQL map into an internal data structure (i.e. directly accessible), the filter initializes its particles on the first valid GNSS position received. This step is shown in the red box in the top part of the PF box in Fig. 2. Particles are generated around this position, in a circular pattern to cover the full area corresponding to the associated *Horizontal Protection Level* (HPL) (see Fig. 3), circumventing any bias the GNSS fix might have. The HPL denotes an area in which, at a given risk level, the true position is guaranteed to be found. Modern receivers are now beginning to implement these kinds of computations, as integrity is becoming an increasingly important part of localization [19]. This step initializes the 2D coordinates (x^i, y^i) part of the particle state.

Each particle is then matched to the closest link, following a *point-to-curve* method that selects the link candidate with the lowest Euclidean distance to the particle. This process accounts for most of the computational load during the initialization step, since the distance of every particle to the different link candidates needs to be evaluated. Concerning the vehicle heading, it is assumed that the vehicle is traveling on-road, and that this road is present in the map. It is thus reasonable to initialize the heading of each particle to correspond to the bearing of its matched link. This process sets the values of the rest of the state (ψ^i, ml^i), finalizing the initialization of the particles.

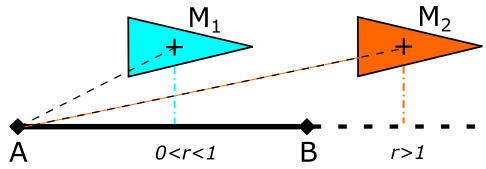

Fig. 4. Particle at time $t = 1$ (M_1) remains on the current segment $[AB]$, while at time $t = 2$ (M_2) it has left it. This is determined by calculating the ratio r described by (4).

This initialization ensures that all the links present in the map around this position are considered, if enough particles are provided. This is the problem of particle density: the larger the initialization area, the more particles are needed to maintain an acceptable particle density, essential for an efficient PF. Some particles are clearly created outside the drivable area represented in the map. This illustrates the fact that the particles are not strongly constrained on the map and can evolve in the 2-dimensional space, not only on the centerlines. The filter consequently has spatial flexibility, limited only by the decreasing likelihood of particles (for example one that gets too far away from a link). These particles will be quickly eliminated by the filter during the update step, following the natural evolution of their likelihood.

D. Main Loop

This second step takes over after the filter has been initialized (see the bottom part of the PF box in Fig. 2). The 2D pose evolves using (3) (state prediction step in Fig. 2). By design, the computations required during the matching update step are far fewer than during the initialization step: further iterations do not require as many distance calculations. The matching evolves thanks to an efficient use of the map data.

1) Nominal Case: Let M_i be the position of a particle at time step i and $[AB]$ be the matched segment. To determine if the matched segment needs to be updated, the ratio

$$r = \left(\overrightarrow{AB} \cdot \overrightarrow{AM}\right) / \left|\left|\overrightarrow{AB}\right|\right|^2 \tag{4}$$

is computed: if $r > 1$ (respectively $r < 0$), the particle has left the current segment and has to be associated with the next one (respectively the previous one) by simply using the connectedness information of the map. This situation is illustrated in Fig. 4. Based on this information, the ratio r becomes the only value requiring computation, which improves the efficiency of the main loop.

2) Case of Multiple Successors: Situations can arise where the current link has multiple successors. In this case, the filter follows a thorough exploration strategy: when leaving the current segment, a clone particle is created for each successor and matched accordingly (see Fig. 5). This effectively covers all hypotheses at the lane bifurcation. The cloned particles are

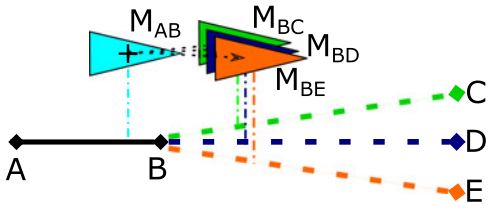

Fig. 5. Lane bifurcation where particle cloning occurs: the particle M_{AB} is duplicated into M_{BC}, M_{BE} and M_{BD} in order to do a thorough search.

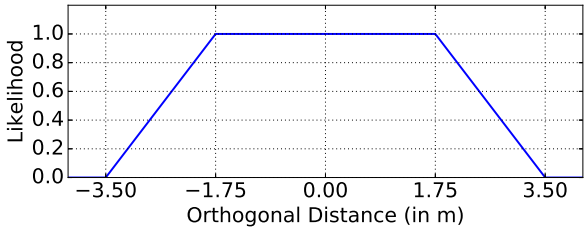

Fig. 6. Trapezoidal likelihood function for orthogonal distance (based on the French regulations relating to the width of roads). The car is equally likely to be in any lateral position on the roadway.

exact copies of the original, and the evolution model together with the map data will allow a selection to be made, so that only the most likely hypotheses among the clones are retained.

3) Lateral Lane Change: Another possible evolution is a change to an adjacent lane. This is made possible as the particles are not constrained to evolve only on the links but also laterally. Using lane marking information from the map (see Section III-A), the lane width can be determined. When a particle leaves the boundaries of its matched link, it will be transferred to the adjacent link, if one exists.

All the information needed for the matching to evolve is contained in the map structure, making the main loop filtering highly efficient for the purposes of MM.

E. Likelihood Calculation and Resampling

The final step in the PF is the likelihood update and resampling step. These are necessary to keep a consistent particle set. The likelihood of a particle is expressed through its weight: the higher the weight, the more likely the hypothesis. It is updated after each iteration of the filter. Since the MM algorithm is as independent from the GNSS as possible, the likelihood computation is done by comparing the particle state to the prior map information. This is the way dead-reckoning is map-aided.

1) Map Likelihood: A Sampling Importance Resampling (SIR) PF is used, as described by Arulampalam *et al.* [13], but maintaining a recursive weight calculation such that:

$$w_t^i = w_{t-1}^i \cdot p(z_t|X_t^i) \tag{5}$$

where z_t represents the measurements, which correspond to two metrics computed using the map. The first metric is the heading difference $(\Delta\psi^i)$ between the particle and the link, as used by Merriaux *et al.* [20]. This metric is a good likelihood indicator for map-matching, allowing the links whose heading evolution does not correspond to the dead-reckoned estimate to be eliminated quickly. A Gaussian distribution is assumed such as

$$f_\psi(\Delta\psi^i) \sim \mathcal{N}(0, \sigma_\psi) \tag{6}$$

to represent the particle likelihood with respect to its heading. It is centered on 0 (maximum likelihood if the particle heading is the same as its link) with a typical standard deviation of $\sigma_\psi = 15^\circ$ (value set experimentally during the tuning stage of the method). Note that this heading metric cannot discriminate parallel links.

The second metric is the orthogonal distance $(d^\perp)$ to the centerline: this takes into account that the farther a particle is from the roadway, the less likely it is. To reflect the fact that a car is equally likely to be located in any lateral position on the roadway, the likelihood function is defined as

$$f_d(d^\perp) = \begin{cases} 1 & \text{if } |d^\perp| < L \\ \max\{1 - \frac{|d^\perp| - L}{m}, 0\} & \text{otherwise} \end{cases} \tag{7}$$

where L is half the width of a driving lane, and m is the margin on either side of the lane where the likelihood decreases linearly, forming a trapezoidal distribution (see Fig. 6). We chose this shape in order to have efficient computation in real-time. When two lanes are adjacent, the two functions overlap so that there is no discontinuity when a particle changes lane: the top part of the adjacent lane's trapezoid begins where the current lane's trapezoid ends. In other words, the sloping parts only apply when there is no other lane adjacent to the current lane, because otherwise the particle would change its matching and apply the likelihood function of the new lane.

Given that the two errors are considered conditionally independent, (5) can be computed as

$$w_t^i = w_{t-1}^i \cdot f_\psi(\Delta\psi^i) \cdot f_d\left(d^\perp\right) \tag{8}$$

2) Lateral Information from a Camera: The smart camera used in the system is able to detect the ego-lane markings and return them using polynomial models. For each detected lane marking, the camera returns the coefficients of a third-degree polynomial [21] (see (9)) approximating the equation of a clothoid, where x and y are coordinates in the camera's working frame $(C, \overrightarrow{x_c}, \overrightarrow{y_c})$ (see Fig. 7).

$$y = \frac{C_3}{6} \cdot x^3 + \frac{C_2}{2} \cdot x^2 + C_1 \cdot x + C_0 \tag{9}$$

In practice, the highest coefficients (the curvature C_2 and curvature derivative C_3) are often very noisy and do not provide reliable information. Therefore, only C_0 (the lateral offset, in meters) and C_1 (the line heading, in Rad.) are used, the lane marking being considered as a straight segment. This approximation still gives a good estimate of the line position and orientation with respect to the car. Note that the algorithm would also work with other sources of information (e.g., lidar-based lane detection) that are able to provide lane detection in a similar manner.

As presented in Sec. III-A, lane markings are additional information included in the map. They follow the same geometric representation as the centerlines (i.e. polylines). Each driving lane directly references the lane markings (if they exist) that delimit it, and it is therefore possible to obtain the equation $y = C_1 \cdot x + C_0$ for the current left and right markings.

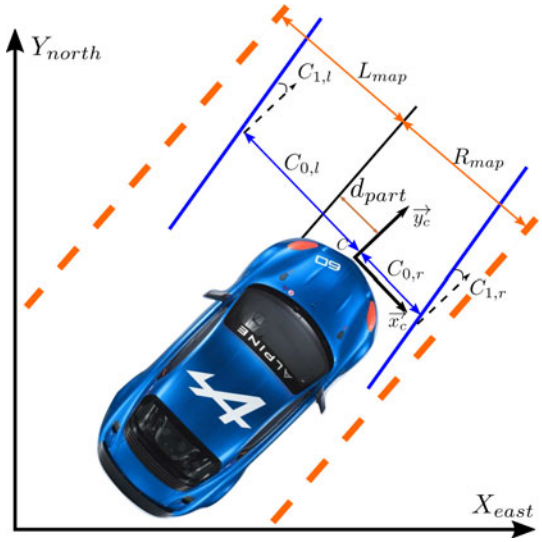

Fig. 7. Reference frames used by the camera for describing lane markings. Dashed orange lines represent lane marking data from the map. Camera detections are represented in blue. The lane centerline is in black.

This exteroceptive information, if available, is used during the update step of the filter to calculate the likelihood of each particle on the basis of richer information [22]. Note that the lane markings are not always detected by the camera. This can be caused by difficult conditions for the sensor, for instance, bad lighting, a wet road surface, faded markings, etc. The camera also performs best when moving in a straight line, since the marking tracking can be challenging in high-curvature roads [23]. In these cases, the camera indicates a low confidence level and the likelihood calculation reverts to the map-only likelihood described previously.

If marking detection is available, lateral positioning in the lane is taken into account. The same procedure as the original filter is followed but slightly modified: with marking information, the likelihood will be centered on the relative lateral positioning according to the detection [16]. To do this, two position ratios are computed, describing the lateral position of the vehicle in its lane. The first position ratio uses the marking detection from the camera, i.e. the signed distances $C_{0,l}$ and $C_{0,r}$ (respectively the C_0 coefficients of the left and the right lane marking detection, and respectively negative and positive, see Fig. 7) and is defined as:

$$r_{\text{cam}}^{\text{lat}} = \frac{C_{0,l}}{C_{0,l} - C_{0,r}} \tag{10}$$

The second position ratio corresponds to the map data: for each particle (i.e. position hypothesis) the theoretical ratio is computed using the distances L_{map} and R_{map} of the left and right lane markings to the centerline (both unsigned), and the signed distance d_{part} of the particle to the centerline (positive to the right):

$$r_{\text{map}}^{\text{lat}} = \frac{L_{\text{map}} + d_{\text{part}}}{L_{\text{map}} + R_{\text{map}}} \tag{11}$$

The car within the limits of the lane markings will thus have a ratio between 0 (when on the left-hand marking) and 1 (on the right-hand marking). Absolute lane marking distances are not used so as to address a possible scaling factor discrepancy between the lane detection provided by the camera and the map information. In other words, if an incorrect lane width is stored in the map, the ratios are still valid. The physical meaning of these ratios is a percentage of the driving lane width. For example, a ratio of 0.1 is 10% of the width of a lane, which might for instance correspond to a distance of 0.35 m on a lane that is 3.5 m wide.

If the camera observations are in accordance with the particle position on the map, the two ratios will be approximately the same. To verify this, the difference between them is used to compute the likelihood:

$$\Delta r = r_{\text{map}}^{\text{lat}} - r_{\text{cam}}^{\text{lat}} \tag{12}$$

The closer to zero the value is, the better the particle fits the camera observation. The likelihood function is thus set as a Gaussian distribution centered on 0 (i.e. a null difference meaning $r_{\text{map}}^{\text{lat}} = r_{\text{cam}}^{\text{lat}}$, a perfect fit), described by $\mathcal{N}(0, \sigma_{\text{dist}})$, with $\sigma_{\text{dist}} = 0.1$. This difference in ratio would correspond to a 10% error (0.35 m on a 3.5 m lane). The value is set empirically, as the camera does not provide any quantitative indicator of accuracy.

In the same way, instead of considering the car being aligned with the road segment, the heading coefficients from the camera $C_{1,r}$ and $C_{1,l}$ are used to calculate the particle heading likelihood: the mean heading C_1 is computed and serves as a reference for the particle heading. The heading likelihood is described by $\mathcal{N}(0, \sigma_{\text{head}})$ with $\sigma_{\text{head}} = 15°$, set empirically.

With exteroceptive lateral information, the likelihood function follows the lateral movement of the vehicle, therefore containing the particle spread in this direction. This gives a much better lateral distribution (tighter) and even naturally allows lane changing maneuvers. Moreover, it allows a better description of the car's path in the carriageway. For instance, if the vehicle is one meter away from the center of the lane, the particles fitting this behavior will have the best likelihood score.

3) GNSS Gating: This last step of the likelihood calculation ensures that the filter stays consistent by killing particles that stray too far from the position returned by the GNSS receiver. It simply sets the weight of these diverging particles to 0 in order to eliminate them from the filter during a resampling step. To keep the influence of the GNSS fix to a minimum, the gate radius (after which a particle is eliminated) is set to the Horizontal Protection Level (HPL) returned by the GNSS receiver. In this paper, the HPL has been set to a fixed value of 50 m, commonly used for a 10^{-4} risk in urban environment [19]. Note that with additional information from the GNSS receiver (e.g., positioning residuals), a real-time value can be computed, representing the actual GNSS status more accurately.

F. Resampling

As noted in Section IV-E1, the SIR algorithm is slightly modified to use a recursive likelihood calculation. In the original version a systematic resampling takes place at each iteration. The MM developed here relies on a certain exploration of the

possible matching hypotheses in order to avoid *sample impoverishment*, which is inconsistent with a too frequent resampling that would eliminate possible good hypotheses, reducing the variance of the particle set. Therefore, to resample only when necessary, the number of effective particles N_{eff} can be estimated as follows [24]:

$$N_{\text{eff}} = \frac{1}{\sum_i (w_k^i)^2} \tag{13}$$

If N_{eff} falls below a given threshold (typically $2/3$ of the total number of particles), a resampling procedure is applied. The classic resampling method used for PF draws new particles randomly from the original set with a probability proportional to the weight of the original particles. This randomness can cause an impoverishment by drawing exclusively from a subset of the particles, as successive drawings are independent and uncontrolled. A low variance resampling [25] is therefore preferred, since this favors a good distribution of the retained particles. First, a correspondence between each particle is established with the interval $[0, 1]$ (each number in the interval corresponds to a single particle). This is done by summing the particle weights consecutively to create an empirical cumulative density function (the weights sum to 1). The principle is then to only perform a single random draw to determine a seed s in the interval $[0; N^{-1}[$, N denoting the total number of particles and then add repetitively N^{-1} to it, generating numbers corresponding to particles. Using this, the N resampled particles $\widehat{\text{Part}^i}$ can be determined from the original set Part by:

$$\widehat{\text{Part}^i} = \text{Part}^j \tag{14}$$

for $i = \{0, ..., N-1\}$ with

$$j = \arg\min_k \sum_{m=0}^{k} w^m \geq \left(s + \frac{i}{N}\right) \tag{15}$$

V. Integrity Monitoring

A. Map-Matching Integrity

A multi-hypotheses MM (such as the one presented here) respects an integrity level if the set of hypotheses provided as a solution contains the correct one, with respect to a given risk. In other words, an algorithm with a risk of 10^{-2} (i.e. 1%) has to return the actual map-matched position (i.e. the matching ground truth) 99% of the time. As advances in intelligent vehicles continue to be made, integrity monitoring for safety reasons is becoming an increasingly important topic in the literature [26], [27].

Applied to an HD map described by polylines such as the map used in this paper, where an elementary geometrical descriptor is a segment (a link is a list of connected segments, each delimited by two shapepoints), the map-matched segment of the position ground truth has to be part of the set of hypotheses returned by the MM algorithm, according to the given risk. As a consequence, most of the time the proposed MM algorithm returns a set of solutions describing multiple hypotheses.

To check the integrity of the MM, redundancy is needed. For this, the position computed by the GNSS receiver is used. By design, the MM algorithm is quite independent of GNSS, since GNSS is only used for gating the particles based on the HPL which is very reliable.

The method we propose for checking the integrity is to use the GNSS fix in a fault detection step. This approach is called *internal integrity monitoring*. The fault detection method that we use is based on a probabilistic approach using Mahalanobis distances. This involves manipulating the covariance matrices of the different estimates involved in the process.

B. Covariance Estimation

Manipulating the covariance of the errors of the different estimates gives a good overview of the situation in terms of uncertainty.

1) GNSS Covariance: The covariance of the estimation error of a GNSS fix is generally returned by the receiver itself, giving an estimate of the uncertainty affecting the positioning computation.

A 1-sigma uncertainty ellipsoid is in general given (e.g., in the NMEA GST sentence) with the semi-major and semi-minor axes and the orientation ϕ of the error ellipsoid. In the ellipsoid frame, the covariance matrix is diagonal:

$$\Sigma_{\text{GNSS}} = \begin{bmatrix} \sigma^2_{\text{GNSS}_x} & 0 \\ 0 & \sigma^2_{\text{GNSS}_y} \end{bmatrix} \tag{16}$$

To get the covariance in an east-north frame, a simple rotation of an angle ϕ is applied:

$$\hat{\Sigma}_{\text{GNSS}} = R^T . \Sigma_{\text{GNSS}} . R \tag{17}$$

with R the 2D rotation matrix of angle ϕ.

In practice the covariance matrix is generally overconfident, since it does not take into account effects such as atmospheric perturbations or ephemeris errors. Covariance estimates can then be off by few meters. One way of handling this is to add a constant variance (e.g., 1 m) to the GNSS covariances before doing the rotation to compensate for this kind of error.

2) Hypotheses Covariances: Each hypothesis j being a set of n_j weighted particles, an estimate of the covariance matrix is:

$$\Sigma_{\text{hyp}_j} = \begin{bmatrix} \sigma_x^{j2} & \sigma_{xy}^j \\ \sigma_{xy}^j & \sigma_y^{j2} \end{bmatrix} \tag{18}$$

with

$$\begin{cases} \sigma_x^{j2} = a_j . \sum_{i=0}^{n_j} w^i (x^i - \bar{x}_j)^2 \\ \sigma_y^{j2} = a_j . \sum_{i=0}^{n_j} w^i (y^i - \bar{y}_j)^2 \\ \sigma_{xy}^{j} = a_j . \sum_{i=0}^{n_j} w^i (x^i - \bar{x}_j)(y_i - \bar{y}_j) \end{cases} \tag{19}$$

and

$$a_j = \frac{1}{1 - \sum_{i=0}^{n_j} (w^i)^2}; \; \bar{x}_j = \sum_{i=0}^{n_j} w^i x^i; \; \bar{y}_j = \sum_{i=0}^{n_j} w^i y^i$$

These formulas assume normalized weights by hypothesis ($\sum_{i=0}^{n_j} w^i = 1$). Each hypothesis Hyp_j is assumed then to

follow a normal distribution, so $Hyp_j \sim \mathcal{N}(\bar{X}_j, \Sigma_{\text{hyp}_j})$ with $\bar{X}_j = [\bar{x}_j, \bar{y}_j]^T$.

C. Consistency Metric

The goal now is to find a decision variable in order to determine whether the GNSS fix is consistent with the hypotheses. Otherwise, it is very likely that either the GNSS fix or the map-aided dead-reckoned hypotheses are faulty. Since the GNSS fix and the means of the hypotheses can be considered independent, the Mahalanobis distance between the two estimates can be calculated by:

$$D_{M_j}(\bar{X}_j) = \sqrt{\left(\bar{X}_j - X_{\text{GNSS}}\right)^T \Sigma^{-1} \left(\bar{X}_j - X_{\text{GNSS}}\right)} \quad (20)$$

with X_{GNSS} being the GNSS fix expressed in the local Cartesian frame and, as a consequence of the independence error of the GNSS with the one of the hypotheses:

$$\Sigma = \hat{\Sigma}_{\text{gnss}} + \Sigma_{\text{hyp}_j} \quad (21)$$

The Mahalanobis distance is a good metric for evaluating the consistency between two uncertain positions by taking into account the covariances of the variables [28].

D. Fault Detection

One interesting property of the Mahalanobis distance is that if $X \sim \mathcal{N}(0, I)$ is a standardized normal distribution, then $D_M^2(X) \sim \chi_p^2$ (with $p = 2$ in this case, as the localization problem is in 2D). This does not hold if the estimate X is biased or inconsistent. We will apply this principle as a way of detecting faults: the Mahalanobis distance is used as a decision variable in a χ^2 test in order to determine the presence of an incoherence.

Let H_0 be the null hypothesis that "the position estimate X is not affected by any fault". If H_o is verified, then $D_M^2(X)$ follows a centered χ_2^2 distribution. To test H_0, a χ^2 test is applied using the Mahalanobis distance as the decision variable and setting a critical value T given a significance level, equivalent in this case to a probability of false alarm (p_{fa}, the probability of detecting a fault when none is present). Here we choose $p_{fa} = 0.01$, and the corresponding critical value for 2 degrees of liberty is $T = 9.2103$. $D_M^2(X)$ is compared to the critical value T: H_0 is considered to be satisfied if $D_M^2(X) < T$, and rejected otherwise, validating the hypothesis H_1 that "a fault is detected".

1) Testing the Hypotheses: At each iteration of the filter, the consistence of the MM is tested. For this, a χ^2 test is applied to each hypothesis Hyp_j separately using the GNSS fix. Each D_{M_j} is compared to the chosen critical value in order to flag the hypotheses not satisfying H_0 as having a fault (i.e. either the hypothesis is wrong or the GNSS fix is biased). The set of coherent hypotheses is then determined. Note that the coherence is with respect to the GNSS fix, which means that rejected matching hypotheses may in fact be correct but GNSS errors are causing them to be rejected. The χ^2 test indicates here the coherence of the situation depicted by the filter with the GNSS fix, not the correctness of the MM: eliminating a hypothesis does not necessarily mean it is incorrect, but that the situation does not allow the positioning system to be trusted.

2) Eliminating Low-Weight Hypotheses: Another selection is made on the set to eliminate the least likely hypotheses. This is to avoid having to consider low-likelihood hypotheses that nevertheless satisfied H_0. Such a situation can arise for example when a bias suddenly affects the GNSS (e.g., multipath) and closely matches a low-weight hypothesis, resulting in a small Mahalanobis distance. A minimum weight (e.g., 0.1) is chosen to eliminate the least likely hypotheses.

E. "Use/Don't Use" Decision

The remaining solution set contains only high-likelihood hypotheses consistent with GNSS. A prudent integrity test is then implemented and the decision for the localization system is set to:

1) *Use* if the remaining set contains only one single hypothesis. In this case, no ambiguity remains, the GNSS positioning is in accordance with the MM results, and the filter gives a high likelihood to the hypothesized position.
2) *Don't Use* if the remaining set contains multiple hypotheses or none. Either the filter was unable to resolve every ambiguity and cannot decide between these hypotheses, or all hypotheses have been eliminated due to a default detection and/or a too low likelihood.

VI. Results

A. Experimental Setup

A C++ implementation of the algorithm was developed using the Pacpus framework[1], providing easy integration with the Heudiasyc Laboratory's test vehicle and offline data replay. An experimental vehicle was used for the acquisition of real road data. The car was equipped with a Septentrio AsteRx2 GNSS receiver (used as single frequency, standalone receiver) and DR information was accessible directly from the vehicle CAN bus, as well as from a Mobileye EyeQ2 system.

The algorithm was tested using Pacpus data replay capability. The algorithm runs in real-time conditions with 2000 particles on an AMD A8-4500M CPU@1.90 GHz with 8 Go RAM.

The 4-km test route is shown in Fig. 8. It is representative of a peri-urban trip (2 × 2-lane roads, including roundabouts). Some high-rise buildings are present, as well as open-sky conditions at the bottom of the figure (urban bypass).

For greater clarity, the results presented in this section only show the top 2 hypotheses. This does not affect the conclusion of the research, since 95% of the dataset results in the MM returning 2 or fewer hypotheses. The other 5% generate a third hypothesis only momentarily and it is quickly discarded. The weight of these discarded hypotheses is never above 0.1 (as shown in Fig. 9), and so they are eliminated by the filter integrity check.

B. Particle Filter Performance

The filter performs quite well in these test conditions. Fig. 10 shows the evolution of the hypothesis weights during the test.

[1] developed at Heudiasyc. More info at https://pacpus.hds.utc.fr

Fig. 8. Left: test GNSS trace (in purple) over the lane map (in blue). This 4-km route is representative of a peri-urban trip (2-lane roadways, presence of buildings in certain areas as well as open-sky conditions in others). Large GNSS errors are clearly visible. Right: satellite view of the area (image from Google Maps).

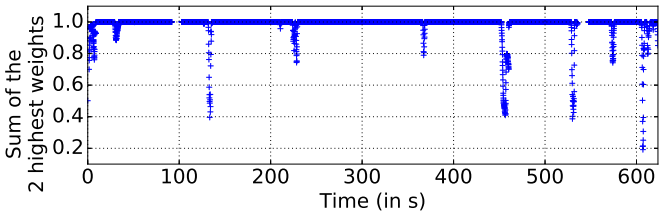

Fig. 9. The sum of two highest weights. 95% of the time the sum is higher than 0.95. When more hypotheses are present, they are quickly eliminated (in the order of 10 seconds).

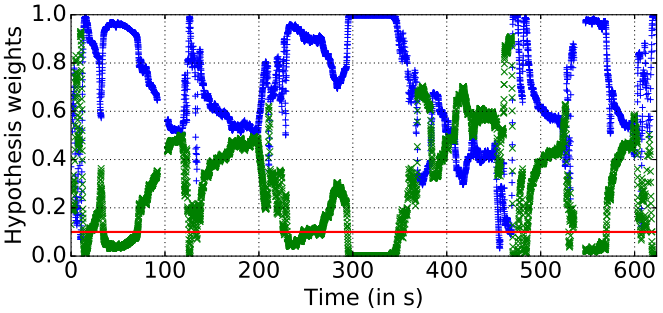

Fig. 10. Evolution of the weights of hypotheses. The blue curve ("+" signs) is the correct matching. For most of the time it is the most likely hypothesis. The red line denotes a 0.1 weight threshold.

TABLE I
MATCHING METRICS ON THE SET OF MATCHING HYPOTHESES

Metric	%
Set containing the correct hypothesis	100
Set of 2 or fewer hypotheses	95.0
Correct best hypothesis	84.6

The hypothesis corresponding to the correct matching is shown in blue, using "+" signs. The red line denotes the 0.1 weight threshold used during the integrity checking phase.

Table I shows performance metrics of the filter: during our tests, the MM always returned the correct matching hypothesis

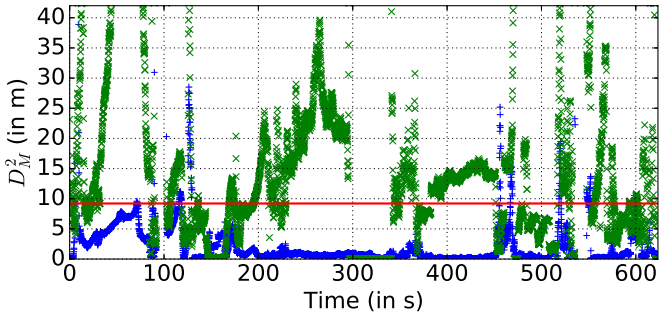

Fig. 11. Squared Mahalanobis distances for the matching hypotheses (the correct hypothesis shown as blue "+"). The red line denotes the critical value for $p_{fa} = 0.01$.

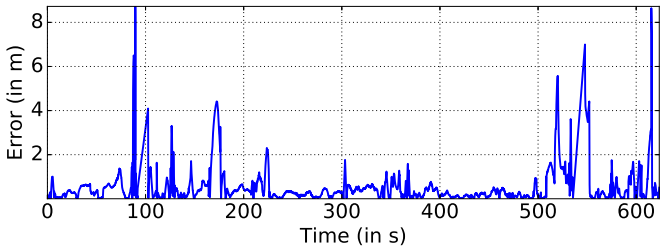

Fig. 12. GNSS error during the test. Spikes are clearly visible, corresponding areas with poor GNSS reception (high-rise buildings nearby).

included in the solution set. Moreover, for 84.6% of the time, this hypothesis was the one with the highest weight. Note that in cases where an incorrect hypothesis had the highest weight, the correct hypothesis generally had a comparable weight, as can be seen in Fig. 10 at $t = [400, 460]$. The filter converged to one or two hypotheses 84.6% of the time. This percentage takes into account the initial convergence and lane forking that produces a larger number of hypotheses.

An interesting event occurs at $t = [300, 340]$, where the correct hypothesis is the only one in the solution set. This is made possible by the use of camera information. Ambiguity is resolved via a lane-changing detection: the filter originally has 2 hypotheses, on the left and right lane. The car is moving from the right lane into the left lane. The filter updates accordingly using the exteroceptive information from the camera system: the right lane particles evolve without issue, changing to the left lane through the adjacency information from the map. This leads the left lane particles off road on the left of the roadway: their weights diminish greatly in comparison to the others and are finally eliminated from the filter.

C. Integrity Check

Fig. 11 shows the evolution of the Mahalanobis distances. For reference, Fig. 12 shows the GNSS error throughout the test. The statistics for the χ^2 test and the weight rejection (coherent solution rejected due to a too low weight value) are shown in Table II. It is clear that both tests are effective in removing the wrong hypotheses, with however some unavoidable false alerts. Note again that rejecting the correct matching only means that the positioning system should not be trusted, and this can be caused by a GNSS error: it is the case at $t = 127$ s. The

TABLE II
REJECTIONS BASED ON χ^2 TEST AND WEIGHT THRESHOLD

% rejection	χ^2 test	Weight based
Correct matching	3.7	0.7
Other hypothesis	57.5	28.7

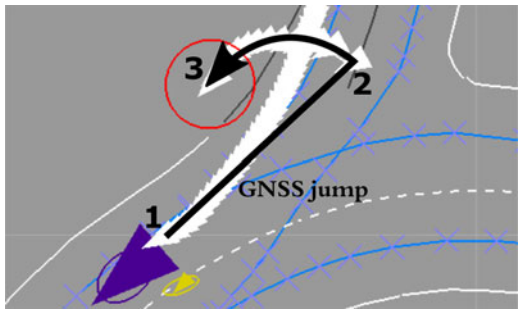

Fig. 13. Screenshot of the situation at $t = 127$ s. A biased GNSS position causes the two matching hypotheses to be rejected.

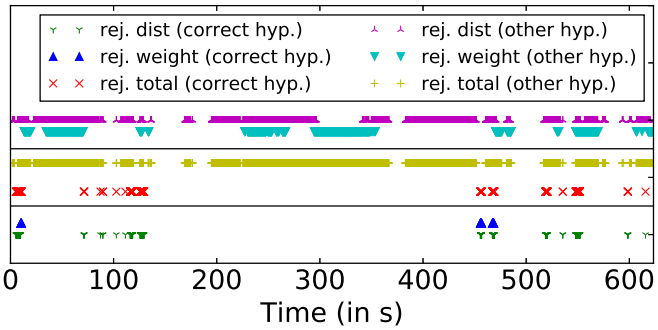

Fig. 14. Rejection classification by cause, for the correct matching and the others. A point indicates that the corresponding hypothesis was rejected at this time step for the given reason.

situation is described in Fig. 13. This screenshot of the actual program running shows that the rejection of the correct matching (violet, larger triangle) is due to a GNSS fix jump (depicted by the arrow between points 1 and 2). The current position (point 3) is affected by a GNSS bias and is not coherent with any matching hypothesis. At that time step, no hypothesis remains in the solution set after the integrity check.

The different causes of rejection are summarized in Fig. 14, which shows the rejections happening at each time step. Each rejection cause is plotted separately on a different line (cf. legend). The bottom two lines show the rejections due to the weight and to the Mahalanobis distance of the correct hypothesis, while the top two correspond to other hypotheses. The two center lines sum up both types of rejections to give an overview of the situation, which confirms the higher number of rejections for incorrect matching. Finally, Fig. 15 shows the result of the "Use/Don't Use" classification. Table III shows that the algorithm has an availability of 65.6% (the amount of time during which the filter flags the correct matching as "Use"). For 34.9% of the time the algorithm flags the positioning as "Don't Use", where it has detected a discrepancy between the map-aided dead-reckoning and the GNSS fix. Finally, the misdetection rate is only 0.54% (the filter classifies an incorrect position as "Use").

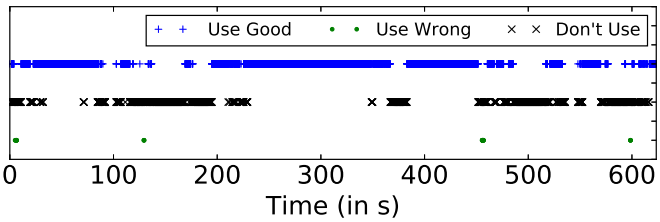

Fig. 15. "Use/Don't Use" decisions by the integrity monitoring method. It can be seen that misdetections ("Use Wrong") are rare. Positioning availability corresponds to "Use Good".

TABLE III
USE/DON'T USE STATISTICS FOR THE INTEGRITY MONITORING METHOD

	Don't Use	Correct Use	Incorrect Use
%	34.9	65.6	0.54

VII. CONCLUSION

MM integrity is becoming an important topic for intelligent vehicles that use information obtained from maps in safety applications. In this paper we have presented a PF-based MM algorithm developed to retain all the likely matching hypotheses throughout the map-aided dead-reckoning, and which includes an integrity check procedure that uses redundant GNSS positions to detect faults. Using real road data, the filter was shown experimentally to respect the integrity of its solution set (i.e. always returning a set containing the correct matching). A new method for internal integrity monitoring was developed using a Mahalanobis distance as decision variable for detecting faults: the distance of the GNSS fix to each matching hypothesis is computed in order to test the coherence of the map-matching. The computation of the covariance matrices of the hypotheses obtained from the particle set is quite straightforward and well adapted to this computation. This allows the positioning system to be flagged as either "Use" or "Don't Use" at a given time step. Experimentally, this system provides only 0.54% of misdetections and has a 65.6% availability (correct hypothesis tagged as "Use") with the low cost sensors used in the experiments. While the algorithm may be over-pessimistic in difficult conditions (e.g., a dense urban environment with multi-lane roads), it is a good indicator for determining whether the positioning system requires complementary information to make it reliable (e.g., multipath detection and correction).

The integrity monitoring method currently runs in snapshot mode (i.e. it uses data from the current time step only). Integrating the temporal dimension could improve the performance of the algorithm. For instance, the evolution of the GNSS position and of the matching hypotheses could provide better information on the source of an incoherence. We also intend to improve implementation performance by parallelizing the execution, which will include porting to graphics processing units (GPU) to make use of the parallel nature of the PF.

ACKNOWLEDGMENT

This paper was carried out within SIVALab, a shared laboratory between Renault and Heudiasyc UMR 7253 UTC/CNRS, using an experimental setup provided by the Equipex Robotex (ANR-10-EQPX-44-0).

REFERENCES

[1] E. D. Kaplan and C. J. Hegarty, *Understanding GPS: Principles and Applications*. Norwood, MA, USA: Artech House, 2006.
[2] F. Li, P. Bonnifait, J. Ibanez-Guzman, and C. Zinoune, "Lane-level map-matching with integrity on high-definition maps," in *Proc. IEEE Intell. Veh. Symp.*, Jun. 2017, pp. 1176–1181.
[3] S. M. Oh, S. Tariq, B. N. Walker, and F. Dellaert, "Map-based Priors for Localization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.,* 2004, pp. 2179–2184.
[4] A. U. Peker, O. Tosun, and T. Acarman, "Particle filter vehicle localization and map-matching using map topology," in *Proc. IEEE Intell. Veh. Symp.,* 2011, pp. 248–253.
[5] Y. Gu, L. T. Hsu, and S. Kamijo, "Passive sensor integration for vehicle self-localization in urban traffic environment," *Sensors*, vol. 15, no. 12, pp. 30199–30220, 2015.
[6] D. Betaille, R. Toledo-Moreo, and J. Laneurit, "Making an enhanced map for lane location based services," in *Proc. 11th Int. IEEE Conf. Intell. Transp. Syst.*, Oct. 2008, pp. 711–716.
[7] P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving," in *Proc. IEEE Intell. Veh. Symp.*, Jun. 2014, pp. 420–425.
[8] Jie Du and M. Barth, "Next-generation automated vehicle location Systems: Positioning at the lane level," *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 1, pp. 48–57, Mar. 2008.
[9] D. Betaille and R. Toledo-Moreo, "Creating enhanced maps for lane-level vehicle navigation," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 4, pp. 786–798, Dec. 2010.
[10] M. A. Quddus, W. Y. Ochieng, and R. B. Noland, "Current map-matching algorithms for transport applications: State-of-the art and future research directions," *Transp. Res. Part C, Emerg. Technol.*, vol. 15, no. 5, pp. 312–328, Oct. 2007.
[11] R. Toledo-Moreo, D. Betaille, F. Peyret, and J. Laneurit, "Fusing GNSS, dead-reckoning, and enhanced maps for road vehicle lane-level navigation," *IEEE J. Sel. Topics Signal Process.*, vol. 3, no. 5, pp. 798–809, Oct. 2009.
[12] M. A. Quddus, W. Y. Ochieng, and R. B. Noland, "Integrity of map-matching algorithms," *Transp. Res. Part C, Emerg. Technol.*, vol. 14, no. 4, pp. 283–302, Aug. 2006.
[13] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 174–188, Feb. 2002.
[14] F. Gustafsson *et al.*, "Particle filters for positioning, navigation, and tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 425–437, Feb. 2002.
[15] J. Levinson, M. Montemerlo, and S. Thrun, "Map-based precision vehicle localization in urban environments," in *Robotics: Science and Systems III*. Cambridge, MA, USA: MIT Press, Jun. 2007.
[16] J. Rabe, M. Hubner, M. Necker, and C. Stiller, "Ego-lane estimation for downtown lane-level navigation," in *Proc. IEEE Intell. Veh. Symp.*, Jun. 2017, pp. 1152–1157.
[17] I. Szottka, "Particle filtering for lane-level map-matching at road bifurcations," in *Proc. 16th Int. IEEE Conf. Intell. Transp. Syst.*, Oct. 2013, pp. 154–159.
[18] P. Bonnifait, J. Laneurit, C. Fouque, and G. Dherbomez, "Multi-hypothesis map-matching using particle filtering," in *Proc. 16th World Congr. ITS Syst. Serv.*, Sep. 2009, pp. 1–8.
[19] J. Cosmen-Schortmann, M. Azaola-Saenz, M. Martinez-Olague, and M. Toledo-Lopez, "Integrity in urban and road environments and its use in liability critical applications," in *Proc. IEEE/ION Position, Location Navig. Symp.*, 2008, pp. 972–983.
[20] P. Merriaux, Y. Dupuis, P. Vasseur, and X. Savatier, "Fast and robust vehicle positioning on graph-based representation of drivable maps," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2015, pp. 2787–2793.
[21] Z. Tao and P. Bonnifait, "Sequential data fusion of GNSS pseudoranges and dopplers with map-based vision systems," *IEEE Trans. Intell. Veh.*, vol. 1, no. 3, pp. 254–265, Sep. 2016.
[22] F. Li, P. Bonnifait, and J. Ibanez-guzman, "Estimating localization uncertainty using multi-hypothesis map-matching on high-definition road maps," in *Proc. 20th Int. IEEE Conf. Intell. Transp. Syst.*, Yokohama, Japan, 2017, pp. 1408–1413.
[23] Z. Kim, "Robust lane detection and tracking in challenging scenarios," *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 1, pp. 16–26, Mar. 2008.
[24] N. Bergman, "Recursive bayesian estimation navigation and tracking applications," Ph.D. dissertation, Linköping Univ., Linköping, Sweden, 1999.
[25] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2002.
[26] M. Jabbour, P. Bonnifait, and V. Cherfaoui, "Map-matching integrity using multihypothesis road-tracking," *J. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 189–201, Oct. 2008.
[27] C. Fouque and P. Bonnifait, "Matching raw GPS measurements on a navigable map without computing a global position," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 2, pp. 887–898, Jun. 2012.
[28] Y. Bar-Shalom and X.-R. Li, *Estimation and tracking: Principles, Techniques, and Software*. Norwood, MA, USA: Artech House, 1993.

Franck Li received the Engineering and M.Sc. degrees in computer science from the Université de Technologie de Compiègne (UTC), Compiègne, France, in 2013. He is currently working toward the Ph.D. degree at the Heudiasyc Laboratory, UMR 7253, UTC, in an industrial partnership with the French car manufacturer Renault. His topic of research is the use of high-definition road maps for the intelligent vehicle and the integrity of positioning.

Philippe Bonnifait received the Ph.D. degree in automatic control and computer science from the École Centrale de Nantes, Nantes, France, in 1997. He is currently a Professor with the Computer Science and Engineering Department, Université de Technologie de Compiègne, Compiègne, France. Since 1998, he has been with Heudiasyc UMR 7253, a joint research unit between UTC and CNRS. His research interests include intelligent vehicles, and high-integrity positioning and map-matching for autonomous navigation in structured outdoor environments.

Javier Ibañez-Guzmán (M'89) received the Ph.D. degree from the University of Reading, Reading, U.K., on a SERC-UK fellowship, and the M.S.E.E. degree from the University of Pennsylvania, Philadelphia, PA, USA, as a Fulbright scholar. In 2011, he was visiting scholar with the University of California, Berkeley, Berkeley, CA, USA (CITRIS), working on connected vehicle applications. He is currently a member of the technical staff with the Renault s.a.s., carrying out work on autonomous vehicle navigation technologies and driving assistance systems. Formerly, he was Senior Scientist at a National Research Institute in Singapore, where he spearheaded work on autonomous ground vehicles operating in unstructured environments. He has several publications and patents in the robotics and automotive domains. He has successfully supervised a number of Ph.D. students. He is a C.Eng. (U.K.) and a Fellow of the Institute of Engineering Technology (U.K.).