Hindawi Publishing Corporation
EURASIP Journal on Embedded Systems
Volume 2007, Article ID 39350, 12 pages
doi:10.1155/2007/39350

Research Article

Real-Time Implementation of a GIS-Based Localization System for Intelligent Vehicles

Philippe Bonnifait, Maged Jabbour, and Gérald Dherbomez

Heudiasyc UMR CNRS 6599, Université de Technologie de Compiègne, BP 20529, 60205 Compiègne Cedex, France

Received 31 October 2006; Revised 17 March 2007; Accepted 2 May 2007

Recommended by Roger Reynaud

This paper presents a loosely coupled fusion approach that merges GPS data, dead-reckoned sensors, and GIS (geographical information system) data. The GPS latency is compensated for by the DR sensors and the use of an xPPS signal. The fusion of the estimate with the map data is spatial-triggered, while the fusion with the GPS is time-triggered. We present a strategy that relies on pose tracking which is reinitialized when GPS data become incoherent. Particular attention is given to the representation of the road map and to the management of a cache memory for efficiency purposes. We report experiments carried out with our equipped car and a GIS whose characteristics are well adapted to embedded constraints.

1. INTRODUCTION

Dynamic localization is a key issue for intelligent vehicles performing driving assistance tasks or autonomous navigation in the presence of uncertainty and variability in their external environment.

Global navigation satellite-based systems (GNSS) such as GPS, Glonass and, in the near future, Galileo are interesting key components in localization. Unfortunately, when high integrity and high availability are required, a GNSS receiver alone is not sufficient for intelligent vehicles, since satellites can be masked, and since the propagation of the signal can suffer from multitracks. One way of overcoming this problem is to use dead-reckoned (DR) sensors or inertial units. These can be sufficient when outages are brief, but they cannot cope with certain situations, such as in urban areas when navigating in an urban canyon. In such cases, the drift of DR localization can be too large for the needs of the task. This problem can be addressed through the use of extra exteroceptive sensors like video cameras or laser scanners. A number of studies have demonstrated the feasibility of this approach. For instance, Royer et al. [1] have developed a navigation system for a Cycab that can control the trajectory of the vehicle with respect to a learned trajectory, using only a monocular vision system. The principle is as follows: in the learning stage, the vehicle extracts and localizes characteristic features of the environment; while navigating, it matches the features it is detecting in the current view with those that have been learned. A precise localization is subsequently computed using a Ransac method.

An ideal localization system is an embedded system that is able to deal with all of the following technologies: GNSS, DR sensors, and exteroceptive sensors detecting natural landmarks.

When implementing such a system, a key issue is the management of the landmarks that have been characterized and localized previously. The question is now how to organize the landmark information for an intelligent vehicle moving in a large area containing many roads. The usual answer to this question is to group landmarks within local maps, a local map being a set of landmarks considered as a monolithic entity because of (i) memory constraints arising from the use of embedded systems, (ii) the need to download or update a limited amount of data from a remote server, and (iii) the connections that exist between the landmarks, essential to compute a location.

Localization with respect to a digital map describing the road network is an essential task for intelligent vehicles [2, 3]. The user of a vehicle usually specifies its itinerary by indicating the destination address. In this case, the geocoding facility of the GIS is very useful when converting an address like “10, Albert Road” into global (x, y) coordinates.

The GIS can also be used for the management of landmarks by making use of the road descriptions stored in the map database. An efficient technique is to group the landmarks in local maps corresponding to the different roads [4]. Indeed, the local maps can have the same identification numbers (hereafter denoted as ID) as the roads. From a real-time point of view, this is essential since the local maps can be stored in a look aside database (LADB) and easily be retrieved and loaded into the vehicle's memory for localization purposes.

This paper focuses on absolute localization and the ID retrieval problem. In particular, it deals with real-time constraints and positioning integrity. The implementation of the precise localizer is not considered here. Unlike many industrial prototypes that have mainly low frequency (often 1 Hz), this work also deals with the development of high-frequency systems. Moreover, we consider a matching strategy that does not rely on the use of a precomputed route, since our system is not limited to applications dealing only with navigation.

We consider a vehicle equipped with an odometer (we use the ABS sensors of the rear wheels), a fiber-optic gyrometer, an L1 (single-frequency) differential GPS receiver, and embedded GIS software managing a standard road map database (NavTeQ in this particular case [2, 3]).

During the first stage, the DR sensors are merged with the GPS fixes using a loosely coupled Kalman filter approach. Thanks to the predictor/estimator mechanism in the pose tracking process, GPS latency is eliminated. The second stage involves fusion with the map data. A request is sent to the GIS server (embedded or remote) that returns the roads contained in a box whose center and size have been instantiated in the request. Then, a road selection method (also called map matching) is used to select the most likely segment and finally this segment is merged with the state estimate. The management of a road cache memory is an important issue. If it is too small, the road selection can fail. If it is too large, the selection will be uselessly time-consuming. In addition, the management of the map has to be done spatially and just when necessary.

The paper is organized as follows. In Section 2, the localization method fusing GPS and DR sensors is described and the method for compensating the GPS latency is introduced. The fusion of the map data is then presented in Section 3, and the complete algorithm, the map representation, and the road selection strategy are described. We look at the management of the road map's cache memory and show that the right road is always present in the cache, while the worst-case execution time is respected and inaccuracies are taken into account. Finally, Section 4 is devoted to real experiments that illustrate the performance of the fusion process, the latency, and cache management.

2. GPS AND DR SENSORS FUSION

Localization using DR sensors and GPS data can be achieved via a multisensor fusion approach, that is, an approach that explicitly takes into account inaccuracies to handle redundancy and complementarity.

From the real-time point of view, efficient methods are those that rely on state observation because of the predictor/estimator mechanism that can be implemented as a recurrent task (termed *complex task* by Kopetz [5]) that only needs to keep in memory the state vector between two sampling times.

Let us first consider the fusion of DR sensors with GPS. The GPS fixes are projected in a local frame tangential to the surface of the earth.

This fusion problem can be expressed by a discrete state-space representation, sampled with respect to time:

$$\begin{aligned} X(t_k) &= f(X(t_{k-1}), U(t_k)), \\ Y_g(t_k) &= g(X(t_k)), \end{aligned} \tag{1}$$

where

(i) $t_k = t_0 + k \cdot Te$;
(ii) $X(t_k) = [x(t_k)y(t_k)\theta(t_k)]^T$ is the pose (position and heading) of the vehicle in the projected frame;
(iii) the evolution model corresponds to the DR model;
(iv) $U(t_k) = [\delta(t_k)\omega(t_k)]^T$ is the vector of the elementary traveled distance and elementary rotation measured by the DR sensors;
(v) $Y_g(t_k) = [x_g(t_k)y_g(t_k)]^T$ is the GPS measurement vector after projection.

The evolution model of the center of the rear axle can be expressed by

$$\begin{aligned} x(t_k) &= x(t_{k-1}) + \delta(t_k) \cdot \cos\left(\theta(t_{k-1}) + \frac{\omega(t_k)}{2}\right), \\ y(t_k) &= y(t_{k-1}) + \delta(t_k) \cdot \sin\left(\theta(t_{k-1}) + \frac{\omega(t_k)}{2}\right), \\ \theta(t_k) &= \theta(t_{k-1}) + \omega(t_k). \end{aligned} \tag{2}$$

If the GPS antenna is located at the origin of the mobile frame, then the observation equation becomes

$$Y_g(t_k) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \cdot X(t_k). \tag{3}$$

A very popular approach for fusing localization data is the Bayesian framework. An extended Kalman filter (EKF) is often used. Unscented Kalman filtering (UKF) [6] is a new approach, very popular currently, because it can more precisely estimate the error covariance than an EKF, especially if this covariance is large with respect to the nonlinearity. In this paper, the equations of the filters are not detailed. For more information, the reader can see [7].

In order to implement a Kalman state observer, we propose to choose the maximum sampling period for two reasons:

(a) reduction of processing workload,
(b) it is well adapted to handle GPS latency.

Tessier et al. [8] have proposed an architecture for the fusion of delayed observations using data buffers: each piece of data being timestamped and buffered together with the filter estimations. Once a new piece of data appears, the fusion is computed in the past and the current estimation is recomputed from the buffered data. We propose a strategy that compensates for the GPS latency by using an xPPS (xpulses per second) signal, a GPS receiver running in synchronized mode, and a predictor implementation of the state observer. This strategy is more efficient than Tessier's, but works only for small delays. It consists in delaying to the maximum the use of the GPS data in the algorithm in order to leave for it enough time to compute and transmit its results. If the sampling period is higher than the worst-case GPS latency, one solution is to perform the GPS estimation stage first with each new sample (see Figure 1). By this way, the first step at time t_k is to correct the pose $X(t_{k-1})$ using $Y_g(t_{k-1})$. Once this has been done, the prediction stage provides an estimate of state $X(t_k)$ using the DR sensors. This is the output of the filter that works as a one-sample DR predictor.

It should be noted that a good initialization of the Kalman filter, especially of the heading, accelerates significantly its convergence. Moreover, in order to filter GPS jumps due to multitracks (especially in urban areas), a coherence test based on a Mahalanobis distance is applied in the correction stage of the filter [4].

The strategy we have chosen can be expressed as follows. When few GPS fixes are inconsistent with the DR predictions, we suppose that it is the GPS that is faulty. Otherwise, that is, when GPS and DR prediction are inconsistent for a long time, the localizer is reinitialized as shown in Figure 2. The initialization consists in waiting for a new GPS fix. The tracking is done by an EKF, the implementation of which follows the chronogram of Figure 1.

This strategy is also robust when initializing or reinitializing the system with bad GPS data. Let us study the system behavior when this case happens. Two cases can occur with a bad reinitialization, depending on the validity of the information contained in the GST NMEA sentence.

Case 1. The position is bad and the confidence ellipsoid is large. If a new good GPS fix arrives, it will be considered consistent, and therefore used to correct the previous estimate.

Case 2. The position is bad and the confidence ellipsoid is small (inconsistent data). The filter will not be able to detect this at once, since the initial covariance is small. When good GPS fixes are restored, the filter will consider them as bad data until a new reinitialization occurs. It should be noted that multitracks are often very short for a moving receivers. So, the probability of the filter being reinitialized in this case remains low.

To obtain a high positioning frequency (e.g., a video application client working at 60 Hz and requiring positioning data at the same rate), a client-server mechanism can be implemented. Between two low-rate pose computations, the positioning server extrapolates the previously computed pose using the known linear and rotational speeds (cf. (2)).

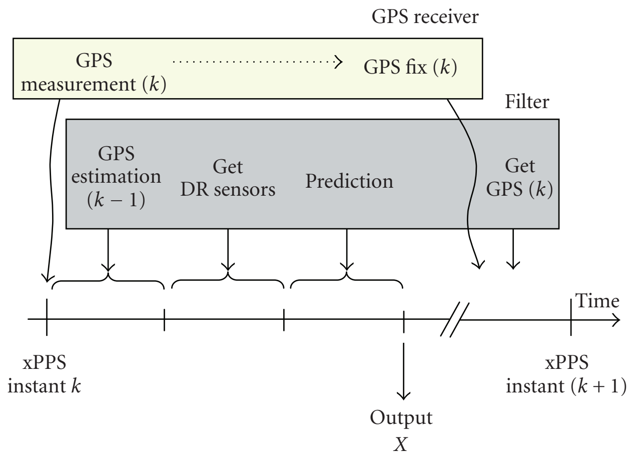


FIGURE 1: Chronogram of the state observer for GPS latency compensation. Because of the synchronized mode of the GPS receiver, it takes its pseudorange measurements at the xPPS instant but provides a solution when it has completed its computation.

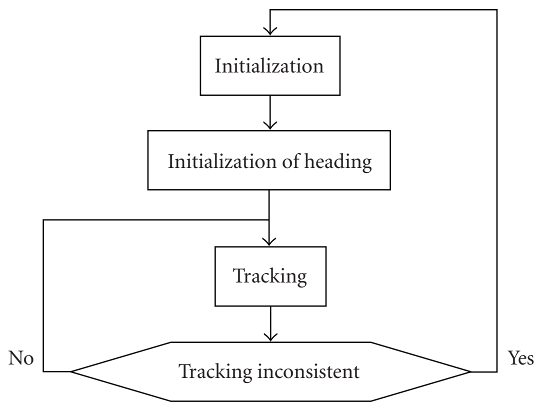


FIGURE 2: Synopsis of the states of the localizer.

This mechanism also allows the transmission of positioning data to different clients at different request rates, using the same position computations.

If the vehicle is motionless, the heading is not observable. In this case, to avoid error, the previously computed pose is maintained, and no prediction or updating is performed until the vehicle starts to move again. In practice, we check that the odometers' counters have changed since the previous step. This corresponds to a spatial sampling condition equal to the sensor's resolution, which is here about 2 cm.

3. GPS, GIS DATA, AND DR SENSORS FUSION

Because of GPS outages occurring in urban areas for instance, an effective means of correcting DR localization drift is to use map data as an observation in the filtering process. This can be done by serializing the two correction stages, as shown in Figure 3. The GPS correction stage is carried out at

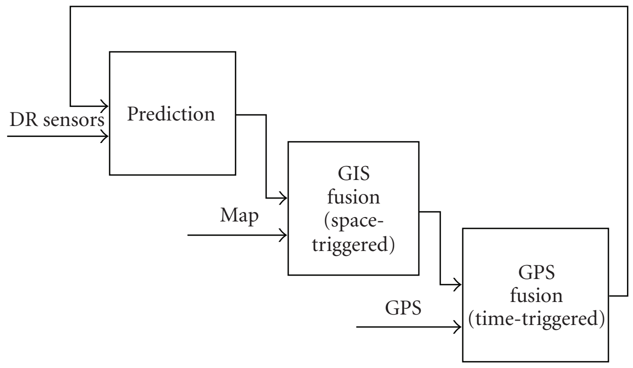


Figure 3: Hierarchical fusion strategy.

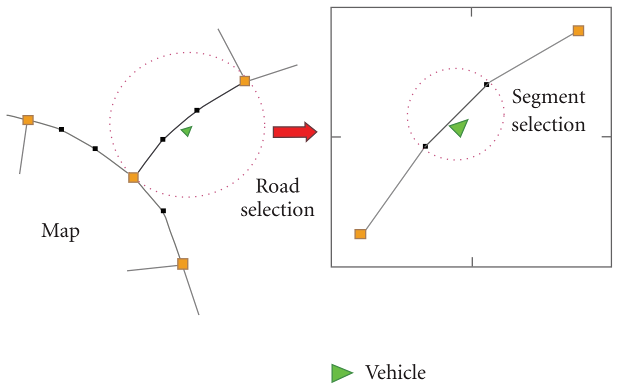


Figure 4: Road selection and segment selection.

the end in order to apply the same GPS latency management strategy as before.

Localization on a map is a problem that can have several solutions, for instance while approaching junctions, when the quality of GPS positioning is poor, or given a map with a poor absolute precision. Particle filtering has shown interesting characteristics in relation to this problem, particularly because of its ability to manage several hypotheses [9]. Nevertheless, its real-time implementation [10] is time consuming and its convergence with a small number of particles is not guaranteed because of the degeneracy problem that can occur. An alternative would be doing a pose tracking with the most likely road. To illustrate this, let us explore the concepts of roads and segments.

A road map is usually a set of digitized roads described by polylines represented by their centerline. Their topological information is very good, while their geometry is often rough. Each junction is represented by a *node. Shape points* are used to enhance the geometry description. By definition, a road is a polyline linking two nodes. A segment is defined as the linear interpolation between two points being either nodes or shape points (see Figure 4).

3.1. GIS data fusion

For real-time computation purposes, we adopt a monohypothesis approach: one road only is used in the tracking process. If the filter should select the wrong road, this error is detected thanks to the GPS, and the filter is reinitialized.

An important characteristic of map matching is its *spatial* nature. A time-triggered approach is not well adapted for this problem since it is not elapsed time but traveled distance (also called abscissa curvilinear) that is important for the convergence. Moreover, many approaches rely on data fusion approaches that suppose independence of the errors. If the vehicle is motionless, then the same map data can be used several times, violating the independence hypothesis. For these reasons, map matching can be formulated by a space-triggered state-space description.

Let suppose first that a candidate segment has been selected:

$$\begin{aligned} X(l_i) &= f(X(l_{i-1}), U(l_i)), \\ Y_m(l_i) &= h(X(l_i)), \end{aligned} \tag{4}$$

where

(i) $l_i = l_0 + i \cdot Le$ is ith sample of the traveled distance from the beginning, Le the sampling distance;
(ii) $X(l_i) = [x(l_i)\ y(l_i)\ \theta(l_i)]^T$ is the pose of the vehicle in the frame of the map;
(iii) the evolution model is the DR navigation model;
(iv) $U(l_i) = [\delta(l_i)\ \omega(l_i)]^T$ is the vector of the elementary traveled distance and elementary rotation measured by the DR sensors;
(v) $Y_m(l_i) = [x_m(l_i)\ y_m(l_i)]^T$ is a point (which we term map measurement) that corresponds to the projection of the estimated position onto the most likely segment (see (5) and Figure 5):

$$Y_m(l_i) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \cdot X(l_i). \tag{5}$$

The fusion of the estimate with the map is performed during a Kalman estimation stage. The covariance associated with the map observation is modeled by an ellipsoid around the selected segment as shown in Figure 5 [11]. The center of the ellipsoid is Y_m, the orthogonal projection on the segment of the last estimated location. In the frame associated with the segment, the longitudinal inaccuracy is far greater than the lateral inaccuracy. Theoretically, the longitudinal inaccuracy can be chosen as large as possible, even infinite for a long segment. In practice, we consider a one-sigma value in the order of the length of the segment.

Before doing the correction stage, a consistency test is applied to check if the selected road is correct. If not, a road selection stage or a segment selection stage is carried out.

3.2. Road and segment selections

In order to fuse the road information with the estimated position, the system needs to know the most likely segment.

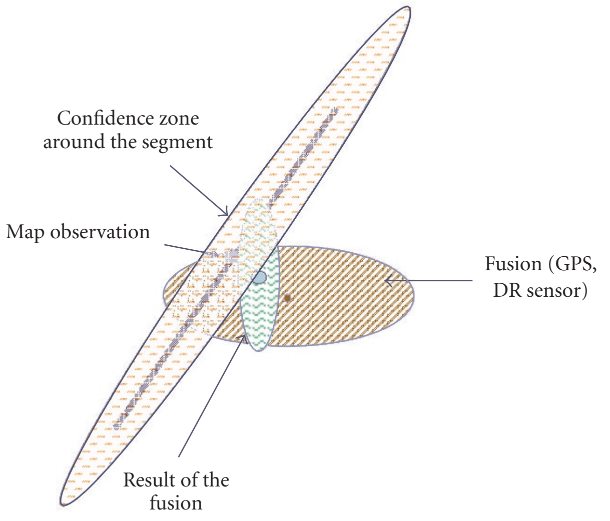


Figure 5: Fusion of the estimate with the selected segment.

The following situations can arise:

(a) there is no selected road (initialization);
(b) the system needs to select the most likely segment of the current road;
(c) the vehicle is approaching the end of a road.

Cases (a) and (c) are road-selection problems, while (b) is a segment-selection issue.

A number of solutions for solving case (a) exist in the literature [11]. Where a high level of integrity is required, the process may be particularly sophisticated, although in this paper, where our concern is efficiency, we propose a simple approach. First, the road segments whose orientation is compatible with the vehicle heading are selected. The heading information stored in the road map structure is used here to accelerate this processing (Section 3.5). Then, in this set, the segment with the smallest distance from the estimated position is chosen.

Now, let us suppose that the selected road is consistent with the current pose. The goal is to select the most likely segment of the road (if the road is a polyline with at least two segments). The most likely segment is simply the closest one.

Finally, suppose that the vehicle is approaching the end of the road. This road selection consists in selecting the most likely road connected to the previously selected road. This connectivity information is once again contained in the array of road IDs connected to the road origin and endpoint, in the road map structure. An efficient strategy is to select the road whose direction corresponds most closely to the heading estimation.

Given that the risk of selecting the wrong road is high when the vehicle's estimated location is close to a node (ambiguity area), we use a careful strategy: the map correction is not computed and the connected road selection is not performed until the vehicle leaves the ambiguity area. This helps to reduce erroneous matches, especially when the map contains large errors.

```
(A) Initialization(·)
Wait4GPS(·)
Heading_initialization(·)
$\Delta$4MF = 0 //Distance for map fusion
road = select_new_road($x_k$, map_cache)
Counter=0 //GPS outliers

(B) Tracking loop trigged at each xPPS
$\Delta_k$=Get_traveled_distance(); $\delta_k = \Delta_k - \Delta_{k-1}$
$\Omega_k$=Get_heading_rotation(); $\omega_k = \Omega_k - \Omega_{k-1}$
If $\delta_k \neq 0$ //the vehicle is moving
 Then
 ($x_{k-1}$, Counter) = correction($x_{k-1}$, y_gps$_{k-1}$)
 If (Counter > threshold) Then Initialization(); break;
End
 $x_k$=prediction($x_{k-1}$, $\delta_k$, $\omega_k$)
 If ($\Delta_k$ − $\Delta$4MF>Le)
   $\Delta$4MF=$\Delta_k$
   (Seg, matched_pt) = select_seg($x_k$, road)
   TH = Map_error + HE; //ambiguity zone size
   If (get_dist_to_road_node(matched_pt)>TH) Then
     ($x_k$, Inconsistency)=fusion($x_k$, Seg)
     If (Inconsistency is bad) then
        road = select_new_road($x_k$, map_cache)End
   Else //zone of ambiguity
        road = select_connected_road($X_k$, road)
   End
 End
End
Function new_road = select_connected_road($X_k$,
current_road)
Selects the road which is connected to road and is most
consistent with $X_k$, Eventually, returns current_road

Function road = select_new_road($x_k$, map_cache)
Selects the most consistent road with $X_k$

Function seg = select_seg($x_k$, current_road)
Selects the most likely segment of the current road

Function dist=get_dist_to_road_node(matched_pt)
Returns the distance to the current matched point
```

Algorithm 1

3.3. Algorithm

The global fusion algorithm is spatial- and time-triggered. It uses the serial fusion strategy shown in Figure 3, implements the GPS latency management described in Section 2, and has the same running modes as the GPS and DR fusion algorithm (Figure 2). We now consider its pose tracking strategy (see stage B Algorithm 1). The fundamental trigger is the xPPS signal obtained from the GPS receiver (the xPPs is assumed always to be available, even during long outages).

The traveled distance and heading rotation are first obtained from the DR sensors. If the vehicle is moving, the correction of the previous prediction is computed using the previous GPS fix. Using a Mahalanobis test, an incoherence counter (see *Counter*) is incremented (when bad GPS data is detected) or reinitialized (each time a coherent GPS data is received). The predicted state is computed. If the traveled distance since the last fusion with the map is higher than *Le* (spatial sampling period), a segment selection is performed. The Kalman correction stage with respect to the selected segment is performed only if the vehicle is not in the zone of ambiguity (see *function get_dist_to_road_node(matched_pt)>TH*). The zone of ambiguity, defined by the threshold *TH*, includes an estimation of the map error (assumed to be known, e.g., 10 meters) and an estimation of the accuracy of the estimated position (termed HE, i.e., horizontal error). HE is defined to be the 1-sigma error circle, that is, the maximum eigenvalue of the position covariance matrix. If the fusion of the segment is not consistent (see (x_k, *Inconsistency*) = fusion(x_k, *Seg*)), then a new road selection stage occurs. While the vehicle is located in the zone of ambiguity, a connected road is sought (see *select_connected_road*(X_k, *roa*)).

It will be noticed that the tracking mode of the localizer depends only on the coherence of the GPS points with respect to the state estimate. By reinitializing the localizer, this mechanism solves map-matching mistakes.

One should also note that the GPS bad data and the map-matching errors are not handled in the same way, since the GPS outliers can come from a change in satellite constellation, multitracks, and are rather brief, while the matching errors are derived from an erroneous segment-selection mechanism.

3.4. Integrity and behavior analysis of the algorithm

Integrity can be defined as the confidence which can be placed in the correctness of the information supplied by the whole system. The most robust way to ensure that the positioning information is valid is to have a multilayer series of checks [12]. It is important to have integrity checking at the end-user level because this is the only place where all information used to form the position solution are present. For a road vehicle, real-time integrity estimation is very challenging because it has to take into account the degradation of the satellite visibility due to the environment of the vehicle and map errors.

Here, there are several problems to tackle:

(i) convergence of the method,
(ii) tracking divergence detection due to map and positioning errors,
(iii) bad GPS fixes arising from multitracks for instance,
(iv) map cache management which means here to keep the estimate in the cache with a guaranty zone.

The last point is studied in Section 3.6.

3.4.1. Nonobservable situation

If the vehicle is motionless, the heading is not observable. In this case, thanks to the spatial triggering, no computation is done. This prevents any drift.

3.4.2. Road tracking

For simplicity, let us suppose that the road is described by an infinite polyline that corresponds to the right road. If the GPS is available and is coherent with the DR sensors, then the positioning is good and the map matching is trivial. If there is no GPS, the map observation is only able to correct the transversal DR drift [13]. It is well known [14] that lateral drift exceeds longitudinal drift. So, the map observation that we have proposed models correctly this phenomenon, since the map ellipsoid has a small lateral standard deviation and a large longitudinal.

3.4.3. Convergence in case of a bad road selection

Suppose now that the filter has been initialized with an incorrect road (because of a bad road selection in a zone of ambiguity, or because of bad GPS fixes). The only property that can be proved is that if a bad choice occurs, the system is able to detect it after a bounded duration or traveled distance.

There are two cases.

(a) The GPS is good and coherent: if the location is inside a zone of ambiguity, the fusion with the map is not performed and the road-selection error is undetectable. As soon as the vehicle leaves this zone, the fusion with the map will be incoherent and a new road selection is done (*select_new_road()*).
(b) The GPS is poor: as soon as good GPS is restored, the filter will consider it as a bad data and after several steps (counter > threshold), the filter will be reinitialized.

3.4.4. Robustness in relation to GPS outliers

Suppose now that the GPS receiver suffers from multitracks. If this phenomenon is short, then the incoherent fixes will be rejected (counter < threshold). Please note that the map can help in detecting GPS errors. If not, the filter will be reinitialized with bad GPS fixes and remains in that state until good GPS is restored.

3.4.5. Robustness in relation to bad map data

Let us consider the case where the map is very bad (e.g., in intertown situations). In this case, there will never be any map fusion and the system will continuously carry out new road selections without making any map fusion.

3.5. Map representation

In order to facilitate the road selection and fusion processes, the map representation is enhanced. Let us consider its structure.

The extracted map is a vector of road structures; each road structure element contains the following:

(i) road or street *name,*

(ii) *ID*, that is, a unique identifier for each road: this is indispensable for navigation and map-matching purposes,

(iii) *speed limit*,

(iv) *driving direction* information: this indicates whether the road is a one-way segment (from east to west or west to east) or a two-way road. This field also indicates if the road is restricted to pedestrians or cars,

(v) number of shape points that describe the geometry of the road,

(vi) *shape points* coordinates array,

(vii) *heading angles* of the segments defined by the shape points,

(viii) number of connected segments to the road *origin pointn*,

(ix) IDs array of the connected segments to the road origin point,

(x) number of connected segments to the road *endpoint*,

(xi) IDs array of connected segments to the road endpoint.

The connectedness is described by the arrays containing the segments IDs.

3.6. Map cache memory management

Let us suppose that we have a GIS server, which can be either embedded or remote. For an efficient map management, let us consider a map cache memory corresponding to an area whose shape is a square. Such a geographical zone is easy to extract since no distance has to be computed: roads are extracted using only tests on the road segment coordinates. The management of the cache memory is a parallel task with the filter computations.

The center c and the semilength r of the side of the square have to be specified in the request made by the fusion client. When a request is received, all the roads partially or completely included within the square of center c and side length $2r$ are extracted.

For an intelligent map cache memory management, two problems have to be dealt with: center and semilength management.

Since the vehicle continues to move while the map cache requests are made, a predicted request center has to be computed to take into account the worst-case server process and transmission delay (denoted as $T_{\text{GIS}} = T_{\text{Extraction}} + T_{\text{Transmission}}$). The map extraction therefore involves anticipation: we use the vehicle's estimated heading and speed.

Let us denote by $X_c = [x_c\ y_c]^T$ the coordinates of the center of the cache memory. Once the vehicle is close to the limit of the current cache (see Figure 6), a request is sent to the server.

The condition of the request is given by (6) where the distance between the current vehicle position and the center of the cache is compared to semilength of the cache. λ is a parameter necessary for managing the request to the server in realtime. It corresponds to the proportion of r from which

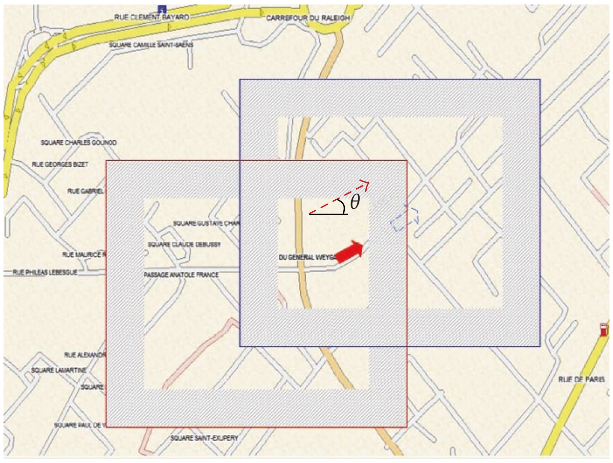


FIGURE 6: Extrapolation of the cache center while approaching the limit of the current cache (the gray zone corresponds to parameter Δ).

a request has to be made before leaving the area of the current cache:

$$\|X_c(t_{k-1}) - X(t_k)\| \geq \lambda \cdot r. \tag{6}$$

If the request condition is verified, the new center position is computed using a nominal speed v, for instance 50 Km/h in urban areas:

$$\begin{aligned} x_c(t_k) &= x(t_k) + T_{\text{GIS}} \cdot v \cdot \cos(\theta(t_k)), \\ y_c(t_k) &= y(t_k) + T_{\text{GIS}} \cdot v \cdot \sin(\theta(t_k)). \end{aligned} \tag{7}$$

While the request is being processed, the vehicle does not remain stationary. The risk is that it might leave the current cache area. A security zone is characterized by the following condition on the traveled distance:

$$T_{\text{GIS}} \cdot v \leq (1-\lambda) \cdot r. \tag{8}$$

This leads to the following condition:

$$\lambda \leq 1 - v \cdot \frac{(T_{\text{transmission}}(r) + T_{\text{extraction}}(r))}{r}. \tag{9}$$

An interesting value is the distance (denoted as Δ) before leaving the region of interest:

$$\begin{aligned} \Delta &= (1-\lambda) \cdot r, \\ \Delta &> v \cdot T_{\text{GIS}}(r). \end{aligned} \tag{10}$$

Suppose now that the maximum map inaccuracy (denoted as E_M) is known and the localization inaccuracy (denoted as E_L) is estimated in real time by the Kalman filter. In order to have a reliable road selection, Δ must verify

$$\Delta > E_M + E_L + v \cdot T_{\text{GIS}}(r). \tag{11}$$

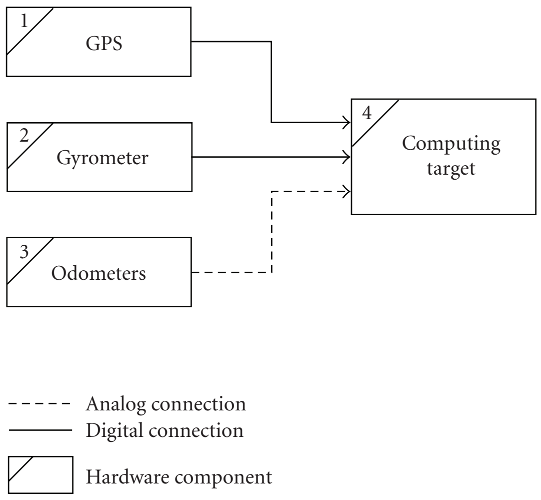


Figure 7: Architecture diagram of the hardware components.

The management of the semilength of the square is the second most important issue in cache management. If r is too small, then the road selection can fail due to localization and map errors. Otherwise, if r is too large, then

(i) the memory cache retrieval can be long (especially if the GIS server is remote);
(ii) the size of the cache can exceed the maximum size of the target's RAM;
(iii) the road-selection procedure can be uselessly time-consuming.

In summary, the cache memory management depends mainly on parameters Δ and r, which have to be determined regarding the implementation. A case study will be presented in the next section.

4. EXPERIMENTS

4.1. Embedded real-time platform

In this section, we describe the different hardware and software components used to test and validate our localization system.

The hardware components are separated into two categories: sensors and computing resources. The architecture diagram in Figure 7 shows the interaction between these components.

The components of Figure 7 are as follows.

(i) A computing target (4): a Shoebox PC with a low electrical power consumption processor (Intel Pentium M 1.5 GHz, 512 MB RAM), with a 12 V power supply and running Windows XP.
(ii) A GPS receiver (1): a Trimble AG132 DGPS with EGNOS/RTCM corrections. It is connected to the computer via a serial RS232 link at 38400 bauds.
(iii) A gyrometer 2: a fiber-optic KVH ECore2000 connected via a serial RS232 link at 9600 bauds.
(iv) Two odometers whose values are obtained thanks to the ABS sensors of the rear wheels of the car. The sinusoidal signal generated by the rotation of each wheel is applied to a national instruments 16-bit counter. This provides measurements of the distances covered by the two wheels (1 top corresponds to 2 centimeters).

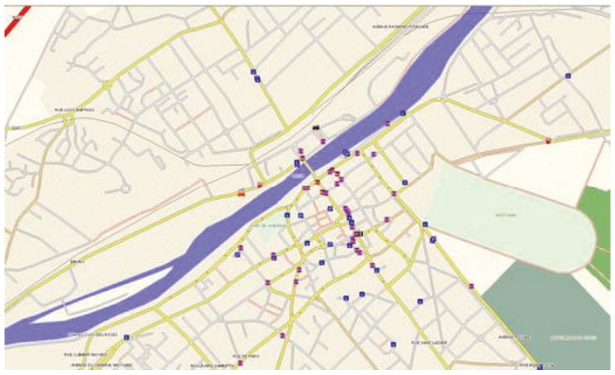


Figure 8: Map rendering using BeNomad SDK.

The interfacing programs were developed in C++, and all the data are timestamped and stored in a binary format for rapid prototyping purposes. Our goal is to have timestamps as close as possible to the sensor measurements in order to fit the model. To obtain the data via the serial link, we used an asynchronous driver which enables the data to be timestamped upon their arrival at the port and eliminates the decoding time. For the GPS, two kinds of data are received:

(a) on the Rx pin, NMEA0183 frames which contain the navigation data;
(b) an xPPS signal on the ring indicator pin.

The GIS used by the map-matching module is based on a software development kit (SDK) provided by BeNomad [15], see Figure 8. This SDK is completely object-oriented and cross-platform (Windows, Linux). It is also available for embedded targets such as PDAs or smart phones running Windows CE and Windows Mobile. The maps are size-optimized and provided in the SVS (scalable vector system) file format. For our prototype, we used a NavteQ geographical database converted to SVS format. The SVS format is very compact: the file size for the whole of the town of Compiegne is only 68 KB, and that for the entire OISE department is only 3 MB.

4.2. GPS latency experiments

Real-time experiments were carried out on a specialized test track in Versailles using one of our experimental cars (Figure 9).

In order to compute estimation errors, a L1/L2 Thales Navigation GPS receiver was used in a post-processed kinematic mode working with a local base (a Trimble MSi 7400). This system was able to give reference positions with a 1 Hz sampling rate. Since the constellation of the satellites was sufficiently good throughout the

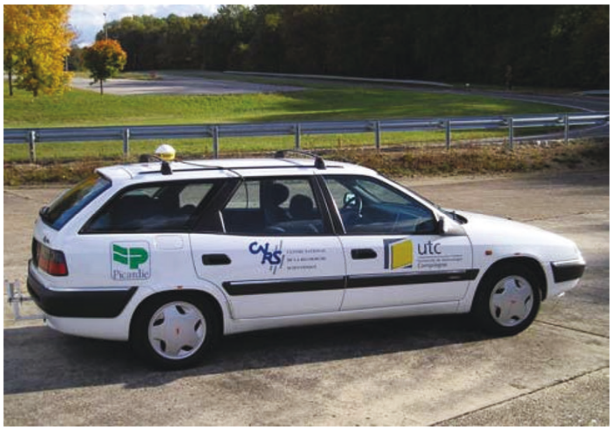


FIGURE 9: Used experimental car and experimental test track.

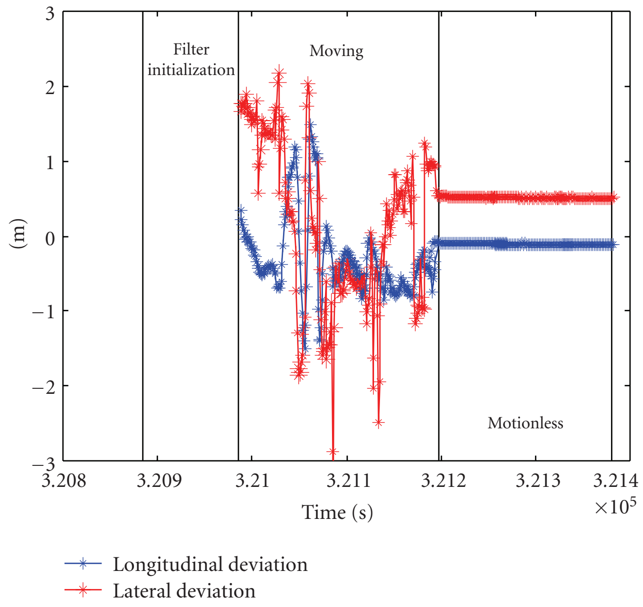


FIGURE 10: Filter output errors (maximum speed 90 Km/h).

trials, all the kinematic ambiguities were fixed. Accuracy was therefore guaranteed to within a few centimeters. The synchronization between this reference and the outputs of the localizer was achieved using the GPS timestamps.

Before implementing the filter, we measured the latency of the Ag132 receiver using an oscilloscope. We observed a 180 millisecond maximum latency corresponding to the worst case, that is, with 8 satellites used in the computation. Therefore, in order to implement the GPS latency management of Section 2, the receiver was tuned to 5 Hz, which guarantees that the GPS data are available before the next xPPS signal.

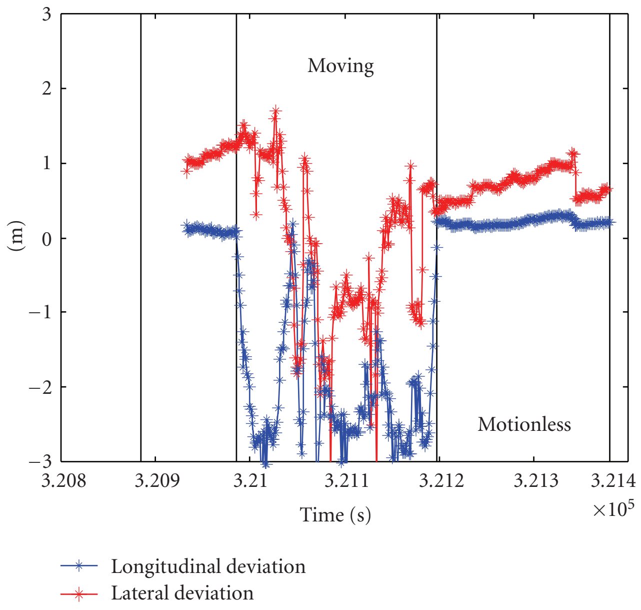


FIGURE 11: GPS output errors on the same trial.

Figures 10 and 11 are courtesy of Kais et al. [16], featuring the output of our localizer and the rough GPS files. Theses plots correspond to the longitudinal and lateral errors in a Frenet's frame.

Itcan be seen in Figure 11 that the longitudinal GPS error can be as high as 3 meters because of its latency, while this error is significantly reduced by the filter thanks to the predictor and the synchronization strategies used. Moreover, it is important to notice the efficient filtering of the multisensor localizer while the vehicle is motionless.

4.3. *Map cache experiments*

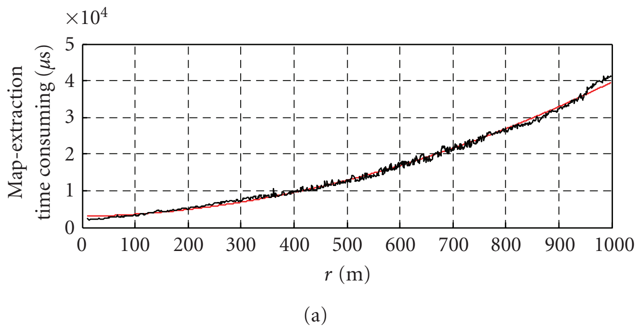
As a source digital map, we used a NavteQ database. In this map, coordinates are integers in centimeters, in the French Lambert 93 coordinate system.

We took a position in a downtown area (city hall of Compiègne). This position is the center of the GIS request. The semilength r of the square search has been instantiated from 10 m to 1 Km with a 1 m step.

Figure 12 shows the map-extraction time-consuming process and the road-selection time-consuming process versus the client request radius. For each plot, a second-order approximation was made, plotted in bold blue for each subplot. As a matter of fact, theses processes are dependent on the size of the area of interest, proportional to r^2.

We note that the two processes are very efficient and also that the road selection time is relatively small when compared to the map extraction time. Even if r is large, the road selection duration never exceeds 0.5 millisecond, which confirms that it is negligible compared to the sampling period of tracking process (Te = 200 milliseconds).

Figure 13 plots the size of the extracted map cache regarding the request radius. Once again, an expansion of the



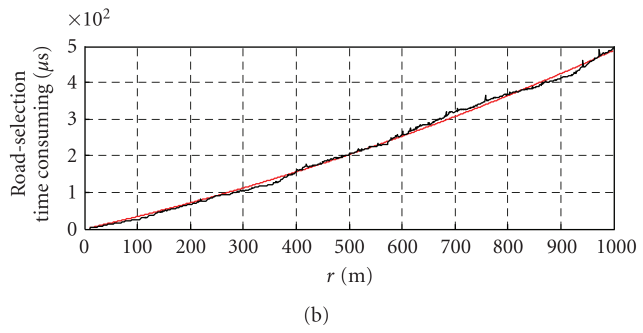


FIGURE 12: Map-extraction and road-selection time consuming for an urban area.

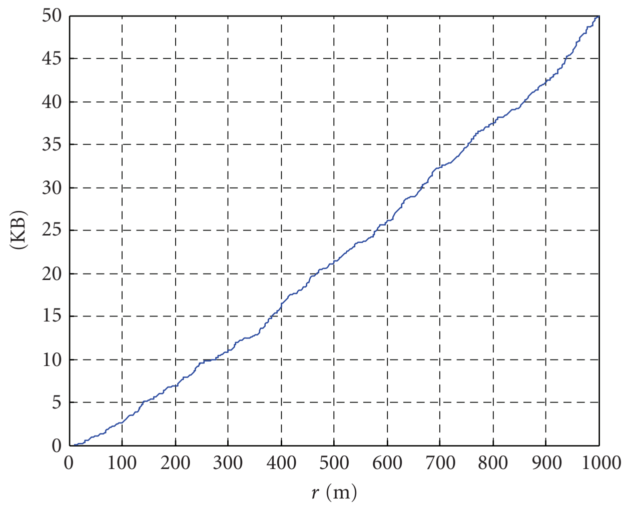


FIGURE 13: Map cache size regarding the size r of the request.

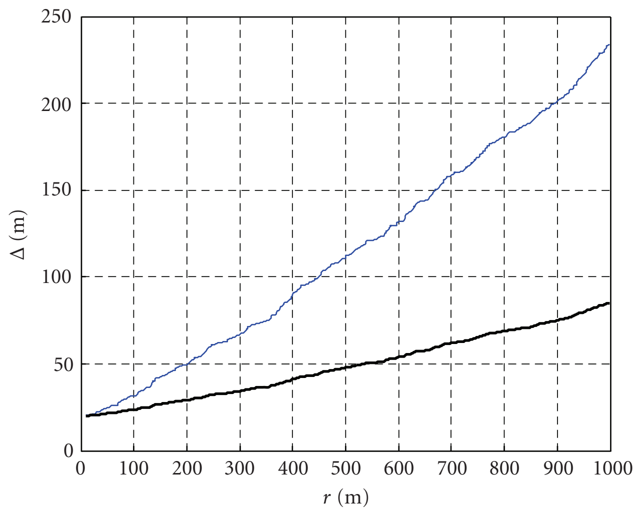


FIGURE 14: Δ versus r in the case of a remote server: in bold black 3G com link and in thin blue GPRS link.

search area implies a four-fold increase in the size of the map cache. Moreover, we can see that for a maximum radius of 1000 m, the cache size does not exceed 50 KB, which is not a constraint for the real-time target under consideration.

These results indicate that neither the size nor the computation times (extraction and road selection) are hard constraints for the cache management. r can be set very high.

Let us consider now the computation of Δ.

Suppose that the maximum map inaccuracy is $E_M = 15$ m and the localization inaccuracy is $E_L = 5$ m (as indicated by the experiments described in Section 4.2).

First of all, let us assume that the GIS server is embedded in the target and provides its data by using a shared memory or the middleware SCOOT-R [17] for instance. We choose $r = 1000$ m.

On our target, the local transmission time between the server and the client was measured $T_{\text{transmission}}(1000\,\text{m}) = 745$ microseconds, and $T_{\text{map extraction}} = 41$ milliseconds (from Figure 12). Suppose that the vehicle speed equals 30 m/s $= 108$ km/h. Thanks to (11), we can compute that the request has to be sent at a distance $\Delta = 15+5+(745\cdot 10^{-6}+41\cdot 10^{-3})\cdot 30 = 21 \cdot 25$ m only before leaving the cache area. Therefore, the map cache management is mainly constrained by the localization and map errors.

Let us suppose now that the GIS server is remote and let us consider 2 kinds of network: 3G (3rd generation) and GPRS (general packet radio service) connection [18]. For realistic considerations, we took the half of the maximum bandwidth: for GPRS, we took the half of 115.2 Kb/s, and for 3G, the half of 384 Kb/s.

Figure 14 shows Δ versus r that gives the distance to the cache limit that it is necessary to respect in order to obtain reliable road selection. This plot was estimated numerically by using (11) where $T_{\text{extraction}}(r)$ was the second-order approximate of the time extraction process in function of r, and $T_{\text{transmission}}(r)$ was the transmission delay of the extracted map within the network. This delay depends on the size of the map and on the bandwidth of the network.

In summary, the methodology for cache memory management is as follows.

(i) Estimate the road selection duration versus r.

(ii) Choose a value of r such that the duration is compatible with

  (a) the sampling period of the localizer,
  (b) RAM size of the target,
  (c) the map error,
  (d) the localizer error.

(iii) Compute Δ corresponding to the implantation of the GIS server.

In our case, the road selection routine not being very time consuming, and the size of the cache memory not being a limitation, we suggest using a large cache area (r in the order of 1 km) in order to obtain a reliable behavior.

5. CONCLUSION

This paper has considered the real-time implementation of a localization system that uses DR sensors, GPS, and GIS data following a loosely coupled paradigm. Such a system is a key component for intelligent vehicles since it can constitute the basis for precise localization or the development of advanced driving assistance systems. Its main outputs are the pose of the vehicle in the projected frame of the map and the ID of the most likely road in case of successful map matching (otherwise an off-road situation is detected).

An efficient implementation relies on pose tracking using Kalman filtering, after a good initialization. Because of the GPS latency, we have proposed a strategy that involves delaying the GPS correction stage until last, the filter outputting DR predictions with small latency.

The multisensor data fusion problem has been modeled as a space-time problem with two pose trackers. Because of the GPS receiver, a time-triggered approach is necessary. The trigger is the xPPS signal. For the road-selection problem, the spatial nature is essential. Therefore, a mechanism for triggering the filter according to the vehicle displacement has been developed. At each fusion step, a coherence test is applied. If the GPS is incoherent for several samples, the localizer is reinitialized. This guarantees that wrong map matches can be corrected.

A key aspect of this system is the map representation for an efficient road selection. We have seen that our proposed representation is very pertinent to this consideration. Another issue is the map cache memory management that is performed in parallel by considering spatial, rather than temporal, conditions. This is essential for an embedded system, since such a strategy gives rise to computations only when necessary. Two aspects have to be dealt with: the center and the semilength of the square search area. They are linked together and we have identified the key points. The first step consists in evaluating the duration of the road selection with respect to the size of the area. Then a compromise has to be found between map-matching integrity and workload (in case the GIS server is embedded) or workload and communication (if the GIS server is remote).

This prototype is currently used for the management of visual landmarks memory. Thanks to the use of the filter, it is possible to send requests to the positioning server at a video rate.

One perspective of this work is to use an electronic horizon to manage the cache memory. An electronic horizon is a graph of the accessible roads from the current pose.

ACKNOWLEDGMENT

This research has been carried out within the framework of the European FP6 Integrated Project CVIS, *Cooperative Vehicle Infrastructure Systems*, started in February 2006 for 4 years.

REFERENCES

[1] E. Royer, J. Bom, M. Dhome, B. Thuilot, M. Lhuillier, and F. Marmoiton, "Outdoor autonomous navigation using monocular vision," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '05)*, pp. 1253–1258, Edmonton, Canada, August 2005.

[2] http://www.navteq.com/.

[3] http://www.teleatlas.com/.

[4] M. Jabbour, Ph. Bonnifait, and V. Cherfaoui, "Enhanced local maps in a GIS for a precise localisation in urban areas," in *Proceedings of the 9th IEEE International Conference on Intelligent Transportation Systems (ITSC '06)*, pp. 468–473, Toronto, Ontario, Canada, September 2006.

[5] H. Kopetz, "Time-triggered real-time computing," in *Proceedings of the 15th World Congress of the International Federation of Automatic Control (IFAC '02)*, IFAC Press, Barcelona, Spain, July 2002.

[6] S. J. Julier and J. K. Uhlmann, "New extension of the Kalman filter to nonlinear systems," in *Proceedings of the 11th International Symposium on Aerospace/Defence Sensing, Simulation and Controls*, Orlando, Fla, USA, April 1997.

[7] R. G. Brown and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering: With MATLAB Exercises and Solutions*, John Wiley & Sons, New York, NY, USA, 1996.

[8] C. Tessier, C. Cariou, C. Debain, F. Chausse, R. Chapuis, and C. Rousset, "A real-time, multi-sensor architecture for fusion of delayed observations: application to vehicle localization," in *Proceedings of the 9th IEEE International Conference on Intelligent Transportation Systems (ITSC '06)*, pp. 1316–1321, Toronto, Ontario, Canada, September 2006.

[9] F. Gustafsson, F. Gunnarsson, N. Bergman, et al., "Particle filters for positioning, navigation, and tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 425–437, 2002.

[10] S. Niklas, "Real time implementation of map aided positioning using a Bayesain approach," M.S. thesis, Linköping University, Linköping, Sweden, December 2002.

[11] M. E. El Najjar and Ph. Bonnifait, "A road-matching method for precise vehicle localization using belief theory and Kalman filtering," *Autonomous Robots*, vol. 19, no. 2, pp. 173–191, 2005.

[12] T. Walter and P. Enge, "Weighted RAIM for precision approach," in *Proceedings of the 8th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GPS '95)*, vol. 2, pp. 1995–2004, Palm Springs, Calif, USA, September 1995.

[13] S. Wijesoma, K. W. Lee, and J. I. Guzman, "On the observability of path constrained vehicle localisation," in *Proceedings*

of the 9th IEEE International Conference on Intelligent Transportation Systems (ITSC '06), pp. 1513–1518, Toronto, Ontario, Canada, September 2006.

[14] A. Kelly, "Linearized error propagation in odometry," *The International Journal of Robotics Research*, vol. 23, no. 2, pp. 179–218, 2004.

[15] http://www.benomad.com/.

[16] M. Kais, Ph. Bonnifait, D. Bétaille, and F. Peyret, "Development of loosely-coupled FOG/DGPS and FOG/RTK systems for ADAS and a methodology to assess their real-time performances," in *Proceedings of IEEE Intelligent Vehicles Symposium (IV '05)*, pp. 358–363, Las Vegas, Nev, USA, June 2005.

[17] K. Chaaban, M. Shawky, and P. Crubillé, "A distributed framework for real-time in-vehicle applications," in *Prceedings of the 8th IEEE International Conference on Intelligent Transportation Systems (ITSC '05)*, pp. 925–929, Vienna, Austria, September 2005.

[18] P. R. Muro-Medrano, D. Infante, J. Guillo, J. Zarazaga, and J. A. Banares, "A CORBA infrastructure to provide distributed GPS data in real time to GIS applications," *Computers, Environment and Urban Systems*, vol. 23, no. 4, pp. 271–285, 1999.