Adaptive Fuzzy-Network-Based C-Measure Map-Matching Algorithm for Car Navigation System

Sinn Kim and Jong-Hwan Kim, *Member, IEEE*

***Abstract*—Most car navigation systems estimate the car position from dead reckoning and the Global Positioning System (GPS). However, because of the unknown GPS noise, the estimated position has an undesirable error. To solve this problem, a map-matching method is introduced, which uses a digital road map to correct the position error. In this paper, a novel adaptive-fuzzy-network-based C-measure algorithm is proposed, which can find the exact road on which a car moves. The C-measure algorithm is easy to calculate, and calculation time does not increase exponentially with the increase of junctions. For the experiments, a car navigation system is implemented with a small number of sensors. The real road experiments demonstrate the effectiveness and applicability of the proposed algorithm and the developed car navigation system.**

***Index Terms*—Fuzzy logic, learning systems, road vehicle location monitoring, terrain mapping.**

I. Introduction

FOR THE LAST decade, much work has been concentrated on car navigation systems. The first one developed merely gave the car position in text form. However, recently developed systems give the shortest route as well as the car position on a graphic screen [1]–[3]. These systems are expected to communicate with other traffic management systems to get traffic information in the near future [4].

The main role of the car navigation system is to estimate the car position. Dead reckoning has been widely used for estimating the mobile vehicle's position. It estimates the position by integrating the displacements at every sampling time. Since this method is based on the integration of the sensory outputs, it is subject to major accumulations of errors caused by wheel slippage, surface roughness, sensor errors, etc. To remedy this problem in the field of mobile robotics, absolute positioning methods using beacons or landmarks have been developed [5]–[7]. Ultrasound sensors [8], [9] or laser range finders [10]–[12] were used in the absolute positioning. Image information is also an effective tool to calculate the absolute position [13], [14]. These methods estimate the absolute position, which is fused with the estimated position of dead reckoning. The Kalman filter is the most widely used in the fusion.

The same idea can be applied to estimating the car position. However, it is hard to use the above absolute positioning methods directly, because a car is used outdoors, whereas a conventional mobile robot is used indoors. Image information can be used for guidance to follow a road, but it cannot give the position of a car by itself [15]–[17]. In this regard, the Global Positioning System (GPS) is the most attractive one for the car navigation system. This is because the position can be calculated on the globe if more than four satellites are detected [18]. However the GPS signal is corrupted by selective availability noise (S/A noise) for civil users. The characteristic of S/A noise is unknown. This is why it is difficult to combine the GPS and the dead reckoning directly. In combining the GPS and dead reckoning, the Kalman filter can be used. The estimated position by the Kalman filter is proved to be optimal if the system is linear and the noise is white Gaussian [19]. It should be noted that as the noise of the GPS is not white Gaussian, the estimated position from the Kalman filter is not optimal. It leads to position error. To reduce the error, many integration methods have been suggested [18], [20], [21]. The Differential GPS (DGPS) is also a good algorithm to reduce the error [22].

Map matching is another useful approach to correct the error, as reliable road maps are readily available [23], [24]. Map matching is a scheme for finding the car position on the road. As the car is always running on the road, the positioning error of the filter can be eliminated if we know the road on which the car is. The geometric approach, two types of which are arc-to-point mapping and arc-to-arc mapping, is the most popular map-matching algorithm [25]. In this algorithm, it is difficult to calculate the nearest road and it requires case-by-case calculations, which leads to a time-consuming tuning process. Another map-matching algorithm is to use conditional probability. This algorithm is more robust than the geometric method and it can recover from false positioning quickly. However, it requires more computation time and more memory to store the car trajectory.

In this paper, a novel map-matching algorithm is proposed which requires little computation time and can find the exact road on which a car moves. To reduce the complexity, the map-matching problem is formulated as identifying the road not the point on the road. The C-measure is defined to represent the certainty of the car's existence on the corresponding road. The road is identified by comparing the C-measures. For a more robust and exact map matching, an adaptive fuzzy network (AFN)

Manuscript received November 29, 1998; revised May 18, 2000. Abstract published on the Internet September 6, 2000. An earlier version of this paper was presented at the 1999 IEEE 8th International Fuzzy Systems Conference, Seoul, Korea, August 22–25.

The authors are with the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology, Taejon 305-701, Korea (e-mail: johkim@vivaldi.kaist.ac.kr).

Publisher Item Identifier S 0278-0046(01)10008-0.

is employed to obtain the C-measure. The convergent learning rule is provided for the AFN. As it takes little computation time and the implementation is easy, the proposed algorithm is applicable to car navigation systems. A car navigation system is implemented with a small number of sensors for experiments. The real road experiments demonstrate the effectiveness and applicability of the proposed algorithm and the developed car navigation system.

In Section II, the hardware implementation of the car navigation system is described. In Section III, the C-measure and the map-matching structure are presented. Section IV is devoted to the presentation of the novel AFN-based C-measure algorithm. The convergent learning rule and its proof are also provided. Section V shows some experimental results with the developed car navigation system. Concluding remarks follow in Section VI.

II. Implementation of Car Navigation System

A car navigation system consists of several subsystems, such as a positioning system, a route guidance system, a communication system, and a user interface system. The main role of a car navigation system is to find the car position as precisely as possible. In this paper, a car navigation system merely means the positioning system which is one of subsystems. The car navigation system is implemented with a small number of sensors (a vehicle speed sensor, a gyroscope, and a GPS receiver). An extended Kalman filter is used as a navigation filter and a recursive least-squares algorithm is used in the GPS receiver. The kinematic model of a car is simplified to reduce computational burden. This simplification may produce an error. However, the error is small and will be reduced by the map-matching algorithm. The kinematic model of the car and the implementation of the navigation filter are presented in Appendixes I and II, respectively.

To read the sensor data as well as to calculate the car position simultaneously at every sampling time, the developed navigation system consists of two parts, a sensor handling part and a filter part. A 32-bit microprocessor TMS320C32 is used for the filter part and the 8-bit processor 89C52 is used for the sensor handling part. A dual-input/output buffer is used for communication between the two. By adopting two processors, it is easy to append more sensors and the computational burden is distributed on the two processors effectively. All the programs for the sensor part and the filter part are coded in C-language. As mentioned above, the developed car navigation system is a subsystem that has the positioning ability only. The estimated position from the developed car navigation system is sent to the notebook computer through RS232C serial communication, and the map matching is performed in the notebook computer. The map matched position is displayed on the notebook monitor. The developed car navigation system is shown in Fig. 1.

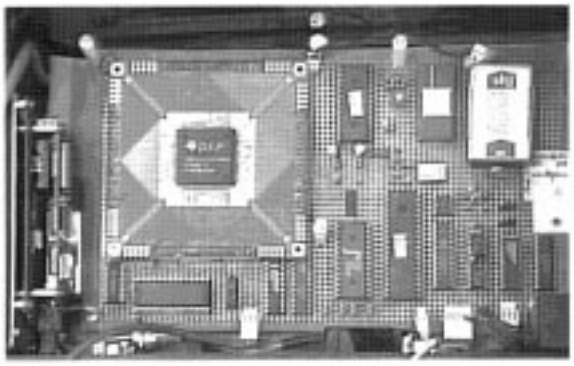


Fig. 1. Developed car navigation system.

III. C-Measure Map-Matching Algorithm

In a digital map, a road is defined as a single strip starting from one node and ending at the other. A node is the point where a road starts or ends. A junction, where more than two roads meet, is also a node. A road has only two nodes at both ends. Map matching is a scheme to find the car position on the road with the filtered car position of a navigation filter. This problem is very complicated and difficult to solve, if we try to solve it by handling all points on the road directly. In this paper, we redefine the map-matching problem to reduce its complexity. The map-matching problem is formulated as identifying the road not the point on the road. After identifying the road, we can find the car position on the road by orthogonal projection [28]. The orthogonal projection gives the point that is nearest to the filtered car position on the road.

If there is a measure which represents the possibility or certainty of the existence of a car on a specific road, it will be very easy to identify the road on which the car is moving by comparing the measures of all the roads. The C-measure of the ith road represents the certainty of existence of a car on that road. If we define the C-measure of the ith road at the kth iteration step as $C_i(k)$, a simple decision rule can be derived as follows:

IF $C_i(k) \geq C_T$,
THEN *there exists a car on the ith road*,
ELSE *there does not exist a car on the ith road*,

where C_T is a threshold value.

The above idea is employed in the proposed map-matching algorithm. If there is more than one road whose C-measure is greater than C_T, then the road which has the greatest C-measure is selected. The next section describes how to design such C-measure.

A. C-Measure Design

There are two important issues to be considered when one decides the road where the car is. The C-measure should be associated with these issues. The first one is that the distance between the filtered car position obtained from the navigation filter and its projected position on the road should be small, and the second is that the shape of the road should be similar to the trajectory of the filtered car position.

Let $p_c = (x_c, y_c)$ be the car position and $p_r = (x_r, y_r)$ be the project position on the road. The velocity $\vec{v_c}$ and the projected

velocity $\vec{v_r}$ of the car are calculated by the displacements of p_c and p_r. These two properties can be implemented as

$$D(p_c, p_r) = \frac{1}{1 + \dfrac{||p_c - p_r||^2}{\sigma^2}} \tag{1}$$

$$V(\vec{v_c}, \vec{v_r}) = \frac{(\vec{v_c} \cdot \vec{v_r})^2}{||\vec{v_c}||^2 ||\vec{v_r}||^2} = \cos^2 \theta_\delta \tag{2}$$

where

σ standard deviation of the navigation filter error;
θ_δ angle between $\vec{v_c}$ and $\vec{v_r}$;
$\vec{v_c} \cdot \vec{v_r}$ inner product of two vectors.

Because the shape of the road is the same as the trajectory of the projected position, the similarity between the road and the trajectory of the filtered car position can be obtained from $\vec{v_c}$ and $\vec{v_r}$. These two functions give the values in [0, 1]. For notational simplicity, $D(k)$ and $V(k)$ will be used to represent $D(p_c, p_r)$ and $V(\vec{v_c}, \vec{v_r})$ at the kth step, respectively.

Initially, the C-measure is defined as

$$C(k) = \alpha D(k) + \beta V(k) \tag{3}$$

where $\alpha > 0$ and $\beta > 0$. However, we may not obtain satisfactory results with the above definition, because it will be sensitive to noise.

In addition to (1) and (2), there is another important property which should be incorporated. When a car is moving on a road, it cannot jump or fly to another road, so it can never change the road unless it goes through a junction. Abrupt change of the road cannot ever happen. This property is called the continuous property. The C-measure should be designed to represent this property as well as those two properties mentioned above. The continuous property can be implemented by defining the C-measure in a recursive form. Finally, the C-measure is defined as follows:

$$C(k+1) = \alpha D(k) + \beta V(k) + \gamma C(k) \tag{4}$$

where $\gamma > 0$.

B. Characteristics of C-Measure and Parameter Selection

Three parameters (α, β, and γ) are needed in the C-measure, and another parameter C_T is required in selection of the road. To be effective, the C-measure should meet the following criteria.

1) $C(k)$ should maintain a certain value and should be less affected by the noise of the navigation filter.
2) $C(k)$ of the true road should be distinguished distinctly from those of other roads.

The true road is defined as the road where the car is.

To satisfy the first requirement, γ should be selected between 0–1 so that the C-measure has a finite steady-state value. The steady-state value C_{ss} is obtained as

$$C_{ss} = \frac{1}{1-\gamma}(\alpha D + \beta V). \tag{5}$$

We will examine the characteristics of the C-measure with C_{ss} instead of $C(k)$ itself, because $C(k)$ will eventually converge to C_{ss} as k approaches to infinity. If $D(k)$ and $V(k)$ are constants, C_{ss} will be constant and it will satisfy the first requirement. However, since C_{ss} is affected by noise of the navigation filter, it is a random variable, not a constant. The statistical properties of C_{ss} should be investigated to test whether the requirements are satisfied or not.

The two most important statistical characteristics are the mean and the variance. As C_{ss} is a linear combination of D and V, if we assume that D and V are uncorrelated, the mean and the variance of C_{ss} are given by

$$E[C_{ss}] = \frac{1}{1-\gamma}(\alpha E[D] + \beta E[V]) \tag{6}$$

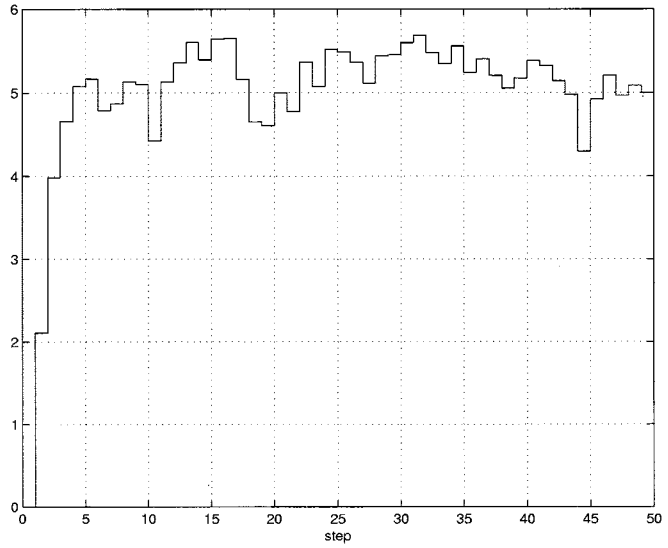
$$\mathrm{var}[C_{ss}] = \frac{1}{(1-\gamma)^2}(\alpha^2 \mathrm{var}[D] + \beta^2 \mathrm{var}[V]) \tag{7}$$

where $E[D]$, $\mathrm{var}[D]$ are the expectation and the variance of $D(k)$, respectively, and $E[V]$, $\mathrm{var}[V]$ are the expectation and the variance of $V(k)$, respectively. If we assume that the error of the filtered position is Gaussian, $E[D]$, $\mathrm{var}[D]$, $E[V]$, and $\mathrm{var}[V]$ can be calculated [29].

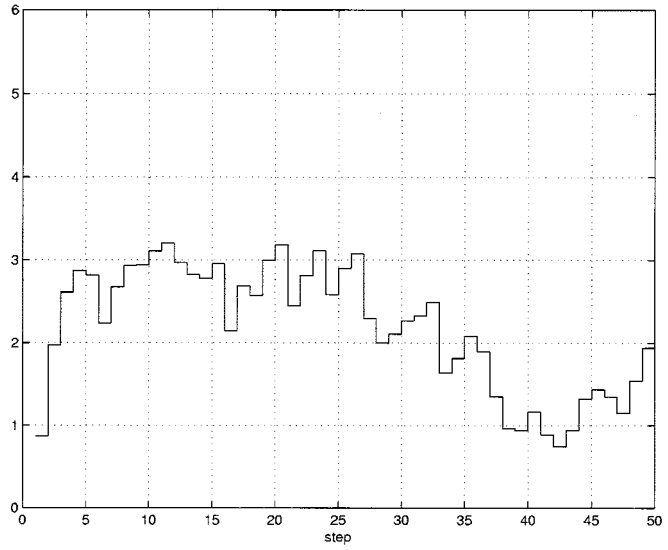
The requirements of the C-measure can be restated with the mean and the variance of C_{ss}. $\mathrm{var}[C_{ss}]$ should be small and $E[C_{ss}]$ of the true road must be distinctly different from the others. As mentioned above, γ should be selected between 0–1. If γ is close to 0, the C-measure will be sensitive to noise. If γ is close to 1, the C-measure will show a slow response. As a result, $C(k)$ of the true road may become smaller than those of others. γ should be selected to avoid the above phenomena. α and β should be selected to satisfy the remainder of the requirements. To ensure the small variance, α and β should be small because $\mathrm{var}[C_{ss}]$ is directly proportional to α^2 and β^2. However, if they are too small, $C(k)$ of the true road will be hardly distinguishable. Also, α should satisfy $\alpha > \beta$ such that much weight would be imposed on $D(k)$, as $V(k)$ would not make a critical contribution to making $C(k)$ of the true road distinctive because the shapes of most roads are similar. Considering the above discussios, $\alpha = 2$, $\beta = 1$, and $\gamma = 0.5$ are selected.

Now, we have to decide the threshold value C_T. It can be decided by examining the probability distribution of C_{ss} or through computer simulations. If the probability distribution is known, we can decide C_T with a desired probabilistic accuracy. However, it will be conservative as several assumptions are required to calculate the probability distribution function of C_{ss}. It is more practical and efficient to use computer simulations. By evaluating the performance of different threshold values in simulations, an appropriate threshold value can be decided.

Fig. 2(a) shows the C-measure of the true road and Fig. 2(b) shows the C-measure of another road which runs parallel to the true road and is 100 m away. From this simulation result, C_T will be determined. The position error of the filter was assumed to be a zero-mean Gaussian noise whose standard deviation was 50 m. In this case, the predicted values of $E[C]$ and $\mathrm{var}[C]$ are about 5.2 and 0.45. These predicted values are almost the same as those of the simulation. If we set $C_T = 3.0$, $C(k)$ of the true road is greater than C_T as shown in Fig. 2(a). Thus, the positioning error of the filter does not make any mismatches. If we raise C_T to $C_T = 5.0$, it is observed that the true road is missing rather frequently because $C(k)$ of the true road often becomes smaller than C_T as shown in Fig. 2(a). If we select a lower value for C_T, the chance to select the wrong road becomes



(a)



(b)

Fig. 2. Simulation results. (a) $C(k)$ of the true road. (b) $C(k)$ of another road.

higher as shown in Fig. 2(b). From this simulation result, we selected $C_T = 3.0$. In this case, the true road was identified in 2 s. From this result, the C-measure is found to be an effective measure for the map matching.

The performance depends on the parameters a lot. In particular, α and β perform an important role to make the C-measure of the true road distinctive. It is a difficult and cumbersome job to find the optimal values. To solve this problem, an AFN is employed to obtain the C-measure in Section IV.

C. Map-Matching Algorithm

As discussed in the previous section, the map-matching problem is redefined as identifying a road where the car is, and the corresponding road can be identified by comparing the C-measure. In the proposed map-matching algorithm, two different modes (the position-fixing mode and the tracking mode) are provided. The position-fixing mode is to find the true road and the tracking mode is to track the road found in the position fixing mode. In the position-fixing mode, $C(k)$s of all the roads around the car are calculated to identify the true road. If the maximum $C(k)$ is smaller than C_T, there are no possible roads and the position fixing mode is kept active. If $C(k)$ of the ith road is the maximum value and $C_i(k) \geq C_T$, then it is declared that there exists a car on the ith road. The ith road is called the tracked road. If the tracked road is identified, the mode is changed to the tracking mode. The car position on the tracked road is obtained by the orthogonal projection as discussed in Section III-A. Fig. 3 shows the proposed map-matching structure.

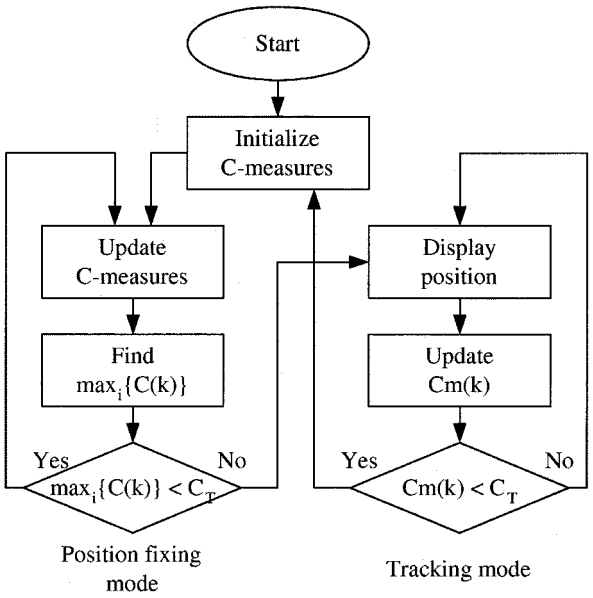
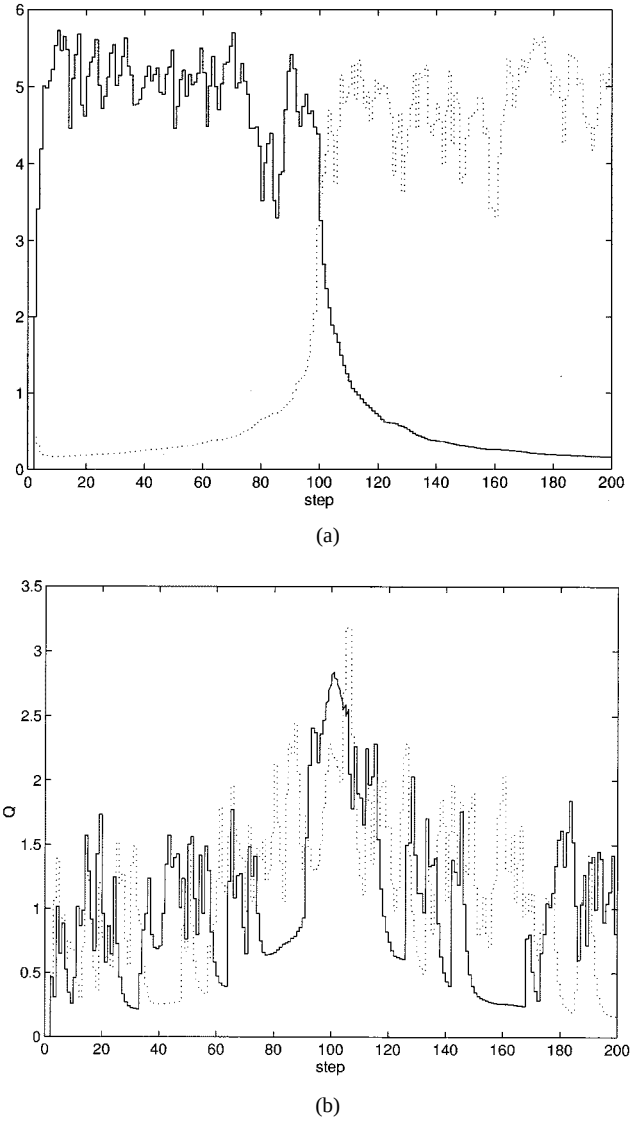


Fig. 3. Map-matching structure.

In the tracking mode, it is declared that the car exists on the tracked road unless $C(k) < C_T$. If $C(k)$ of the tracked road is smaller than C_T, it is declared that the car does not exist on the tracked road any more, and then $C(k)$s of all the roads are initialized to 0 and the mode is switched to the position-fixing mode. In the tracking mode, only one C-measure of the tracked road is calculated, and it takes very little computation time. When the car reaches a junction, it will leave the tracked road and will enter another road that is connected to the tracked road via the junction. To decide whether a car has entered a junction or not, another threshold value, C_J, is used. If $C(k)$ of the tracked road is smaller than C_J and the distance between the junction and the filtered car position is smaller than L, it is declared that a car is at the junction. To prevent loss of the tracked road while crossing a junction, the C-measures of the linked roads via the junction are updated. The C-measures of all the linked roads are initialized to $C(k)$ of the tracked road. After updating $C(k)$s of all the linked roads, the road that has the maximum $C(k)$ is selected and tracked continuously.

D. Simulation

For the simulation, it was assumed that there were four roads which were connected via a junction, and they were straight and perpendicular to each other. The car was moving on road 1 with



(a)

(b)

Fig. 4. When a car moves through a junction. (a) $C(k)$s of the selected roads. (b) $C(k)$s of the unselected roads.

a constant velocity. It would go to road 3 via the junction. The filtered position error was assumed to be a zero-mean Gaussian noise whose standard deviation was 50 m.

Fig. 4(a) shows $C(k)$s of the two selected roads before and after the junction. The solid line represents $C(k)$ of road 1 and the dotted line is that of road 3. Road 1 was not a tracked road initially, as it was in the position-fixing mode at first. After two steps, road 1 was declared as a tracked road and the mode was changed to the tracking mode. After identifying the tracked road, the tracking mode was pursued during the simulation, because $C(k)$ of the tracked road was always greater than C_T. In Fig. 4(a), $C(k)$ of road 1 is decreasing at the junction, while that of road 3 is increasing. Hence, road 3 was selected as a tracked road while a car moved through the junction. Fig. 4(b) shows $C(k)$s of the unselected roads. $C(k)$s of the unselected roads were always smaller than those of the selected roads. There were no mismatching points. From the simulation result, we can see that the proposed map-matching algorithm is effective.

IV. AFN-Based C-Measure Algorithm

Even though the proposed C-measure showed a good performance in computer simulations, there is the parameter selection problem in the proposed C-measure, as mentioned in Section III-B. It is difficult to find the optimal values of α and β which make $C(k)$ of the true road distinctive. Even if the optimal values are found, they will vary as the road profile changes. Hence, a parameter adaptation scheme is required to deal with the variations in road profiles. The point is that it is difficult to derive an adaptation law, as there is no mathematical model. Only the characteristics of the parameters are known. In this case, fuzzy logic is a useful tool, as it does not need an exact mathematical model. From this reason, an AFN is employed to obtain the optimal C-measure.

A. AFN-Based C-Measure

Let us rewrite the C-measure as

$$C(k+1) - \gamma C(k) = u(k) \tag{8}$$

$$u(k) = \alpha D(k) + \beta V(k). \tag{9}$$

This difference equation has the same structure as the first-order discrete— time system, whose control input is $u(k)$. In Section III, the linear summation $\alpha D(k) + \beta V(k)$ was used for $u(k)$.

It is possible to modify $u(k)$ to improve the performance of the C-measure as in the controller design. In some situations, it will take much time for $C(k)$ to be greater than C_T. If a product term is used for $u(k)$ as $V(k) \times D(k)$ instead of $\alpha D(k) + \beta V(k)$, the time can be reduced, or, if α and β are increased, a much faster response will be obtained. Let us assume that a nonlinear function, $f^*(D, V)$ is the desired function that can enhance the performance. Then, we can obtain the better C-measure by using $f^*(D, V)$ for $u(k)$ instead of $\alpha D(k) + \beta V(k)$. However, it is not easy to get a mathematical model of $f^*(D, V)$. Fortunately, it is known that an AFN is possible to model any continuous function within a desired accuracy [30]. We will approximate $f^*(D, V)$ with the AFN, and the network output will be used for $u(k)$ in (8). Thus, the AFN-based C-measure is defined as follows:

$$C(k+1) = \gamma C(k) + y(k) \tag{10}$$

where $y(k)$ is the output of the AFN.

B. Proposed AFN

There are many variations in the AFN according to their adaptation structures. The proposed network changes the consequence part by changing the centers of output membership functions. Neither input space partitions nor input membership functions are changed. Due to the simple structure, the parameter vector to be tuned is linearly related to the output and it is possible to train online. Also, the learning rate is very fast.

Let $\mu_D(x)$ be the membership function defined on a continuous universe of D, $\mu_V(x)$ be that of V, and $\mu_y(y)$ be that of the output. If all the membership functions are symmetric and the multivariate fuzzy input sets form a partition of unity ($\sum_i \mu_X(x) = 1$), then the defuzzified output of the network,

$y(D, V)$ can be written as (11), shown at the bottom of the page, where y_i^c is the center of the ith output membership function. y_i^c is the parameter to be tuned in the learning.

When the input and output fuzzy set is a singleton, (11) is simplified as follows:

$$y(D, V) = \sum_{i=1}^{n} \mu_{D^i}(D)\mu_{V^i}(V)y_i^c \tag{12}$$

$$= \zeta_1\theta_1 + \zeta_2\theta_2 + \cdots + \zeta_n\theta_n \tag{13}$$

$$= \zeta^T(D, V)\theta \tag{14}$$

where $\theta^T = [y_1^c\, y_2^c \cdots y_n^c]$ is a vector of parameters to be tuned and $\zeta^T(D, V) = [\zeta_1\zeta_2\cdots\zeta_n]$ is a vector of static terms and n is the number of rules. The learning algorithm can be derived from (14).

Let $\{D(k), V(k), y^*(k)\}$ be a training set where $D(k)$ and $V(k)$ are inputs of the network and $y^*(k)$ is the desired output of the network. The learning rule is obtained as

$$\Delta\theta(k-1) = \eta\epsilon_y(k-1)\zeta(D, V)$$
$$\theta(k) = \theta(k-1) + \Delta\theta(k-1) \tag{15}$$

where $\eta > 0$ and $\epsilon_y(k) = y^*(k) - y(k)$. After learning, $\theta(k)$ will converge to the optimal value which optimizes the given performance index $J = (1/\gamma)\epsilon_y^2(k)$. The proof can be found in [30].

C. Learning Rule for Map Matching

The learning rule given in (15) is associated with the output error $\epsilon_y(k)$. Therefore, to use (15), a reference model or a true output value is needed to calculate $\epsilon_y(k)$. However, in the C-measure map-matching algorithm, there is neither reference model nor true value. The above learning rule cannot be used.

For the proposed map-matching algorithm, we will derive a learning rule which does not require the output error. The learning purpose is to find $\theta(k)$ such that the following criteria are met.

1) $C(k)$ of the true road is the largest one among the possible roads so that the true road can be selected.
2) $C(k)$ of the true road is larger than C_T.

The learning is continued until both 1) and 2) are satisfied.

Let the mth road be the true road. If we assume that there exists no noise in the filtered position, the following theorem can be obtained.

Theorem 1: If the learning rule is given as follows:

$$\Delta\theta(k) = \begin{cases} 0, & \text{if } C_m(k) \geq C_T \\ \eta(\zeta_m - \zeta_i), & \text{if } C_m(k) < C_i(k) \\ \eta\zeta_m, & \text{if } C_m(k) < C_T \end{cases} \tag{16}$$

$$\theta(k+1) = \theta(k) + \Delta\theta(k) \tag{17}$$

where $\eta > 0$ and ζ_i is the vector of static terms of ith road, then $\theta(k)$ converges to the desired value in finite steps.

Proof: Suppose that there exists an ith road whose C-measure is greater than that of the true road (mth road) at $k = k_0$. Then, from (16), $\theta(k)$ will be adapted as

$$\Delta\theta(k) = \eta(\zeta_m - \zeta_i). \tag{18}$$

From (14) and (18), the difference of $y_i(k+1)$ and $y_m(k+1)$ is given as follows:

$$\begin{aligned} &y_i(k+1) - y_m(k+1) \\ &\quad = (y_i(k) + {\zeta_i}^T\Delta\theta(k)) - (y_m(k) + {\zeta_m}^T\Delta\theta(k)) \\ &\quad = (y_i(k) - y_m(k) + (\zeta_i - \zeta_m)^T\Delta\theta(k) \\ &\quad = (y_i(k) - y_m(k)) - \eta\|\zeta_i - \zeta_m\|^2 \end{aligned} \tag{19}$$

where $\|\zeta_i - \zeta_m\|^2 = (\zeta_{i1} - \zeta_{mi})^2 + (\zeta_{i2} - \zeta_{m2})^2 + \cdots + (\zeta_{in} - \zeta_{mn})^2$.

As we assume that there exists no noise, $\|\zeta_i - \zeta_m\|^2 > 0$, (19) can be written as

$$\begin{aligned} &(y_i(k+1) - y_m(k+1)) - ((y_i(k) - y_m(k)) \\ &\qquad = -\eta\|\zeta_i - \zeta_m\|^2 < 0. \end{aligned} \tag{20}$$

From (20), it can be proved that there exists a finite k_{f1} such that $y_m(k) > y_i(k)$ for $k > k_{f1}$. From (10), we can conclude that there exists k_{f2} such that $C_m(k) > C_i(k)$ for $k > k_{f2} > f_{f1}$, and then, $\theta(k)$ does not change any more if $C_m(k) \geq C_T$.

If $C_m(k)$ is the largest and $C_m(k) < C_T$, $\theta(k)$ is adapted as follows:

$$\Delta\theta(k) = \eta\zeta_m. \tag{21}$$

From (14) and (21), $y_m(k+1) - y_m(k)$ is given as

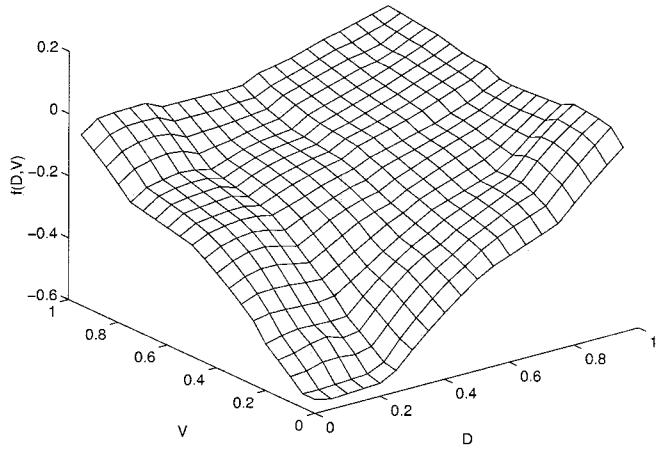
$$y_m(k+1) - y_m(k) = \zeta_m^T\Delta\theta = \eta\|\zeta_m\|^2. \tag{22}$$

As we assume that there exists no noise, $\|\zeta_m\|^2 > 0$, (22) can be written as

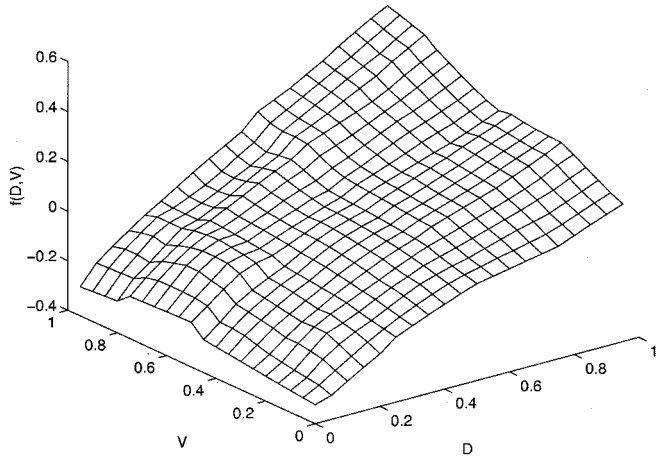
$$y_m(k+1) - y_m(k) > 0. \tag{23}$$

Equation (23) shows that $y_m(k)$ is a monotonic increasing sequence. There exists a finite k_{g1} such that $y_m(k) > 0$ for $k > k_{g1}$. Eventually, there exists a finite k_{g2} such that $C_m(k) \geq C_T$ for $k > k_{g2} > k_{g1}$, because $y_m(k) > 0$ for $k > k_{g1}$. And after $C_m(k) \geq C_T$, $\theta(k)$ will not change as $\Delta\theta(k) = 0$. From this,

$$y(D, V) = \frac{\displaystyle\int_{D\times V} \mu_D(D)\mu_V(V)\sum_i\sum_j \mu_{D^i}(D)\mu_{V^j}(V)y_i^c\, dV\, dD}{\displaystyle\int_{D\times V} \mu_D(D)\mu_V(V)\, dV\, dD} \tag{11}$$



(a)



(b)

Fig. 5. Output of the AFN-based C-measure. (a) Before learning. (b) After learning.

we can conclude that $\theta(k)$ converges to the desired value in finite steps. ■

While deriving the learning rule, we assume that the filtered position has no noise. However, actually, it is corrupted by noise. It can cause a small amount of fluctuation of the parameter around the desired value. To prevent this phenomena, time-varying learning rate $\eta(k)$ can be used as follows:

$$\eta(k) = \eta_0 \exp(-k) \tag{24}$$

where $\eta_0 > 0$.

D. Simulation

Fig. 5 shows the output of the AFN before and after learning. The universes of V and D were divided into five regions such as *very big, big, medium, small, and very small*, and 25 rules were used. The consequence part was generated by approximating the linear combination $\alpha V(k) + \beta D(k)$. The constant learning rate $\eta = 0.25$ was used for training. Fig. 5 shows that the distance of the car position and the road is more emphasized after learning.

The performance of the AFN-based C-measure algorithm was evaluated using several test patterns. The number of mismatches should be small for the exact map matching, and the mismatching interval should be short for fast recovery from the wrong position. The number of mismatches according to their mismatching intervals is shown in Table I before and after learning. After learning, the number of mismatches is substantially reduced and the mismatching intervals are very short.

TABLE I
NUMBER OF MISMATCHES

Mismatching intervals	Number of mismatches	
	Before learning	After learning
1 sec.	4	3
2 sec.	3	2
3 sec.	2	0
4 sec.	1	0
5 sec.	1	0
total	11	5

V. EXPERIMENTS

Many experiments were performed on urban streets, which have different features such as small roads, wide roads, crowded roads, tunnels, etc. The developed navigation system discussed in Section II was used. A GPS antenna and a gyroscope were mounted on the center of the car. To get the translational velocity of the car, the speed meter was used, which gives six pulses per rotation of the wheel. No additional sensors were used. The extended Kalman filter was used as a navigation filter. The filtered position of the car was sent to a 486DX notebook computer, where the map matching was performed. For the map matching, the trained AFN was used.

The area can be classified into three regions, type A, type B, and type C. The type-A region is an open area, where there are few roads and few buildings or structures that can degenerate the GPS performance. This kind region is a very good area for the map matching. Any map-matching algorithms give good results in this area. The type-B region has tunnels or high buildings which prevent the GPS receiver from receiving the satellite signals properly. The filtered position error increases in this region. Even though the position error increases, it is not too difficult for map matching to find the correct position, as there are few roads. The type-C region is a very complicated area. All the roads are very small and their shapes are complicated. The width of most roads is less than 5 m. Buildings beside roads are too close for the GPS receiver to receive the satellite signals. As a result, the filtered position error is very large. In this area, it is difficult to use the geometry-based map-matching method.

Fig. 6(a) shows the map-matching results of the proposed algorithm in the type-A region, where (343 093 m, 419 796 m) is the coordinate of the upper left corner and (344 426 m, 418 796 m) is the coordinate of the lower right corner. The circles are the filtered car positions obtained from the navigation system and the rectangles are the map-matched car positions. There were no mismatching points. Fig. 6(b) shows the experimental results in the type-B region. As the car moved into the tunnel,

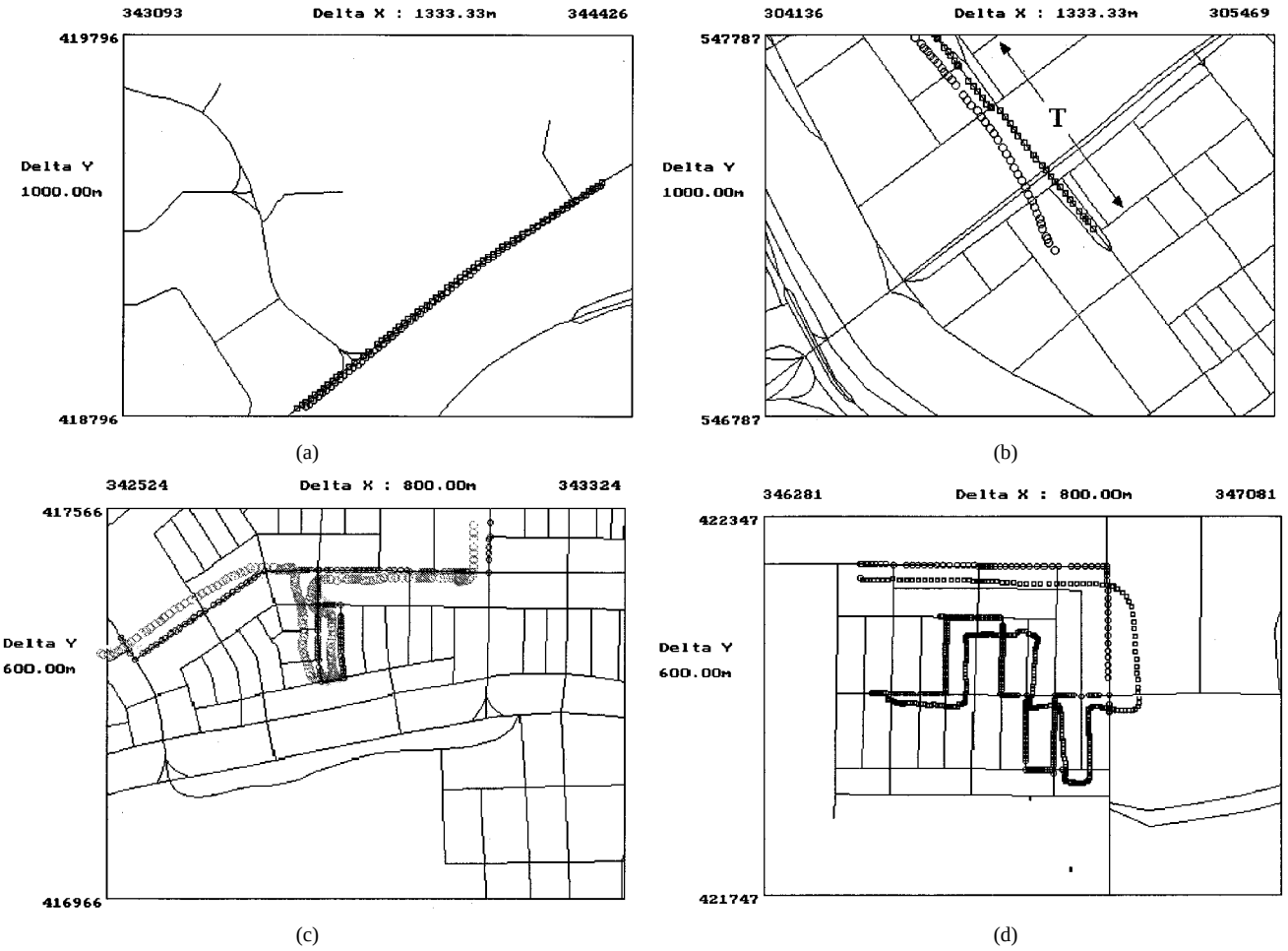


Fig. 6. Experimental results. (a) Map matching in the type-A region. (b) Map matching in the type-B region. (c) Map matching in the type-C region. (d) Map matching in the type-C region.

"T" in the figure, the filtered position error grew rapidly, but the map-matched position was on the exact road.

As the car moved into the type-C region, the filtered position error increased greatly. However, the proposed AFN-based C-measure algorithm could find the exact road as in Fig. 6(c). The filtered position error was more than 40 m. This error is almost the same as the distance between the two adjacent roads. Fig. 6(d) shows another result, where the distance between the two adjacent roads is less than 20 m and the shape of the roads is almost the same. In this case, the filtered position error was larger than the distance between the two adjacent roads. Therefore, it is liable to find the wrong road. However, the proposed algorithm could find the exact road. In the worst case, the map-matching error was less than 15 m.

From the experimental results, we can see that the exact position of the car can be obtained by the proposed map-matching algorithm, even though the filtered position error is larger than the distance between the two adjacent roads.

VI. Conclusions

In this paper, we have proposed a novel AFN-based C-measure map-matching algorithm and its learning rule. For experiments, a car navigation system was developed with a small number of sensors, such as a GPS receiver, a gyroscope, and a speed meter. The effectiveness of the proposed map-matching algorithm was verified with several experiments. In the experiments, the map-matched car position was very precise. Even though the filtered position error was larger than the distance between two adjacent roads, the exact road could be found by the proposed map-matching algorithm. Even in the worst case, the map-matching error was less than 15 m, when the filtered position error was more than 50 m.

The proposed map-matching algorithm can be applied at any environment because of its learning ability, and the proposed algorithm needs little computation time. Moreover, the computation time does not increase exponentially with the increase in the number of junctions, but other algorithms increase exponentially. This feature makes it suitable for a real-time car navigation system.

Appendix I
Kinematic Modeling

Three sensors are used in the developed car navigation system. They are a speed meter, a gyroscope, and the GPS receiver. The characteristics of sensors should be associated with the kinematic model of a car.

The car's kinematic model is simplified to reduce the computational load. We use the kinematic model of a two-wheeled

mobile robot [31]. If we assume that a car moves on a two-dimensional plane, the kinematic equation is given as

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ \theta(k+1) \end{bmatrix} = \begin{bmatrix} x(k)+v(k)\Delta t\cos\theta(k) \\ y(k)+v(k)\Delta t\sin\theta(k) \\ \theta(k)+\omega(k)\Delta t \end{bmatrix} + \begin{bmatrix} w_x(k) \\ w_y(k) \\ w_\theta(k) \end{bmatrix} \tag{25}$$

where $x(k)$, $y(k)$ and $\theta(k)$ are the position and the heading angle of a car, respectively, $v(k)$ and $\omega(k)$ are the translational and angular velocities of a car, respectively, and $[w_x(k)\, w_y(k)\, w_\theta(k)]^T$ is an error vector. Errors are caused by noisy velocity data, slippage, accelerating rate change during the sampling time, etc. These errors are bounded because the power of a car is limited. By the central limit theory, the net effect of these errors can be considered as Gaussian noise [29].

To obtain the translational velocity, a speed meter is used. It counts the number of rotations of a wheel. The radius of the wheel should be known to calculate the translational velocity. It is estimated by the Kalman filter. As it is constant, the state equation can be written as

$$r(k+1) = r(k) + w_r(k) \tag{26}$$

where $r(k)$ is the radius of a wheel and $w_r(k)$ is Gaussian noise. $w_r(k)$ is added to adopt the Kalman filtering algorithm. $v(k)$ is calculated as

$$v(k) = \frac{2\pi n(k) r(k)}{\Delta t} \tag{27}$$

where $n(k)$ is the number of rotations of a wheel.

To get the angular velocity of a car, a gyroscope is used. If its rotational center is vertical to the plane where the car moves, the angular velocity from the gyroscope is the same as the angular velocity of the car. No rotational matrix is needed. The output of the gyroscope has an offset bias. We should compensate for it. It is known that the offset bias of the gyroscope is modeled as follows [32]:

$$\begin{bmatrix} \omega_a(k+1) \\ \omega_b(k+1) \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} \omega_a(k) \\ \omega_b(k) \end{bmatrix} + \begin{bmatrix} w_{\omega a}(k) \\ w_{\omega b}(k) \end{bmatrix} \tag{28}$$

where

h_ij	a constant parameter;
$\omega_b(k)$	offset bias of the gyroscope;
$\omega_a(k)$	augmented state;
$[w_{\omega a}(k)\, w_{\omega b}(k)]^T$	a Gaussian noise vector.

The angular velocity is obtained as

$$\omega(k) = \omega_g(k) - \omega_b(k) \tag{29}$$

where $\omega_g(k)$ is the output of the gyroscope.

From the above discussion, the state-space representation including car kinematics and sensor models can be described as

$$\mathbf{X}(k+1) = A\mathbf{X}(k) + \mathbf{f}(\mathbf{X}(k), v(k), \omega(k)) + w(k) \tag{30}$$

where

$$\mathbf{X}(k) = \begin{bmatrix} x(k) \\ y(k) \\ \theta(k) \\ r(k) \\ \omega_a(k) \\ \omega_b(k) \end{bmatrix}$$

$$A = \begin{bmatrix} \mathbf{I}_{4\times4} & \mathbf{0}_{4\times2} \\ \mathbf{0}_{2\times4} & \begin{matrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{matrix} \end{bmatrix}$$

$$\mathbf{f}(\cdot) = \begin{bmatrix} v(k)\Delta t\cos\theta(k) \\ v(k)\Delta t\sin\theta(k) \\ (\omega_g(k)-\omega_b(k))\Delta t \\ \mathbf{0}_{3\times1} \end{bmatrix}$$

$$w(k) = \begin{bmatrix} w_x(k) \\ w_y(k) \\ w_\theta(k) \\ w_r(k) \\ w_{\omega a}(k) \\ w_{\omega b}(k) \end{bmatrix}.$$

APPENDIX II
NAVIGATION FILTER

The posture $(x_{\mathrm{GPS}}(k), y_{\mathrm{GPS}}(k), \theta_{\mathrm{GPS}}(k))$ and the speed $(v_{GPS}(k))$ can be obtained from the GPS receiver. Although the measurement noise is not Gaussian noise, it can be assumed Gaussian noise during a short time interval because it has a large time constant. The measurement equation is given as follows:

$$\begin{aligned} z(k) &= H(k)\mathbf{X}(x) + \nu(k) \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \dfrac{2\pi n(k)}{\Delta t} & 0 & 0 \end{bmatrix} \mathbf{X}(k) + \nu(k) \end{aligned} \tag{31}$$

where $\nu(k)$ is a Gaussian noise vector.

Based on (30) and (31), the extended Kalman filter is used to estimate the position of a car. Let the estimated states be $\hat{X}(k)$ and $\mathbf{E}(\nu(t)\nu^*(t-\tau)) = R\delta(\tau)$ and $\mathbf{E}[w(t)w^*(t-\tau)] = Q\delta(\tau)$. Using the well-known extended Kalman filter, the position of a car is estimated as follows:

$$\begin{aligned} \hat{X}(k) &= \hat{X}^-(k) + K(k)(z(k) - \hat{z}(k)) \\ \hat{X}^-(k) &= A\hat{X}(k-1) + f\left(\hat{X}(k-1),\, v(k),\, w(k)\right) \\ \hat{z}(k) &= H(k)X^-(k) \\ K(k) &= P^-(k)H^T(k)[H(k)P^-(k)H^T(k) + R(k)]^{-1} \\ P^-(k) &= \Phi(k-1)P(k-1)\Phi^T(k-1) + Q(k-1) \\ P(k) &= (I - K(k)H(k))P^-(k) \\ \Phi(k-1) &= A + \left.\frac{\partial f(X(k), v(k), w(k))}{\partial X(k)}\right|_{X(k)=\hat{X}(k-1)} \end{aligned}$$

where $\hat{X}(k)$ is the estimated state.

REFERENCES

[1] Y. Fujita, T. Nawaoka, and E. Hirohata, "Development of a mapping and guidance database for automobile navigation system," in *Proc. IEEE Vehicular Technology Conf.*, 1991, pp. 869–874.

[2] A. Bastiaansen, "Interchangeability standard for digital maps open the market for car navigation systems," in *Proc. IEEE Conf. Intelligent Transportation Systems*, 1997, pp. 520–524.

[3] G. K. H. Pang, K. Takahashi, T. Yokota, and H. Takenaga, "Adaptive route selection for dynamic route guidance system based on fuzzy-neural approaches," *IEEE Trans. Veh. Technol.*, vol. 48, pp. –<AUTHOR: PAGES?>, <AUTHOR: MONTH?> 1999.

[4] T. Nakahara and N. Yomoto, "ITS development and deployment in Japan," in *Proc. IEEE Conf. Intelligent Transportation Systems*, 1997, pp. 631–636.

[5] C. Sossai, P. Bison, G. Chemello, and G. Trainito, "Sensor fusion for localization using possibility theory," *Control Eng.Practice*, vol. <AUTHOR: VOLUME?>, no. 7, pp. 773–782, 1997.

[6] D. L. Boley, E. S. Steinmetz, and K. T. Sutherland, "Robot localization from landmarks using recursive total least squares," in *Proc. IEEE Int. Conf. Robotics and Automation*, vol. 2, 1996, pp. 1381–1396.

[7] M. Betke and L. Gurvits, "Mobile robot localization using landmarks," *IEEE Trans. Robot. Automat.*, vol. 13, pp. 251–263, Apr. 1997.

[8] H. Wehn and P. R. Bélanger, "Ultrasound-based robot position estimation," *IEEE Trans. Robot. Automat.*, vol. 13, pp. 682–692, Oct. 1997.

[9] C.-C. Tsai, "A localization system of a mobile robot by fusing dead-reckoning and ultrasonic measurements," *IEEE Trans. Instrum. Meas.*, vol. 47, pp. 1399–1404, Oct. 1998.

[10] U. Larsson, J. Forsberg, and Å Wernersson, "Mobile robot localization: Integrating measurements from a time-of-fight laser," *IEEE Trans. Ind. Electron.*, vol. 43, pp. 422–431, June 1996.

[11] J. Hancock, E. Hoffman, R. Sulliva, D. Ingrimarson, D. Langer, and M. Hebert, "High-performance laser range scanner," in *Proc. SPIE Intelligent Transportation Systems*, 1997, pp. –<AUTHOR: PAGES?>.

[12] J. Hancock, M. Hebert, and C. Thorpe, "Laser intensity-based obstacle detection," in *Proc. Int. Conf. Intelligent Robots and Systems*, 1998, pp. 1541–1546.

[13] L. Delahoche, C. Pégard, B. Marchic, and P. Vasseur, "A navigation system based on an omnidirectional vision sensor," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, vol. 2, 1997, pp. 718–724.

[14] J. M. Armingol, L. Moreno, A. de la Escalera, and M. A. Salichs, "Landmark perception planning for mobile robot localization," in *Proc. IEEE Int. Conf. Robotics and Automation*, May 1998, pp. 3425–3430.

[15] T. M. Jochem, D. A. Pomerleau, and C. E. Thorpe, "Vision based intersection navigation," in *Proc. IEEE Intelligent Vehicles Symp.*, 1996, pp. 391–396.

[16] M. Rombaut and D. Meizel, "Dynamic data temporal multisensor fusion in the Prometheus ProLab2 Demonstrator," in *Proc. IEEE Int. Conf. Robotics and Automation*, vol. 4, <AUTHOR: YEAR?>, pp. 3576–3583.

[17] M. Maurer and E. D. Dickmanns, "A system architecture for autonomous visual road vehicle guidance," in *Proc. IEEE Conf. Intelligent Transportation Systems*, 1997, pp. 578–583.

[18] B. W. Parkinson and J. J. Spiker Jr. *et al.*, *Global Positioning System Theory and Applications*. <AUTHOR: LOCATION OF INSTITUTE?>: American Inst. Aeronautics and Astronautics, 1996.

[19] M. S. Grewal and A. P. Andrew, *Kalman Filtering: Theory and Practice*. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[20] S. Ishikawa, Y. Suwa, N. Nakahara, T. Ito, T. Yamamoto, Y. Murakami, and T. Yasuda, "Hybrid GPS for land vehicle," in *Proc. ION GPS-94*, 1994, pp. 1301–1306.

[21] R. Da and G. Dedes, "Nonlinear smoothing of dead reckoning data with GPS measurements," *Proc. ION GPS-95*, pp. 1285–1294, Jan. 1995.

[22] K. Kobayashi, K. C. Cheok, K. Watanabe, and F. Munekata, "Accurate differential global positioning system via fuzzy logic Kalman filter sensor fusion technique," *IEEE Trans. Ind. Electron.*, vol. 45, pp. 510–518, June 1998.

[23] M. Arikawa, "Personal dynamic maps based on distributed geographic information servers," in *Proc. Vehicle Navigation and Information Systems Conf.*, 1994, pp. 591–596.

[24] H. Claussen and R. Bosch Gmbh, "Status and directions of digital map databases in Europe," in *Proc. Vehicle Navigation and Information Systems Conf.*, 1993, pp. 25–28.

[25] B.-C. Lee, "A study on the correction of positioning accuracy of car navigation system and map matching algorithm," in *Proc. 5th World Congr. Intelligent Transport Systems*, Seoul, Korea, Oct. 1998, pp. –<AUTHOR: PAGES?>.

[26] S. Kim, J.-H. Kim, and I.-H. Hyun, "Development of a map matching algorithm for car navigation system using fuzzy Q-factor algorithm," in *Proc. World Congr. Intelligent Transport Systems*, Seoul, Korea, Oct. 1998, pp. –<AUTHOR: PAGES?>.

[27] S. Kim and J.-H. Kim, "Q-factor map matching method using adaptive fuzzy network," *Proc. IEEE Int. Conf. Fuzzy Systems*, vol. 2, pp. 628–633, 1999.

[28] D. G. Luenberger, *Optimization By Vector Space Methods*. New York: Wiley, 1968.

[29] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*. New York: McGraw-Hill, 1991.

[30] M. Brown and C. Harris, *Neurofuzzy Adaptive Modeling and Control*. Englewood Cliffs, NJ: Prentice-Hall, 1994.

[31] G. Campion, G. Bastin, and B. D'Andrea-Novel, "Structural properties and classification of kinematic and dynamic models of wheeled mobile robot," *IEEE Trans. Robot. Automat.*, vol. 12, pp. 47–62, Jan. 1996.

[32] B. Barshan and H. F. Durrant-Whyte, "Inertial navigation systems for mobile robots," *IEEE Trans. Robot. Automat.*, vol. 11, pp. 328–342, June 1995.

Sinn Kim received the B.S. and M.S. degrees in electrical engineering in 1994 and 1996, respectively, from Korea Advanced Institute of Science and Technology, Taejon, Korea, where he is currently working toward the Ph.D degree in electrical engineering.

His research interests include localization of mobile vehicles, robust observers, and intelligent control.

Mr. Kim was the Recipient of the Best Post Award at the 1995 IEEE International Conference on Evolutionary Computation, held in Nagoya, Japan.

Jong-Hwan Kim (S'85–M'88) received the B.S., M.S., and Ph.D. degrees in electronics engineering from Seoul National University, Seoul, Korea, in 1981, 1983, and 1987, respectively.

Since 1988, he has been with the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology, Taejon, Korea, where he is currently a Professor. He was a Visiting Scholar at Purdue University, West Lafayette, IN, from September 1992 to August 1993. He is an Associate Editor of the *International Journal of Intelligent and Fuzzy Systems*. He is one of the Co-Founders of the Asia-Pacific Conference on Simulated Evolution and Learning and the General Chair of the Congress on Evolutionary Computation 2001. He is currently serving as the President of both the Federation of International Robot-Soccer Association (FIRA) and IROC. He was the Guest Editor of the Special Issue on MiroSot'96 of the *Journal of Robotics and Autonomous Systems* and on Soccer Robotics of the *Journal of Intelligent Automation and Soft Computing*. His research interests are in the area of evolutionary multi-agent robotic systems.

Dr. Kim is an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION. He is included in the Barons 500 Leaders for the New Century as the Founder of FIRA and of IROC for the Robot Olympiad. He was the recipient of the 1988 Choongang Young Investigator Award from the Choongang Memorial Foundation, the LG YonAm Foundation Research Fellowship in 1992, the Korean Presidential Award in 1997, and the SeoAm Foundation Research Fellowship in 1999.