Autonomous Passive Localization Algorithm for Road Sensor Networks

Jaehoon (Paul) Jeong, *Member*, *IEEE*, Shuo Guo, *Student Member*, *IEEE*,
Tian He, *Member*, *IEEE*, and David H.C. Du, *Fellow*, *IEEE*

Abstract—Road networks are one of important surveillance areas in military scenarios. In these road networks, sensors will be sparsely deployed (hundreds of meters apart) for the cost-effective deployment. This makes the existing localization solutions based on the ranging ineffective. To address this issue, this paper introduces a novel approach based on the passive vehicular traffic measurement, called Autonomous Passive Localization (*APL*). Our work is inspired by the fact that vehicles move along routes with a known map. Using binary vehicle-detection time stamps, we can obtain distance estimates between any pair of sensors on roadways to construct a virtual graph composed of sensor identifications (i.e., vertices) and distance estimates (i.e., edges). The virtual graph is then matched with the topology of the road map, in order to identify where sensors are located on roadways. We evaluate our design outdoors on Minnesota roadways and show that our distance estimate method works well despite traffic noises. In addition, we show that our localization scheme is effective in a road network with 18 intersections, where we found no location matching error, even with a maximum sensor time synchronization error of 0.07 sec and a vehicle speed deviation of 10 km/h.

Index Terms—Sensor network, passive localization, road network, binary vehicle detection, time stamp, prefiltering, graph matching.

1 Introduction

Road networks are one of important infrastructures under surveillance in military operations. For the surveillance of these road networks, the localization of sensors is a prerequisite, providing target positions. In the military scenarios, it has been envisioned that for the fast and safe deployment, unmanned aerial vehicles drop a large number of wireless sensors into road networks around a target area. For the localization, many solutions have been proposed, using 1) precise range measurements (e.g., TOA [1], TDOA [2], and AOA [3]) or 2) connectivity information (e.g., Centroid [4], APIT [5], SeRLoc [6], and Robust Quads [7]) between sensors for sensor localization. To cover a large area in road networks, sensors have to be sparsely deployed (hundreds of meters apart) to save costs. In this sparse deployment, since sensors cannot reach each other either through ranging devices (e.g., Ultrasound signals can only propagate 20-30 feet) or single-hop RF connectivity, the previous solutions become ineffective.

To address this issue, we propose an Autonomous Passive Localization (APL) algorithm for extremely sparse wireless sensor networks. This algorithm is built upon an observation: Military targets normally use roadways for maneuver; therefore, only the sensors near the road are actually useful for surveillance. The sensors away from the roadway can only be used for communication, since targets are out of their sensing range. In other words, their localization is unimportant. In such a scenario, the research question becomes *how sensors on/near a road can identify their positions in a sparse deployment without any pairwise ranging or connectivity information*.

The high-level idea of our solution is to use vehicles on roadways as natural events for localization. The solution would be trivial if all sensors are equipped with sophisticated vehicle identification sensing devices, because measuring the distance between two sensors by multiplying vehicles' average speed by Time Difference on Detection (TDOD) between two sensors corresponding to the *same* vehicle is relatively easy. Obviously vehicle identification sensors would be costly in terms of hardware, energy, and computation. Therefore, the challenging research question becomes *how to obtain locations of the sensors, using only binary detection results without the vehicle identification capability in sensors*. Note that vehicle identification sensors can generate a unique signature for each vehicle type [8]. With this signature and time stamps, the distance estimation is trivial between sensors, so the localization is also trivial.

Our main idea is as follows: Through statistical analysis of vehicle-detection time stamps, we can obtain distance estimates between any pair of sensors on roadways to construct a virtual graph composed of sensor identifications (i.e., vertices) and distance estimates (i.e., edges). This virtual graph is then matched with the topology of the known road map. Thus, this unique mapping allows us to identify where sensors are located on roadways.

Specifically, our localization scheme consists of three phases: 1) the estimation of the distance between two arbitrary sensors in the same road segment; 2) the construction of the connectivity of sensors on roadways; and 3) the identification of sensor locations through matching the constructed connectivity of sensors with the

• *The authors are with the Department of Computer Science and Engineering, University of Minnesota, Twin Cities, EE/CS Building, 200 Union Street SE, Minneapolis, MN 55455.*
E-mail: {jjeong, sguo, tianhe, du}@cs.umn.edu.

Manuscript received 10 May 2010; revised 5 July 2010; accepted 26 July 2010; published online 7 Dec. 2010.
Recommended for acceptance by S. Nikoletseas.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2010-05-0278.
Digital Object Identifier no. 10.1109/TC.2010.260.

graph model for the road map. Our key contributions in this paper are as follows:

- A new architecture for autonomous passive localization using only the binary detection of vehicles in the road networks. Unlike previous approaches, *APL* is designed specially for sparse sensor networks where long-distance ranging is difficult, if not impossible.
- A statistical method to estimate road segment distance between two arbitrary sensors, based on the concept of the TDOD. For the distance estimation, the TDOD operation uses the correlation between the time stamps of sensors geographically close to each other. Though the TDOD operation was first proposed in our earlier work *APL* [9], the validity of TDOD operation is analyzed in this paper.
- A prefiltering algorithm for selecting only robust *edge distance estimates* between two arbitrary sensors in the same road segment. Unreliable *path distance estimates* are filtered out for better accuracy. The prefiltering proposed by our earlier work *APL* [9] can handle only the scenario where sensors are deployed only at intersections, however, this paper provides an enhanced prefiltering to handle the scenario where sensors are deployed both at intersections and in the middle of road segments.
- A graph matching algorithm for matching the sensor's identification with a position on the road map of the target area. The graph matching uses the isomorphic structure between the road network and the sensor network.
- Considerations on practical issues, such as time synchronization error, vehicle detection missing, and duplicate vehicle detection.

The rest of this paper is organized as follows: In Section 2, we summarize related work for the localization in wireless sensor networks. Section 3 describes the problem formulation for our Autonomous Passive Localization. Our APL system design is described in Section 4. In Section 5, we discuss practical issues that can affect our localization scheme in practice. Section 6 evaluates our APL algorithm in realistic settings. We conclude this paper along with future work in Section 7.

2 Related Work

Many localization schemes have previously been proposed, and they can be categorized into three classes: 1) Range-based localization schemes, 2) Range-free localization schemes, and 3) Event-driven localization schemes. Range-based schemes require costly hardware devices (e.g., ultrasound ranging device) to accurately estimate the distance between nodes, along with the additional energy consumption. The Time of Arrival (TOA) (e.g., GPS [1]) and Time Difference of Arrival (TDOA) schemes (e.g., Cricket [2] and AHLoS [10]) measure the propagation time of the signal and estimate the distance based on the propagation speed. Since ultrasound signals usually propagate only 20-30 feet, TDOA is not quite suitable for sparse networks. The Angle of Arrival (AOA) schemes [3] estimate the positions of the nodes by sensing the direction from which a signal is received. The Received Signal Strength Indicator (RSSI) schemes [11], [12] use either theoretical or empirical models to estimate the distance based on the loss of power during signal propagation. The RSSI can use the same hardware used for communication, but its ranging accuracy is not good due to the communication radio irregularity [13], [14]. Since both AOA and RSSI are also constrained by their effective distance, they are not appropriate for the localization in sparse road sensor networks that is the target in this paper.

The range-free localization schemes try to localize sensors without costly ranging devices. One of the most popular range-free schemes is based on the anchor-based scheme. The main idea is that the nonanchors can determine their locations using the overlapping region of communication areas for the anchors [4], [5], [15], [16], [17]. However, since these schemes require a dense deployment of anchors to give beacon signals, these solutions are not applicable to localization in sparse road sensor networks.

Recently, a series of event-driven localization schemes have been proposed to simplify the functionality of sensors for localization and to provide high-quality localization. The main idea of these schemes is to use artificial events for sensor localization that are generated from the event scheduler [18], [19], [20], [21], [22]. Although their effective range can reach hundreds of meters, additional external devices and manual operations are needed to generate artificial events. On the other hand, our localization scheme is a new branch of event-driven localization schemes. Because our localization scheme is based on the natural events of moving vehicles, event delivery is not problematic.

The graph matching, as one key component in this paper, has intensively been researched in the past [23], [24], [25]. The isomorphic graph matching is performed for matching two graphs with the same structure [26]; that is, both graphs have the same number of vertices and the same edge connection structures. Umeyama [23] proposed an isomorphic graph matching algorithm for two isomorphic graphs whose edge weights are a little different. The proposed algorithm uses an eigen-decomposition approach to deal with this weighted graph matching. In the case where a graph is a subgraph of another graph, that is, the subgraph isomorphism, the subgraph matching is performed. Ullmann [25] proposed an algorithm for this subgraph matching. The proposed algorithm has the limitation that the graphs cannot have attributes, such as edge weight. To deal with graph attributes in the subgraph matching, Cordella et al. [24] proposed a deterministic matching method for verifying both isomorphism and subgraph isomorphism. In our APL, Umeyama's method is used for the graph matching based on the isomorphism between the road network and the sensor network.

3 Problem Formulation

We consider a network model where sensors are placed at both intersection points and nonintersection points on road networks. The objective is to localize wireless sensors deployed in road networks only with a road map and binary vehicle-detection time stamps taken by sensors as shown in Fig. 1a. Section 3.1 lists definitions for APL and Section 3.2 lists assumptions.

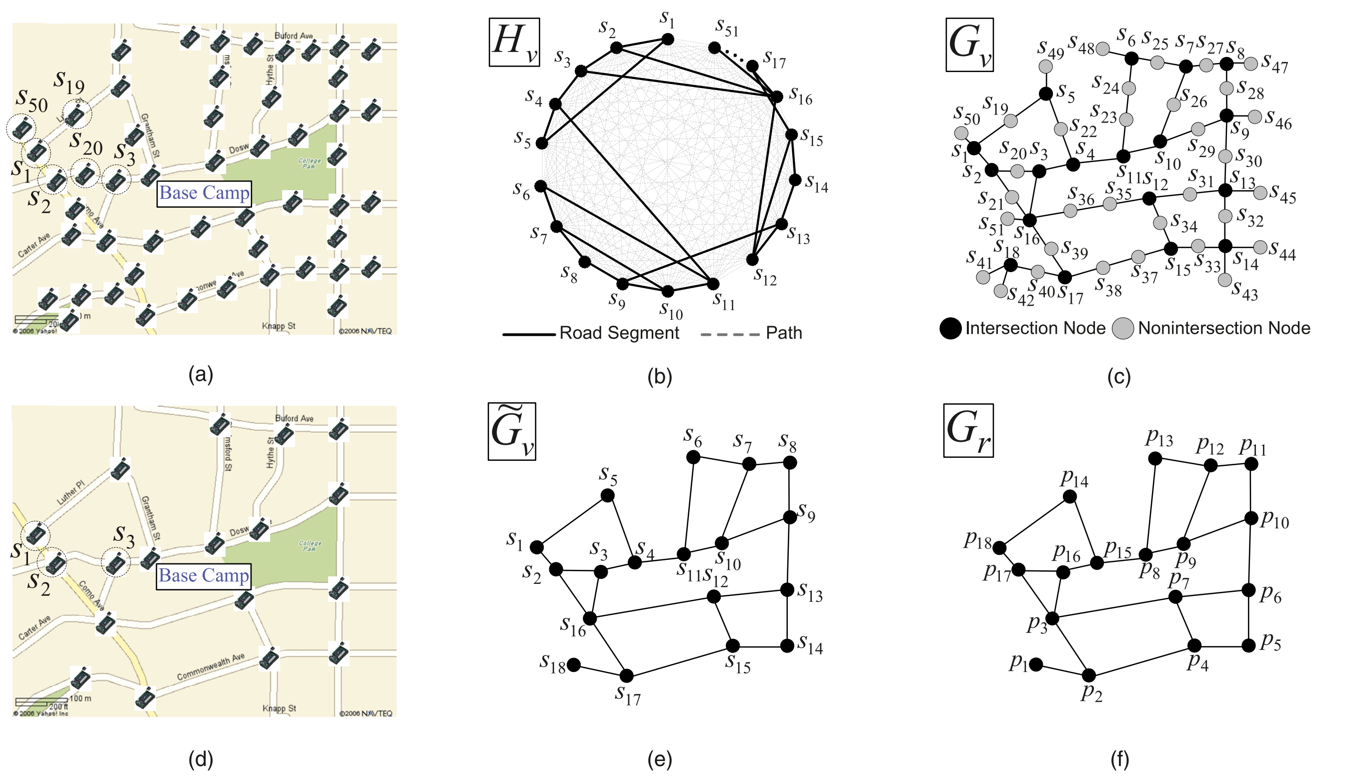


Fig. 1. Wireless sensor network deployed in road network. (a) Road Network with Wireless Sensors. (b) Virtual Topology of Wireless Sensors: $H_v = (V_v, M_v)$. (c) Virtual Graph representing Sensor Network: $G_v = (V_v, E_v)$. (d) Road Network only with Intersection Nodes of Virtual Graph. (e) Reduced Virtual Graph consisting of Intersection Nodes of Virtual Graph: $\tilde{G}_v = (\tilde{V}_v, \tilde{E}_v)$. (f) Real Graph corresponding to Road Map: $G_r = (V_r, E_r)$.

3.1 Definitions

We define nine terms as follows:

1. **Neighboring nodes.** Sensors geographically adjacent to a sensor on the road network regardless of the connectivity by the communication range of a sensor. In Fig. 1a, sensors s_2, s_{19}, and s_{50} are the neighboring nodes of sensor s_1.
2. **Intersection nodes.** Sensors placed at an intersection and having more than two neighboring sensors (i.e., degree ≥ 3). In Fig. 1a, sensors s_1, s_2, and s_3 are intersection nodes.
3. **Nonintersection nodes.** Sensors placed at a nonintersection and having one or two neighboring sensors. In Fig. 1a, sensors s_{19}, s_{20}, and s_{50} are nonintersection nodes.
4. **Virtual Topology.** Let *Virtual Topology* be $H_v = (V_v, M_v)$, where $V_v = \{s_1, s_2, \ldots, s_n\}$ is a set of sensors in the road network, and $M_v = [v_{ij}]$ is a matrix of path length v_{ij} for sensors s_i and s_j. Fig. 1b shows a virtual topology of sensors in the road network, shown in Fig. 1a. M_v is a complete graph, since there is an edge between two arbitrary sensors. We define the edge of the virtual topology as *virtual edge*. In Fig. 1b, among the virtual edges, a solid thick line represents an *edge estimate* (i.e., road segment) between two sensors, which means that they are adjacent on the road network as *neighboring nodes*. The dotted thin line represents a *path estimate* between two sensors, which means that they are not adjacent on the road network.
5. **Virtual Graph.** Let *Virtual Graph* be $G_v = (V_v, E_v)$, where $V_v = \{s_1, s_2, \ldots, s_n\}$ is a set of sensors in the road network, and $E_v = [v_{ij}]$ is a matrix of road segment length v_{ij} between sensors s_i and s_j. Fig. 1c shows a virtual graph of the sensor network deployed on the road network shown in Fig. 1a, where the black node represents an intersection node and the gray node represents a nonintersection node.
6. **Reduced Virtual Graph.** Let *Reduced Virtual Graph* be $\tilde{G}_v = (\tilde{V}_v, \tilde{E}_v)$, where $\tilde{V}_v = \{s_1, s_2, \ldots, s_m\}$ is a set of sensors placed only at intersections in the road network, and $\tilde{E}_v = [v_{ij}]$ is a matrix of road segment length v_{ij} between intersection nodes s_i and s_j. The reduced virtual graph $\tilde{G}_v$ is obtained by deleting nonintersection nodes and their edges from the virtual graph G_v through the degree information in G_v. Refer to Section 4.4.1. For example, Fig. 1e shows a reduced virtual graph consisting of only intersection nodes of virtual graph in Fig. 1c.
7. **Real Graph.** Let *Real Graph* be $G_r = (V_r, E_r)$, where $V_r = \{p_1, p_2, \ldots, p_n\}$ is a set of intersections in the road network around the target area, and $E_r = [r_{ij}]$ is a matrix of road segment length r_{ij} for intersections p_i and p_j. Real Graph can be obtained through map services, such as Google Earth and Yahoo Maps. Fig. 1f shows a real graph corresponding to the road network whose intersection points have intersection sensor nodes, shown in Fig. 1d. The real graph is *isomorphic* to the reduced virtual graph $\tilde{G}_v$ shown in Fig. 1e [26].

8. **Shortest Path Matrix.** Let *Shortest Path Matrix* for $G = (V, E)$ be M such that $M = [m_{ij}]$ is a matrix of the shortest path length between two arbitrary nodes i and j in G. M is computed from E by the All-Pairs Shortest Paths algorithm, such as the *Floyd-Warshall algorithm* [27]. We define M_r as the shortest path matrix for the real graph $G_r = (V_r, E_r)$, and define M_v as the shortest path matrix for the virtual graph $G_v = (V_v, E_v)$.
9. **APL Server.** A computer that performs the localization algorithm with binary vehicle-detection time stamps collected from the sensor network.

3.2 Assumptions

The localization design of *APL* is based on the following assumptions:

- Sensors have simple sensing devices for binary vehicle detection without any costly ranging or GPS devices [28]. Each detection is a tuple (s_i, t_j), consisting of a sensor ID s_i and time stamp t_j.
- There exists an ad hoc network consisting of sensors or a Delay Tolerant Network (DTN) for wireless sensors to deliver vehicle-detection time stamps to the *APL server*. For such a DTN, vehicles as *data mules* [29] can construct Vehicular Ad Hoc Networks (VANETs) for delivering the time stamps to the *APL server* through the VANET forwarding schemes, such as VADD [30] and TBD [31].
- Sensors are time-synchronized at the millisecond level. For this time synchronization accuracy in sparse road sensor networks, the time synchronization protocol in [32] can be used along with vehicles for time information sharing in the DTN scenario.
- The *APL server* has road map information for the target area under surveillance and can construct a real graph consisting of intersections in the road network.
- Vehicles pass through all road segments on the target road networks. The vehicle mean speed is close (but not identical) to the speed limit assigned to roadways. The standard deviation of vehicle speed is assumed to be a reasonable value, based on real road traffic statistics obtained from transportation research [33].
- Sensors are deployed into the target road network such that each road intersection has one intersection node and also each intersection node has at least three neighboring nodes whose road segments are different from each other.

4 APL System Design

4.1 System Architecture

We use an asymmetric architecture for localization as in Fig. 2 in order to simplify the functionality of sensors for localization. As simple devices, sensors only monitor road traffic and register vehicle-detection time stamps into their local repositories. A server called the *APL server* processes the complex computation for localization. Specifically, the localization procedure consists of the following steps as shown in Fig. 2:

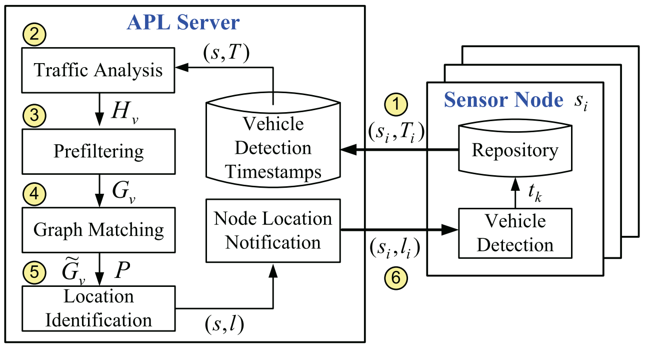


Fig. 2. APL system architecture.

- **Step 1.** After road traffic measurement, sensor s_i sends the *APL server* its vehicle-detection time stamps along with its sensor ID, i.e., (s_i, T_i), where s_i is sensor ID and T_i is time stamps.
- **Step 2.** The traffic analysis module estimates the road segment length between two arbitrary sensors with the time stamp information, constructing a virtual topology $H_v = (V_v, M_v)$, where V_v is the vertex set of sensor IDs, and M_v is the matrix containing the distance estimate of *every sensor pair*.
- **Step 3.** The prefiltering module converts the virtual topology H_v into a virtual graph $G_v = (V_v, E_v)$, where V_v is the vertex set of the sensor IDs, and E_v is the adjacency matrix of the estimated road segment lengths.
- **Step 4.** The graph matching module constructs a reduced virtual graph $\tilde{G}_v = (\tilde{V}_v, \tilde{E}_v)$ from the virtual graph G_v, where $\tilde{V}_v$ is a set of only intersection nodes among V_v, and $\tilde{E}_v$ is a set of edges whose endpoints both belong to $\tilde{V}_v$. $\tilde{G}_v$ is *isomorphic* to the real graph $G_r = (V_r, E_r)$. The graph matching module then computes a permutation matrix P, making the reduced virtual graph $\tilde{G}_v = (\tilde{V}_v, \tilde{E}_v)$ be isomorphic to the real graph $G_r = (V_r, E_r)$.
- **Step 5.** The location identification module determines each sensor's location on the road map by applying the permutation matrix P to both the reduced virtual graph $\tilde{G}_v$ and the real graph G_r. Through this mapping, node location information (s, l) is constructed such that s is the sensor ID vector, and l is the corresponding location vector; that is, $l_i = (x_i, y_i)$, where s_i is the sensor ID, x_i is the x-coordinate, and y_i is the y-coordinate in the road map.
- **Step 6.** With (s, l), the *APL server* sends each sensor s_i its location with a message (s_i, l_i).

In the rest of this section, we describe the technical content of each step. We start with the second step, because the operations in Step 1 are straightforward.

4.2 Step 2: Traffic Analysis for Road Segment Length Estimation

In order to estimate road segment lengths, we found a key fact that vehicle arrival patterns in one sensor are statistically maintained at neighboring sensors close to the sensor. This means that the more closely the two sensors are located, the more correlated the vehicle-detection time stamps are. Consequently, we can estimate road segment length with

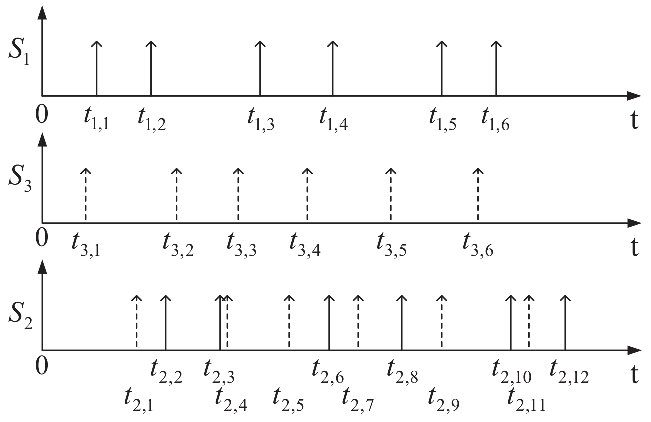

S_1
$t_{1,1}$ $t_{1,2}$ $t_{1,3}$ $t_{1,4}$ $t_{1,5}$ $t_{1,6}$
S_3
$t_{3,1}$ $t_{3,2}$ $t_{3,3}$ $t_{3,4}$ $t_{3,5}$ $t_{3,6}$
S_2
$t_{2,1}$ $t_{2,2}$ $t_{2,3}$ $t_{2,4}$ $t_{2,5}$ $t_{2,6}$ $t_{2,7}$ $t_{2,8}$ $t_{2,9}$ $t_{2,10}$ $t_{2,11}$ $t_{2,12}$
0
t

Fig. 3. Detection sequence for vehicles at sensors s_1, s_3, and s_2.

estimated movement time between two adjacent sensors using the correlation of the time stamp sets of these two sensors, along with the vehicle mean speed (i.e., speed limit given on the road segment). Through both outdoor test and simulation, we found that we can estimate the lengths of road segments used by vehicles during their travels on roadways only with vehicle-detection time stamps.

4.2.1 Time Difference on Detection Operation

In this section, we explain the TDOD operation on binary vehicle-detection time stamps. The TDOD operation for time stamp sets T_i and T_j from two sensors s_i and s_j is defined as follows:

$$d_{hk}^{ij} = |t_{ih} - t_{jk}|, \tag{1}$$

where $t_{ih} \in T_i$ for $h = 1, \ldots, |T_i|$ is the hth time stamp of sensor s_i and $t_{jk} \in T_k$ for $k = 1, \ldots, |T_j|$ is the kth time stamp of sensor s_j. We define a quantized TDOD as follows:

$$\hat{d}_{hk}^{ij} = g\big(d_{hk}^{ij}\big), \tag{2}$$

where g is a quantization function to map the real value of d_{hk}^{ij} to the discrete value. The interval between two adjacent quantization levels is defined according to *the granularity of the time difference*, such as 1 second, 0.1 second, or 1 millisecond. The number m of quantization levels (i.e., q_k for $k = 1, \ldots, m$) is determined considering the expected movement time of vehicles in the longest road segment of the relevant road network.

We define *frequency* as the count of a discrete time difference. After the TDOD operation for two time stamp sets from two sensors, the quantization level with the *highest frequency* (i.e., $\hat{d}^{ij}$) is regarded as the movement time of vehicles for the roadway between these two sensors s_i and s_j as follows:

$$\hat{d}^{ij} \leftarrow \arg\max_{q_k} f(q_k), \tag{3}$$

where f is the frequency of quantization level q_k for $k = 1, \ldots, m$; that is, in (3), the value of $\hat{d}^{ij}$ is set to the quantization level q_k with the maximum frequency. The movement time on the road segment can be converted into road segment length using the formula $l = vt$, where l is the road segment's length, v is the vehicle mean speed, and t is the vehicle mean movement time on the road segment.

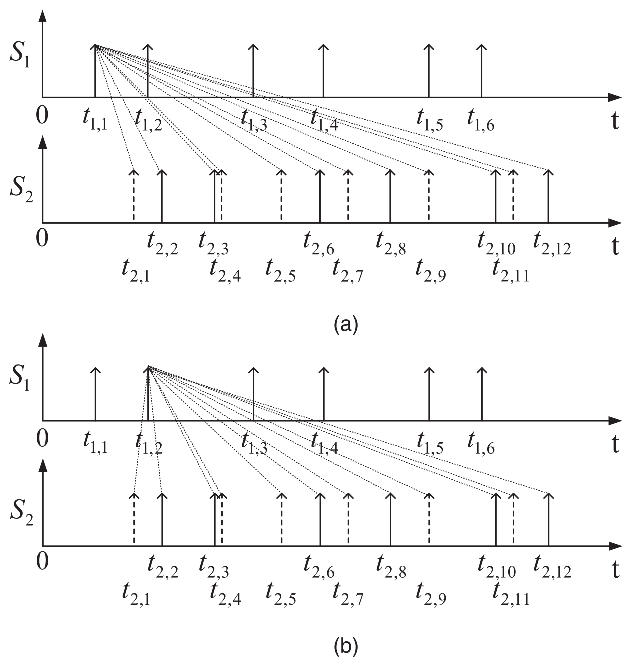

S_1
$t_{1,1}$ $t_{1,2}$ $t_{1,3}$ $t_{1,4}$ $t_{1,5}$ $t_{1,6}$
S_2
$t_{2,1}$ $t_{2,2}$ $t_{2,3}$ $t_{2,4}$ $t_{2,5}$ $t_{2,6}$ $t_{2,7}$ $t_{2,8}$ $t_{2,9}$ $t_{2,10}$ $t_{2,11}$ $t_{2,12}$
0
t
(a)
(b)

Fig. 4. Time Difference on Detection for sensors s_1 and s_2. (a) TDOD between time stamps $t_{1,1}$ and $t_{2,i}$. (b) TDOD between time stamps $t_{1,2}$ and $t_{2,i}$.

For example, Fig. 3 shows the detection sequence for vehicles at intersection nodes s_1, s_2, and s_3 in Fig. 1a, where s_2 is a common neighbor of s_1 and s_3. Fig. 4 shows the TDOD operation for nodes s_1 and s_2 that is a kind of Cartesian product for two time stamp sets. Fig. 5 shows the histogram [34] obtained by the TDOD operation for two time stamp sets. The time difference value (7.3 sec) with the highest frequency indicates the estimated movement time between two nodes. For the theoretical analysis of the validity of TDOD, please refer to Section 4.2.2.

We performed outdoor test to verify whether our TDOD operation could give good estimates for road segment lengths in terms of vehicle movement time. The results of outdoor test indicate that our TDOD can give reasonable road segment length indicators. Fig. 6 shows the road map of local roadways in Minnesota for outdoor test. The test

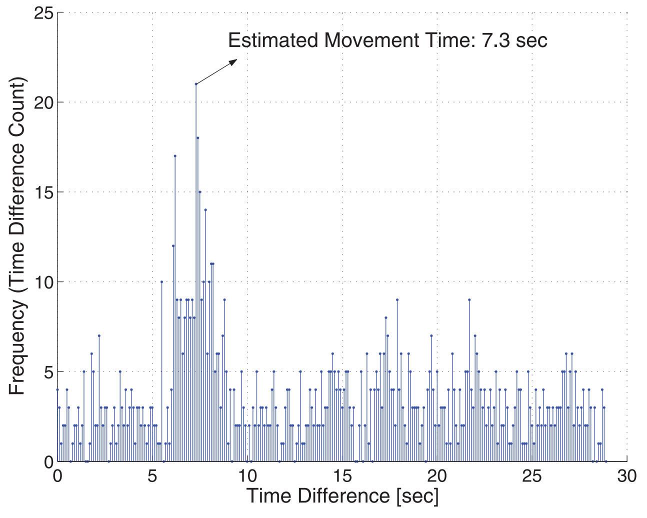


Fig. 5. Estimation of movement time through TDOD operation.

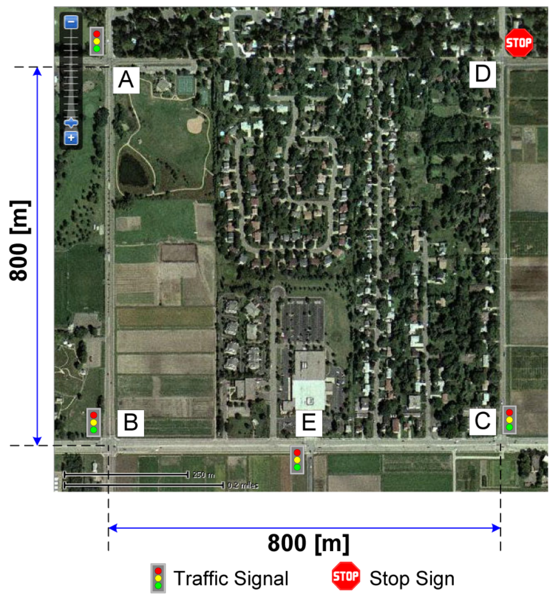


Fig. 6. Road network for outdoor test.

roadways consist of four intersections A, B, C, and D. Speed limits on these road segments are 40 or 30 MPH, as shown in Table 1. We performed vehicle detection manually for more accurate observation; note that it is hard to get accurate vehicle detections at intersections with the current motes due to the sensor capability and mote's physical size, so the development of the vehicle-detection algorithm based on motes is left as future work. As shown in Table 1, for the road segments of $A \leftrightarrow B$, $C \leftrightarrow D$, and $D \leftrightarrow A$, the expected movement time is close to the measured movement time obtained by TDOD operation. On the other hand, the road segment of $B \leftrightarrow C$ has a relatively bigger difference between the expected movement time and the measured movement than the other road segments. This is because $B \leftrightarrow C$ has a traffic signal in the middle of the road segment, that is, at intersection E in Fig. 6. Due to this traffic signal, some of vehicles moving on $B \leftrightarrow C$ stop at intersection E, so they take a longer travel time than the expected one. Therefore, if a sensor is deployed at each intersection according to our assumption in Section 3.2, the expected movement time will be close to the measured movement time.

From Table 1, it can be seen that the estimated movement times are close to the expected movement times; note that even though the manual measurement can introduce some human errors, this experimental result shows the significant evidence that the TDOD can provide us with the estimates accurate enough to perform the localization. Note that the road network topology is square in Fig. 6, but the validity of the TDOD operation in the irregular road network topology in Fig. 1a is shown through the simulation in Section 6.

Therefore, with the TDOD operation, the distance estimates between two arbitrary nodes can be obtained for the virtual edges in the virtual topology, as shown in Fig. 1b. Note that this TDOD operation does not classify sensors into either intersection nodes or nonintersection nodes. One important observation is that as two sensors are geographically closer to each other, the TDOD operation on their time stamps gives a better movement time estimate between the two sensors.

TABLE 1
Outdoor Test Results

Road Segment	Distance	Speed Limit	Expected Movement Time	Measured Movement Time
$A \leftrightarrow B$	800 m	40 MPH	44.7 sec	43 sec
$B \leftrightarrow C$	800 m	40 MPH	44.7 sec	54 sec
$C \leftrightarrow D$	800 m	40 MPH	44.7 sec	43 sec
$D \leftrightarrow A$	800 m	30 MPH	59.7 sec	56 sec

4.2.2 Analysis of Movement-Time-Estimation Error

In this section, we analyze the probability that the TDOD operation gives a wrong movement time estimation for an edge rather than an accurate movement time estimation. This probability is defined as the *movement-time-estimation error probability*. In this section, we will show that this movement-time-estimation error probability is very small in the TDOD operation. First, we will describe a scenario for the TDOD operation. Second, in such a scenario, we will compute the movement-time-estimation error probability.

First of all, Fig. 7a shows a scenario for the TDOD operation for the vehicle arrival time stamps for sensors s_1 and s_2 that are adjacent with each other, as shown in Fig. 1a. In this figure, m vehicles arrive at s_1 and only n vehicles among them move to s_2; note that at the time diagram at s_1,

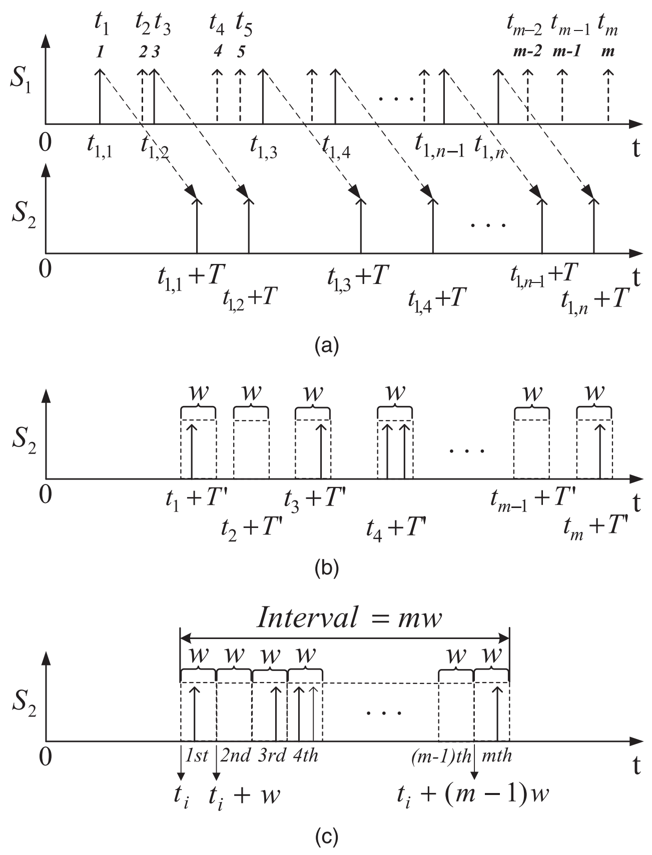

t_1 t_2 t_3 t_4 t_5 t_{m-2} t_{m-1} t_m
1 2 3 4 5 m-2 m-1 m
S_1
$t_{1,1}$ $t_{1,2}$ $t_{1,3}$ $t_{1,4}$ $t_{1,n-1}$ $t_{1,n}$
S_2
$t_{1,1}+T$ $t_{1,2}+T$ $t_{1,3}+T$ $t_{1,4}+T$ $t_{1,n-1}+T$ $t_{1,n}+T$
(a)
w
t_1+T' t_2+T' t_3+T' t_4+T' $t_{m-1}+T'$ t_m+T'
(b)
$Interval = mw$
1st 2nd 3rd 4th (m-1)th mth
t_i t_i+w $t_i+(m-1)w$
(c)
0
t

Fig. 7. TDOD analysis for sensors s_1 and s_2. (a) Vehicle Travel from Sensor s_1 to Sensor s_2 for Vehicle Movement Time T. (b) Vehicle Counting at Time Windows for Noisy Estimate T' at Sensor s_2. (c) Vehicle Counting in Aggregated Time Window of Noisy Estimate T' at Sensor s_2.

the solid arrows denote vehicles moving to s_2 and the dotted arrows denote vehicles moving to other sensors. Suppose that the road segment of (s_1, s_2) has the length of l and the average vehicle speed (e.g., speed limit) is v. Let T be the real movement time for the edge (s_1, s_2) such that $T = l/v$. Clearly, as shown in Fig. 7a, n vehicle moving to s_2 will arrive at s_2 after time T. Thus, the TDOD operation on two time stamp sets of s_1 and s_2 must have at least n pairs of movement time T.

In order to compute the *movement-time-estimation error probability*, we define T' as noisy movement time estimate for the edge (s_1, s_2) that is an arbitrary time. As shown in Fig. 7b, for m arrivals at s_1, we consider m time windows corresponding to noisy estimate T' for the arrivals; note that the window's width w is the granularity of the time difference that determines the interval between two adjacent quantization levels, as mentioned in Section 4.2.1. The window's center corresponds to the sum of the arrival time stamp t_i at s_1 and the noisy estimate T' for $i = 1..m$. For example, for time stamp t_1 at s_1, the window's center is $t_1 + T'$. With these windows, we can count the vehicles corresponding to pairs for the noisy estimate T'. That is, in Fig. 7b, the frequency for T' is the number of the vehicle arrivals within the windows for T' at s_2. Thus, with T and T', we can redefine the *movement-time-estimation error probability* as the probability that the noisy estimate T' has a higher frequency than the real movement time T.

Now, with the setting in Fig. 7b, we can compute the *movement-time-estimation error probability* with the vehicle arrival process. We model this vehicle arrival process as *Poisson process* with arrival rate λ for each sensor. Note this modeling is valid by the following two reasons; 1) the *Kolmogorov-Smirnov test* can accurately approximate the statistics of vehicle interarrival time based on the empirical data for a real roadway into an *exponential distribution* [35] and 2) an *exponential distribution* for the interarrival time is equivalent to a *Poisson distribution* for the arrival number within a unit time [36].

Under the Poisson process, in Fig. 7b, the first property is that the number of arrivals for one window is independent of that of another window [36]. The second property is that the number of arrivals are determined by the length of time interval [36]. From these two properties, the counting process of vehicle arrivals within windows of Fig. 7b can be generalized into the Poisson process of Fig. 7c where the window's time interval is w, the number of windows is m, the time interval of the aggregated window is mw, and the arrival rate is λ. Let P_{error} be movement-time-estimation error probability. Let N be the random variable of the number of vehicle arrivals within the aggregated window, as shown in Fig. 7b. Let m be the number of vehicles arriving at sensor s_1. Let n be the number of vehicles arriving at sensor s_2 from sensor s_1. Let λ be the vehicle arrival rate for sensor s_2. Thus, the *movement-time-estimation error probability* P_{error} can be computed as follows:

$$\begin{aligned} P_{error} &= P[N > n] \\ &= 1 - P[N \le n] \\ &= 1 - \sum_{k=0}^{n} \frac{e^{-\lambda m w}(\lambda m w)^k}{k!}. \end{aligned} \tag{4}$$

TABLE 2
Simulation Configuration

Parameter	Description
Number of sensors N	51 sensors (from s_1 to s_{51}) are deployed in the road network.
Vehicle speed v	$v \sim N(\mu_v, \sigma_v)$ where $\mu_v = 50$ km/h and $\sigma_v = \{1, 2, 3, ..., 14\}$ km/h. The maximum speed is $\mu_v + 3\sigma_v$ and the minimum speed is $\mu_v - 3\sigma_v$. The default of (μ_v, σ_v) is $(50, 5)$.
Vehicle inter-arrival time T	$T \sim Exponential(\lambda)$ where λ is the vehicle arrival rate. The default of T is $\frac{1}{\lambda} = 30$ sec.
Vehicle travel path length L	Let $d_{u,v}$ be the shortest path distance from source position u to destination position v in the road network. $L \sim N(\mu_d, \sigma_d)$ where $\mu_d = d_{u,v}$ meters and $\sigma_d = 500$ meters.
Time synchronization error ϵ	$\epsilon \sim Uniform(-\epsilon_{max}, \epsilon_{max})$ where $\epsilon_{max} = \{0, 0.1, 0.2, ..., 0.7\}$ sec. The default of ϵ_{max} is 0.01 sec.

We compute P_{error} for edge (s_1, s_2) in the sensor network shown in Fig. 1a. Through the simulation with the parameter setting in Table 2, we obtain four parameters m, n, λ, and w to compute P_{error}. The movement-time-estimation error probability is $P_{error} = 8.93 * 10^{-14} \simeq 0$ where $m = 188$, $n = 60$, $\lambda = 0.05$, and $w = 2$ seconds. Therefore, since P_{error} is very small, we can claim that the TDOD operation can give an accurate movement time estimation with a high probability.

4.2.3 Enhancement of Road Segment Length Estimation

We found that an estimate close to a road segment's length cannot always be obtained by the maximum frequency through the TDOD operation discussed in Section 4.2.1. The reason is that there can exist some noisy estimates with higher frequencies than an expected good estimate. Note that even though the probability that a noisy estimate (i.e., random movement time estimate) has a higher frequency than an accurate movement time estimate is very small (as discussed in Section 4.2.2), this case can still happen with a small probability.

In order to resolve this problem, we introduce a more robust estimation method called *aggregation method* where the mean of several adjacent time differences becomes a new TDOD value, and the sum of frequencies of those is the corresponding frequency. This is based on an observation that time differences close to a real time difference (i.e., movement time needed by a vehicle with the vehicle mean speed on a road segment) have relatively high frequencies by the TDOD operation for two time stamp series, as shown in Fig. 4. On the other hand, we observe that a noisy estimate with the highest frequency occurs randomly, and its neighbor estimates have relatively low frequencies. This method based on TDOD aggregation is called *Aggregation Method* and the previous simple TDOD is called *Nonaggregation Method*. We determine the aggregation window size proportionally to standard deviation σ_v of the vehicle speed, such as $c \cdot \sigma_v$ for $c > 0$. Thus, in *Aggregation Method*, the *movement-time-estimation error probability* in Section 4.2.2 will be extremely small.

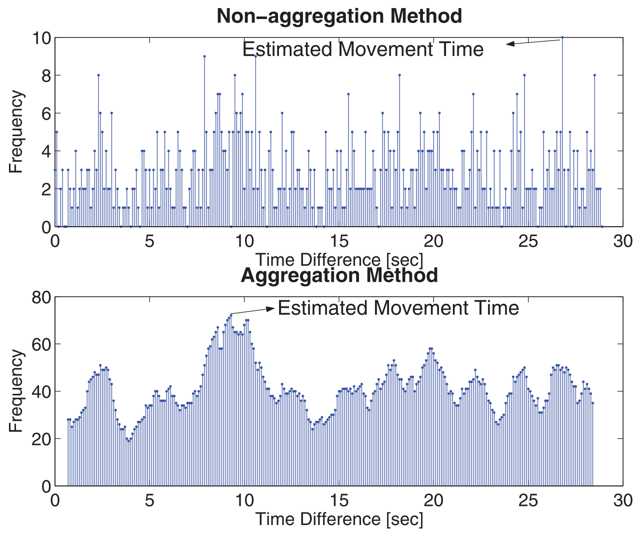


Fig. 8. Comparison between nonaggregation method and aggregation method.

Fig. 8 shows the comparison between the nonaggregation method and aggregation method through simulation. The aggregation window size is 10 such that the vehicle speed deviation σ_v is 10 km/h and the window size factor c is 1. Starting from the time difference value of zero in the histogram for *Nonaggregation Method*, we choose a representative of the adjacent time difference values within the aggregation window size as the mean of them, and then sum their frequencies into the representative's frequency. We then move the window to the right by the unit of time difference value and repeat the computation of the representative and frequency. Thus, the histogram for *Aggregation Method* is obtained by this *moving window*.

We found that for the road segment between sensors s_2 and s_3 in Fig. 1a whose real time difference is 9.36 sec with the vehicle speed $\mu_v = 50$ km/h, the nonaggregation method makes a wrong estimate (i.e., 26.8 sec), but the aggregation method makes a correct estimate (i.e., 9.3 sec). Thus, this aggregation method can be used to obtain good estimates for road segment lengths in a virtual topology.

4.3 Step 3: Prefiltering Algorithm for Virtual Graph

The prefiltering algorithm is performed to make a virtual graph that has only edge estimates among virtual edges (i.e., distance estimates) in the virtual topology (e.g., Fig. 1b) obtained from the TDOD operations in Section 4.2. Our prefiltering algorithm consists of two prefilterings: 1) *Relative Deviation Error* and 2) *Minimum Spanning Tree*.

We explain the prefiltering procedure and the effect of two prefilterings on a virtual topology using Fig. 9. As shown in Fig. 9a, there is a partial road network of the entire one shown in Fig. 1a containing sensors $\{s_1, s_2, s_3, s_4, s_5, s_{19}, s_{20}, s_{22}\}$. In the virtual topology, two arbitrary sensors among them have a distance estimate, as shown in Fig. 9b. Using the prefiltering based on the relative deviation error, we remove the virtual topology's edges corresponding to inaccurate path estimates, and we then construct a virtual graph, shown in Fig. 9c. Next we apply the prefiltering based on the minimum spanning tree to the virtual graph, so the virtual graph containing only the edge estimates is constructed by removing accurate path estimates, as shown

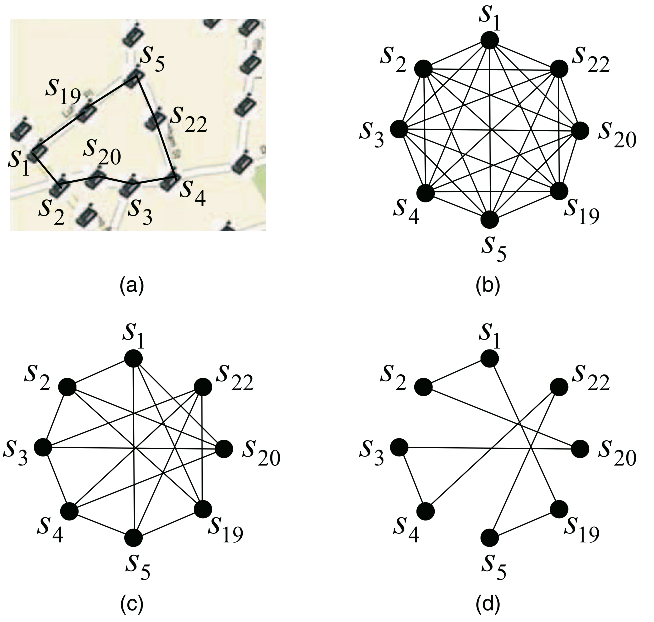


Fig. 9. Procedure of prefiltering for obtaining virtual graph. (a) Road Network with the following Sensors: $\{s_1, s_2, s_3, s_4, s_5, s_{19}, s_{20}, s_{22}\}$. (b) Virtual Topology for the following Sensors: $\{s_1, s_2, s_3, s_4, s_5, s_{19}, s_{20}, s_{22}\}$. (c) Virtual Graph after Prefiltering based on the Relative Deviation Error. (d) Virtual Graph after Prefiltering based on the Minimum Spanning Tree.

in Fig. 9d. In this section, we explain the idea of these two prefilterings for obtaining the virtual graph $G_v = (V_v, E_v)$ from virtual topology $H_v = (V_v, M_v)$ in detail.

4.3.1 Prefiltering Based on the Relative Deviation Error

Large errors in path estimates will significantly affect our future steps. An example is as follows: We know that the smallest entry in M_v must be an edge when no large error occurs, since path lengths are always the sum of more than one edge length. However, when there are large errors in M_v, they can have any value in M_v, that is, either a large value or a small value. In this case, the smallest entry is no longer guaranteed to be an edge estimate because a path estimate can be perturbed to be the smallest one by a large error. As a result, it is very important to filter out all the entries having large errors at first, regarding them as path estimates.

We define *Relative Deviation* (ϕ) as the ratio of the standard deviation (σ) to the mean (μ), that is, $\phi = \sigma/\mu$; note that this is known as the *coefficient of sample variation* in statistics and is used to compare the amount of variance between populations with different means. To compute both the mean μ and the standard deviation σ of each entry in M_v, we use multiple estimation matrices of M_v per measurement time with the same duration. Note that outliers are at first eliminated from the multiple estimation matrices of M_v in order to let μ and σ be less affected by these outliers and then be more robust statistics; the estimates are regarded as outliers when they are less or greater than the δ percent (e.g., 20 percent) of the median of all the sample measures for an entry in M_v; note that the outlier threshold δ is determined as 20 percent through the empirical results in simulations.

In order to compute the relative deviations of the estimates, we divide the vehicle-detection time stamps into time windows (e.g., every half an hour) and perform the TDOD operation for the time stamps of two arbitrary sensors within the same time window. We then compute the relative deviations of the virtual edge estimates for each pair of sensors. If the relative deviation is greater than a certain threshold ε (e.g., 10 percent), the corresponding entry is regarded as a path estimate, and it is replaced with ∞, indicating that this entry is a path estimate. Note that in our prefiltering, the threshold ε is set to the ratio of the known vehicle speed deviation to the vehicle speed limit in the road network. This threshold is empirically determined in order to allow the prefiltering to work considering the real vehicle speed deviation.

Note that for certain reasons (e.g., outliers and measurement errors), the edge estimate can be regarded as path estimate due to a big relative deviation error, leading to being filtered out. In this case, we cannot perform the isomorphic graph matching since the sensor network topology (i.e., reduced virtual graph) and the road network topology (i.e., real graph) are not isomorphic any more. However, this almost does not happen under the realistic setting in simulations in Section 6.

4.3.2 Prefiltering Based on the Minimum Spanning Tree

Suppose that there are n sensors in the virtual topology $H_v = (V_v, M_v)$ where V_v is the vertex set and M_v is the $n \times n$ adjacency matrix of the virtual topology. Prefiltering based on the Minimum Spanning Tree consists of three steps: 1) The first step identifies the first $n-1$ edges of the virtual graph. 2) The second step identifies the remaining edge candidates of the virtual graph. 3) The third step filters out the path estimates among the edge candidates of the virtual graph.

Step 1. We select $n-1$ edges from M_v that make a Minimum Spanning Tree (MST) for the virtual topology by using a Minimum Spanning Tree algorithm, such as *Prim's algorithm* [27]. We can prove that the $n-1$ edges that form the MST are definitely edge estimates as follows:

Let $M_v(i,j)$ be the entry of matrix M_v where i is the row index and j is the column index.

- *Case 1.* The smallest entry must be an edge because the path length is the sum of more than one edge length.
- *Case 2.* Suppose we have found m edges, where $1 \leq m < n-1$. Let N be a set of the corresponding nodes of the m edges. We then choose the smallest entry $M_v(i,j)$ that satisfies $i \notin N$, and $j \in N$. $M_v(i,j)$ must be an edge by the following reason: If $M_v(i,j)$ is not an edge, another node k must exist such that $M_v(i,j) = M_v(i,k) + M_v(k,j)$. 1) If $k \in N$, then $M_v(i,k) < M_v(i,j)$, which contradicts our assumption that $M_v(i,j)$ is the smallest entry. 2) If $k \notin N$, then $M_v(k,j) < M_v(i,j)$, which also contradicts our assumption that $M_v(i,j)$ is the smallest.

Step 2. We find all of the other edge candidates of the virtual graph $G_v = (V_v, E_v)$ from the virtual topology $H_v = (V_v, M_v)$, as shown in Fig. 1c. First, the edge set E_v is initialized to have $n-1$ edges obtained by the previous step. Then, with E_v, the shortest path matrix M'_v is computed for the shortest path between an arbitrary pair of nodes. We use the fact that $M'_v(i,j) \geq M_v(i,j)$. For an arbitrary pair of nodes i and j, $M'_v(i,j)$ is the shortest path

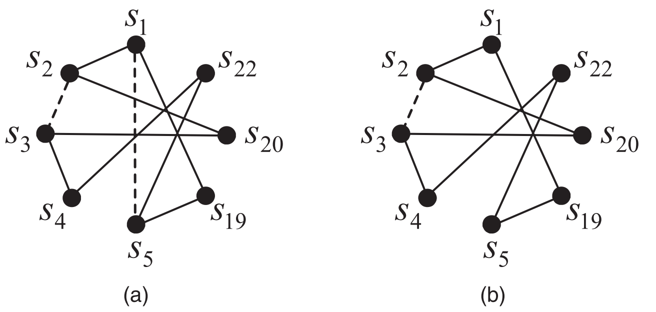


Fig. 10. Procedure of filtering out path estimates. (a) Virtual Graph after Step 2. (b) Virtual Graph after filtering edge $e_{1,5}$.

created only by $n-1$ edges, while $M_v(i,j)$ is the one created from more edges; that is, $M_v(i,j)$ might be shorter than $M'_v(i,j)$. In Theorem A.1 in Appendix A, we prove that $M_v(i,j)$ must be an edge estimate if it is the smallest one among all of the entries in M_v that satisfies $M_v(i,j) < M'_v(i,j)$, since there is no entry with a large error after the previous filtering based on the *relative deviation error*. Consequently, $M_v(i,j)$ is the nth edge estimate. We update the edge set E_v for M'_v by adding this new edge to E_v, and then recompute the matrix M'_v using this new E_v. We repeat this process until M'_v and M_v are exactly the same. In this way, we can find out edge candidates for the other edges of E_v from M_v.

Step 3. We filter out the path estimates among the edge candidates of the virtual graph, which are not filtered out from the prefiltering based on Relative Deviation Error in Section 4.3.1 and the prefiltering of Steps 1 and 2. Fig. 10 shows the procedure of filtering out path estimates from the virtual graph after Step 2. In this figure, edges $e_{1,5}$ and $e_{2,3}$ are path estimates in the sensor network as shown in Fig. 9a. The idea of filtering these path estimates is to check whether there exists a path consisting of shorter edges than a path estimate or not. If there exists such a path, the path estimate can be deleted from the virtual graph. Otherwise, it remains in the virtual graph. For example, for edge $e_{1,5}$ in Fig. 10a, there exists a path consisting of two edges $e_{1,19}$ and $e_{19,5}$ whose lengths are shorter than $e_{1,5}$. Thus, the edge $e_{1,5}$ can be deleted from the virtual graph. This checking can be performed with the shortest path algorithm (e.g., *Floyd-Warshall algorithm*) after deleting the edge $e_{1,5}$ from the virtual graph. For edge $e_{2,3}$ in Fig. 10b, the same procedure can be performed.

Note that an edge candidate may be a real edge in the sensor network. In this case, there exists an ambiguity whether this edge is a real edge or a path estimate. Thus, in order to perform the graph matching correctly in Section 4.4, it is needed to delete these ambiguous edges that have alternative paths consisting of other shorter edges from the real graph corresponding to the road network (e.g., Fig. 9a).

4.4 Step 4: Graph Matching

In this section, we explain how to construct a reduced virtual graph from the virtual graph constructed by the prefiltering in Section 4.3, and then how to match the reduced virtual graph and the real graph that are *isomorphic* to each other [26].

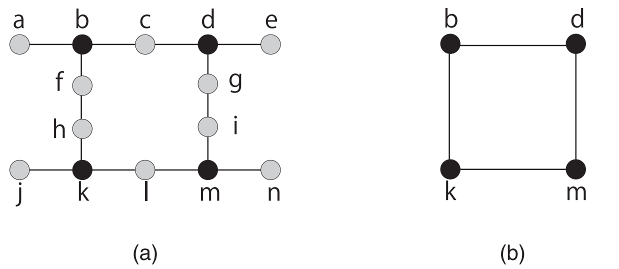


Fig. 11. Construction of reduced virtual graph. (a) Virtual Graph with four Intersection Nodes $\{b, d, k, m\}$. (b) Reduced Virtual Graph after deleting Nonintersection Nodes.

4.4.1 Construction of Reduced Virtual Graph

Let $G_v = (V_v, E_v)$ be a virtual graph. In order to perform isomorphic graph matching, two graphs should be isomorphic. Since the virtual graph G_v returned from the prefiltering module has more vertices and edges than the real graph G_r, we cannot perform isomorphic graph matching directly. From the observation that each intersection node has at least three neighboring sensors, a reduced virtual graph $\tilde{G}_v = (\tilde{V}_v, \tilde{E}_v)$ is made from the virtual graph G_v as follows:

Let N be a set of nonintersection nodes of G_v. Let $d_{G_v}(u)$ be the degree of u in the graph G_v. Let e_{uv} be the edge whose endpoints are u and v for $u, v \in V_v$. Let $l(e)$ be the length of the edge $e \in E_v$. We perform the following for all $u \in N$:

- If $d_{G_v}(u) = 1$, then delete u from G_v and delete an edge whose one endpoint is u from G_v.
- If $d_{G_v}(u) = 2$, then delete u from G_v, merge the two edges e_{ux} and e_{uy}, whose one endpoint is u, into one edge e_{xy}. The length of the edge e_{xy} is set to $l(e_{ux}) + l(e_{uy})$.

For example, Fig. 11 shows the construction of a reduced virtual graph from a virtual graph in which a set of intersection nodes is {b,d,k,m} and a set of nonintersection nodes is {a,c,e,f,g,h,i,j,l,n}. After removing nonintersection nodes and dealing with the corresponding edges, the final reduced virtual graph consists of four intersection nodes b, d, k, and m, as shown in Fig. 11b.

Finally, we can perform the graph matching between the sensor network and the road network since the reduced virtual graph is isomorphic to the real graph; that is, two graphs have the same structure for one-to-one mapping. We will explain how to perform the graph matching in the next section.

4.4.2 Weighted Graph Matching

Since the reduced virtual graph's $\tilde{E}_v$ and the real graph's E_r are isomorphic, our graph matching can be defined as searching for the $n \times n$ permutation matrix P to satisfy the following, in which P is the row permutation matrix, and P^T is the column permutation matrix:

$$\Phi(P) = \|E_r - P\tilde{E}_v P^T\|_2^2, \tag{5}$$

$$\leftarrow \arg\min_{\hat{P}} \Phi(\hat{P}), \tag{6}$$

$$\hat{E}_v \leftarrow P\tilde{E}_v P^T. \tag{7}$$

Let P be an $n \times n$ optimal permutation matrix of (6) in terms of the minimum estimation error. The result $\hat{E}_v$ of (7) is a matrix isomorphic to E_r where indices in both matrices indicate the node identifiers; that is, the sensor ID in $\tilde{E}_v$ corresponds to the intersection ID in E_r for $i = 1, \dots, n$. This optimization problem is called the Weighted Graph Matching Problem (WGMP). In order to get the exact solution P, allowing the global minimum of $\Phi(P)$, all of the possible cases should be checked. Since this is a purely combinatorial problem, the algorithm based on combination has the time complexity of $O(n!)$ for n nodes. Consequently, this is an unfeasible approach in reality. We need to use approximate approaches to give an accurate permutation matrix P, such as an eigen-decomposition approach to WGMP [23], known as an optimal approach. For our graph matching purpose, we adopt the eigen-decomposition approach that has polynomial time complexity.

We investigated the effect of the real vehicle mean speed different from the speed limit on roadways. The conclusion is that as long as all of the road segments have the same constant scaling factor for their mean speeds, our localization algorithm works well regardless of the distribution of the vehicle mean speed during traffic measurement! In other words, our algorithm works even though the actual speeds are unknown. In the case where each road segment has a different scaling factor according to unbalanced congestion conditions, our algorithm does not work well. To address this issue, we suggest to conduct measurements under a light road traffic condition, such as during night. Without congestion, we expect that all of the road segments tend to have the same constant scaling factor for their mean speeds. We have detailed proof in Theorem B.1 in Appendix B.

4.5 Step 5: Node Location Identification

In this section, we explain how to identify the location of each intersection node with the permutation matrix obtained through the graph matching in Section 4.4, and then how to identify the location of each nonintersection node.

4.5.1 Localization of Intersection Nodes

We perform the identification of each intersection node's location with the permutation matrix P returned from the graph matching module. Let $\sigma(s)$ be the permutation function corresponding to the permutation matrix P such that

$$\sigma : s \in \{1, \dots, n\} \rightarrow p \in \{1, \dots, n\}, \tag{8}$$

that is, $p = \sigma(s)$ where s is the sensor ID and p is the intersection ID corresponding to s. With the permutation function in (12), we can identify the intersection ID (p) on the road map for each intersection node (s).

4.5.2 Localization of Nonintersection Nodes

In the previous section, we know the positions of the intersection nodes. Now we localize the positions of the nonintersection nodes of degree 2 or 1. For nonintersection nodes of degree 2, using E_v of the virtual graph G_v, we begin from an intersection node u, and then create a path from u to another intersection node v, that is, $u \rightarrow a_1 \rightarrow a_2 \rightarrow \cdots \rightarrow a_m \rightarrow v$. All a_i for $i = 1, \dots, m$ are nonintersection nodes whose degrees are 2. Since we have already

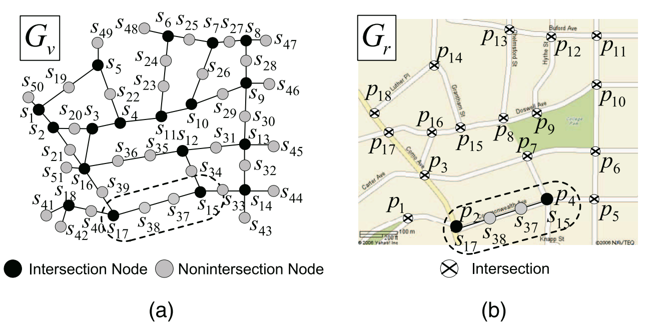


Fig. 12. Localization of nonintersection nodes. (a) Virtual Graph for Localization of Nodes. (b) Nonintersection Node Localization in Real Graph.

localized nodes u and v, and all of these a_i must be placed on the edge from u to v on the reduced virtual graph $\tilde{G}_v$, as shown Fig. 1e, we can know the positions of these a_i by looking at the length information in E_v of the virtual graph G_v, as shown in Fig. 1c. We repeat this procedure until we localize all of the nonintersection nodes in the virtual graph.

For example, Fig. 12 shows the localization of nonintersection nodes of degree 2. Two nonintersection nodes s_{37} and s_{38} are known to be located between intersection nodes s_{15} and s_{17} from the adjacency matrix E_v of the virtual graph G_v, as shown in Fig. 12a. We can identify the locations of intersection nodes s_{15} and s_{17} through the localization of intersection nodes in Section 4.5.1; the nodes s_{15} and s_{17} are placed at intersections p_4 and p_2, respectively. Thus, this intersection node localization lets us know that two nonintersection nodes s_{37} and s_{38} are sequentially placed between intersections p_4 and p_2, as shown in Fig. 12b.

For nonintersection nodes of degree 1, let w be a nonintersection node of degree 1. Since w is adjacent to an intersection node $u \in E_v$, it can be known to which intersection w is adjacent in a road network. For example, since nonintersection nodes s_{41} and s_{42} are adjacent to an intersection node s_{18}, it can be known that they are adjacent to an intersection p_1.

5 Practical Issues

In this section, we discuss the following practical issues for the deployment of our localization scheme in real road networks: 1) Time synchronization error, 2) Vehicle detection missing and duplicate vehicle detection, and 3) Intersection node missing. They are important because they affect the localization accuracy in reality. For these issues, the impacts on the localization and the solutions are discussed.

5.1 Sensor Time Synchronization Error

The inaccuracy of the time stamps should be considered, due to time synchronization errors among sensors. That is, sensor nodes might have different times at a certain level (e.g., millisecond). Let τ be the exact time. Let τ_i be the time of sensor s_i such that $\tau_i = \tau + \epsilon_i$ and ϵ_i is a uniform random variable in the interval $[-\epsilon_{max}, \epsilon_{max}]$. If the time error is small, such as c milliseconds in which c is a small constant, then the road segment length estimation through the time difference will not be affected so much. For example, suppose that the average vehicle speed is v, and some road segment's length is l. In the case of perfect time synchronization, our time difference scheme will estimate the movement time t in the road segment such that $t \approx l/v$. In the case where two adjacent sensors have time errors $-\epsilon_{max}$ and ϵ_{max}, respectively, the estimated movement time $\hat{t}$ will be approximately $t + 2\epsilon_{max}$. Consequently, if ϵ_{max} is small, then the movement time $\hat{t}$ will be close to t, leading to a reasonable estimate for the APL localization. We will show the effect of the time synchronization error in Section 6.

5.2 Vehicle Detection Missing and Duplicate Vehicle Detection

There might be vehicle detection missing or duplicate vehicle detection due to some noises. Since our algorithm uses many vehicle-detection time stamps, the missing of some vehicle detections does not affect the road segment length estimation. We will show the effect of the detection missing for the whole localization accuracy through simulation in Section 6. The conclusion for the detection missing probability is that our localization scheme has no localization error even under a reasonably high detection missing probability (e.g., 0.25) at each sensor.

Also, we will investigate the effect of the duplicate detection in Section 6. The conclusion for the duplicate detection is that our localization scheme has no localization error even under a very high duplicate detection probability (e.g., 1) at each sensor. The duplicate vehicle detection has positive effect. This is because it lets the multiple vehicle-detection time stamps registered into neighboring sensors, so it can contribute more to the detection frequency corresponding to the right road segment estimate.

5.3 Intersection Node Missing

Our prefiltering and graph matching algorithm works well when nonintersection nodes are missing or out of function in road segments. This is because our algorithm is based on intersection nodes. However, when the missing of intersection nodes exists after the sensor deployment, the reduced virtual graph will no longer be isomorphic to the real graph and so the graph matching algorithm will not work. Thus, the isomorphic graph matching cannot be performed to localize the intersection nodes in the road network.

To deal with the graph matching for this nonisomorphism between the real graph and the reduced virtual graph, the subgraph matching can be investigated [24], [25]. However, in the case of the intersection node missing, it may be very hard to perform the subgraph matching with the sensor network topology (i.e., reduced virtual graph) and the road network topology (i.e., real graph). This is because the reduced virtual graph can contain a subgraph that cannot be matched with any subgraph of the real graph for the road network. Therefore, we leave this issue as future work.

6 Performance Evaluation

As we explain in the introduction, there is no other solution appropriate to our scenario for localization in road networks. Instead of comparing our schemes with other

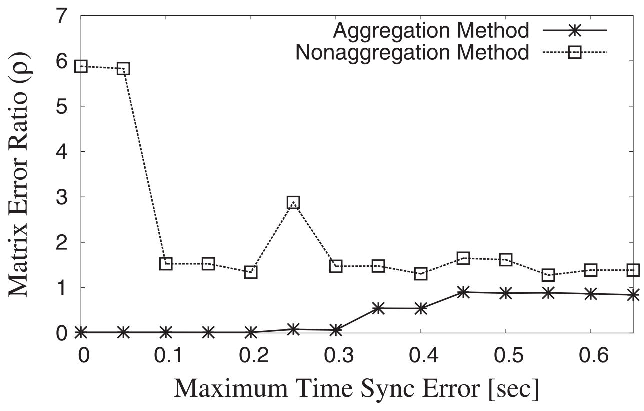


Fig. 13. Maximum Time Sync Error versus Matrix Error Ratio according to aggregation type.

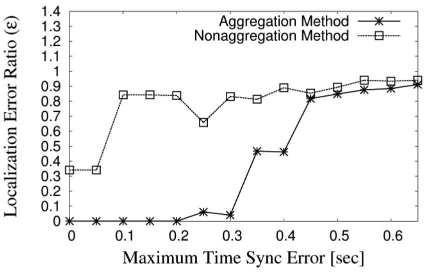
state-of-the-art schemes, we investigate the impact of the following five parameters on our localization scheme:

1. Maximum time synchronization error (ϵ_{max}),
2. Vehicle speed standard deviation (σ_v),
3. Vehicle interarrival time ($1/\lambda$),
4. Detection missing probability (α), and
5. Duplicate detection probability (β).

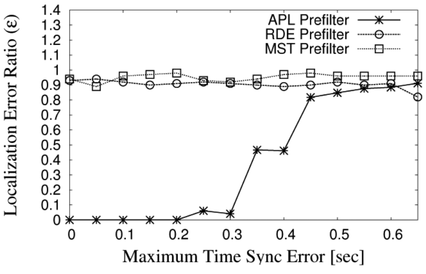
Simulation uses the map of a real road network as shown in Fig. 1a. Intersection nodes and nonintersection nodes are deployed as shown in Fig. 1a. The system parameters are selected based on a typical military scenario [37]. Unless mentioned otherwise, the default values in Table 2 are used. Our vehicle mobility model guarantees that the whole road network is covered by vehicular traffic as follows: Most of vehicles arrive at the perimeters of the road network and randomly choose one destination placed at the perimeters, moving toward the destination. When a vehicle arrives at an entrance randomly chosen, it moves toward an exit randomly chosen with a vehicle speed with a certain speed deviation described in Table 2. To reflect the realistic traffic model, the travel path is not the shortest path, but the path with some detour according to the vehicle travel path length model shown in Table 2.

To obtain high statistical confidence for the localization, road traffic is measured during the simulation time of 10 hours. From this road traffic measurement, a matrix M_v is created for the virtual topology as the average of 20 matrices M_{vs} that are adjacency matrices of the virtual topology created from the same measurement time, such as half an hour. Note that the number of 20 is determined as the sample size for the mean movement time per road segment. For example, the sample size n is 15 for 95 percent confidence interval where the movement time standard deviation σ is 10 seconds and the margin of error m is 5 seconds; $n = (1.96 * \sigma/m)^2$ [34]. We use 20 for a more accurate estimate instead of 15. Thus, M_v is the all-pairs shortest path estimation matrix for the virtual topology.

In this section, we present two kinds of performance evaluations as follows: *First*, we compare the aggregation-based estimation method with the nonaggregation-based estimation method for the estimation accuracy for road segment length in Section 6.1. *Second*, we evaluate the



(a)



(b)

Fig. 14. The impact of maximum time synchronization error (ϵ_{max}). (a) Aggregation Type versus Localization Error. (b) Prefiltering Type versus Localization Error.

performance of prefiltering types that use the aggregation-based estimation method and the same graph matching algorithm in Section 6.2.

6.1 Performance Comparison between Road Segment Estimation Methods

We compare the performance of localization schemes according to the following two road segment estimation methods:

1. Aggregation Method, and
2. Nonaggregation Method.

We define two performance metrics as follow: First, for the estimation accuracy comparison, the *Matrix Error Ratio (ρ)* is defined as the ratio of the sum of the entries of the absolute difference of two matrices (i.e., E_r and E_v) to the sum of the entries of reference matrix (i.e., E_r). Second, for the localization accuracy comparison, the *Localization Error Ratio (ϵ)* is defined as the ratio of the number of incorrectly localized sensors to the number of all sensors deployed on the road network. After the road segment estimation based on TDOD, we perform the prefiltering algorithm described in Section 4.3 and the graph matching algorithm described in Section 4.4 in order to evaluate the *Matrix Error Ratio* and *Localization Error Ratio*.

For the maximum time synchronization error, as shown in Fig. 13, the aggregation method outperforms the nonaggregation method in that the *Matrix Error Ratio* of the aggregation method is less than that of the nonaggregation method. This is why the aggregation method can give more accurate localization than the nonaggregation method, as shown in Fig. 14a. Note that the nonaggregation method is more sensitive to random noises due to time synchronization error than the aggregation method, and so such random noises make the *Matrix Error Ratio* the nonaggregation method have a random pattern according to the maximum time synchronization error, as shown in Fig. 13.

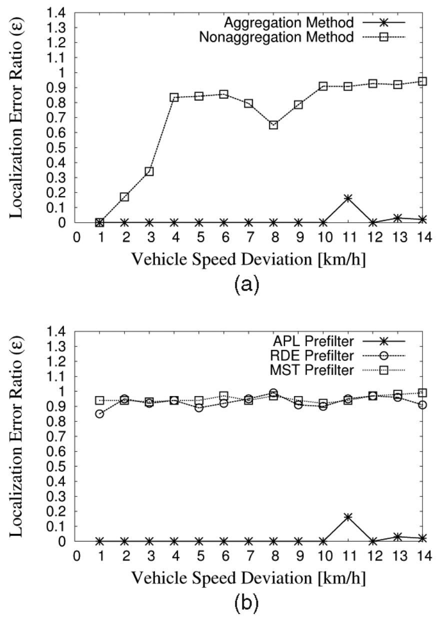


Fig. 15. The impact of vehicle speed deviation (σ_v). (a) Aggregation Type versus Localization Error. (b) Prefiltering Type versus Localization Error.

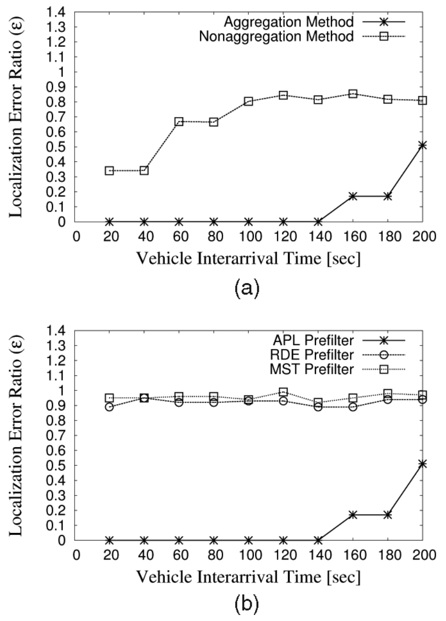


Fig. 16. The impact of vehicle interarrival time ($1/\lambda$). (a) Aggregation Type versus Localization Error. (b) Prefiltering Type versus Localization Error.

From Fig. 14a, it can be seen that our localization works well in the case in which the maximum time synchronization error is less than 0.25 seconds. We can claim that our localization scheme can work in the real environment, since the state-of-the-art time synchronization protocol for sparse wireless sensor networks can give the accuracy at the millisecond level [32].

For the vehicle speed deviation, as shown in Fig. 15a, the aggregation method can outperform the nonaggregation method in terms of localization error ratio; note that the aggregation method has no localization error up to the vehicle speed deviation of 10 km/h. This speed deviation greater than 10 km/h is the value out of the operational region for our localization scheme, so the greater speed deviation leads to the higher localization error ratio. However, considering the real statistics [33] that the vehicle speed deviation on four-lane roadways is 9.98 km/h, and the vehicle speed deviation on two-lane roadways is 8.69 km/h, it can be claimed that our localization can work well in the real environment, since our localization scheme works with the vehicle speed deviation less than 11 km/h.

For the vehicle interarrival time, as shown Fig. 16a, we see that it does not affect the performance of our localization scheme up to 140 seconds. The reason is that our TDOD operation based on the aggregation method can give accurate estimates for road segment lengths, as long as the vehicle interarrival time is long enough to allow road traffic to cover all of the road segments; note that when the vehicle interarrival time is so short (e.g., 1 second), the time stamps among sensors lose the correlation, so it is difficult to estimate the distance among them through TDOD operation. However, in fact, most people drive their vehicles with the interarrival time longer than one second for their safety, so we can claim that our localization works under normal driving condition.

6.2 Performance Comparison among Prefiltering Types

We compare the performance of localization schemes, according to the following three prefiltering types:

1. *RDE Prefilter.* Prefiltering based on the Relative Deviation Error described in Section 4.3.1,
2. *MST Prefilter.* Prefiltering based on the Minimum Spanning Tree described in Section 4.3.2, and
3. *APL Prefilter.* Prefiltering based on both the Relative Deviation Error and the Minimum Spanning Tree.

Each prefiltering type uses a matrix M_v created by the aggregation-based road segment method. After the prefiltering step and the construction step of a reduced virtual graph $\tilde{G}_v = (\tilde{V}_v, \tilde{E}_v)$, the same graph matching algorithm described in Section 4.4 is applied to the output matrix $\tilde{E}_v$ in order to evaluate the *Localization Error Ratio*.

From Figs. 14b and 15b, our localization with *APL Prefilter* works well under reasonable, real environments in which the maximum time synchronization error is less than 0.25 sec and the vehicle speed deviation is less than 11 km/h. As we can see in both figures, one missing of *RDE Prefilter* and *MST Prefilter* cannot allow the accurate localization under the reasonable, real environment. This is because the missing of either *RDE Prefilter* or *MST Prefilter* cannot produce a correct sensor network graph (i.e., reduced virtual graph) for the graph matching with the road network graph (i.e., real graph). As a result, we need to use the combination of these two prefilters for filtering out path estimates.

For the vehicle interarrival time, as shown in Fig. 16b, *APL Prefilter* has no localization error in the whole interval from 20 to 140 seconds. However, as shown in the figure, one missing out of two Prefiters makes a lot of localization error in the same reason with the maximum time synchronization error and vehicle speed deviation in Figs. 14b and 15b.

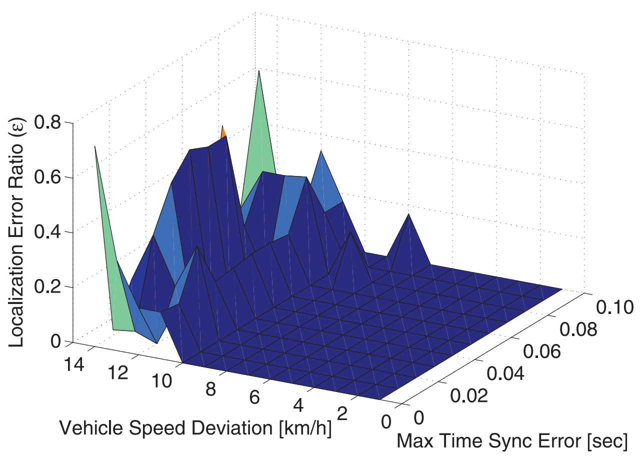


Fig. 17. APL operational region for vehicle speed deviation and maximum time synchronization error.

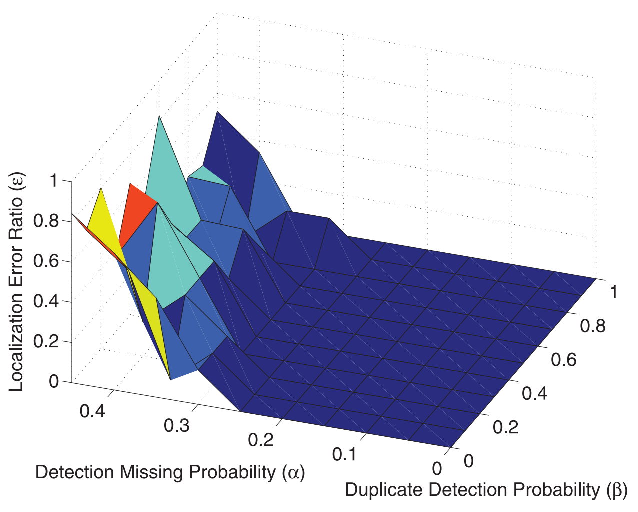


Fig. 18. APL operational region for detection missing probability and duplicate detection probability.

6.3 APL Operational Region

We evaluate *APL* to see what range of time synchronization error and vehicle speed deviation it works well in. Fig. 17 shows the *APL operational region* that contains the range of the maximum time synchronization error and the vehicle standard deviation to allow a perfect localization under the simulation environment given in Table 2. Our localization scheme works well in the case in which the vehicle standard deviation is less than 10 km/h, regardless of the maximum time synchronization error from 0 to 0.07 sec. This threshold for the vehicle standard deviation is close to the real statistics of the vehicle speed deviation (e.g., 9.98 km/h for four-lane roadways) [33]. For the vehicle interarrival time, our localization works well as long as the vehicular traffic density (e.g., interarrival time less than 140 sec) allows the road network to be fully covered by vehicles for the distance estimation. Thus, the vehicle speed deviation is one dominant factor of the performance in APL.

Also, we investigated what effects the detection missing and the duplicate detection have for the whole localization accuracy by modeling the detection missing event and the duplicate detection event as *Bernoulli trial*. Fig. 18 shows the *APL operational region* that contains the range of the detection missing probability (α) and the duplicate detection probability (β) to allow a perfect localization under the simulation environment given in Table 2. The result is that our localization scheme has no localization error under the simulation setting in Table 2 with the detection missing probability α from 0 to 0.25 at each sensor and with the duplicate detection probability β from 0 to 1 at each sensor, respectively. Thus, it can be claimed that our localization scheme can work in the real road networks with noises and the detection missing probability is another dominant factor in APL.

7 Conclusion

In sparse road sensor networks, sensors cannot effectively obtain pairwise ranging distance or connectivity information for the purpose of localization. To address this issue, this work introduces an autonomous passive localization scheme, called *APL*, using only binary sensors. Our APL system performs a localization using vehicle-detection time stamps along with the road map of the target area. The key idea is to use the statistics of vehicle-detection time stamps to obtain distance estimates between any pair of sensors on roadways to construct a virtual graph, which is then matched with the topology of road map, in order to identify where sensors are located in the target road network. We observe that path estimates are less accurate than edge estimates, therefore, it is necessary to conduct prefiltering before graph matching can be successfully conducted. Through both the outdoor test and simulation, we show that our localization scheme is robust to realistic setting, considering the time synchronization error, vehicle speed deviation, and different vehicular traffic intensity along with sensor detection missing probability and duplicate detection probability. Our APL system shows the encouraging results in the localization for the road network fully covered by vehicular traffic for the distance estimation along with one vehicle speed distribution. As future work, we will investigate the handling of intersection node missing in the case where some sensors are missing at intersections in the road network.

Appendix A

Valid Edge Selection Based on Minimum Spanning Tree

Theorem A.1. *Suppose that M is an $n \times n$ matrix, and $M(i,j)$ is the shortest path length from node i to node j in a graph G that has n nodes. Let A be the set of all edges in graph G. Suppose that M' is another $n \times n$ matrix, and $M'(i,j)$ is the shortest path length from node i to node j in another graph G' that has n nodes such that $G' \subset G$. Let A' be the set of all edges in graph G' such that $A' \subset A$. If $M(i,j)$ is the smallest entry that satisfies $M(i,j) < M'(i,j)$ such that $M(i,j) \in A$ and $M(i,j) \notin A'$, then $M(i,j)$ must be* ***an edge length****.*

Proof. We prove the claim using contradiction. Let $G = (V,E)$ such that V is the node set of G, and E is the edge set of G. Let e_{uv} be an edge whose endpoints are u and v for $u, v \in V(G)$. Let e_k be the kth shortest edge in the edge set A where $A = E(G)$ in terms of edge length. Let $l(e_i)$ be the length of edge e_i.

Suppose that $M(i,j)$ is the length of the shortest path between nodes i and j rather than the length of the edge between nodes i and j. It must be the sum of more than one edge length in A as follows:

$$M(i,j) = \sum_{k=1}^{m} l(e_k),\ e_k \in A. \tag{9}$$

Similarly, $M'(i,j)$ must be the sum of more than one edge length in A'. We know that $M(i,j) \le M'(i,j)$ for all $i,j \in V(G)$, since $A' \subset A$. From the given condition $M(i,j) < M'(i,j)$, the following two cases for e_k are considered:

The first case. If there exists an e_k for $k = 1,\dots,m$ in (9) such that 1) $e_k \notin A'$ and 2) the entries $M(u,v)$ and $M'(u,v)$ corresponding to the endpoints of the edge $e_k = e_{uv}$ satisfy $M(u,v) < M'(u,v)$, then $M(u,v) < M(i,j)$. This contradicts the minimality of $M(i,j)$ such that $M(i,j) \in A$ and $M(i,j) \notin A'$.

The second case. If there exists no e_k for $k = 1,\dots,m$ in (9) such that 1) $e_k \notin A'$ and 2) the entries $M(u,v)$ and $M'(u,v)$ corresponding to the endpoints of the edge $e_k = e_{uv}$ satisfy $M(u,v) < M'(u,v)$, then this means that A' has the same edges e_k for $k = 1,\dots,m$ as A has. Thus, $M(i,j) = M'(i,j)$, since A' can construct the shortest i,j-path with the same edges as A has. This contradicts the condition $M(i,j) < M'(i,j)$.

From these two cases, it is concluded that $M(i,j)$ must be **an edge length**. □

Appendix B

Graph Matching Independent of Vehicle Mean Speed

Theorem B.1. *Let P, E_r, and $\tilde{E}_v$ be $n \times n$ real matrices. If P is an $n \times n$ optimal permutation that minimizes the following 2-norm square:*

$$P = arg \min_{\hat{P}} \|E_r - P\hat{E}_v P^T\|_2^2, \tag{10}$$

then, P is also an $n \times n$ optimal permutation that minimizes the following 2-norm square:

$$P = arg \min_{\hat{P}} \|E_r - Pc\hat{E}_v P^T\|_2^2, \forall c \in \mathbb{R}^+. \tag{11}$$

Proof. Let $E_r = (r_{ij})$ and $\tilde{E}_v = (v_{ij})$ for $1 \le i,j \le n$. Let the permutation function $\sigma(x)$ be a map corresponding to the optimal permutation matrix P

$$\sigma : x \in \{1,\dots,n\} \rightarrow y \in \{1,\dots,n\}, \tag{12}$$

that is, $y = \sigma(x)$. Thus, the 2-norm square in (10) can be represented using the summation and permutation function as follows:

$$\Phi(P, E_r, \hat{E}_v) = \|E_r - P\hat{E}_v P^T\|_2^2 = \sum_{i=1}^{n}\sum_{j=1}^{n}(r_{ij} - v_{\sigma(i)\sigma(j)})^2. \tag{13}$$

Also, the 2-norm square in (11) can be represented as follows:

$$\begin{aligned}\Phi(P, E_r, c\hat{E}_v) &= \|E_r - Pc\hat{E}_v P^T\|_2^2 \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n}(r_{ij} - cv_{\sigma(i)\sigma(j)})^2.\end{aligned} \tag{14}$$

Let $\bar{\sigma}(x)$ be the arbitrary permutation function corresponding to an arbitrary permutation matrix $\bar{P}$. Since P is an optimal permutation, the following inequality always holds:

$$\begin{aligned}&\Phi(P, E_r, \hat{E}_v) - \Phi(\bar{P}, E_r, \hat{E}_v) \le 0, \\ &\Rightarrow \sum_{i=1}^{n}\sum_{j=1}^{n}(r_{ij} - v_{\sigma(i)\sigma(j)})^2 - \sum_{i=1}^{n}\sum_{j=1}^{n}(r_{ij} - v_{\bar{\sigma}(i)\bar{\sigma}(j)})^2 \le 0, \\ &\Rightarrow \sum_{i=1}^{n}\sum_{j=1}^{n}(-2r_{ij}v_{\sigma(i)\sigma(j)} + 2r_{ij}v_{\bar{\sigma}(i)\bar{\sigma}(j)}) \le 0.\end{aligned} \tag{15}$$

In the same way, from (14), if we take the difference between two 2-norm squares for P and $\bar{P}$, then P is also an optimal permutation matrix of (14) due to (15) as follows:

$$\begin{aligned}&\Phi(P, E_r, c\hat{E}_v) - \Phi(\bar{P}, E_r, c\hat{E}_v) \\ &= \sum_{i,j}^{n}(r_{ij} - cv_{\sigma(i)\sigma(j)})^2 - \sum_{i,j}^{n}(r_{ij} - cv_{\bar{\sigma}(i)\bar{\sigma}(j)})^2 \\ &= c\sum_{i,j}^{n}(-2r_{ij}v_{\sigma(i)\sigma(j)} + 2r_{ij}v_{\bar{\sigma}(i)\bar{\sigma}(j)}) \le 0,\ \forall c \in \mathbb{R}^+.\end{aligned} \tag{16}$$

From Theorem B.1, as long as all of the road segments have the same constant scaling factor c for their mean speeds, our localization algorithm works well regardless of the distribution of the vehicle mean speed during traffic measurement. □

Acknowledgments

This research is supported in part by the US National Science Foundation (NSF) grants CNS-0917097/0845994/0720465/1016350. The authors also receive the facility support from MSI and DTC at the University of Minnesota.

References

[1] B.H. Wellenhoff, H. Lichtenegger, and J. Collins, *Global Positions System: Theory and Practice*, fourth ed., Springer Verlag, 1997.

[2] N.B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The Cricket Location-Support System," *Proc. ACM MobiCom*, Aug. 2000.

[3] D. Niculescu and B. Nath, "Ad Hoc Positioning System (APS) Using AOA," *Proc. IEEE INFOCOM*, Mar. 2003.

[4] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-Less Low Cost Outdoor Localization for Very Small Devices," *IEEE Personal Comm. Magazine*, vol. 7, no. 5, pp. 28-34, Oct. 2000.

[5] T. He, C. Huang, B.M. Blum, J.A. Stankovic, and T. Abdelzaher, "Range-Free Localization Schemes in Large-Scale Sensor Networks," *Proc. ACM MobiCom*, Sept. 2003.

[6] L. Lazos and R. Poovendran, "SeRLoc: Secure Range-Independent Localization for Wireless Sensor Networks," *Proc. Third ACM Workshop Wireless Security (WiSe)*, Oct. 2004.

[7] D. Moore, J. Leonard, D. Rus, and S. Teller, "Robust Distributed Network Localization with Noise Range Measurements," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems (SenSys)*, Nov. 2004.

[8] S. Park, S.G. Ritchie, and W. Cheng, "Automated Real-Time Vehicle Classifier Development Based on Vehicle Signature," *Proc. IEEE Int'l Conf. Granular Computing (GrC)*, Aug. 2008.

[9] J. Jeong, S. Guo, T. He, and D. Du, "APL: Autonomous Passive Localization for Wireless Sensors Deployed in Road Networks," *Proc. IEEE INFOCOM,* Apr. 2008.

[10] A. Savvides, C.C. Han, and M.B. Srivastava, "Dynamic Fine-Grained Localization in Ad-Hoc Networks of Sensors," *Proc. ACM MobiCom,* July 2001.

[11] P. Bahl and V.N. Padmanabhan, "RADAR: An In-Building RF-Based User Location and Tracking System," *Proc. IEEE INFOCOM,* Mar. 2000.

[12] M. Youssef, M. Mah, and A.K. Agrawala, "Challenges: Device-Free Passive Localization for Wireless Environments," *Proc. ACM MobiCom,* Sept. 2007.

[13] G. Zhou, T. He, S. Krishnamurthy, and J.A. Stankovic, "Impact of Radio Irregularity on Wireless Sensor Networks," *Proc. ACM MobiSys,* June 2004.

[14] T. He, C. Huang, B.M. Blum, J.A. Stankovic, and T. Abdelzaher, "Range-Free Localization Schemes for Large Scale Sensor Networks," *Proc. ACM MobiCom,* Sept. 2003.

[15] C. Taylor and A.R.J. Bachrach, "Simultaneous Localization, Calibration, and Tracking in an ad Hoc Sensor Network," *Proc. Fifth IEEE/ACM Int'l Conf. Information Processing in Sensor Networks (IPSN),* Apr. 2006.

[16] N.B. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller, "Anchor-Free Distributed Localization in Sensor Networks," MIT Technical Report no. 892, Apr. 2003.

[17] M. Li and Y. Liu, "Rendered Path: Range-Free Localization in Anisotropic Sensor Networks with Holes," *Proc. ACM MobiCom,* Sept. 2007.

[18] Z. Zhong and T. He, "MSP: Multi-Sequence Positioning of Wireless Sensor Nodes," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems (SenSys),* Nov. 2007.

[19] Z. Zhong, D. Wang, and T. He, "Sensor Node Localization Using Uncontrolled Events," *Proc. 28th IEEE Int'l Conf. Distributed Computing Systems (ICDCS),* June 2008.

[20] R. Stoleru, T. He, J.A. Stankovic, and D. Luebke, "A High-Accuracy, Low-Cost Localization System for Wireless Sensor Networks," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems (SenSys),* Nov. 2005.

[21] R. Stoleru, P. Vicaire, T. He, and J.A. Stankovic, "StarDust: A Flexible Architecture for Passive Localization in Wireless Sensor Networks," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems (SenSys),* Nov. 2006.

[22] K. Römer, "The Lighthouse Location System for Smart Dust," *Proc. ACM MobiSys,* May 2003.

[23] S. Umeyama, "An Eigendecomposition Approach to Weighted Graph Matching Problems," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 10, no. 5, pp. 695-703, Sept. 1988.

[24] L.P. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 26, no. 10, pp. 1367-1372, Oct. 2004.

[25] J.R. Ullmann, "An Algorithm for Subgraph Isomorphism," *J. Assoc. for Computing Machinery,* vol. 23, pp. 31-42, 1976.

[26] D.B. West, *Introduction to Graph Theory,* second ed., Prentice Hall, 2000.

[27] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms,* second ed., MIT Press, 2003.

[28] L. Gu et al., "Lightweight Detection and Classification for Wireless Sensor Networks in Realistic Environments," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems (SenSys),* Nov. 2005.

[29] C.-J. Jiang, C. Chen, J.-W. Chang, R.-H. Jan, and T.C. Chiang, "Construct Small Worlds in Wireless Networks Using Data Mules," *Proc. IEEE Int'l Conf. Sensor Networks, Ubiquitous and Trustworthy Computing (SUTC),* June 2008.

[30] J. Zhao and G. Cao, "VADD: Vehicle-Assisted Data Delivery in Vehicular Ad Hoc Networks," *IEEE Trans. Vehicular Technology,* vol. 57, no. 3, pp. 1910-1922, May 2008.

[31] J. Jeong, S. Guo, Y. Gu, T. He, and D. Du, "TBD: Trajectory-Based Data Forwarding for Light-Traffic Vehicular Networks," *Proc. 29th IEEE Int'l Conf. Distributed Computing Systems (ICDCS),* June 2009.

[32] K. Römer, "Time Synchronization in Ad Hoc Networks," *Proc. ACM MobiHoc,* Oct. 2001.

[33] V. Muchuruza and R. Mussa, "Traffic Operation and Safety Analyses of Minimum Speed Limits on Florida Rural Interstate Highways," *Proc. Mid-Continent Transportation Research Symp.,* Aug. 2005.

[34] D.S. Moore and G.P. McCabe, *Introduction to the Practice of Statistics,* fifth ed., Freeman, 2006.

[35] N. Wisitpongphan, F. Bai, P. Mudalige, and O.K. Tonguz, "On the Routing Problem in Disconnected Vehicular Ad Hoc Networks," *Proc. IEEE INFOCOM Mini-Symposia,* May 2007.

[36] M. DeGroot and M. Schervish, *Probability and Statistics,* third ed., Addison-Wesley, 2001.

[37] T. He et al., "An Energy-Efficient Surveillance System Using Wireless Sensor Networks," *Proc. ACM MobiSys,* Jun. 2004.

Jaehoon (Paul) Jeong received the PhD degree under Professor David H.C. Du and Professor Tian He from the Department of Computer Science and Engineering at the University of Minnesota in 2009. He received the BS degree from the Department of Information Engineering at Sungkyunkwan University in Korea and the MS degree from the School of Computer Science and Engineering at Seoul National University in Korea, in 1999 and 2001, respectively. He is a member of the IEEE.

Shuo Guo received the BS in electronic engineering at Tsinghua University in 2006 and is currently a PhD candidate in the Department of Electrical and Computer Engineering at the University of Minnesota, Twin Cities. Her research includes wireless sensor networks, vehicular ad-hoc networks, and real-time and embedded systems. She received a best paper award at IEEE MASS 2008. She is a student member of the IEEE.

Tian He is currently an associate professor in the Department of Computer Science and Engineering at the University of Minnesota-Twin City. He received the PhD degree under Professor John A. Stankovic from the University of Virginia in 2004. Dr. He is the author and coauthor of over 90 papers in premier sensor network journals and conferences with over 4000 citations. His publications have been selected as graduate-level course materials by over 50 universities in the United States and other countries. He is a member of the IEEE.

David H.C. Du received the BS degree in mathematics from National Tsing-Hua University, Taiwan, R.O.C. in 1974, and the MS and PhD degrees in computer science from the University of Washington, Seattle, in 1980 and 1981, respectively. He is currently the Qwest Chair professor at the Computer Science and Engineering Department, University of Minnesota, Minneapolis. His research interests include cyber security, sensor networks, multimedia computing, storage systems, and high-speed networking. He is a fellow of the IEEE.