Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 07-016

APL: Autonomous Passive Localization for Wireless Sensors Deployed in Road Networks

Jaehoon Jeong, Shuo Guo, Tian He, and David Du

July 02, 2007

APL: Autonomous Passive Localization for Wireless Sensors deployed in Road Networks

Jaehoon Jeong, Shuo Guo†, Tian He and David H.C. Du
Department of Computer Science and Engineering, University of Minnesota
†Department of Electrical and Computer Engineering, University of Minnesota
jjeong@cs.umn.edu, guoxx080@umn.edu, {tianhe,du}@cs.umn.edu

Abstract

In road networks, sensor nodes are deployed sparsely (hundreds of meters apart) to save costs. This makes the existing localization solutions based on the ranging ineffective. To address this issue, this paper introduces an autonomous passive localization scheme, called APL. Our work is inspired by the fact that vehicles move along routes with a known map. Using vehicle-detection timestamps, 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 road map, in order to identify where sensors are located in roadways. We evaluate our design in local roadway and show that our distance estimate method works well. In addition, we show that our localization scheme is effective in a road network with eighteen intersections, where we found no location matching error, even with a maximum sensor time synchronization error of 0.3[sec] and the vehicle speed deviation of 10[km/h].

Categories and Subject Descriptors

C.2.4 [**Computer Communication Networks**]: Distributed Systems; C.3 [**Special Purpose and Application Based Systems**]: Real-time and embedded systems

General Terms

Algorithms, Design, Architecture, Measurement, Experimentation

Keywords

Wireless Sensor Networks, Localization

1 Introduction

Wireless sensor networks have been researched for a variety of military applications, such as surveillance and target tracking. In particular, we have interest in the localization of wireless sensors for military applications in road networks around a target area. We consider a scenario that unmanned aerial vehicles drop a large number of wireless sensors into road networks around a target area, as shown in Figure 1. We need only sensors on the road to work for military applications in road networks within the target area. On the other hand, sensors out of the road are regarded as unimportant or useless, since they cannot perform surveillance for roadways. In this scenario, our problem is how sensors on a road can recognize their positions on the road network, which is referred to as the localization of sensors. Localization is a prerequisite step for most military applications, including surveillance and target tracking, since the location of sensor reporting is something that should be identified for the sensor's report to be useful.

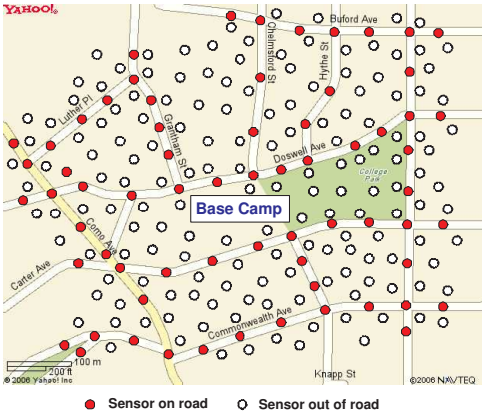


Figure 1. Localization Scenario: Wireless sensors are dropped by unmanned aerial vehicles for the surveillance of road networks around a target area (i.e., Base Camp).

Road networks have unique characteristics different from two-dimensional open fields in that most legacy localization schemes have focused on the following premises: (a) There exist vehicles moving on roadways; and (b) Roadways are routes on which vehicles move. Also, since we consider ad-hoc, military scenarios on road networks, we throw a large number of wireless sensors in the sky using airplanes for the deployment of sensors on road networks, so the cost per sensor should be considered. For road networks having these unique characteristics, legacy localization schemes have limitations. First of all, we can categorize legacy schemes into three kinds as follows: (a) Range-based schemes (e.g., TOA and TDOA [2, 25]); (b) Range-free schemes (e.g., APIT); and (c) Artificial event-based schemes (e.g., StarDust and Spotlight [21, 22]). First, for the TOA scheme in [25], wireless sensors need GPS devices that are costly and require additional energy consumption. This approach will increase the cost of sensor deployment and the energy consumption rate of sensors due to GPS devices. The TDOA scheme in [2] has the limitation of requiring expensive ranging devices and assumes that sensors are closely placed for ranging based on electrical waves or ultrasound. For example, in the TDOA scheme based on ultrasound, ranging devices cannot be used for the localization of sensors placed far away on the roadway, since ultrasound signals usually propagate only 20-30 feet. In order for the TDOA scheme based on ultrasound to work on roadways, it will require a dense deployment of sensors, thus leading to high-cost deployment. Second, range-free schemes (e.g., APIT [7]) are based on anchor nodes with GPS devices for self-localization. Non-anchor nodes without

GPS devices identify their locations using an intersection area of communication circles of neighboring anchor nodes. This approach cannot give accurate locations to non-anchor nodes on the road when the anchor nodes are placed out of the road. In order to let most roads have anchor nodes and let non-anchor nodes be surrounded by anchor nodes, a large number of anchor nodes should be dropped uniformly within a target area. Consequently, this deployment will be costly. In the same way as range-based schemes, when the spacing between the sensors is so large for communication, the beacon signal from the anchor nodes for localization cannot reach non-anchor nodes, so the range-free schemes cannot work well, either. Third, the localization schemes based on artificial event generation (e.g., StarDust and Spotlight [21, 22]) are difficult to use in large-scale road networks, since it is hard to generate artificial events in a large area. Also, in the case where road networks are large, it is very hard to let artificial events reach all sensors. Consequently, it is difficult to apply these artificial event-based schemes to localization for sensor networks deployed on roadways.

The challenge in the localization on road networks is to use the unique characteristics of road networks to cope with the limitations of legacy localization schemes. There are challenges for localization on road networks as follows:

- How can we use the vehicles moving only through roadways, as natural events for localization?

We observe that we can use vehicles moving on roadways as natural events for localization; that is, when sensors on the road detect moving vehicles, they regard vehicle detections as localization events. One challenge is how to use the timestamps of binary vehicle detection for localization [13], where binary vehicle detection indicates only vehicle detection, without any identification of the vehicle. If two sensors separated with some spacing can identify detections of the same vehicle using costly hardware for vehicle identification [16], it is easy to measure the distance between these two sensors by the difference of the timestamps corresponding to the vehicle from these two sensors, given the vehicle's speed. However, if we try to estimate the distance between the sensors without such costly vehicle identification hardware, such an estimation will be one challenge. The other challenge is how to use the fact that vehicles move only through roadways. We observe that the roadways can be regarded as the possible area where sensors can detect moving vehicles. Consequently, we can use the road map of the roadways around a target area as useful information for localization in order to identify the possible locations of the sensors that detect the moving vehicles. Thus, if with the vehicle-detection timestamps and the road map we can identify which road segments sensors are placed on a road map, along with the relative positions within the road segments, we can easily identify the locations of sensors on roadways.

In this paper, we propose an *Autonomous Passive Localization (APL)* based on the binary vehicle-detection timestamp and the road map. Our localization scheme consists of three phases: (a) the estimation of the distance between two arbitrary sensors in the same road segment; (b) the construction of the connectivity of sensors on roadways; (c) the identification of sensor locations through matching the constructed connectivity of sensors with the graph model for the road map.

Our contributions in this paper are as follows:

- A new architecture for autonomous passive localization in road networks in Section 3.1. Simple sensors on roads autonomously cooperate to localize their positions on roadways through the passive measurement of vehicle detection as a natural event.
- An estimation method for road segment distance among two arbitrary sensors in Section 3.2. We propose a simple, efficient estimation method, based on the difference of the vehicle-detection timestamps of these two sensors.
- A prefiltering algorithm for selecting only road segment distance estimates between two arbitrary sensors in the same road segment in Section 3.3. In the case where two arbitrary sensors are not in the same road segment and are located far away from each other, the distance estimate based on our estimation method might be inaccurate. The prefiltering algorithm eliminates such a distance estimate between two arbitrary sensors not in the same road segment.
- A graph-matching algorithm for matching the sensor's identification with a position at a graph model for the road map of the target area in Section 3.4. We reduce our localization problem to a traditional graph-matching problem, then use an optimal graph-matching algorithm.

The rest of this paper is organized as follows. Section 2 describes the problem formulation for our localization. Section 3 explains the system design for *Autonomous Passive Localization*. In Section 4, we discuss several considerations that affect our localization scheme in reality. Section 5 shows the performance evaluation of our algorithm. In Section 6, we explain other related work. We summarize our work with our future work in Section 7.

2 Problem Formulation

We consider a realistic model where sensors are placed at both intersection points and non-intersection points on road networks. The objective is to localize wireless sensors deployed in road networks only with a road map and binary vehicle-detection timestamps taken by sensors placed at road intersection points, as shown in Figure 2(a).

2.1 Definitions and Assumptions

We define eight terms as follows:

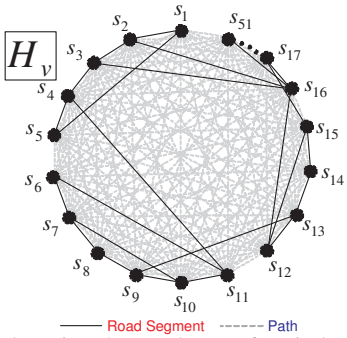
Intersection Node Let *Intersection Node* be the sensor placed at an intersection on the road and having more than two neighboring sensors (i.e., degree ≥ 3). In Figure 2(a), sensors a and c are intersection nodes.

Non-intersection Node Let *Non-intersection Node* be the sensor placed at a non-intersection and having one or two neighboring sensors. In Figure 2(a), sensors b and d are non-intersection nodes.

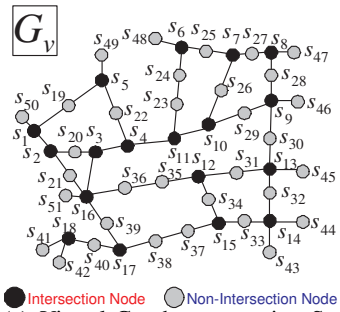
Virtual Topology Let *Virtual Topology* be $H_v = (V_v, M_v)$, where $V_v = \{s_1, s_2, ..., s_n\}$ is a set of sensors in the road network, and $M_v = [v_{ij}]$ is a matrix of path length v_{ij} for sensors s_i and s_j. Figure 2(b) shows a virtual topology of sensors to the road network, shown in Figure 2(a). M_v is a complete simple graph, since there is a path between two arbitrary sensors. We define the edge of the virtual topology as *virtual edge*. In Figure 2(b), among the virtual edges, a solid black line represents an *edge estimate* between two sensors, which means that these two sensors are adjacent on the road network. The dotted gray line represents a *path estimate* between two sensors, which means that these two sensors are not adjacent on the road network.

Virtual Graph Let *Virtual Graph* be $G_v = (V_v, E_v)$, where $V_v = \{s_1, s_2, ..., s_n\}$ 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

(a) Road Network with Wireless Sensors

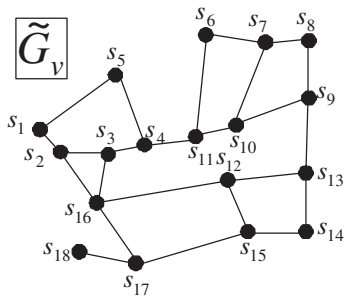


(b) Virtual Topology of Wireless Sensors: $T_v = (V_v, M_v)$

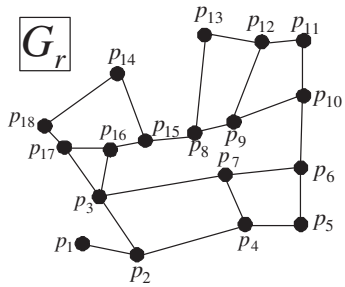


(c) Virtual Graph representing Sensor Network: $G_v = (V_v, E_v)$

(d) Road Network only with Intersection Nodes of Virtual Graph



(e) Reduced Virtual Subgraph consisting of Intersection Nodes of Virtual Graph: $\tilde{G}_v = (\tilde{V}_v, \tilde{E}_v)$



(f) Real Graph corresponding to Virtual Subgraph: $G_r = (V_r, E_r)$

Figure 2. Wireless Sensor Network deployed in Road Network

and s_j. Figure 2(c) shows a virtual graph of the sensor network deployed on the road network shown in Figure 2(a), where the black node represents an intersection node and the gray node represents a non-intersection node.

Reduced Virtual Subgraph Let *Reduced Virtual Subgraph* be $\tilde{G}_v = (\tilde{V}_v, \tilde{E}_v)$, where $\tilde{V}_v = \{s_1, s_2, ..., s_m\}$ is a set of sensors placed 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 subgraph $\tilde{G}_v$ is obtained by deleting the non-intersection nodes of the virtual graph G_v with the degree information in E_v and by connecting two edges connected to each non-intersection node when the non-intersection node has degree 2. When the non-intersection node has degree 1, it is deleted along with its edge. Refer to Section 3.4.1 for more detailed procedure. Figure 2(e) shows a reduced virtual subgraph consisting of only intersection nodes of virtual graph in Figure 2(c) and Figure 2(d) shows the road network representing only intersection nodes.

Real Graph Let *Real Graph* be $G_r = (V_r, E_r)$, where $V_r = \{p_1, p_2, ..., p_n\}$ 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. Figure 2(f) shows a real graph corresponding to the road network shown in Figure 2(d), and the real graph is isomorphic to the reduced virtual subgraph graph $\tilde{G}_v$ shown in Figure 2(e).

Shortest Path Matrix Let *Shortest Path Matrix* for graph $G = (V, E)$ be M such that $M = [m_{ij}]$ is a matrix of the shortest path length between two arbitrary nodes i and j in graph G. M is computed from E by the All-Pairs Shortest Paths algorithm, such as the *Floyd-Warshall algorithm* [4]. In this paper, 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 $G_v = (V_v, E_v)$.

APL Server Let *APL Server* be an ad-hoc server deployed in the road network that performs the localization algorithm with binary vehicle-detection timestamps collected from the sensor network.

Our assumptions are as follows:

- Sensors have simple sensing devices for binary vehicle detection without any costly ranging or GPS devices [6, 13]. Each detection consists of a sensor ID and timestamp, that is, (s_i, t_j) for $i, j \in \mathbb{N}$.
- Sensors are time-synchronized at a certain level (e.g., millisecond [ms]) [5, 12].
- 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.
- There is an ad-hoc network or a delay tolerant network for wireless sensors to deliver vehicle-detection timestamps to the *APL* server.
- The vehicle mean speed is close to speed limit assigned to roadways. However, we deal with the situation that the mean speed might shift from the speed limit according to traffic condition.
- Vehicles pass through all road segments on the target road networks and move at a certain level of speed change. That is, the standard deviation of vehicle speed is assumed to be a reasonable value, based on real road traffic statistics [14].

2.2 Main Idea

Suppose that one sensor is placed as intersection node at a unique intersection point in the road network, as shown in Figure 2(a). We construct a virtual topology, as shown in Figure 2(b), which has a distance estimate between two arbitrary sensors in roadways with detection timestamps using the distance estimation method discussed in Section 3.2. The main idea of the estimation method is based on an observation that there is the correlation between vehicle-detection timestamps from two adjacent sensors. This correlation gives a road segment length estimate. We found that in the virtual topology, as shown in Figure 2(b), edge estimates for adjacent sensors in the sensor network are accurate, which are represented as solid black lines in Figure 2(b). However, most path estimates for non-adjacent sensors are inaccurate, which are represented as dotted gray lines in Figure 2(b).

In order to make the virtual topology become a graph corresponding to the sensor network deployed on the road network, as shown in Figure 2(c), we remove path estimates from the virtual topology through the prefiltering algorithm described in Section 3.3. The main idea of the prefiltering algorithm is that path estimates have more variance than edge estimates, and path estimates are the sum of edge estimates. The refined virtual topology is called *virtual graph*, as shown in Figure 2(c), which is the subgraph of the virtual topology only with the edge estimates. The virtual graph is not isomorphic to the real graph of Figure 2(f) yet since the virtual graph has both intersection nodes and non-intersection nodes, so it has more nodes and edges than the real graph. However, the virtual graph contains a subgraph isomorphic to the real graph. We need to find this subgraph for the graph-matching with the real graph.

In order to find such a subgraph of the virtual graph isomorphic to the real graph, we remove the non-intersection nodes and deal with their edges from the virtual graph in order to make a reduced virtual subgraph only with intersection nodes corresponding to intersection points in the road network, as shown in Figure 2(e). At this point, the reduced virtual subgraph and the real graph are isomorphic, so the reduced virtual subgraph's vertices can be matched with the real graph's vertices by finding a permutation matrix to let them be isomorphic. This graph-matching for the permutation matrix belongs to a traditional graph matching problem described in Section 3.4.

With the permutation matrix to let the reduced virtual subgraph and the real graph be isomorphic, we can identify the location of each intersection node with the corresponding vertex in the real graph. As shown in Figure 2(e) and Figure 2(f), we can see that the real graph and virtual graph are isomorphic since there is a bijection between the vertices of the real graph and those of the virtual graph. For example, sensors s_1 and s_2 are matched with intersections p_{18} and p_{17}, respectively. After the localization of intersection nodes, we localize the non-intersection nodes using the the observation that they are always placed between two intersection nodes in the virtual graph, as shown in Figure 2(c). The detailed procedure of node-location identification is discussed in Section 3.5.

3 APL System Design

In this section, we explain our system architecture for autonomous passive localization, the estimation method to measure distance between two arbitrary sensors, the prefiltering algorithm to convert a virtual topology into a virtual graph, the graph-matching algorithm to find a permutation matrix letting the reduced virtual subgraph and real graph be isomorphic, and sensor location identification using the found permutation matrix.

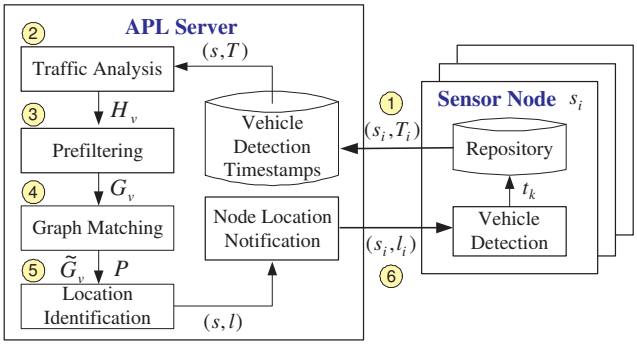


Figure 3. APL System Architecture

3.1 System Architecture

We use an asymmetric architecture for localization as in Figure 3 in order to simplify the functionality of sensors for localization. As simple devices, sensor nodes only monitor road traffic and register vehicle-detection timestamps in their local repositories. An ad-hoc server called the *APL server* processes the complex computation for localization.

The localization procedure consists of the following phases:

- **Step 1**: After road traffic measurement, sensor s_i sends the *APL server* its vehicle detection timestamps, along with its sensor ID, i.e., (s_i, T_i), where s_i is the sensor ID and T_i is the timestamps. The *APL server* collects road traffic measurement data comprised of sensor ID vector s and corresponding timestamps T from the sensor network.
- **Step 2**: The traffic analysis module estimates the length of the road segment between two arbitrary sensors and constructs a virtual topology $H_v = (V_v, M_v)$, where V_v is the vertex set of the sensor IDs, and M_v is the matrix containing the distance estimate between two arbitrary sensors.
- **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 subgraph $\tilde{G}_v = (\tilde{V}_v, \tilde{E}_v)$ from the virtual graph G_v, where $\tilde{V}_v$ is a set of 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 $G_r = (V_r, E_r)$. Then the graph-matching module computes a permutation matrix P, making the reduced virtual subgraph $\tilde{G}_v = (\tilde{V}_v, \tilde{E}_v)$ 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 along with matrix P, the virtual graph G_v, the reduced virtual subgraph $\tilde{G}_v$ and the real graph G_r. It creates node location information (s, l), where s is the sensor ID vector, and l is the corresponding location vector, that is, $l_i = (x_i, y_i)$, where 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).

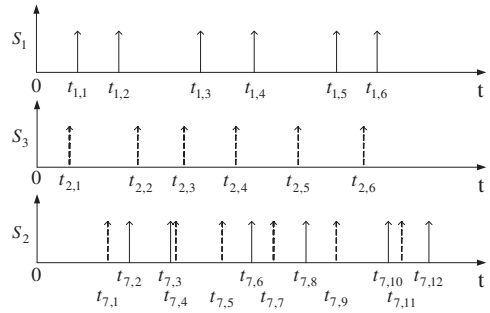
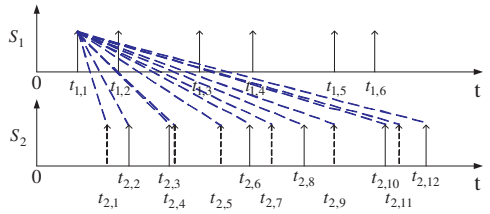
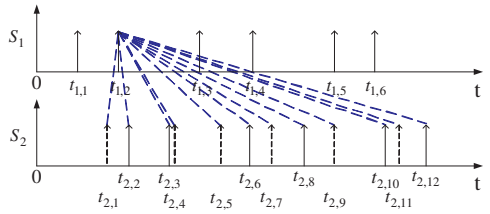


Figure 4. Detection Sequence for Vehicles at Sensors s_1, s_3, and s_2



(a) Time Difference between Timestamps $t_{1,1}$ and $t_{2,i}$



(b) Time Difference between Timestamps $t_{1,2}$ and $t_{2,i}$

Figure 5. Time Difference Operation for Sensors s_1 and s_2

3.2 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 arrival timestamps are. Consequently, we can estimate road segment length with estimated movement time between two adjacent sensors using the correlation of the timestamp 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 based SMPL [11], we found that we can estimate the lengths of road segments used by vehicles during their travels on roadways only with vehicle-detection timestamps, given the vehicle mean speed. The time difference operation for timestamp 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 $h = 1,...,|T_i|$ is the h-th timestamp of sensor s_i and $t_{jk} \in T_k$ for $k = 1,...,|T_j|$ is the k-th timestamp of sensor s_j. We define a quantized time difference operation as follows:

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

where g is a quantization function to map the real value of d_{hk}^{ij} to the discrete value in order to compute the frequency (i.e., vehicle detection count) per time difference 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 $k = 1,...,m$) is determined considering the expected movement time of vehicles in the longest road segment of the relevant road network.

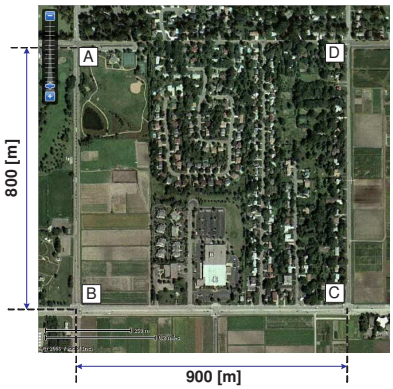


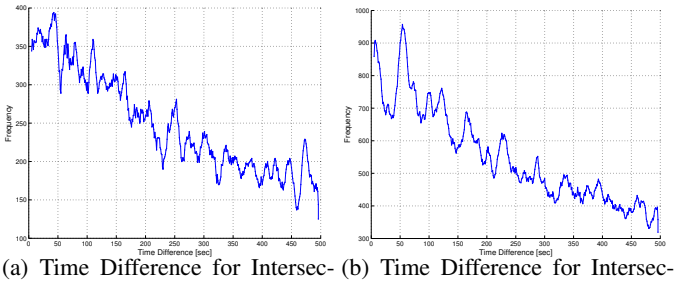
Figure 6. Road Networks for Outdoor Test

After the time difference operation for two timestamp 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 $k = 1,...,m$. 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. For example, Figure 4 shows the detection sequence for vehicles at intersection nodes s_1, s_2, and s_3 in Figure 2(c), where s_2 is a common neighbor of s_1 and s_3. Figure 5 shows the time difference operation for nodes s_1 and s_2 that is a kind of Cartesian product for two timestamp sets. For example, Figure 4 shows the detection sequence for vehicles at intersection nodes s_1, s_2, and s_3 in Figure 2(c), where s_2 is a common neighbor of s_1 and s_3. Figure 5 shows the time difference operation for nodes s_1 and s_2 that is a kind of Cartesian product for two timestamp sets.

We performed outdoor test to verify whether our time difference operation can give good estimates for road segment lengths in terms of vehicle movement time. The results of outdoor test indicate that our time difference operation can give reasonable road segment length indicators. Figure 6 shows the road map of local roadways for outdoor test. The test roadways consist of four intersections A, B, C, and D. Road segments $\overline{AB}$ and $\overline{CD}$ have the length of about 800[m] and road segments $\overline{BC}$ and $\overline{DA}$ have the length of about 900[m]. Speed limit on these road segments is 64[km/h] (or 40[mph]). 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 is our future work. We registered timestamps for vehicle detection for 15 minutes whenever vehicles passed through the center point of the intersection. The numbers of vehicle detec-



(a) Time Difference for Intersections A and B

(b) Time Difference for Intersections B and C

(c) Time Difference for Intersections C and D

(d) Time Difference for Intersections D and A

Figure 7. Outdoor Test Results in Real Road Networks

Table 1. Outdoor Test Results

Road Segment	Distance	Expected Movement Time	Measured Movement Time
A and B	800 [m]	45[sec]	43[sec]
C and D	800 [m]	45[sec]	43[sec]
B and C	900 [m]	51[sec]	54[sec]
D and A	900 [m]	51[sec]	56[sec]

tion at intersections A, B, C, and D are 89, 208, 195, and 92, respectively. Figure 7 shows the vehicle movement times with the highest frequency for four road segments. Clearly, we can see the dominant peaks indicating that their time differences are the vehicle movement times. Table 1 shows the expected movement times and measured movement times for these four road segments. We can see that the estimated movement times are close to the expected movement times. Through outdoor test, we found that our estimation method works well even under heterogenous traffic density (i.e., traffic density order for intersections: $B > C > D > A$) during a short time (i.e., 15-minute measurement). Thus, we can estimate vehicle movement time on each road segment through the timestamps and time difference operation, which can be converted into road segment length with speed limit.

Consequently, with the time difference, we can make a virtual topology, as shown in Figure 2(b), containing the distance between two arbitrary nodes. We call this estimated distance in the virtual topology as *virtual edge*. Since we do not know the exact connectivity among sensors in the virtual topology at this point, we cannot identify which distances indicate the sensor network's edges that represent road segments between sensors. Through both outdoor test and simulation, we found that in virtual graph there is a large error in the virtual edge between two arbitrary nodes corresponding to a path in the virtual topology. On the other hand, there is a very small error in the edge between two arbitrary nodes that is an edge in the virtual topology. We explain how to deal with these errors in Section 3.3.

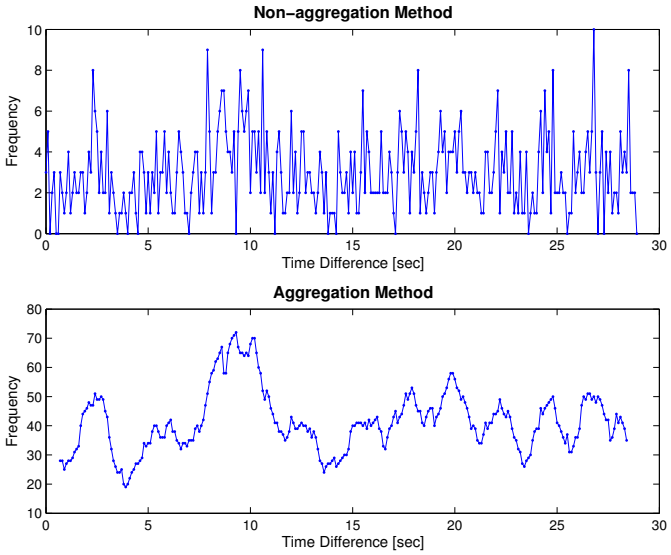


Figure 8. Comparison between Non-aggregation Method and Aggregation Method

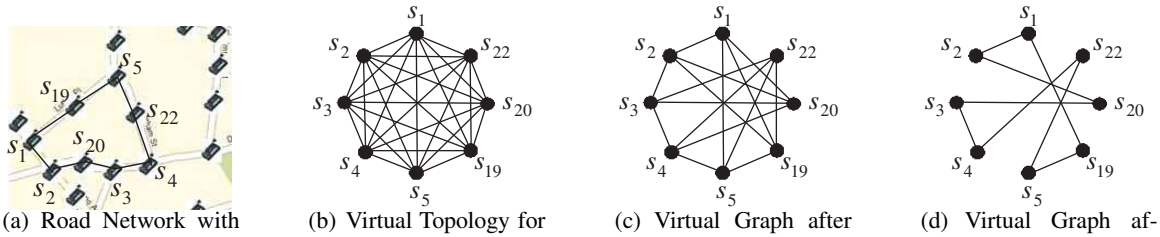
3.2.1 Enhancement of the Road Segment Length Estimation

We found that an estimate close to real road segment length cannot always be obtained by the maximum frequency through the time difference operation discussed previously. The reason is that there are some noisy estimates with higher frequencies than an expected good estimate. In order to resolve this problem, we introduce an aggregation method where the mean of several adjacent time differences become a new time difference 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 in terms of the count of the corresponding time difference obtained by the time difference operation for two timestamp series, as shown in Figure 5. On the other hand, we observe that a noisy estimate with the highest frequency occurs randomly, and its neighbor estimates have relatively low frequencies. We call this method based on time difference aggregation as the *Aggregation Method* and call the previous simple time difference as the *Non-aggregation Method*.

We determine the aggregation window size with the standard deviation σ_v of the vehicle speed. The current formula for the aggregation window size Δ is as follows:

$$\Delta \leftarrow \sigma_v w, \tag{4}$$

where σ_v is the standard deviation of the vehicle speed (e.g., 5 [km/h]), and w is the window size factor (e.g., 5). Starting from the time difference value of zero, we choose a representative of the adjacent time difference values within the optimal aggregation window size as the mean of them, and 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. For example, we show this phenomenon using our simulation for the road network, as shown in Figure 2. Figure 8 shows the comparison between the non-aggregation method and aggregation method. We found that for the road segment between sensors s_2 and s_3 whose time difference is 9.36[sec], the non-aggregation method



(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

Figure 9. Procedure of Prefiltering for obtaining Virtual Graph

makes a wrong estimate (i.e., 26.8[sec]), but the aggregation method makes a correct estimate (i.e., 9.3[sec]). Thus, we use this aggregation method to obtain good estimates for road segment lengths in virtual topology.

3.3 Step 3: Prefiltering Algorithm for a Virtual Graph

We observe that the time difference operation discussed in Section3.2 gives large errors in path estimates between two arbitrary sensors in virtual topology. The reason is that when two sensors are separated far from each other, the correlation between the two timestamp sets from them is reversely proportional to the distance between the two sensors. On the other hand, the edge estimates (i.e., estimates for road segments) produced by the time difference operation are much more accurate. From this observation, we filter out all inaccurate path estimates from the virtual topology, except for edge estimates so that the virtual topology is converted into a virtual graph. However, there still remain accurate path estimates of two sensors separated from each other by approximately two or three road segments. We can filter out the accurate path estimates using the fact that the shortest estimate should usually be an edge estimate, and a path estimate consists of such edges. Thus, our prefiltering algorithm consists of two prefilterings:

1. Prefiltering based on the *Relative Deviation Error* and
2. Prefiltering based on the *Minimum Spanning Tree*.

We explain the prefiltering procedure and the effect of two prefilterings on virtual topology using Figure 9. As shown in Figure 9(a), there is a partial road network of the entire one shown in Figure 2(a) containing sensors $\{s_1, s_2, s_3, s_4, s_5, s_{19}, s_{20}, s_{22}\}$. In virtual topology, two arbitrary sensors among them have a distance estimate, as shown in Figure 9(b). Using 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 Figure 9(c). Next we apply 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 in Figure 9(d). 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.

3.3.1 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 several edge lengths. However, when there are large errors in M_v, the perturbed data can be any value regardless of what the accurate estimate value is for the entry in M_v. In this case, we will no longer regard the smallest entry as an edge estimate rather than a path estimate perturbed by a large error. As a result, it is very important to filter out all the entries that have large errors, regarding them as path estimates.

We define *Relative Deviation* (ϕ) as the ratio of the standard deviation (σ) to the mean (μ), that is, $\phi = \sigma/\mu$. To compute both the mean and the standard deviation of each entry in M_v, We use multiple estimation matrices of M_v per measurement time with the same duration. In order to compute the relative deviations of the estimates, we divide the vehicle-detection timestamps into time windows (e.g., every one hour) and perform the time difference operation for the timestamps 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., $\varepsilon = 5\%$), the corresponding entry is regarded as a path estimate, and it is replaced with ∞, indicating that this entry is a path estimate.

3.3.2 Prefiltering based on the Minimum Spanning Tree

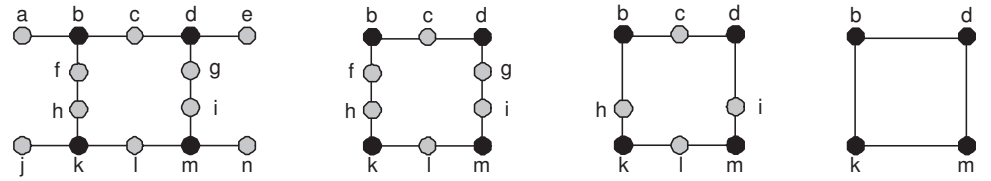
Suppose that there are n sensors in the virtual topology. Let M_v be the $n \times n$ adjacency matrix of the virtual topology. Prefiltering based on the Minimum Spanning Tree consists of the following two steps:

1. Finding of the First n-1 Edges of the Virtual Graph and
2. Finding of All of the Other Edges of the Virtual Graph.

First, 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 algorithm* [4]. The $n-1$ edges that form the MST are definitely edge estimates. Let $M_v(u,v)$ be the entry of matrix M_v where u is the row index and v is the column index. A brief proof is as follows:

1. The smallest entry must be an edge because the path length is the sum of several edge lengths.
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 $M_v(u,v)$ that satisfies $u \notin N$, and $v \in N$. $M_v(u,v)$ must be an edge. If $M_v(u,v)$ is not an edge, there must exist another node q such that $M_v(u,v) = M_v(u,q) + M_v(q,v)$. If $q \in N$, then $M_v(u,q) < M_v(u,v)$, which contradicts our assumption that $M_v(u,v)$ is the smallest entry. If $q \notin N$, then $M_v(q,v) < M_v(u,v)$, and it also contradicts our assumption that $M_v(u,v)$ is the smallest.

Second, in order to find all of the other edges of the virtual graph $G_v = (V_v, E_v)$, as shown in Figure 2(c), with $n-1$ edges



(a) Virtual Graph: Intersection Node Set={b,d,k,m} and Non-intersection Node Set={a,c,e,f,g,h,i,j,l,n}

(b) Deletion of Non-intersection Nodes {a,e,j,n} with degree 1

(c) Deletion of Non-intersection Nodes {f,g} with degree 2

(d) Deletion of Non-intersection Nodes {c,h,i,l} with degree 2

Figure 10. Construction of the Reduced Virtual Subgraph

obtained by the previous step, we compute the shortest paths between all pairs of nodes and create a new matrix M'_v. We use the fact that $M'_v(i,j) \geq M_v(i,j)$ because for an arbitrary pair of nodes i and j, $M'_v(i,j)$ is the shortest path created only by $n-1$ edges, while $M_v(i,j)$ is created from more edges; that is, $M_v(i,j)$ might be shorter than $M'_v(i,j)$. It is proven from Theorem A.1 in Appendix A 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 large error after the previous filtering. Consequently, $M_v(i,j)$ is the n-th edge estimate we found. We update the set of edges by adding this new edge, and we also update the matrix M'_v using the new set. We repeat this process until M'_v and M_v are exactly the same. In this way, we can find out all of the other edge estimates of E_v from M_v.

3.4 Step 4: Graph Matching

In this section, we explain how to construct a reduced virtual subgraph from the virtual graph, and then how to match the reduced virtual subgraph and real graph that are isomorphic to each other.

3.4.1 Construction of the Reduced Virtual Subgraph

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. We observe that every intersection node in the virtual graph, as shown in Figure 2(c), has a degree greater than 2, since each intersection on the road are connected to at least 3 road segments, so each intersection node has at least three neighboring sensors. With this observation, we make a reduced virtual subgraph from the virtual graph as follows:
Let $G_v = (V_v, E_v)$ be a virtual graph. Let N be a set of non-intersection 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, Figure 10 shows the construction of a reduced virtual subgraph from a virtual graph in which a set of intersection nodes is {b,d,k,m} and a set of non-intersection nodes is {a,c,e,f,g,h,i,j,l,n}. After removing non-intersection nodes and dealing with the corresponding edges, the final reduced virtual subgraph consists of four intersection nodes b, d, k, and m.

We should note that Theorem A.1 is based on the assumption that if there is an edge between two intersection nodes in a real graph, it must correspond to the shortest path in the shortest path matrix M_r of the real graph. That is, $E_r(i,j) = M_r(i,j)$. However, it is not always the case in the real world. In the case where for nodes p_i and p_j, there exists a path between nodes p_i and p_j shorter than an edge of the nodes p_i and p_j, that is, $E_r(i,j) > M_r(i,j)$, our algorithm sets $\tilde{E}_v(i,j)$ to 0, which means that there is no edge between nodes p_i and p_j. At this time, E_r and $\tilde{E}_v$ are no longer isomorphic to each other. We need to deal with this situation in the following two approaches:

1. Approach 1: Before applying the graph matching algorithm to E_r and $\tilde{E}_v$, we modify E_r as follows. We can compare all of the entries in E_r and M_r and set $E_r(i,j) = 0$ if $E_r(i,j) > M_r(i,j)$.
2. Approach 2: We use $\tilde{E}_v$ to generate another all-pair shortest paths matrix $\tilde{M}_v$. Unlike M_v obtained from the measurement, $\tilde{M}_v$ is accurate and isomorphic to M_r. As a result, we can use $\tilde{M}_v$ and M_r instead of $\tilde{E}_v$ and E_r in the graph matching algorithm.

3.4.2 Weighted Graph Matching

Since the reduced virtual subgraph'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:

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

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

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

Let P be an $n \times n$ optimal permutation matrix of Eq. 6 in terms of the minimum estimation error. The result $\hat{E}_v$ of Eq. 7 is a matrix isomorphic to E_r where indices in both matrices indicate the node identifications; that is, the sensor ID in $\tilde{E}_v$ corresponds to the intersection ID in E_r for $i = 1, ..., n$. This optimization problem is called the Weighted Graph Matching Problem (WGMP). In order to get the exact solution P, allowing for 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 a 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 eigendecomposition approach to WGMP [24], known as an optimal approach. For our graph matching purpose, we adopt the eigendecomposition approach that has polynomial time complexity.

3.4.3 Effect of the Real Vehicle Mean Speed different from the Speed Limit on Roadways

The road traffic condition is changed according to time zone, such as rush hour and night. This might affect the vehicle mean speed. Thus, we investigate how the changed vehicle mean speed affects our localization scheme in this section.

Assume that the vehicle mean speed in each road segment is uniformly scaled up or down according to traffic condition. We found that the the graph matching described in Section 3.4.2 is not affected, even in the case in which vehicles have the mean speed from our assumed vehicle mean speed of the limited speed on roadways. This limited speed is used in the computation of the adjacency matrix E_r of the real graph $G_r = (V_r, E_r)$. The reason is that the different mean speed does not affect the eigenvectors of matrices E_r and $\tilde{E}_v$ used for computing the permutation matrix P in the eigendecomposition approach [24]. Consequently, the permutation matrix P does not change due to the scalar multiplication for E_r or $\tilde{E}_v$, which means that there is no problem in the case in which the mean speed used in matrix E_r is different from that used in matrix $\tilde{E}_v$. This argument can formally be proved as follows:

THEOREM 3.1. *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*

$$P = arg \min_{\hat{P}} \|E_r - P\hat{E}_v P^T\|_2^2, \tag{8}$$

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

$$P = arg \min_{\hat{P}} \|E_r - PcE_v P^T\|_2^2, \forall c \in \mathbb{R}^+. \tag{9}$$

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

$$\sigma : x \in \{1, ..., n\} \to y \in \{1, ..., n\}, \tag{10}$$

that is, $y = \sigma(x)$. Thus, the 2-norm square in Eq. 8 can be represented using the summation and permutation function as follows:

$$\Phi(P, E_r, \hat{E}_v) = \|E_r - P\hat{E}_v P^T\|_2^2 = \sum_{i=1}^{n}\sum_{j=1}^{n}(r_{ij} - v_{\sigma(i)\sigma(j)})^2. \tag{11}$$

Also, the 2-norm square in Eq. 9 can be represented as follows:

$$\Phi(P, E_r, c\hat{E}_v) = \|E_r - PcE_v P^T\|_2^2 = \sum_{i=1}^{n}\sum_{j=1}^{n}(r_{ij} - cv_{\sigma(i)\sigma(j)})^2. \tag{12}$$

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:

$$\begin{aligned}
&\Phi(P, E_r, \hat{E}_v) - \Phi(\bar{P}, E_r, \hat{E}_v) \le 0, \\
\Rightarrow &\sum_{i=1}^{n}\sum_{j=1}^{n}(r_{ij} - v_{\sigma(i)\sigma(j)})^2 - \sum_{i=1}^{n}\sum_{j=1}^{n}(r_{ij} - v_{\bar{\sigma}(i)\bar{\sigma}(j)})^2 \le 0, \\
\Rightarrow &\sum_{i=1}^{n}\sum_{j=1}^{n}(-2r_{ij}v_{\sigma(i)\sigma(j)} + 2r_{ij}v_{\bar{\sigma}(i)\bar{\sigma}(j)}) \le 0.
\end{aligned} \tag{13}$$

In the same way, from Eq. 12, if we take the difference between two 2-norm squares for P and $\bar{P}$, then P is also an optimal permutation matrix of Eq. 12 due to Eq. 13 as follows:

$$\begin{aligned}
&\Phi(P, E_r, c\hat{E}_v) - \Phi(\bar{P}, E_r, c\hat{E}_v) = \\
&\textstyle\sum_{i,j}^{n}(r_{ij} - cv_{\sigma(i)\sigma(j)})^2 - \sum_{i,j}^{n}(r_{ij} - cv_{\bar{\sigma}(i)\bar{\sigma}(j)})^2 = \\
&c\textstyle\sum_{i,j}^{n}(-2r_{ij}v_{\sigma(i)\sigma(j)} + 2r_{ij}v_{\bar{\sigma}(i)\bar{\sigma}(j)}) \le 0, \forall c \in \mathbb{R}^+.
\end{aligned} \tag{14}$$

□

From Theorem 3.1, 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. In the case where each road segment has a different scaling factor according to traffic condition, our algorithm does not work. However, under a light road traffic condition, such as night, we expect that all of the road segments tend to have the same constant scaling factor c for their mean speeds. Thus, our algorithm can be performed during the light road traffic condition.

3.5 Step 5: Node Location Identification

In this section, we explain how to identify the location of each intersection node, and then how to identify the location of each non-intersection node.

3.5.1 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 the permutation function $\sigma(s)$ be a map corresponding to the permutation matrix P

$$\sigma : s \in \{1, ..., n\} \to p \in \{1, ..., n\}, \tag{15}$$

that is, $p = \sigma(s)$ where s is the sensor ID and p is the intersection ID.

Let (x_{s_i}, y_{s_i}) be the coordinate of sensor s_i. Let $loc(\sigma(s_i))$ be the location function that returns the coordinate (x_{s_i}, y_{s_i}) corresponding to the index $\sigma(s_i)$. Thus, the location identification of node s_i is performed as follows:

$$[x_{s_i}, y_{s_i}] \leftarrow loc(\sigma(s_i)) \tag{16}$$

3.5.2 Localization of Non-intersection Nodes

In the previous section, we know the positions of the intersection nodes. Now we localize the positions of the non-intersection nodes. Using E_v, we begin from an intersection node u, and we create a path from u to another intersection node v, that is, $u \to a_1 \to a_2 \to \cdots \to a_m \to v$. All a_i for $i = 1, ..., m$ are non-intersection nodes whose degrees are 2. Since we have already 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 subgraph, as shown Figure 2(d), we can know the positions of these a_i by looking at the length information in E_v of the virtual graph, as shown in Figure 2(c). We repeat this procedure until we localize all of the non-intersection nodes in the virtual graph.

4 Discussions

We discuss several considerations for the deployment of our localization scheme in real road networks and suggest two optimizations for traffic analysis in order to get more accurate edge estimates in the virtual topology and to reduce the traffic measurement time.

4.1 Sensor Time Synchronization Error

The inaccuracy of the timestamps 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 $\tau_i = \tau + \varepsilon_i$ and ε_i is a uniform random variable in the interval $[-\varepsilon_{max}, \varepsilon_{max}]$. 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 in which $t \approx l/v$. In the case where two adjacent sensors have time errors $-\varepsilon_{max}$ and ε_{max}, respectively, the estimated movement time $\hat{t}$ will be approximately $t + 2\varepsilon_{max}$. Consequently, if ε_{max} is small, then the movement time $\hat{t}$ will be close to t.

4.2 Intersection Node Missing

Our prefiltering and graph matching algorithm works well when there are some non-intersection nodes missing or out of function. This is because our algorithm is based on intersection nodes. However, when there is the missing of intersection nodes, the reduced virtual subgraph will no longer be isomorphic to the real graph and so the graph matching algorithm will not work. As a result, we need to find out the missing intersection nodes and add them to the reduced virtual subgraph to make it isomorphic to the real graph.

Assume that every road segment has at least one non-intersection node. This assumption makes sense, since road segments are normally longer than intersections, so sensors can be placed on road segments with high probability by the random deployment based on unmanned aerial vehicles. The basic idea to find out a missing intersection node is as follows: With the degree information in E_v, delete all of the edges incident to nodes whose degrees are one or two. If all of the nodes become isolated in the graph, there is no missing of intersection nodes. Otherwise, there is missing of intersection nodes.

For example, as shown in Figure 2(c), we focus on the intersection where node s_9 is placed. If s_9 is not missing, our prefiltering algorithm in Step 3 of Section 3.1 will make a matrix E_v containing edges $e_{9,28}$, $e_{9,46}$, $e_{9,29}$, and $e_{9,30}$. For the intersection related to s_9, we can see that all of the edges of $\{e_{9,28}, e_{9,46}, e_{9,29}, e_{9,30}\}$ can be deleted from E_v by our algorithm above. However, if s_9 is missing, the prefiltering algorithm will make a matrix E_v including a complete subgraph consisting of four nodes of $\{s_{28}, s_{29}, s_{30}, s_{46}\}$ that are adjacent to s_9. By our algorithm above, the subgraph's six edges of $\{e_{28,46}, e_{28,29}, e_{28,30}, e_{29,46}, e_{29,30}, e_{30,46}\}$ cannot be deleted, since there is no non-intersection node between any pair of these four nodes. As a result, we can find out the missing intersection node s_9 among these four nodes. Since the edge lengths related to s_9 can be easily solved from E_v, we can add s_9 to the virtual graph. In this way, after we find out all the missing intersection nodes, the reduced virtual graph will be isomorphic to the real graph.

4.3 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 timestamps, the missing of some vehicle detections does not affect the road segment length estimation. We investigated the effect of the detection missing for the whole localization accuracy through simulation by modeling the detection event as a Bernoulli trial. The result is that our localization scheme has no localization error under the simulation setting in Section 5 with the detection missing probability from 0 to 0.2 at each sensor.

Also, we investigated what effect the duplicate detection has for the whole localization accuracy through simulation by modeling the duplicate detection event as a Bernoulli trial. The result is that our localization scheme has no localization error under the simulation setting in Section 5 with the duplicate detection probability from 0.1 to 1 at each sensor. This means that the duplicate vehicle detection has positive effect, since the duplicate vehicle detection can contribute more to the detection frequency corresponding to the right road segment estimate.

4.4 Matching Ambiguity due to Topology Symmetricity

In certain cases, some subgraphs in a real graph might have a symmetric topology. Since our graph matching cannot localize sensors correctly, due to the topology symmetricity, we need anchor nodes that are sensors with GPS receivers. We need to determine whether we can localize sensors uniquely by testing the conditions for unique localizability discussed in [1]. If there is a topology symmetricity in the road network, we need to find out a minimum number of anchor nodes and the places to remove the ambiguity for graph matching before applying our localization scheme to the road network.

4.5 Subgraph Matching in the case of Sensor Death or Intersection Node Missing

Wireless sensors can die due to some reasons, such as energy exhaustion and breakdown. There may be missing intersection nodes, due to the random deployment from the airplanes in the sky. This means that we cannot get the reduced virtual subgraph's adjacency matrix isomorphic to the real graph's adjacency matrix. In this case, we can try to use subgraph matching [10], since the reduced virtual subgraph is isomorphic to a subgraph of the real graph. That is, the intersection nodes of the reduced virtual subgraph are matched with the corresponding intersection points of the real graph through a subgraph matching.

4.6 Optimal Time-Difference Window Size for Time Difference Operation

We define the *time-difference window size* as the time duration for the time difference operation of vehicle-detection timestamps in order to obtain a virtual topology. The time duration for the time difference operation is required to be long enough to obtain a virtual topology that has accurate edge estimates. If the time-difference window is longer, it will give a more accurate virtual topology, but it will require more computation time. Thus, we need to determine an optimal time-difference window size for the time operation in terms of the computation cost and the accuracy of the virtual topology.

With an optimal time-difference window size, we divide the timestamps into the multiple sets in order to compute the mean

and deviation of edge estimates in the virtual topology as discussed in Section 3.3.1. For the computation of the mean and deviation, we need to consider how many the timestamp sets are appropriate for the accurate prefiltering based on the relative deviation error. Consequently, the time-difference window size and the number of timestamp sets will determine the traffic measurement time. In our simulation, we divide the timestamps into multiple sets per hour, and compute the mean and deviation of edge estimates.

4.7 Other Issues

There are other issues to study as future work as follows:

- The case in which an exact real graph for intersection nodes is unknown: In reality, it is not easy to construct a real graph in the scenario of aerial sensor deployment. Otherwise, we need to perform subgraph matching [10] between the reduced virtual subgraph and the real graph, since the real graph should be a supergraph for the reduced virtual subgraph.
- The scenario in which multiple sensors are placed at each intersection point in the roadways: We need to select one of the multiple sensors at the same intersection point as representative for localization. We can regard these sensors as belonging to one intersection point if they have similar road segment estimates for the other sensors, except for them.
- Detection of Outlier Sensors in Timestamping: Timestamps may be totally wrong due either to sensor malfunction or a security attack. We need to filter out the timestamps of such sensors in order to perform exact localization.
- Localization for Large-scale Road Networks: For large-scale road networks, it is difficult to compute a large matrix in one *APL* server, so we need to divide the road network into doable areas for localization with multiple *APL* servers in terms of localization accuracy and computation complexity. We can consider a kind of divide-and-conquer approach.
- Localization on Three-Dimensional Roadways: In reality, there are three-dimensional road networks. We can extend our two-dimensional solution to a three-dimensional solution for localization in such road networks if we have the road map for the road networks.
- Traffic Measurement Duration: Since our localization scheme depends on the passive traffic measurement, it will take a long time comparing with other legacy schemes. Thus, we need to determine the *traffic measurement duration* for the road segment estimation in terms of the computation cost and the accuracy of the virtual topology, considering road traffic condition.
- Optimal Aggregation Window Size in Aggregation-based Road Estimation: We need to investigate the function to give an *optimal aggregation window size* in terms of localization accuracy, considering road traffic condition, such as vehicle mean speed and vehicle speed deviation.

5 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 state-of-the-art schemes, we investigate the effect of the following three parameters on our localization scheme:

- The time synchronization error standard deviation,
- The vehicle speed standard deviation, and
- The vehicle interarrival time.

We want to show the operational region of our localization scheme for these three parameters.

We present two kinds of performance evaluations as follows: First, we compare the aggregation-based estimation method with the nonaggregation-based estimation method in terms of the estimation accuracy for road segment length. For the estimation accuracy, the *Matrix Error Ratio* 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, we evaluate the performance of each localization method consisting of a combination of the aggregation-based estimation method and prefiltering types below that use the same graph-matching algorithm specified in Section3.4. The *Localization Error Ratio* is defined as the ratio of the number of incorrectly localized sensors to the number of all sensors deployed on the road network.

The simulation environment is described in Table 2. From road traffic measurement, we create a matrix M_v for the virtual topology as the average of 10 matrices M_{vs} that are adjacency matrices of the virtual topology created from the same measurement time, such as one hour; that is, M_v is the all-pair shortest path estimation matrix for the virtual topology, as shown Figure 2(b).

5.1 Performance Comparison between Road Segment Estimation Methods

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

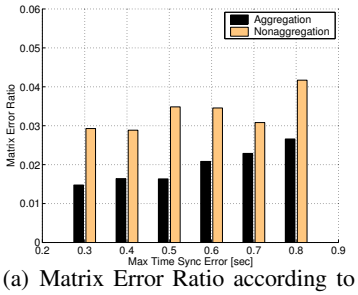
1. The aggregation-based road segment estimation and
2. The nonaggregation-based road segment estimation.

After the estimation, we perform the prefiltering algorithm described in Section 3.3 and the graph matching algorithm described in Section 3.4 in order to evaluate the *Matrix Error Ratio* and *Localization Error Ratio*. For the maximum time synchronization error, Figure 11 shows the performance comparison between the aggregation and non-aggregation methods. For the aggregation method, the *Matrix Error Ratio* is less than 0.03, which indicates that $\tilde{E}_v$ of the reduced virtual subgraph $\tilde{G}_v$ is very close to the E_r of the real graph G_r, as shown in Figure 2, where $\tilde{G}_v$ is a subgraph of the virtual topology H_v. We can see that most *Matrix Error Ratios* of the aggregation method are less than the *Matrix Error Ratios* of the nonaggregation method. That is why the aggregation method gives better localization than the non-aggregation. From Figure 11(b), we can see that our localization works well in the case in which the maximum time synchronization error is less than 0.4 seconds. We can claim that our localization scheme can work in the real environment, since the state-of-the-art time synchronization protocols can give the accuracy at the millisecond level [5, 12].

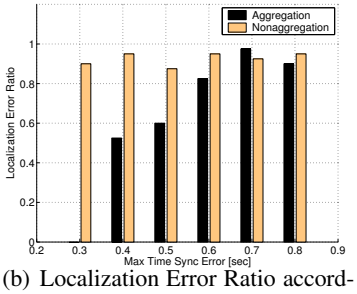
For the vehicle speed deviation, as shown Figure 12, the aggregation method outperforms the nonaggregation method in that the *Matrix Error Ratio* of the aggregation is less that that of the nonaggregation method. That is why the aggregation method can give more accurate localization than the nonaggregation method, except for the vehicle speed deviation of 15[km/h]. This speed deviation of 15[km/h] is the value out of

Table 2. Simulation Environment

Parameter	Description
Number of sensors	40 sensors (from s_1 to s_{40}) are deployed in the road network, as shown in Figure 2.
Simulation time	Sensors perform vehicle detection for 10 hours and store the vehicle-detection timestamps into their repositories.
Time synchronization error	Sensor's time synchronization error conforms to a uniform distribution with the interval $[-\varepsilon_{max}, \varepsilon_{max}]$ where ε_{max}=0.01[sec].
Vehicle speed distribution	Vehicle speed conforms to a Gaussian distribution of $N(\mu_v, \sigma_v^2)$ where $\mu_v = 50$[km/h] and $\sigma_v = 5$[km/h].
Vehicle speed boundary	Vehicle's maximum speed is 80[km/h] and vehicle's minimum speed is 20[km/h].
Vehicle interarrival time	Every vehicle arrives at road network in the interval 120[sec].
Vehicle travel length distribution	Let $d_{u,v}$ be the shortest path distance from source intersection u and destination intersection v in road network. Vehicle's travel path length from u and v conforms a Gaussian distribution of $N(\mu_d, \sigma_d^2)$ where $\mu_d = d_{u,v}$[m] and $\sigma_d = 500$[m].

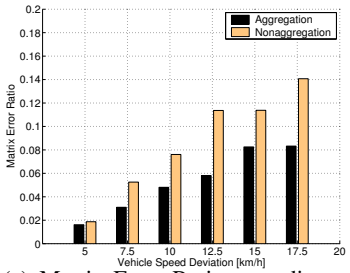


(a) Matrix Error Ratio according to Maximum Time Synchronization Error

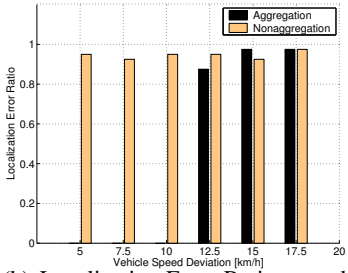


(b) Localization Error Ratio according to Maximum Time Synchronization Error

Figure 11. Performance Comparison between Aggregation and Nonaggregation Methods for Maximum Time Synchronization Error (ε_{max})

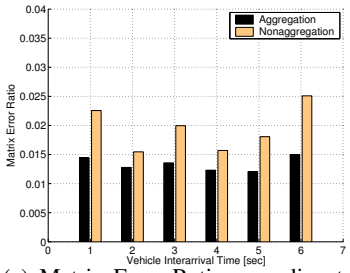


(a) Matrix Error Ratio according to Vehicle Speed Deviation

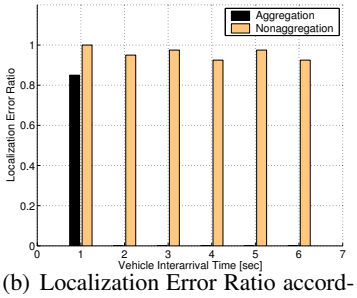


(b) Localization Error Ratio according to Vehicle Speed Deviation

Figure 12. Performance Comparison between Aggregation and Nonaggregation Methods for Vehicle Speed Deviation (σ_v)



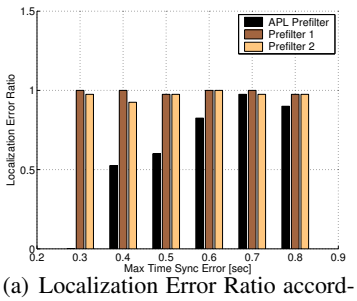
(a) Matrix Error Ratio according to Vehicle Interarrival Time



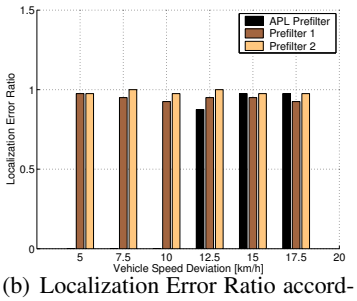
(b) Localization Error Ratio according to Vehicle Interarrival Time

Figure 13. Performance Comparison between Aggregation and Nonaggregation Methods for Vehicle Interarrival Time ($1/\lambda$)

the operational region for our localization scheme, so the corresponding localization error ratio is always a random value close



(a) Localization Error Ratio according to Maximum Time Synchronization Error



(b) Localization Error Ratio according to Vehicle Speed Deviation

Figure 14. Performance Comparison among Prefiltering Types

to 1. However, considering the real statistics [14] that the vehicle speed deviation in four-lane roadways is 9.98[km/h], and the vehicle speed deviation in two-lane roadways is 8.69[km/h], we can claim that our localization can work in the real environment, since our localization scheme works with the vehicle speed deviation less than 10[km/h].

For the vehicle interarrival time, as shown Figure 13, we see that it does not affect the performance of our localization scheme. The reason is that our time difference operation can give accurate estimates for road segment lengths, as long as the vehicle interarrival time is larger than 1 second and it allows for enough road traffic to cover all of the road segments. In fact, most people drive their vehicles with the interval time larger than 1 second for their safety, so we can claim that our localization works in normal driving condition. For the aggregation method, the *Matrix Error Ratio* is less than 0.015, which indicates that $\tilde{E}_v$ of the reduced virtual subgraph $\tilde{G}_v$ is very close to the E_r of the real graph G_r. This is why the aggregation method gives 100% localization, except for 1-second vehicle interarrival time. We can see that all of the *Matrix Error Ratios* of the aggregation method are less than those of the nonaggregation method.

5.2 Performance Comparison among Prefiltering Types

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

1. *Prefilter 1*: Prefiltering based on the minimum spanning tree described in Section 3.3.2,
2. *Prefilter 2*: Prefiltering based on the relative deviation error described in Section 3.3.1, 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 subgraph $\tilde{G}_v = (\tilde{V}_v, \tilde{E}_v)$, the same graph-matching algorithm described in Section 3.4 is applied to the output matrix $\tilde{E}_v$ in order to evaluate the *Localization Error Ratio*. From Figure 14, our localization with *APL Prefilter* works well under reasonable, real environment in which the maximum time synchronization error is less than 0.4[sec], and the vehicle speed deviation is less than 12.5[km/h]. As we can see Figure 14, one missing of the minimum-spanning-tree-based prefilter (i.e., *Prefilter 1*) and the relative-deviation-error-based prefilter (i.e., *Prefilter 2*) cannot allow for the accurate localization under the reasonable, real

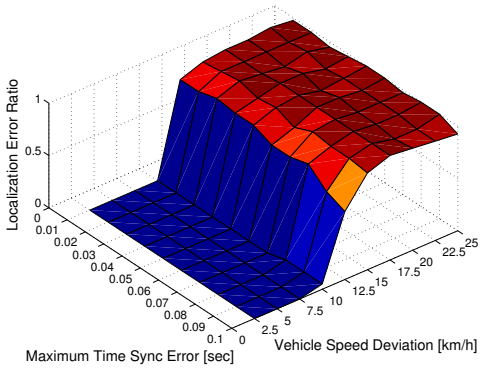


Figure 15. APL Operational Region for Maximum Time Synchronization Error and Vehicle Speed Deviation

environment. That is why we use the combination of two prefilters. From Figure 14(a), the *Prefilter 2* outperforms the *Prefilter 1* according as the maximum time synchronization error increases. On the other hand, Figure 14(b), the *Prefilter 1* outperforms the *Prefilter 2* according as the vehicle speed deviation increases. From these figures, we can conclude that two prefilters cooperate the error reduction in edge estimates for our localization.

Figure 15 shows the *APL operational region* that contains the range of the maximum time synchronization error and the vehicle standard deviation to allow for a perfect localization under the simulation environment given in Table 2. As we can see, 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 0.01 to 0.1[sec]. As we mentioned before, 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) [14]. For the vehicle interarrival time, our localization works well as long as the interarrival time is greater than 1 second. Thus, we can conclude that the vehicle speed deviation is the dominant factor of the performance in our localization scheme. Also, we can claim that our localization scheme can work in the real road networks.

6 Related Work

Many localization schemes have been proposed so far, and they can be categorized into three classes: (a) Range-based localization schemes, (b) Range-free localization schemes, and (c) Event-based localization schemes. They have different assumptions about their respective networks and device capabilities. These assumptions include the device hardware, sig-

nal propagation models, timing, energy requirements, network configuration (i.e., homogeneous vs. heterogeneous), environments (i.e., indoor vs. outdoor), node or beacon density, time synchronization of devices, communication cost, error requirements, node mobility, and localization time. We discuss these legacy localization schemes by categorizing them into the three classes.

6.1 Range-Based Localization Schemes

Range-based schemes require costly hardware devices to estimate the distance between nodes, along with the additional energy consumption for them. The Time of Arrival (TOA) and Angle of Arrival (TDOA) schemes measure the propagation time of the signal, and estimate the distance based on the propagation speed. The most popular use of the TOA technique is GPS, which is a very costly solution for localization in wireless sensor networks [25]. The TDOA based on signal propagation time requires expensive, energy consuming ranging devices, since the devices are required to be time-synchronized at a high granularity [19, 20]. These devices are unfeasible for low-power sensor nodes. The TDOA scheme based on ultrasound requires a dense deployment of sensors, since ultrasound signals usually propagate only 20-30 feet. This dense deployment in large-scale road network is very costly.

The Angle of Arrival (AOA) schemes estimate the positions of the nodes by sensing the direction from which a signal is received [15]. However, the AOA scheme requiring angle estimation devices is also unsuitable for low-power sensor nodes equipped with additional hardware devices for angle estimation, such as an antenna array or several ultrasound receivers.

The Received Signal Strength Indicator (RSSI) schemes use either theoretical or empirical models to estimate the distance based on the loss of power during signal propagation. In the RSSI scheme based on Radio Frequency (RF) wireless signals [2, 9], the signal strength is translated into distance estimate. In [2], several beacon nodes are used to emit RF wireless signals and a map of signal strengths is created. A mobile node can be located by matching its signal strength with the map. However, in real environments where sensors are deployed, there are multi-path fading, irregular signal propagation, and background interference. These are big impediments for the RSSI schemes for the sensor networks. Thus, since the expensive, energy-consuming hardware devices are required for ranging, it is very costly to apply these schemes to tiny low-power sensor nodes deployed in large-scale road networks.

6.2 Range-Free Localization Schemes

To address the limitations of the range-based schemes for the sensor networks, range-free localization schemes have been proposed. The range-free localization schemes try to localize sensors without costly ranging devices. One of the most popular range-free schemes is based on anchor-based scheme, where anchors with their own exact location information broadcast beacon signals to neighboring nodes called non-anchors. The main idea is that the non-anchors can determine their locations using the overlapped region of communication areas for the anchors [3, 7, 23]. However, since these schemes require a dense deployment of anchors to give beacon signals, such a deployment is the overkill for the localization in road networks, where the localization granularity is the possible area, such as intersection points and non-intersection points on the roadways.

Another range-free scheme is anchor-free distributed localization scheme, not using anchors [17]. Sensors initially guess their positions, and iteratively correct their positions by estimating the distances between their neighbors and themselves on the basis of radio connectivity. The correction algorithm for their positions is based on a physics model called mass spring model. Each node tries to minimize the difference between the guessed neighbor distance and the actually measured neighbor distance for each neighboring node. However, this scheme requires a dense deployment of sensors to obtain the accurate distance estimation. This incurs a costly deployment for the sensor networks deployment on the road networks. Thus, in the case in which the spacing between the sensors is large for communication for localization, the range-free localization schemes require the dense deployment of sensors including anchors, leading to the costly deployment in large-scale road networks.

6.3 Event-Based Localization Schemes

Recently, event-based localization schemes have been proposed to simplify the functionality of sensor nodes 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 [8, 18, 22]. Some well-controlled artificial events, such as light, are injected into sensor networks. When sensor nodes detect these events, they report the detection times to a base station that computes complex algorithms for sensor localization, the event-detection timestamps provided by the sensor nodes. However, in outdoor environments, it is very hard to generate and disseminate these artificial events to a large-scale area. In particular, in the road networks of cities, there are many buildings and structures. These are the impediments for the exact delivery of events. This is caused by the dependency of the artificial events. On the other hand, our localization scheme is a new branch of event-based localization schemes. Since our localization scheme is based on natural events of moving vehicles, there is such no problem of the event delivery.

7 Conclusion

We address the localization problem in road networks around a target area with the aerial deployment of sensors with low-cost hardware. We propose an efficient estimation method to estimate road segment lengths only with the vehicle-detection timestamps. Also, we propose prefiltering algorithms, eliminating path estimates with with large errors, and keeping edge estimates. Through the outdoor test and simulation, we show that our localization scheme allows for a sparse deployment of sensors on road networks without costly, energy consuming hardware for localization. In future work, we will develop the vehicle detection algorithm for motes, such as Micaz and XSM, placed on intersections or non-intersections, and then will deploy and test our scheme on large-scale roadways.

8 References

[1] J. Aspnes, T. Eren, D. K. Goldenberg, A. S. Morse, W. Whiteley, Y. R. Yang, B. D. Anderson, and P. N. Belhumeur. A theory of network localization. *IEEE Transactions on Mobile Computing*, 5(12):1663–1678, Dec. 2006.

[2] P. Bahl and V. N. Padmanabhan. Radar: An in-building rf-based user location and tracking system. In *Proceedings of INFOCOM*, Mar. 2000.

[3] N. Bulusu, J. Heidemann, and D. Estrin. Gps-less low cost outdoor localization for very small devices. *IEEE Personal Communications*, 7(5):28–34, Oct. 2000.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (2nd Edition)*. The MIT Press, 2003.

[5] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. In *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI)*, Boston, MA, USA, Dec. 2002.

[6] L. Gu, D. Jia, P. Vicaire, T. Yan, L. Luo, A. Tirumala, Q. Cao, T. He, J. A. Stankovic, T. Abdelzaher, and B. H. Krogh. Lightweight detection

and classification for wireless sensor networks in realistic environments. In *Proceedings of the third Conference on Embedded Networked Sensor Systems (SenSys)*, San Diego, California, USA, Nov. 2005.

[7] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher. Range-free localization schemes for large scale sensor networks. In *Proceedings of MOBICOM*. ACM, Sept. 2003.

[8] T. He, R. Stoleru, and J. A. Stankovic. Spotlight: Low-cost asymmetric localization system for networked sensor nodes. In *Proceedings of the fourth International Conference on Information Processing in Sensor Networks (IPSN)*, Los Angeles, California, USA, Apr. 2005. ACM/IEEE.

[9] J. Hightower, G. Boriello, and R. Want. Spoton: An indoor 3d location sensing technology based on rf signal strength. *University of Washington Technical Report*, (2000-02-02), Feb. 2000.

[10] J. Lladós, E. Marití, and J. J. Villanueva. Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(10), Oct. 2001.

[11] M. MacDougall. *Simulating Computer Systems: Techniques and Tools*. MIT Press, 1987.

[12] M. Maróti, B. Kusy, G. Simon, and Ákos Lédeczi. The flooding time synchronization protocol. In *Proceedings of the second Conference on Embedded Networked Sensor Systems (SenSys)*, Baltimore, Maryland, USA, Nov. 2004.

[13] S. Meguerdichian and M. Potkonjak. Low power 0/1 coverage and scheduling techniques in sensor networks. *UCLA Technical Report*, (030001), Jan. 2003.

[14] V. Muchuruza and R. Mussa. Traffic operation and safety analyses of minimum speed limits on florida rural interstate highways. In *Proceedings of the 2005 Mid-Continent Transportation Research Symposium*, pages 1–10, Ames, Iowa, USA, Aug. 2005.

[15] D. Niculescu and B. Nath. Ad hoc positioning system (aps) using aoa. In *Proceedings of the Conference on Computer Communications (INFOCOM)*, San Francisco, CA, USA, 2003.

[16] C. Oh, A. Tok, and S. Ritchie. Real-time freeway level of service using inductive-signature-based vehicle reidentification system. *IEEE Transactions on Intelligent Transportation Systems*, 6(2):138–146, June 2005.

[17] N. B. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller. Anchor-free distributed localization in sensor networks. *MIT Technical Report*, (892), Apr. 2003.

[18] K. Römer. The lighthouse location system for smart dust. In *Proceedings of MOBISYS*, pages 15–30, San Francisco, CA, USA, 2003.

[19] A. Savvides, C. C. Han, and M. B. Srivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *Proceedings of MOBICOM*, Rome, Italy, July 2001.

[20] A. Savvides, H. Park, and M. B. Srivastava. The bits and flops of the n-hop multilateration primitive for node localization problems. In *Proceedings of the first ACM International Workshop on Wireless Sensor Networks and Application*, Atlanta, GA, USA, Sept. 2002.

[21] R. Stoleru, T. He, J. A. Stankovic, and D. Luebke. A high-accuracy, low-cost localization system for wireless sensor networks. In *Proceedings of the third Conference on Embedded Networked Sensor Systems (SenSys)*, San Diego, California, USA, Nov. 2005.

[22] R. Stoleru, P. Vicaire, T. He, and J. A. Stankovic. Stardust: A flexible architecture for passive localization in wireless sensor networks. In *Proceedings of the fourth Conference on Embedded Networked Sensor Systems (SenSys)*, Boulder, Colorado, USA, Nov. 2006. ACM.

[23] C. Taylor and A. R. J. Bachrach. Simultaneous localization, calibration, and tracking in an ad hoc sensor network. In *Proceedings of the Fifth International Conference on Information Processing in Sensor Networks (IPSN)*, Nashville, TN, USA, Apr. 2006.

[24] S. Umeyama. An eigendecomposition approach to weighted graph matching problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5), Sept. 1988.

[25] B. H. Wellenhoff, H. Lichtenegger, and J. Collins. *Global Positions System: Theory and Practice (4th Edition)*. Springer Verlag, 1997.

A Theorem A.1

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 $G' \subseteq G$. 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 $M(i,j) < M'(i,j)$, then $M(i,j)$ must be the **length of an edge**, which is included in A and excluded in A'.*

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 k-th 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 the lengths of edges in A as follows:

$$M(i,j) = \sum_{k=1}^{m} l(e_k),\ e_k \in A \tag{17}$$

Similarly, $M'(i,j)$ must be the sum of the lengths of edges in A'. We know that $M(i,j) \leq M'(i,j)$ for all $i, j \in V(G)$, since $A' \subset A$. From the given condition $M(i,j) < M'(i,j)$, if there exists an e_k for $k = 1,...,m$ in Eq. 17 such that $e_k \notin A'$, and the $M(u,v)$ 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 $M(i,j)$.

If there exists no e_k for $k = 1,...,m$ in Eq. 17 such that $e_k \notin A'$, and the $M(u,v)$ and $M'(u,v)$ corresponding to the endpoints of the edge $e_k = e_{uv}$ satisfy $M(u,v) < M'(u,v)$, this means that A' has the same edges e_k for $k = 1,...,m$ as A has. Thus, $M(i,j) = M'(i,j)$, since A' can construct the shortest i,j-path with the same edges e_k for $k = 1,...,m$ as A has. This contradicts the condition $M(i,j) < M'(i,j)$. □