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

Jaehoon Jeong, Shuo Guo, Tian He and David Du
Department of Computer Science & Engineering, University of Minnesota
Email: {jjeong,sguo,tianhe,du}@cs.umn.edu

***Abstract*—In road networks, sensors are deployed sparsely (hundreds of meters apart) to save costs. This makes the existing localization solutions based on the ranging be 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 binary 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 outdoor in Minnesota roadways and show that our distance estimate method works well despite of traffic noises. 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.**

I. Introduction

Localization of sensors is a prerequisite step to find target positions for most military applications, including surveillance and target tracking. In these applications, it has been envisioned that for the fast, safe deployment, unmanned aerial vehicles drop a large number of wireless sensors into road networks around a target area. Many localization approaches have been proposed in the context of such a scenario. They use either precise range measurements (e.g., TOA [1], TDOA [2], and AOA [3]) or connectivity information (e.g., Centroid [4], APIT [5], SeRLoc [6], and Gradient [7]), between sensors to locate nodes' locations. Unfortunately, all of them ignore an important fact: To cover a large area, sensors have to be deployed sparsely (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) nor single-hop RF connectivity, 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, it is not important to localize them. Under such a scenario, the research question is *how sensors on/near a road can identify their positions in a sparse deployment without any pair-wise ranging or connectivity information.*

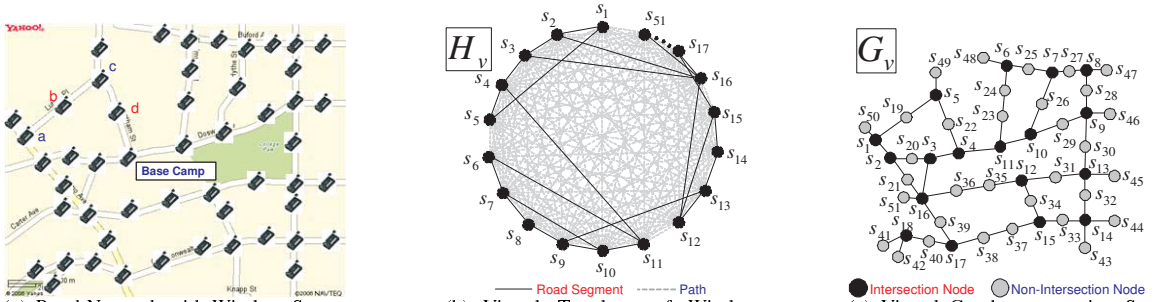
The high-level idea of our solution is to use vehicles on roadways as natural events for localization. The solution would be trivial if all nodes are equipped with sophisticated vehicle identification sensor, because it is relatively easy to measure the distance between two sensors by multiplying vehicles' average speed by Time Difference on Detection (TDOD) between two sensors corresponding to the *same* vehicle. Obviously vehicle identification sensors would be costly in terms of hardware, energy and computation. Therefore, the challenging research question is *how to obtain locations of the sensors, using only binary detection results without the vehicle identification capability in sensors.*

Our main idea is as follows. Through statistical analysis of 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 the known road map. A unique mapping allows us to identify where sensors are located in roadways.

Specifically, 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 key contributions in this paper are as follows:

- A new architecture for autonomous passive localization using only binary detection of vehicles on 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 Time Difference on Detection (TDOD).
- 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.
- A graph-matching algorithm for matching the sensor's identification with a position at the road map of the target area.

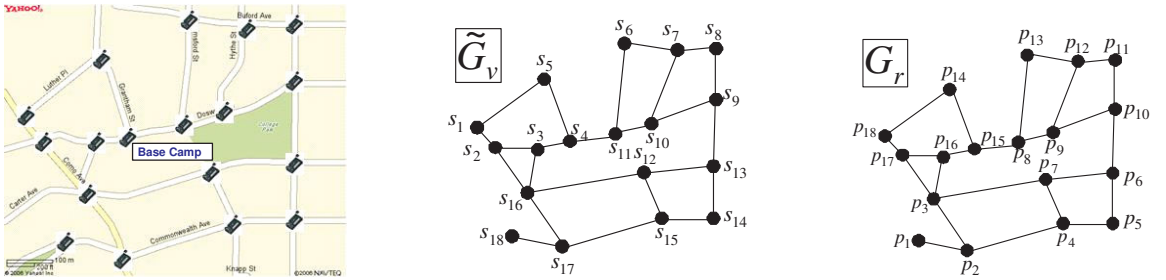
The rest of this paper is organized as follows. Section II describes the problem formulation. Section III explains the system design. Section IV evaluates our algorithm. We summarize related work in Section V and conclude our work in Section VI.



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

(f) Real Graph corresponding to Road Map: $G_r = (V_r, E_r)$

Fig. 1. Wireless Sensor Network deployed in Road Network

II. Problem Formulation

We consider a network 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 as shown in Figure 1(a). We define eight terms as follows:

1. Intersection Nodes Sensors placed at an intersection and having more than two neighboring sensors (i.e., degree ≥ 3). In Figure 1(a), sensors a and c are intersection nodes.

2. Non-intersection Nodes Sensors placed at a non-intersection and having one or two neighboring sensors. In Figure 1(a), sensors b and d are non-intersection nodes.

3. 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 1(b) shows a virtual topology of sensors to the road network, shown in Figure 1(a). M_v is a complete simple graph, since there is an edge between two arbitrary sensors. We define the edge of the virtual topology as *virtual edge*. In Figure 1(b), among the virtual edges, a solid black line represents an *edge estimate* between two sensors, which means that they are adjacent on the road network. The dotted gray line represents a *path estimate* between two sensors, which means that they are not adjacent on the road network.

4. 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 and s_j. Figure 1(c) shows a virtual graph of the sensor network deployed on the road network shown in Figure 1(a), where the black node represents an intersection node and the gray node represents a non-intersection node.

5. 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 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 subgraph $\tilde{G}_v$ is obtained by deleting non-intersection nodes and their edges from the virtual graph G_v through the degree information in G_v. Refer to Section III-D1. For example, Figure 1(e) shows a reduced virtual subgraph consisting of only intersection nodes of virtual graph in Figure 1(c).

6. 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. Real Graph can be obtained through map services, such as Google Earth and Yahoo Maps. Figure 1(f) shows a real graph corresponding to the road network that consists of only intersection points, shown in Figure 1(d). The real graph is *isomorphic* to the reduced virtual subgraph graph $\tilde{G}_v$ shown in Figure 1(e) [8].

7. 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* [9]. 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 $G_v = (V_v, E_v)$.

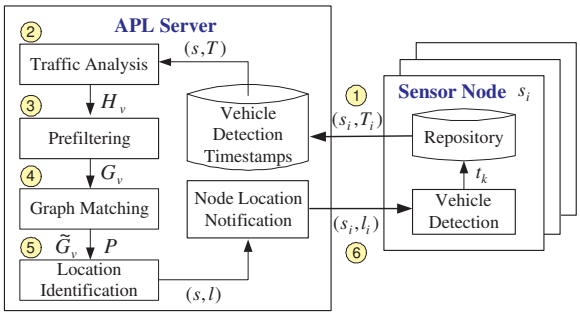


Fig. 2. APL System Architecture

8. APL Server A computer that performs the localization algorithm with binary vehicle-detection timestamps collected from the sensor network.

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 [10]. Each detection is a tuple (s_i, t_j), consisting of a sensor ID s_i and timestamp t_j.
- Sensors are time-synchronized at the millisecond level. This can be achieved easily because many state-of-art solutions [11], [12] can achieve microsecond level accurate.
- 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*.
- 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 [13].

III. APL System Design

A. System Architecture

We use an asymmetric architecture for localization as in Figure 2 in order to simplify the functionality of sensors for localization. As simple devices, sensors only monitor road traffic and register vehicle-detection timestamps 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 Figure 2:

- **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 sensor ID and T_i is timestamps.
- **Step 2**: The traffic analysis module estimates the road segment length between two arbitrary sensors with the timestamp 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 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 by applying the permutation matrix P to both the reduced virtual subgraph $\tilde{G}_v$ 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 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.

B. *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 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, we found that we can estimate the lengths of road segments used by vehicles during their travels on roadways only with vehicle-detection timestamps.

1) Time Difference on Detection (TDOD) Operation: The Time Difference on Detection (TDOD) 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 on Detection (TDOD) 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. 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.

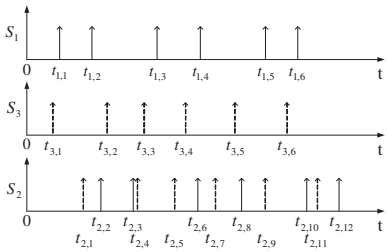
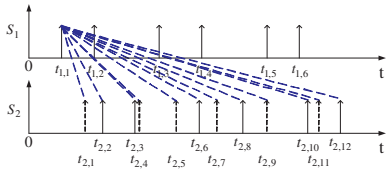
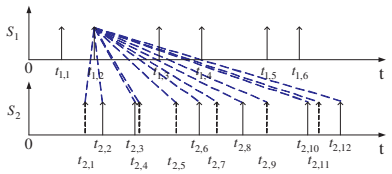


Fig. 3. Detection Sequence for Vehicles at Sensors s_1, s_3, and s_2



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



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

Fig. 4. Time Difference On Detection for Sensors s_1 and s_2

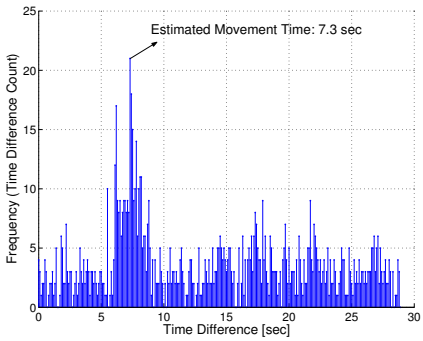


Fig. 5. Estimation of Movement Time through TDOD Operation

We define *frequency* as the count of a discrete time difference. After the TDOD 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 3 shows the detection sequence for vehicles at intersection nodes s_1, s_2, and s_3 in Figure 1(e), where s_2 is a common neighbor of s_1 and s_3. Figure 4 shows the TDOD operation for nodes s_1 and s_2 that is a kind of Cartesian product for two timestamp sets. Figure 5 shows the histogram obtained by the TDOD operation for two timestamp sets. The time difference value (7.3sec) with the highest frequency indicates the estimated movement time between two nodes.

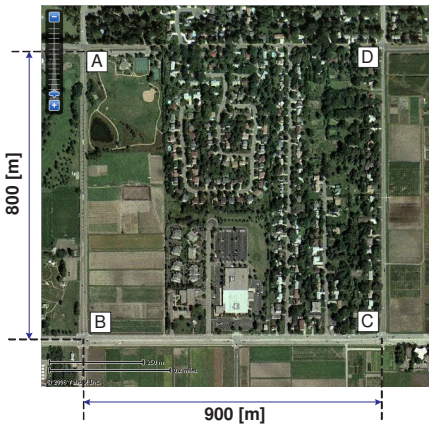


Fig. 6. Road Networks for Outdoor Test

TABLE I
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

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. Figure 6 shows the road map of local roadways in Minnesota for outdoor test. The test roadways consist of four intersections A, B, C, and D. 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 based on motes is our future work.

Table I shows the expected movement times and measured movement times for these four road segments through TDOD. It can be seen that the estimated movement times are close to the expected movement times. Thus, with the TDOD, a virtual topology can be constructed, as shown in Figure 1(b), containing the distance between two arbitrary nodes, called *virtual edge*.

2) 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 TDOD 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 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

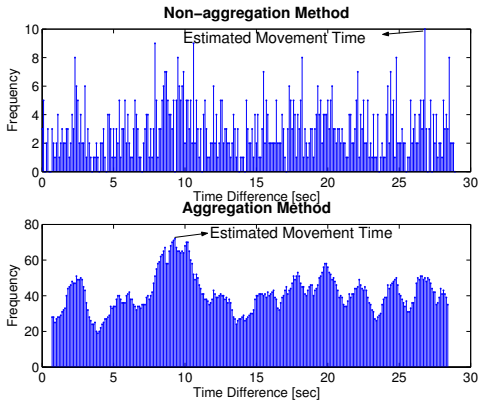
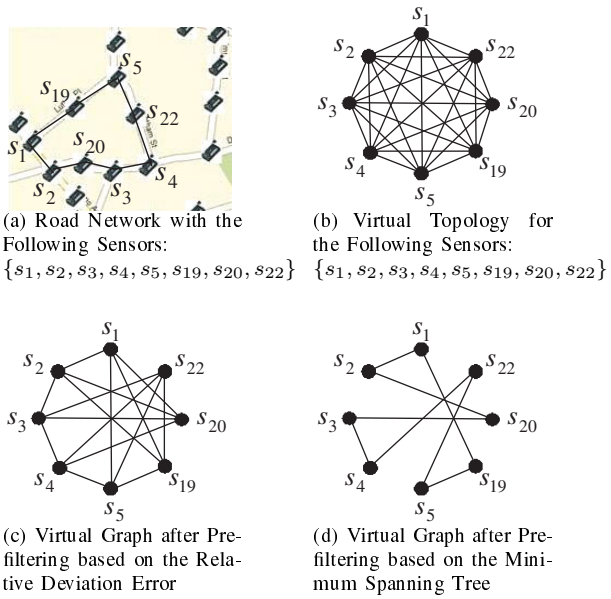


Fig. 7. Comparison between Non-aggregation Method and Aggregation Method

frequencies by the TDOD operation for two timestamp series, as shown in Figure 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 as the *Aggregation Method* and the previous simple TDOD is called as the *Non-aggregation Method*. We determine the aggregation window size proportionally to standard deviation σ_v of the vehicle speed, such as $c \cdot \sigma_v$ for $c > 0$.

Figure 7 shows the comparison between the non-aggregation method and aggregation method through simulation. We found that for the road segment between sensors s_2 and s_3 in Figure 1(e) whose real time difference is 9.36 sec with the vehicle speed μ_v=50 km/h, the non-aggregation 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 is used to obtain good estimates for road segment lengths in virtual topology.



(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

Fig. 8. Procedure of Prefiltering for obtaining Virtual Graph

C. *Step 3*: *Prefiltering Algorithm for a Virtual Graph*

The prefiltering algorithm is performed to make a virtual graph that has only edge estimates from the virtual topology obtained from the TDOD operations in Section III-B. We observe that the TDOD operation discussed in Section III-B 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 TDOD operation are much more accurate due to the high correlation of the timestamps. From this observation, we filter out all inaccurate path estimates from the virtual topology, except for edge estimates so that the virtual topology can be 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 8. As shown in Figure 8(a), there is a partial road network of the entire one shown in Figure 1(a) 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 Figure 8(b). 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 Figure 8(c). 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 in Figure 8(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.

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, they can have any value in M_v, that is, either a large value or a small value. In this case, the smallest entry will be no longer regarded 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 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, $\phi = \sigma/\mu$. 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 TDOD 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., ε = 5%), the corresponding entry is regarded as a path estimate, and it is replaced with ∞, indicating that this entry is a path estimate.

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 two steps: The first step identifies the first $n-1$ edges of the virtual graph, and the second step identifies the remaining edges 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* [9]. We have proved that the $n-1$ edges that form the MST are definitely edge estimates in our technical report [14].

Step 2: In order to find all of the other edges of the virtual graph $G_v = (V_v, E_v)$, as shown in Figure 1(c), with $n-1$ edges 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 $M'_v(i,j) \geq M_v(i,j)$. 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 the one created from more edges; that is, $M_v(i,j)$ might be shorter than $M'_v(i,j)$. In our technical report [14], 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 large error after the previous filtering. Consequently, $M_v(i,j)$ is the n-th edge estimate. 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.

D. *Step 4: Graph Matching*

In this section, we explain how to construct a reduced virtual subgraph from the virtual graph constructed by the prefiltering in Section III-C, and then how to match the reduced virtual subgraph and the real graph that are *isomorphic* to each other [8].

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. From the observation that each intersection node has at least three neighboring sensors, a reduced virtual subgraph $\tilde{G}_v$ is made 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 $l(e_{ux}) + l(e_{uy})$.

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:

$$\Phi(P) = \|E_r - P\tilde{E}_v P^T\|_2^2 \tag{4}$$

$$P \leftarrow \arg\min_{\hat{P}} \Phi(\hat{P}) \tag{5}$$

$$\hat{E}_v \leftarrow P\tilde{E}_v P^T \tag{6}$$

Let P be an $n \times n$ optimal permutation matrix of Eq. 5 in terms of the minimum estimation error. The result $\hat{E}_v$ of Eq. 6 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 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 eigendecomposition approach to WGMP [15], known as an optimal approach. For our graph matching purpose, we adopt the eigendecomposition 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 on this subject. One can refer to our technical report [14] for detailed information.

E. *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 III-D, and then how to identify the location of each non-intersection node.

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

$$\sigma : s \in \{1, ..., n\} \rightarrow p \in \{1, ..., n\}, \tag{7}$$

that is, $p = \sigma(s)$ where s is the sensor ID and p is the intersection ID. With the permutation function in Eq. 7, we can identify the intersection ID (p) on the road map for each intersection node (s).

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 of the virtual graph G_v, we begin from an intersection node u, and we 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 $i = 1, ..., m$ 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 $\tilde{G}_v$, as shown Figure 1(e), 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 Figure 1(c). We repeat this procedure until we localize all of the non-intersection nodes in the virtual graph.

IV. 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 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 SectionIII-D. 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. We just deploy intersection nodes for simplicity.

TABLE II
Simulation Environment

Parameter	Description
Number of sensors	18 sensors (from s_1 to s_{18}) are deployed in the road network, as shown in Figure 1.
Simulation time	Sensors perform vehicle detection for 10 hours and store the vehicle-detection timestamps into their repositories.
Time synch. error	Sensor time synchronization error conforms to a uniform distribution with the interval $[-\epsilon_{max}, \epsilon_{max}]$ 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's maximum speed is 80 km/h and vehicle's minimum speed is 20 km/h.
Interarrival time	Every vehicle arrives at road network according to an exponential distribution with mean interarrival time $1/\lambda = 120$ 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.

The simulation environment based on SMPL [16] is described in Table II. 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-pairs shortest path estimation matrix for the virtual topology.

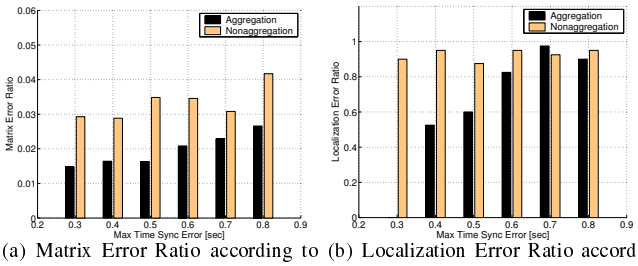
A. Performance Comparison between Road Segment Estimation Methods

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 III-C and the graph matching algorithm described in Section III-D in order to evaluate the *Matrix Error Ratio* and *Localization Error Ratio*.

For the maximum time synchronization error, Figure 9 shows the performance comparison between the aggregation and nonaggregation 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 1, where $\tilde{G}_v$ is a subgraph of the virtual topology H_v. It can be seen 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 nonaggregation method. From Figure 9(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 microsecond level [11], [12].



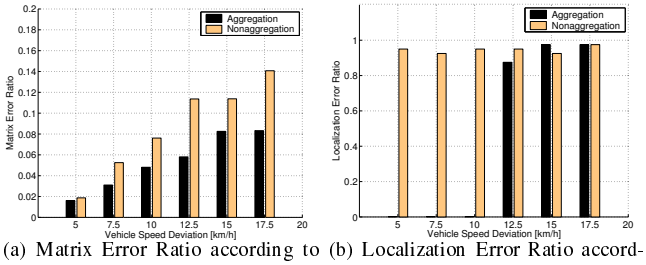
(a) Matrix Error Ratio according to Maximum Time Synchronization Error

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

Fig. 9. Performance Comparison between Aggregation and Nonaggregation Methods for Maximum Time Synchronization Error (ϵ_{max})

For the vehicle speed deviation, as shown Figure 10, the aggregation method outperforms the nonaggregation method in that the *Matrix Error Ratio* of the aggregation method is less that that of the nonaggregation method. Also, that is why the aggregation method can give more accurate localization than the non-aggregation method, except for the vehicle speed deviation of 15 km/h. This speed deviation of 15 km/h is the value out of the operational region for our localization scheme, so the corresponding localization error ratio is always a random value close to 1. However, considering the real statistics [13] 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,

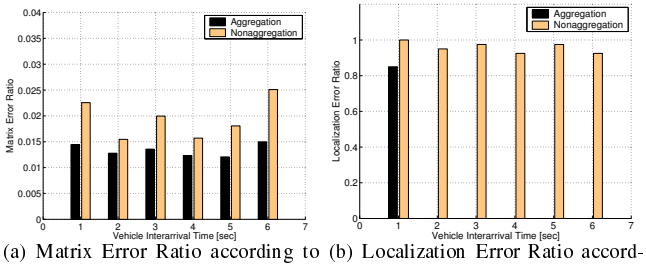
it can be claimed that our localization can work in the real environment, since our localization scheme works with the vehicle speed deviation less than 10 km/h.



(a) Matrix Error Ratio according to Vehicle Speed Deviation (b) Localization Error Ratio according to Vehicle Speed Deviation

Fig. 10. Performance Comparison between Aggregation and Nonaggregation Methods for Vehicle Speed Deviation (σ_v)

For the vehicle interarrival time, as shown Figure 11, we see that it does not affect the performance of our localization scheme. The reason is that our TDOD operation can give accurate estimates for road segment lengths, as long as the vehicle interarrival time is larger than 1 second and it allows enough road traffic to cover all of the road segments. In fact, most people drive their vehicles with the interarrival time longer than 1 second for their safety, so we can claim that our localization works under 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. Also, we can see that all of the *Matrix Error Ratios* of the aggregation method are less than those of the nonaggregation method.



(a) Matrix Error Ratio according to Vehicle Interarrival Time (b) Localization Error Ratio according to Vehicle Interarrival Time

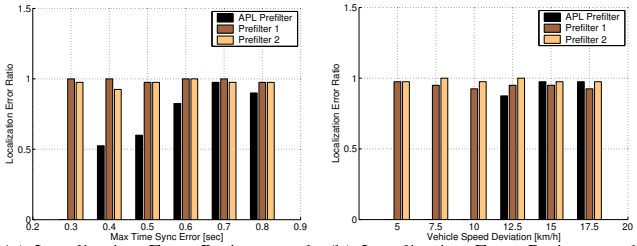
Fig. 11. Performance Comparison between Aggregation and Nonaggregation Methods for Vehicle Interarrival Time ($1/\lambda$)

B. 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 III-C2,
2) *Prefilter 2*: Prefiltering based on the relative deviation error described in Section III-C1, 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 III-D is applied to the output matrix $\tilde{E}_v$ in order to evaluate the *Localization Error Ratio*. From Figure 12, 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 in Figure 12, 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 the accurate localization under the reasonable, real environment. This is why we use the combination of two prefilters.



(a) Localization Error Ratio according to Maximum Time Synchronization Error (b) Localization Error Ratio according to Vehicle Speed Deviation

Fig. 12. Performance Comparison among Prefiltering Types

C. APL Operational Region

We evaluate *APL* to see what range of time synchronization and vehicle speed deviation it works well in. Figure 13 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 II. 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. 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) [13]. For the vehicle interarrival time, our localization works well as long as the interarrival time is greater than 1 second. Thus, the vehicle speed deviation is the dominant factor of the performance in our localization scheme.

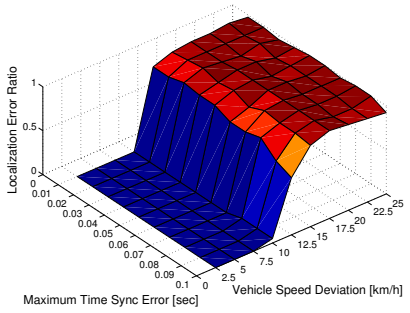


Fig. 13. APL Operational Region for Maximum Time Synchronization Error and Vehicle Speed Deviation

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*. The result is that our localization scheme has no localization error under the simulation setting in Table II with the detection missing probability from 0 to 0.2 at each sensor and with the duplicate detection probability from 0.1 to 1 at each sensor, respectively. Thus, it can be claimed that our localization scheme can work in the real road networks with noises.

We have considered several practical issues in our extended technical report [14] for the deployment of our localization scheme in real road networks: (a) graph matching under intersection node missing and (b) matching ambiguity due to topology symmetricity. Due to space constraints, we cannot explain them in detail.

V. 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-driven 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) (e.g., GPS [1]) and Time Difference of Arrival (TDOA) schemes (e.g., Cricket [2] and AHLoS [17]) measure the propagation time of the signal, and estimate the distance based on the propagation speed. Since ultrasound signals usually propagate only 20$\sim$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 [18] use either theoretical or empirical models to estimate the distance based on the loss of power during signal propagation. Both AOA and RSSI are also constrained by their effective distance.

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. The main idea is that the non-anchors can determine their locations using the overlapped region of communication areas for the anchors [4], [5], [19], [20]. However, since these schemes require a dense deployment of anchors to give beacon signals, these solutions are not applicable for the localization in sparse road 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 [21]–[24]. Although their effective range can reach hundreds of meters, it needs additional external devices and manual operations 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 natural events of moving vehicles, there is no such problem of the event delivery.

VI. Conclusion

In sparse sensor networks, sensors cannot effectively obtain pair-wise 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 the localization using vehicle-detection timestamps along with the road map of target area. As next step, we will perform the test of our *APL* system in real road networks with Motes such as XSM and Micaz.

Acknowledgment

This research was supported by the Digital Technology Center at the University of Minnesota and in part by NSF grant CNS-0626614, CNS-0615063, and CNS-0626609.

References

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

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

[3] D. Niculescu and B. Nath, "Ad Hoc Positioning System (APS) using AoA," in *INFOCOM*. San Francisco, CA, USA: IEEE, Mar. 2003.

[4] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less Low Cost Outdoor Localization for Very Small Devices," *IEEE Personal Communications 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," in *MOBICOM*. ACM, Sep. 2003.

[6] L. Lazos and R. Poovendran, "SeRLoc: Secure Range-Independent Localization for Wireless Sensor Networks," in *WiSe*. ACM, Oct. 2004.

[7] D. Moore, J. Leonard, D. Rus, and S. Teller, "Robust Distributed Network Localization with Noise Range Measurements," in *SENSYS*. ACM, Nov. 2004.

[8] D. B. West, *Introduction to Graph Theory (2nd Edition)*. Prentice Hall, 2000.

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

[10] Lin Gu et al., "Lightweight Detection and Classification for Wireless Sensor Networks in Realistic Environments," in *SENSYS*. San Diego, California, USA: ACM, Nov. 2005.

[11] J. Elson, L. Girod, and D. Estrin, "Fine-Grained Network Time Synchronization using Reference Broadcasts," in *OSDI*. ACM, Dec. 2002.

[12] M. Maróti, B. Kusy, G. Simon, and Ákos Lédeczi, "The Flooding Time Synchronization Protocol," in *SENSYS*. Baltimore, Maryland, USA: ACM, Nov. 2004.

[13] 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*, Ames, Iowa, USA, Aug. 2005.

[14] J. Jeong, S. Guo, T. He, and D. Du, "APL: Autonomous Passive Localization for Wireless Sensors deployed in Road Networks," *Technical Report of University of Minnesota*, no. 07-016, Jul. 2007.

[15] S. Umeyama, "An Eigendecomposition Approach to Weighted Graph Matching Problems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 5, Sep. 1988.

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

[17] A. Savvides, C. C. Han, and M. B. Srivastava, "Dynamic Fine-Grained Localization in Ad-Hoc Networks of Sensors," in *MOBICOM*. Rome, Italy: ACM, Jul. 2001.

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

[19] C. Taylor and A. R. J. Bachrach, "Simultaneous Localization, Calibration, and Tracking in an ad Hoc Sensor Network," in *IPSN*. Nashville, TN, USA: ACM/IEEE, Apr. 2006.

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

[21] Z. Zhong and T. He, "MSP: Multi-Sequence Positioning of Wireless Sensor Nodes," in *SENSYS*. Sydney, Australia: ACM, Nov. 2007.

[22] R. Stoleru, T. He, J. A. Stankovic, and D. Luebke, "A High-Accuracy, Low-Cost Localization System for Wireless Sensor Networks," in *SENSYS*. San Diego, California, USA: ACM, Nov. 2005.

[23] R. Stoleru, P. Vicaire, T. He, and J. A. Stankovic, "StarDust: A Flexible Architecture for Passive Localization in Wireless Sensor Networks," in *SENSYS*. Boulder, Colorado, USA: ACM, Nov. 2006.

[24] K. Römer, "The Lighthouse Location System for Smart Dust," in *MOBISYS*. San Francisco, CA, USA: ACM, May 2003.