SRVC: Sensornets for Remote Vehicle Classification Final Report

METRANS Project (METRANS-07-04)

John Heidemann (PI), Chengjie Zhang, Unkyu Park
USC/Information Sciences Institute

19 July 2012

Disclaimer

Abstract

The motivation of this research is the need for short-term data collection about freight traffic movement in urban areas. We seek to develop automated vehicle classification systems based on networks of small, battery-powered and wireless, intelligent sensors that can be easily deployed with brief setup time (tens of minutes), with accurate vehicle information (as good as or better than human observers), and communicate this information to a central monitoring site. Current approaches are not rapidly deployable, accurate enough, and lack the ability to relay data in real-time to central site. This report summarizes the results of a two-year research effort on sensornets for vehicle traffic classification. At the study outside, our main goals were to understanding the communications requirements for traffic monitoring systems (both short-range wireless inside a traffic sensornet, and wide-area to a central Traffic Management System), developing self-configuring traffic monitoring systems, and integrating prior work with these new results. The key outcomes of our research are to understand communications in a classification system, to carefully evaluate the effects of signature matching, and to further develop self-configuration. Finally, we tested our new approaches through a second major data collection exercise at USC.

Contents

List of Figures

List of Tables

1 Introduction

This report summarizes the research done through the the Sensor for Sensornets for Remote Vehicle Classification (SRVC) project, covering the time period from August 2006 though Sept. 2010. Because of delays starting up the project, discussed in Section 1.3, we did not begin the project until August 2007, and funding allowed us to extend the project through Sept. 2010.

SRVC is applying sensor network to short-term data collection about freight traffic movement in urban areas. We are developing automated vehicle classification systems based on networks of small, battery-powered and wireless, intelligent sensors that can be easily deployed with brief setup time (tens of minutes), with accurate vehicle information (as good as or better than human observers), and communicate this information to a central monitoring site. A prior project, SURE-FT project [17] and SURE-SE project [19] defined this problem, reviewed related work and the current state-of-the-art (summarized in Section 2), and conducted a preliminary experiment at USC to collect data (described in Section 7.1.1). Current approaches are not rapidly deployable, accurate enough, and lack the ability to relay data in real-time to central site. This final report describes our work to address this gap, drawing on material from our technical report [52].

At the study outside, our main goals were to understanding the communications requirements for traffic monitoring systems (both short-range wireless inside a traffic sensornet, and wide-area to a central Traffic Management System), developing self-configuring traffic monitoring systems, and integrating prior work with these new results. The key outcomes of our research are to understand communications in a classification system, to carefully evaluate the effects of signature matching, and to further develop self-configuration.

The reminder of this introduction reviews the need of this research (Section 1.1), limitation of current approaches (Section 1.2). The remainder of the report reviews prior work (Section 2), describes our proposed system (Section 3), the algorithms for single-sensor vehicle classification (Section 4), the algorithms for multi-sensor autoconfiguration (Section 5), and signature matching (Section 6),. We then use data collected from old and new experiments (Section 7) to test our algorithms with simulation (Section 8) and evaluation and analysis (Section 9).

1.1 Background and Justification of Research

1.1.1 Research Directions

The motivation for this work is the need for short-term data collection about freight traffic movement in urban areas. Applications for CALTRANS and METRANS include *(1) monitoring truck traffic to estimate diesel pollution* around port or distribution facilities, and *(2) collection of traffic data to improve traffic models*. We describe applications in more detail below in Section 1.1.2.

We seek to develop automated vehicle classification systems based on networks of small, battery-powered and wireless, intelligent sensors that can be *easily deployed* with brief setup time (tens of minutes), with accurate vehicle information (as good as or better than human observers), and *communicate this information to a central monitoring site*. We review limitations of current approaches (Section 1.2).

The main new research challenges addressed in this work are understanding the communications requirements for traffic monitoring systems (both short-range wireless inside a traffic sensornet, and wide-area to a central Traffic Management System), developing self-configuring traffic monitoring systems, and integrating prior work with these new results. We discuss these research challenges in Section 3.2, 3.4 and 5.

Our work is focused on data collection for METRANS topic area 1: Commercial goods movement and international trade issues in metropolitan areas, with an emphasis on detecting truck traffic to estimate and model potential diesel pollution.

Although data collection is the primary motivation for our work, the technology we propose would also be useful to manage traffic and improve worker safety around construction areas (topic area 4), and to assist in traffic flow in emergency situations such as a major evacuation (topic area 4), or to support data collection for port area monitoring for hazardous substances or terrorist activity (topic area 4).

1.1.2 Need for Interactive Information About Freight Traffic

There is a compelling need to better understand vehicle traffic in our cities. Not only is information about local traffic useful to assist directly in pollution monitoring, urban planning, development planning, and traffic management, but data sources are needed to validate and extend regional models of freight traffic to predict future trends and the effects of policy changes.

Specific applications include:

- to understand the environmental impact of truck queuing and operation
- to consider impacts of new development (such as housing or commercial centers)
- to evaluate development plans or pricing policies [27, 35, 49]
- to develop models of local traffic [16, 3, 46, 47]
- to develop metropolitan models of freight traffic [14, 44]

Concern about the environmental impact of freight traffic is motivation for better data collection. The California law SB 2650 imposes fines on terminal operators if trucks entering the terminal idle for more than 30 minutes. This law is intended to reduce diesel emissions. At present, the law is enforced by a roving officer who estimates idle time based on his assessment of queue length. Estimates of queue length and time spent in the facility could allow for more accurate enforcement. More importantly, if terminal queue lengths could be shared with truck operators as they approach the port area, they could select underserved terminals, reducing queuing times, pollution, and increasing their productivity and income.

There is also a particular need for a better understanding of truck traffic and freight modeling. Since many truck drivers are independent contractors, obtaining data from the drivers is problematic. State highway transportation departments have weigh-in-motion (WIM) stations, but there are relatively few stations (only 84 in all of California), and so they provide little insight to intra-urban traffic. Data about freight traffic on arterial roadways is almost non-existent, because there is no easy way to collect such data.

There are several key problems with automating of the collection of vehicle information. First, is developing a system that can be taken into the field to collect short-term measurements where needed while providing comparable or better accuracy to current approaches (particularly human observers). Our prior work [34] demonstrated that this goal can be achieved by combining the results from multiple independent sensors.

1.2 Limitations of Current Approaches

There are a number of current approaches to observe vehicle traffic. We next briefly review centralized traffic management systems, portable traffic monitoring systems, and how portable systems integrate with centralized TMS. (We cover related work in more detail in Section 2, Literature Review.) In the next section we consider very recent work in applying sensor networks to vehicle traffic monitoring.

Permanent monitoring systems: Because of the high value for the efficient operation of urban transportation systems, there has been extensive investment in permanent monitoring infrastructure, particularly inductive loops and video observation, and a great deal of research on how to best use this infrastructure. We provide an exhaustive survey of this work (as of January 2005) in our final report for a previous METRANS project [19], including a review of the use of inductive loops, video image processing, and weigh-in-motion systems.

While essential to the management of urban traffic, all permanent monitoring systems suffer the limitation that they can *only monitor traffic where they are installed.* Installation of new sensors requires considerable time and expense, typically including placing sensors under pavement, interrupting traffic for an extended duration, and purchasing wired telecommunications equipment from telephone companies or deploying fiber optic communications capability down road right-of-ways. A typical deployment at a new intersection will cost tens-of-thousands of dollars and require weeks of planning. Thus, while an important part of overall traffic management, they do not meet our need for short-term data collection and at many different and changing of locations.

Deployable monitoring systems: The use of *portable* traffic monitoring systems is much more limited. In spite of the large amount research done on traffic sensors, systems for temporary deployment often fall back on simple pneumatic tubes, or manual, human observers, either physically present or interpreting videotape.

There are currently several proposed research systems that provide deployable traffic monitoring systems, including those of Oh *et al.* [31], Sun *et al.* [45], Cheung *et al.* [6, 7], and our prior work [19, 34]. These systems evaluate both the problems of vehicle re-identification and accurate vehicle classification.

Oh *et al.* use an IST Blade sensor [21], targeted at vehicle reidentification at arterial speeds (20–45mph). They propose a neural network for vehicle re-identification using several features derived from the IST sensor signature without computing vehicle speed and length. Sun *et al.* add a neural network to loop sensor output, explicitly trying to differentiate a custom set of categories including cars separate from SUVs, as well as larger vehicles. They report good accuracy, 82–87%, although they do not indicate if their errors are in the difficult-to-distinguish categories or not.

For vehicle classification, Cheung *et al.* instead use custom sensor nodes with magnetometers, measuring the change in the Earth's magnetic field. They use both length-based classification and a novel "Hill Pattern Classification". While they obtain highly accurate vehicle counts (98%), their classification accuracy (82% into 5 FHWA classes corresponding to different axle-count large trucks) [7] is slightly better than ours (74% into 3 FHWA classes corresponding to passenger cars, small trucks, and large trucks). However, it is important to note that their classes are quite distinct and can be distinguished by axles counts and large differences in the length. We have evaluated the use of an IST blade sensor to classify even the most similar FHWA classes (2 and 3, cars vs. small trucks) [34]. We also explore the use of multiple features in a single signal, as well as multiple independent sensors to improve classification accuracy.

Some other semi-mobile traffic monitoring systems exist. A field operational test was conducted using a mobile surveillance and wireless communication system. The system was installed on a trailer that could be placed at roadsides. The core of the system was video image processing (VIP), used for vehicle detection, traffic volume, speed, and occupancy. Wireless radio allowed communications with the local TMC. The system was powered by a propane-powered generator [29]. A portable TMC was developed by the Minnesota DOT as part of a Smart Work Zone project. The system is mounted on portable skids and includes vehicle detection and surveillance, traffic control, driver information (via changeable message signs) and communications [29]. Plans for developing a low cost portable system for wireless measurement and observation at University of Minnesota were recently canceled (personal correspondence with PI). Although relevant, these systems are orders of magnitude larger and more expensive than the approaches envisioned in our work and so they do not allow widespread use.

This prior work demonstrates the feasibility of portable traffic monitoring systems and sets a baseline for the current work. However, while this prior work has demonstrated the effectiveness of traffic monitoring with *off-line* deployable systems, none has integrated traffic monitoring with central traffic management systems, or with on-line control by researchers, and they only begin to address the challenges in making such a system easy to deploy. These areas are focus of our work.

Direct measurement of diesel pollution: Rather than measuring trucks and modeling pollution, one might attempt to directly measure pollution. Unfortunately, sensors for fine particulate matter are large, expensive, and require post-facto measurement. While research efforts are attempting to miniaturize particulate sensors, current state-of-the-art is to measure vehicles model pollution, making deployable vehicle monitoring systems all the more important.

Integration of deployable systems with a central TMS: Although many *permanent* systems tie sensors into a central TMS, it is quite difficult to incorporate easily deployable sensors into centralized systems. Communications cost is a major barrier, since the setup costs of wireline communication (for example, telephone lines) are prohibitive, and the per-byte costs of metro-area wireless (for example, cellular telephones, CDPD, Cellular Digital Packet Data; GPRS, General Packet Radio Service, etc.) are also quite expensive (typically $70–100 per connected computer per month).

Neither do systems today employ multiple, cooperative sensors in a peer-to-peer fashion. Dual inductor loops are common, for example, to estimate speed, loops at an intersection are often connected together, and individual sensors often feed their readings to a central TMS. But larger, deployable systems, as would be needed to monitor a port area, are not available today.

We have discovered only a few examples where sensors communicate among themselves, and in these cases communication is usually only between directly connected, wired sensors, thus greatly limiting the area of coverage. The Raad and Lu sensor array utilizes several sensors located at uniform distance intervals to estimate flow rate, speed profile and other traffic characteristics [39]. The Abreu et al. video-based system goes a step further, assigning agents to each camera and allowing agents to exchange information [1]. This experimental research is closest to our approach, but unlike ours it employs relatively few, fairly powerful nodes (capable of image processing), thus raising questions about cost-effectiveness.

1.3 Project Administration

SRVC was selected for METRANS funding originally to start in Fall 2006 for two years. To use the METRANS support most effectively we have stretched the project timeline, extending the period over which the grant is spent.

We delayed the start of Year 1 from Fall 2006 to Fall 2007, and the end of Year 1 to December 2008. The start was delayed because SRVC could not start until the final report for the PI's prior contract was completed, and because we choose not to begin work on SRVC until we had a PhD student to carry out the work. We therefore began work in earnest in September 2007. Through careful budget management we were able to stretch Year 1 funding through December 2008.

We began Year 2 efforts in January 2008. Because of shifts in graduate student funding (due to requirements of teaching assistantships and other research efforts) we were able to extend research at no additional cost through Sept. 2010.

2 Literature Review

Careful reviews of current work was both given in SURE-SE and SURE-FT projects. Chapter 2 of the SURE-SE final report considers current work on novel sensors, approaches to vehicle classification, and approaches to reidentification [19]. (We briefly summarize this above in Section 1.2.) In SURE-FT, we evaluated the use of multiple small sensors to collaboratively classify vehicles.

The focus of SRVC is to propose a integrated system using multi-sensor for vehicle classification. We therefore review related work in sensor networks below, considering prior work in vehicle tracking, classification, and sensor fusion.

2.1 Sensing for Vehicle Tracking in Constrained Environments

In-road traffic sensors are ubiquitous in most urban environments. The need for efficient traffic flow has sparked significant investment in novel uses of both existing and new sensors. A number of sensor technologies have been considered. Pneumatic tubes and piezoelectric sensors detect wheel crossings; inductive loops and magnetic sensors detect vehicle mass; and infrared, ultrasound, radar or laser ranging, and video, employ different levels of imaging. We survey these elsewhere [19]. Important differentiators here are ease of deployment, robustness, and cost. In-roadway inductive loops are widely used and quite robust, but require construction to deploy. Video approaches remain relatively expensive, both in use and deployment (which may require elevation).

In spite of the large amount of research done on traffic sensors, systems for temporary deployment often fall back on simple pneumatic tubes, or manual, human observers, either physically present or interpreting videotape. For our work we use portable inductive loops because they retain the deployability of tubes but can provide much greater information.

Closest to our work is that of Coifman [9, 8] Cheung *et al.* [6, 7] Kwong *et al.* [26], Oh *et al.* [31], and Sun *et al.* [45]. Coifman proposes a system for freeway deployment [9, 8]. His algorithm looks for short sequences of measured vehicle lengths that exhibit a strong correlation between two stations, namely downstream and upstream sensor nodes. The algorithm is similar to our *Numbering with Resynchronization* (Section 6.1.2), although he employs vehicle length and pattern matching to correct for lane changes. He get about 65% vehicle matching accuracy at highway speeds. We show better correctness (up to 78%), but at much slower speeds. Cheung et al. consider vehicle classification [6, 7], and study the matching problem. They use an array of seven, close spaced, three-axis magnetometers and study seven vehicles on an arterial. They show an impressive 100% re-identification rate. Our results show lower accuracy, but over more than 100 real-world vehicles with sensors at 100 m separation. Kwong et al. use a statistical model of signatures for vehicle re-identification for travel time estimation [26]. They claim no ground truth is needed and their estimated matching rate is about 69%. We instead assume vehicle travel time is relatively stable and our time-stamp based algorithms have recall above 73%. Oh et al. use heterogeneous detection systems for vehicle re-identification [31]. They extract rich feature information from each individual vehicle, match them with a lexicographic optimization algorithm, then use the matching for travel-time estimation. Their approach can be computationally expensive; our approaches are quite simple by comparison and generally suitable to run on-line. Our WETW is similar to their prioritized time window algorithm; but we show other algorithms can do almost as well as it, even without extensive features. Sun et al. use a combined inductive loop detector and image processing [45]. Their result is encouraging, with up to 91% matching rate in the best case. However, the performance is sensitive to fusion weight and video quality from two stations. We discuss parameter sensitivity of our algorithms in Section 9.2. As with Coifman [9], they consider platoon-comparison and assume highway conditions, while we instead study slower traffic.

Three other groups have proposed different method travel time estimation, a part of our time-stamp based algorithms. Jeong et al. provides a method by the frequency of the vehicle time stamp difference between sensors [23]. Dailey [11] and Petty et al. [37] have looked at cross-correlation of raw signatures. Because of signature size, they look at options to downsample or aggregate raw signatures. We show that much less information can provide better correctness (Section 6.3).

We have previously looked at sensor fusion to improve classification accuracy [34]. This work assumed perfect signature matching (an "oracle") and showed that sensor fusion can improve classification accuracy. Here we re-evaluate that work using a realistic signature matching, showing that errors in matching and classification are correlated, so overall accuracy is higher than expected.

2.2 Sensing for Vehicle Tracking in Unconstrained Environments

Much early work in sensornets considered target tracking in unconstrained environments. Early work used dense networks of sensors tracking relatively sparse targets, [43, 40, 32]. Zhao et al. use information theoretic techniques for better vehicle path estimation [53]. Shin et al. consider overlapping targets and use information about distinct targets to clarify the status of targets near each other [42]. As with this prior work, we are concerned with confusion of observations about target near each other (the mis-segmentation in road traffic [34]); although for us the roadway constrains target location (a simplification), greater speed and lower separation complicate our problem. We will discuss our approaches in Section 6.

Computer vision provides an alternative approach to tracking. Pahalawatta et al. employ Affine Gaussian Scale Space to match image according to the feature point detected [33]. The technique was introduced by Baumberg to cope with the situation that naïve direct correlation coefficient comparison is not enough or even impractical [4]. In Section 6.3, we face a similar problem. Besides, they are using "Best-n-match" method, which is infeasible in our scenario because of the uncertainty of incoming vehicle number.

2.3 Sensor Fusion for Improved Accuracy

Sensor fusion is an important approach to exploit multiple sensors. Zhao *et al.* use information theoretic techniques to coordinate cooperation in a multi-sensor environment, selecting the sensor based on maximum information gain [53]. Brooks *et al.* explore collaborative signal processing and identify the level of sensor independence (how correlated or uncorrelated each is with others) as an important issue [5]. Gu *et al.* exploit cluster-based processing to correlate readings from multiple sensors [15].

3 System Description

Our long-term goal is to develop multi-sensor networks that allow automatic accurate vehicle classification and re-identification. This section describes the hardware we propose to deploy to accomplish this task. In the next section we describe the algorithms we have developed for classification and to support deployment.

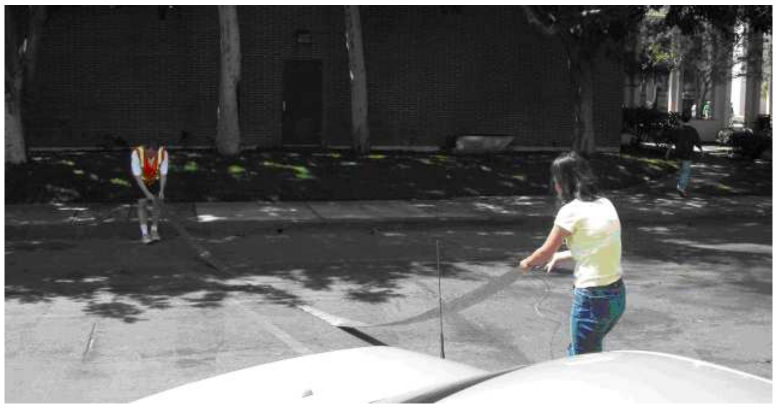


Figure 1: Deployment of a narrow inductive loop for a blade sensor.

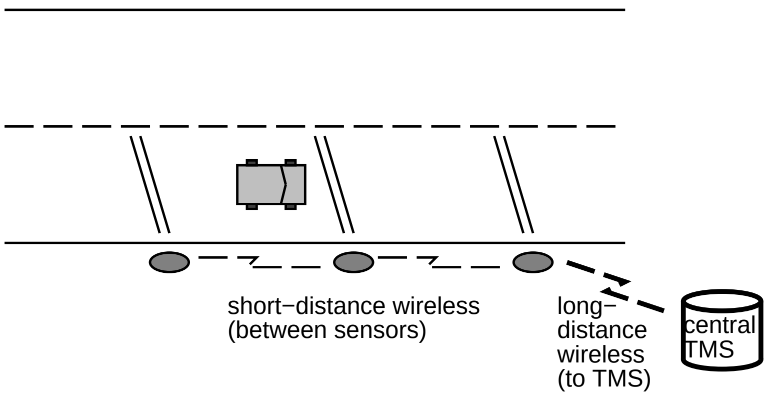


Figure 2: Deployment of three sensor nodes and six sensors along a roadway with both short- and long-range wireless communication.

3.1 System Components

Our focus is a sensor network that is rapidly deployable, low cost, and sufficiently accurate. Our basic sensor network consists of several individual sensor nodes, each connected to a IST-222 high-speed detector card and a pair of Blade inductive loop sensors [20], which was built in SURE-FT [17]. We selected the IST detector card because of its potential for sampling at high resolutions (up to 1.2 kHz), and the Blade inductive loop because of its sensitivity to vehicle features. The IST detector can be integrated with sensor network platforms such as the Intel Stargate [10] via USB, and we expect to use networks such as 802.15.4 for low-power, short-range communication. The systems as a whole should be relatively inexpensive (currently less than US$4000). The complete system is easily deployable since both the sensor nodes and the Blade sensor can operate on battery power. Pairs of inductive loops must be taped down across traffic lanes using asphalt tape, thus requiring a brief interruption of traffic. Figure 1 shows deployment of a pair of loops as part of our trial. Figure 2 shows the logical configuration of our array of traffic sensors: each individual sensor node connects two closely separated blade sensors (a car is shown approaching the middle pair of sensors).

We report here on analysis conducted with data from two locations. First, we discuss our single sensor algorithms using data from one location. We then address multi-sensor fusion using data from two locations. We have not yet integrated our multi-sensor algorithms with communications and evaluate them, both on-line and with post-facto analysis to vary parameters.

System Components Year 1 Status: Although we expect to add software components to our work (for example, approaches to auto-configuration discussed in Section 5), we do not anticipate major changes to system components.

System Components Year 2 Status: In year two we carried out a data collection experiment (Feb. 2009) using our prototype system as described in Section 7.3. We also conducted extensive off-line analysis of the data we collected

3.2 Connectivity

For connectivity, our research goal is to develop traffic classification algorithms that exploit current standards and research protocols to allow cost-effective wireless networks of traffic sensors. There are two sub-problems here: communication between sensors themselves, and communication of results to a central facility. Both of these problems must be solved in ways that are economical and allow battery operation.

Proposed system: To provide low-cost ways to communicate we proposed to explore the use of a tiered network, we connect clusters of sensors together with standards-compliant radios operating at short-ranges in the unlicensed band (such as the recent IEEE 802.15.4 radios that are the basis of the Zigbee stack), and each cluster connects to the Internet via a wide-area wireless data service (such as GPRS, using the cellular telephone network). Sharing the cost of wide-area connectivity is essential to make connectivity economical. Our research effort will be to explore how to implement our traffic classification algorithms on top of current standard protocols for wireless networking such as routing and resource discovery.

Our overriding goal here is to understand the needs that a vehicle traffic classification system places on the wireless network. We have shown the importance of sharing information between multiple sensors, but exactly *what* information is exchanged and *how* it is sent can make a major difference in network requirements.

As an example, Figure 2 shows a potential deployment of three pairs of blade inductive-loop sensors along a roadway. Each sensor will detect and classify vehicles as they pass; the sensors will exchange information about each vehicle to allow subsequent sensors to correct errors seen at earlier sensors [34]. For example, a car moving between two lanes might only partially cross the first sensor—a channelization error that may preclude correct classification at that sensor. If this event is correlated with the same vehicle crossing a later sensor properly (without channelization error), we can prefer the classification from this later sensor and correct for the error of the first. This process requires that the sensors exchange information to allow correlation of readings and selection of the best one.

A correct but naïve way to select from multiple sensor readings is to send raw signatures to a central computer. Given all sensor data, that central site can then make the best possible decision. This approach is naïve, however, because it is very wasteful—the raw sensor data contains a great deal of duplicate information that will be thrown away, and there is no need for a single central site to store *all* the data when it will only be comparing readings at adjacent sensors. We have a complete list of its problems in Section 6.3.

We propose to develop applications that allow sensors to directly share information with each other, and to abstract away raw sensor readings into the minimum information needed to make decisions. We expect to draw on prior work in standard radio hardware (such IEEE 802.15.4, the new, low-power protocol for short-range wireless communication in sensor networks), media access protocols that conserve energy (such S-MAC [51] and B-MAC [38], as well as our recent work on SCP-MAC [50]) protocols for routing information in multiple-hop wireless mesh networks (such as directed diffusion [22, 18], DSDV [36], or DSR [24]), and protocols for resource discovery and coordination (such as directed diffusion [22, 18]). Such work is important because it provides in expensive hardware and well understood protocols that minimize energy consumption; our main task is to understand how they interact with applications for traffic classification. This short-range communication is shown in Figure 2 as the thin dashed lines between sensors themselves.

Communication with TMS: The above approaches suggest how communication *inside* the network of sensors can be accomplished, but not how those sensors should communicate to the central Traffic Management Systems or researcher database on the Internet. At a high level, the simple approach we plan to take is to send the data through the Internet. This approach leverages commercial wide-area wireless services such as GPRS or metropolitan-area 802.11 networks that have been proposed in San Francisco, Philadelphia, and Hermosa Beach. However, again, care must be taken to minimize the monetary and energy costs of Internet connectivity, as well as to maximize reliability. We plan to equip two computers in the traffic monitoring sensor network connections to the wide-area wireless network and the Internet, allowing those nodes to forward results out of the traffic sensornet. This approach reduces both monetary and energy cost, subscriptions to WAN radio service are required for only two computers, and only those computers will consume additional energy. Internet-connectivity also ties in with our work in easy deployment, since the system must automatically identify and use the computers with wide-area connectivity, and it should fail-over from one to the other if one of the computers fails. This long-range communication is shown in Figure 2 as the thick dashed lines between one sensor and the TMS. Finally, we can deploy virtual-private network protocols like IPsec [25] at the gateway to guarantee the integrity of the data as it passes over the commercial Internet.

Connectivity: Year 1 Status Our Year 1 research in connectivity have prototyped autoconfiguration and communication over an 802.11 ("wifi") network, and coordination of multiple nodes with a simple leader selection algorithm. We discuss the auto-configuration aspects of connectivity below in Section 5.4. This work represents a significant step forward from our prior work in SURE-FT where data was analyzed post-facto with no in-field connectivity.

Connectivity: Year 2 Status In Year 2 we carried out data collection using the Year 1 system. We report on our experiences in the field in Section 7.3, including use of wifi for communication between sensors and with the campus network. We explicitly compare the costs and accuracies of moving full raw data and processed data (Section 6.3).

The falling cost of cellular data communications suggest that cellular data can provide wide-area communication provided its cost can be amortized over several sensors. Our experiments show such sharing is feasible.

3.3 Auto-Configuration

Full auto-configuration is a new focus of SRVC. To this end we introduce the concept of a *probe vehicle*—a vehicle of known size (wheelbase) that we can use to calibrate our system when it is first deployed. The probe vehicle is equipped with GPS so that it has an accurate notion of time and its location. Typically, we expect to tape down the inductive loops and then drive the probe vehicle over the sensors. By synchronizing sensor trigger times with the vehicle movement we will know when the sensor is triggered by the probe; we can then calibrate the sensor from what it observes.

Besides sensor calibration, some other internal system parameters (e.g. loop separation distance) and performance-related issue (e.g. time synchronization) also need auto-configuration, in order to build a fully automatic vehicle tracking system.

Auto-configuration: Year 1 Status We have a prototype of probe-vehicle-based autoconfiguration, as described in Section 5.1. In Year 2 we expect to add GPS-based configuration of sensor layout. In Year 1 we have studied inter-sensor separation distance (Section 5.3), time synchronization (Section 5.5).

Auto-configuration: Year 2 Status In Year 2 we further experimented with issues in Section 5.6 and centralized/distributed calibration issue in Section 5.1.

3.4 System Integration

Much of our classification algorithm analysis was carried out with off-line, post-facto analysis of stored traces. This approach has been essential to refine and improve our algorithms with a consistent dataset. However, we are working towards developing an on-line system capable of real-time vehicle classification.

To this end we began integration of our software into a package suitable for on-line operation. In SURE-FT [17], we have completed integration of all single-node processing, including data collection, segmentation,

action	**time required**
physical deployment	30 minutes
probe vehicle collection	10 minutes
auto-configuration execution	< 1 minute

Table 1: Approximate deployment times of integrated system.

and single-node classification. It also includes single-sensor auto-configuration algorithms described in Section 5.1.

Although we planned to use embedded computers running Linux as our sensor nodes, device drivers for the IST blade sensors are currently only available for Windows. Our current system therefore runs on Windows laptops. IST informs us that they are working on Linux drivers, and our code should easily port to an embedded platform when drivers are available.

We tested both classification and sensor auto-configuration algorithm with a short experiment conducted on July 28, 2006 at the roof of ISI parking structure as described in Section 7.1.2 We collected 36 vehicle signatures of three different vehicles. This experiment confirmed that automatic deployment was effective and that our integrated algorithms work well.

Deployment required several steps: first we physically deployed the sensors. We then ran our probe vehicle across the sensors. We run the probe vehicle over the sensor one or two times to run our auto-configuration algorithms. Approximate deployment times of each phase are shown in Table 1. Physical deployment of the a pair of sensors required 30 minutes, during which time the roadway must be closed or partially closed. We expect that could be reduced with more deployment experience. Additional configuration (primarily probe vehicle data collection) can be done after the roadway is opened (although with a mix of vehicles, the operator must currently manually select which signatures are of the probe vehicle).

In addition, we confirmed that our integrated system worked well. All 36 vehicle crossings resulted in positive detections and accurate classification in real time.

Although promising, this experiment is quite preliminary. We must complete integration of communication *between* nodes and support for multi-node collaboration. In addition, this experiment with the integrated software involved only three vehicles and was done in a parking space instead of a real road. Additional experience, including deployment on real-roadways and with a wider selection of vehicles is needed to more fully evaluate the system.

Integration: Year 1 Status In Year 1, a "Master" server, whose function is defined in Section 5.4, has been developed and our test shows that the whole system works as expected.

Integration: Year 2 Status In Year 2, we further investigated the integration of the system and shorten the road deployment time. We developed new multi-sensor software and hardware integration and conducted a field experiment to test deployment times, as reported in Section 7.3.

3.5 Software and Hardware Availability

The software we developed as part of our system and to analyze our results is available on request from the authors. We have provided data from our field experiments to several external researchers over the course of this work.

4 Algorithms for Vehicle Classification

This section summarizes the algorithms developed as part of our prior work in the SURE-FT project (described in its final report [17]). We summarize these here since they serve as the base of our system.

4.1 Algorithm Description

We first review our basic classification algorithms, with single sensors and wheelbase as the classification feature. We then consider the benefits of multiple sensors.

4.1.1 Single-Sensor Classification

Signal processing at an individual sensor is fairly traditional: we begin with noise removal, segment the data into individual vehicles, extract features pertinent to our analysis, then classify vehicles; we examine each of these steps next.

Noise elimination: In some cases we have observed significant crosstalk, environmental noise, and signal drift in our measurements. In addition, communications from the IST card to the host computer is not perfect. Crosstalk arises when inductive loops driven by different sensor cards are close to each other, and also due to slightly different sensor clock rates in the case that multiple sensor cards are attached to the same sensor node.

Drift and noise occur due to temperature change and other environmental effects. We filter each of these out using standard techniques. We observe around a 14% data loss on the USB bus between the sensor and the host (this behavior is a known problem of the specific model of detector cards we were using); we correct for this by interpolating the missing data.

Segmentation: When noise is eliminated, we are left with a continuous signal. We next isolate individual vehicles with a three-step process. We first detect active segments by observing large signal deviations from a running mean. We then merge temporally close active segments to allow for vehicles that have "flat" areas between wheel wells. Finally, we grow segments by half their length at front and back to ensure we capture a complete segment, including leading and trailing features. As a special case, when growing a segment would cause overlap with a neighboring segment, we grow to the midpoint between segments. Ideally, after segmentation, each segment corresponds to exactly one vehicle. In practice we find that occasionally vehicles that are very near to each other appear in a single segment.

Feature extraction: We experimented with several possible features for vehicle classification, including axle count, body width, and wheelbase (axle-to-axle distance). We converged on a two-level set of features. We directly extract the edges of wheels (70% wheels points, described below), then figure these and estimate speed and wheelbase. Our first goal is to determine wheel edges. Figure 3 shows a sample signature of a two-axle car crossing a sensor. Wheels show up as large dips in the signature, the underbody as bumps between the wheels. Because the vehicle crosses the inductive loop at an angle, each wheel of the same axle produces a distinct dip located near the other. We experimented with different algorithms to reliably extract each wheel. Our two main approaches were to identify peaks and to identify large changes in direction. Although peak identification is attractive, consistent results are difficult because peaks tend to be rounded, particularly at higher sampling frequencies (because wheels are round). In addition, depending on the angle the car crosses the sensor, we may get two clearly distinct peaks or a single merged peak. Because of these difficulties we adopted the "steep slope" algorithm shown in Algorithm 1.

It is important that our algorithm adapt to a wider range of vehicle speeds. To do so, we adjust the parameter v based on vehicle speeds. First, we start with a default v value 240 ($0.2R$, where R is the sampling rate) to estimate the vehicle speed, since it is not necessary to separate each wheel well to get speed estimates. Then, we adjust v according to the estimated speed as shown Algorithm 1. If v is too large, we cannot separate the wheel wells; on the other hand, if it is too small, we cannot capture the all wheels in the wheel well. In case of extremely low speed (less than 5mph), we use 680 ($0.4R$) for v to prevent from collapsing every wheel well into one. The other parameters are somewhat arbitrary. We initially chose the S value as 70%, with the goal of identifying a point on the steep part of the signature. We studied values from 30–80% and found they did not affect classification accuracy, as shown in Figure 4.

Speed Estimation: To estimate vehicle speed we compare the time difference when a wheel-point crosses each of the two closely placed loops. Figure 5 shows a time-correlated sample from both of the paired loops for one signature. We compare pairs of shapes (squares or circles), and similar observation at the front and back of the first and last wheel wells. Since the loops are a known distance apart, a speed estimate is simply the distances divided by the time between the same feature at each adjacent loop. To reduce error, we match the wheel-points for the start and end of the first and last wheel wells, giving us four estimates of speed. We then average these values. We discuss how this approach addresses different errors in Section 4.2.2.

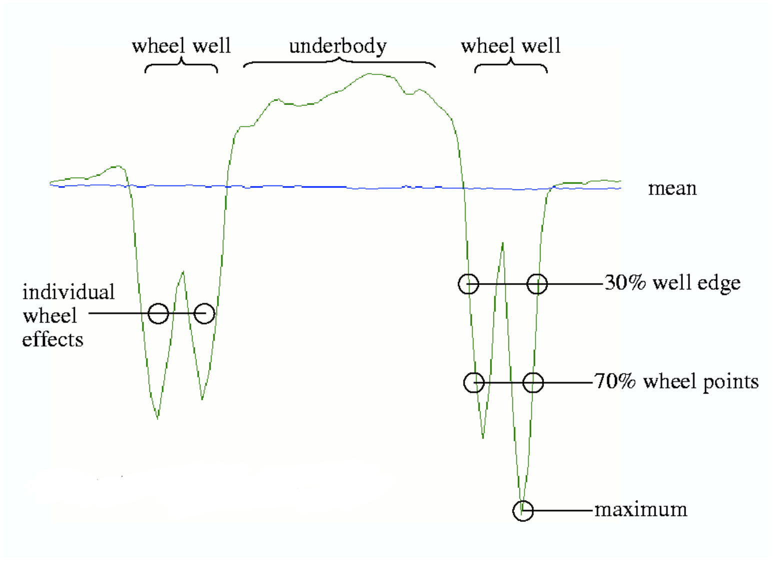


Figure 3: Sample signature indicating vehicle parts (signature #280, site BN)

Algorithm 1 The steep-slope algorithm for wheel edge extraction.

```
Normalize the signature values from 0.0 (lowest underbody) to 1.0 (highest wheel peak)
Compute $m$, the mean of all sensor readings over entire signature
Identify wheel wells by finding the first value greater than $m + (L * m)$ through the last value greater than $m + (L * m)$,
    allowing up to $v$ consecutive values below $m + (L * m)$
For the first and last wheel wells:
    Find the maximum value M in the well
    Define the start-wheel-point as the first value in the well greater than $S * (M - m)$
    Define the end-wheel-point as the last value in the well greater than $S * (M - m)$
Parameters:
$v$ : 2/($VehicleSpeed$) * $SamplingRate$, number of samples allowed below $m + (L * m)$ in a wheel well
(between wheel peaks)
$L$ : 0.3, wheel well start threshold (fraction of mean)
$S$ : 0.7, target for steep slope (fraction between mean and peak)
```

Wheelbase: To estimate wheelbase, we observe the front of the first and last wheel wells: solid shapes (squares or circles) in Figure 5. With two paired sensors, we can get four estimates front and back of sensors 1 and 2.

Classification: Given a speed estimate, we classify vehicles by wheelbase length, the distance from the first to the last wheel axle. Given our wheel-points on the first and last wheel wells, we have four wheelbase estimates on each of our two paired sensors. As with speed estimates, we average these four readings.

Finally, we map length to vehicle classification as described in the next section.

Clearly our classification algorithm is quite simple. A much more sophisticated system would, for example, match the entire signature against a database of known vehicle types. However, even this simple classification system is sufficient to explore the use of multiple sensors to reduce error.

4.1.2 Classification by Wheelbase

Table 2 shows our mapping from length to our two- and three-category classification systems and the relationship to the FHWA classes. We selected the two-category system because automatic classification of truck vs. non-trucks is relatively easy, while we will show that distinguishing cars from SUVs (P from S*) is much more difficult and thus represents a “worst case” for automated classification systems.

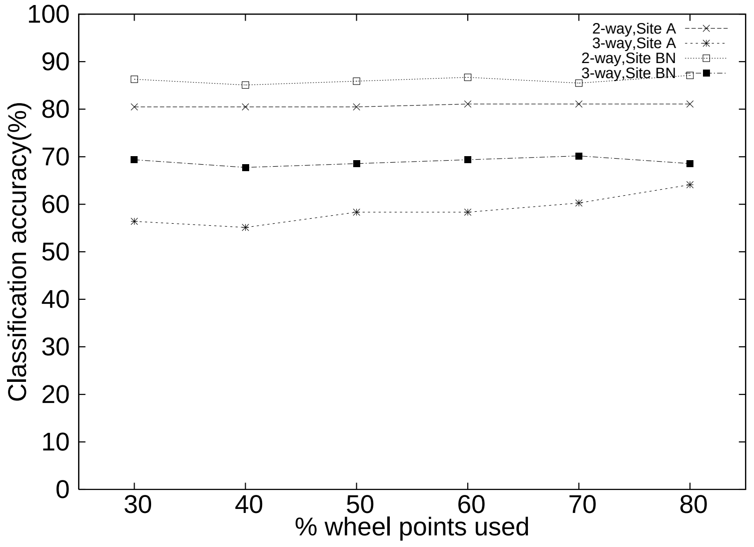


Figure 4: Classification accuracy as a function of wheel point percentage.

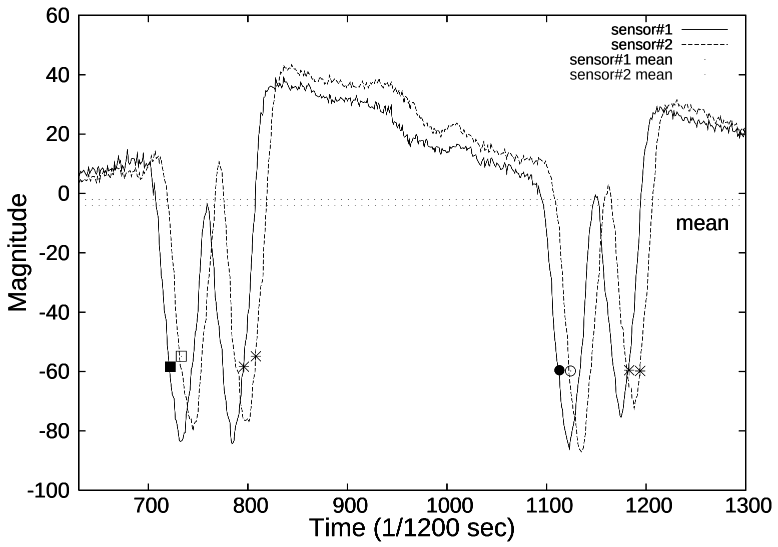


Figure 5: A signature with paired sensors (signature #130, site BN)

Table 2: Length-based vehicle classification

FHWA classes	meaning	symbol	wheelbase length
Two-category system			
2 and 3	non-trucks	non-T	< 170”
4 to 13	commercial trucks	T	≥ 170”
Three-category system			
2	passenger cars	P	< 118”
3	small trucks/SUVs	S*	118 to 170”
4 to 13	commercial trucks	T	≥ 170”

Table 3: Two-category classification with perfect sensors

Class	Total	Correctly classified	Incorrectly classified
non-T	42	41	1
T	5	4	1
Total	47 (100%)	45 (96%)	2 (4%)

Table 4: Three-category classification with perfect sensors

Class	Total	Correctly classified	Incorrectly classified
P	24	24	0
S*	18	12	6
T	5	4	1
Total	47 (100%)	40 (85%)	7 (15%)

A critical problem with the FHWA system is that the boundaries between classes 2 and 3 are indistinct. In fact, the FHWA website says “because automatic vehicle classifiers have difficulty distinguishing class 3 from class 2, these two classes may be combined into class 2” [12]. This problem applies also to human observations (we quantify human accuracy in the SURE-SE); we next consider how length relates to classes.

To evaluate the wheelbase-based classification with perfect sensors we surveyed wheelbase lengths of 47 vehicles from Ford [13] and government sources [2]. Table 3 and 4 show classification accuracies based on wheelbase assuming perfect length determination. Even though we classify vehicles with their exact lengths, not all surveyed vehicles are correctly classified: 96% of vehicle models are correctly classified in two-category classification. In three-category classification only 85% of models are correctly classified, since there are many SUVs on either side of our threshold, regardless of where it is placed.

These results assume static analysis, in that the percentages are based on numbers of vehicle types. In practice, the population of vehicles of each type varies, as does the particular set of vehicles observed at any site. Therefore, dynamic measurements may differ depending on the mix of observations.

4.1.3 Using Multiple Sensors

A defining characteristic of sensor networks is the use of many relatively inexpensive sensors. We therefore wish to explore if multiple sensors can improve the best-possible classification results of a single sensor. Our hypothesis is that classes of errors are independent, so combining values from moderately separated sensors can eliminate these errors.

As shown in Figure 2, we place several pairs of sensors at several places on a roadway. We expect sensors to communicate through a local, low-power, wireless network such as provided by 802.15.4 or similar networks. Sensor nodes will not share raw sensor readings, but instead individual evaluations of vehicle type, coupled with data about their confidence in the classification. Such a system must have several components: a configuration system to automate initial deployment, communications protocols to share information between sensors, a signature matching algorithm to identify which signatures at one sensor correspond to signatures at another sensor, and a classification preference algorithm to select which classification is best. We plan to exploit the simple, constrained topology of the road, so configuration and communications are straightforward as each sensor interacts with its immediate neighbors. Signature matching and classification preference are the keys to improving accuracy with multiple sensors. We discuss these algorithms in Section 4.2.3 after reviewing potential types of error.

4.2 Types of Errors and Error Recovery

To consider how multiple sensors might improve accuracy, we first evaluate the types of error that arise in this application, then consider how to make a single sensor as effective as possible, and finally how multiple sensors can further improve accuracy.

Table 5: Types of errors in vehicle classification

Type of error	**Generality**	**Dependence**	**Addressed**	**Observed**
Environmental noise	General	Either	(In-sensor) or post-facto	Yes
Sensor failure	General	Independent	(Multi-sensor)	No
Insufficient sampling	General	Independent	(Design)	Yes
Vehicle type	Application	Dependent	(Multi-sensor)	No
Mis-channelization	Application	Independent	Multi-sensor	Yes
Imprecise speed	Application	Dependent	Single sensor	Yes
Changing speeds	Application	Independent	Single sensor	Yes
Mis-segmentation	Application	Independent	Multi-sensor	Yes

4.2.1 Types of Error

We review the types of errors we expect in Table 5. There we evaluate each error for its generality, if it is specific to this application or applies to all sensors; dependence, if we expect multiple sensors to exhibit this error consistently or independently; how we address it, in-sensor with multiple estimates or multiple sensors; and if we observed it in our examples.

We observed significant amounts of *environmental noise* in our data, both due to temperature drift and sensor cross talk. A later revision of the IST Blade sensor handles noise elimination in the sensor itself, but for our data collection experiment we filtered noise manually post-facto. Inductive loops respond to vehicle mass relative to its distance from the sensor, thus they are less sensitive to vehicles that are higher off the ground. Loop sensitivity can be controlled by adjusting width, so potentially multiple loops of different widths could detect a wide range of vehicles. For our main experiments we used a loop width of about 4 inches.

We did not observe any *sensor failure* in our system, but it would be an issue for larger deployments.

An *insufficient sampling rate* or too close placement of sensors can result in imprecise speed and length estimates, since a change of a single sample interval corresponds to a noticeable change in estimate. Sampling rate or sensor distance must be adjusted to expected speeds.

Vehicle type errors, refer to different distances of vehicles from the ground. We did observe distance affecting the quality of sensor signatures, however it was not a major cause of mis-classification.

Mis-channelization is when only part of the vehicle crosses the loop because it is changing lanes. *Changing speeds* occur when a vehicle alters its speed over the sensor, making estimation difficult. *Mis-segmentation* occurs when two cars travel so close that they appear to the sensor to be a single vehicle. All of these errors are specific to vehicle classification, but each occurs independently at different sensors and so should be correctable in the sensor network.

4.2.2 Single-Sensor Error Recovery

We used three general techniques to improve individual sensor readings: sharp feature detection, internal consistency checking, and cross-checking with multiple features .

Our early approach identified speed and wheelbase by using peaks of the signature to estimate the exact wheel locations. This approach proved inaccurate at high sampling rates because wheels provide rounded features to the inductive loop, since the loop has an effective low-pass filter, and wheels are round. To address these problems, we shifted to identifying the sharp slopes that correspond to the edges of the wheel peaks, as determined by the S threshold, a fraction of the distance from mean to the peak value (currently 70%). Since the slopes on the edges change rapidly, slopes tolerate much more error than peaks. In fact, we experimented with thresholds corresponding to 30–80% of the distance between the mean and maximum sensor values and found classification accuracy unchanged.

Second, we check the features for internal consistency. Our primary approach is to evaluate where detected wheels are placed inside the wheel well. We expect to get two peaks in each wheel well, corresponding to the left and right wheels. However, if one of the peaks is much smaller than the other, sometimes it is missed by the steep-slope algorithm. For example, in Figure 6, if we miss a right wheel in the first and the left wheel in the last, the estimate will be longer than the proper estimate. One wheel may be omitted from a wheel well

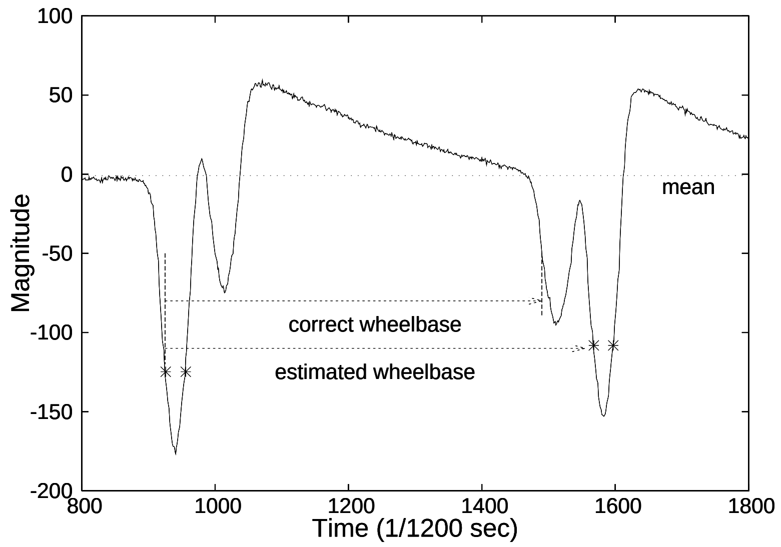


Figure 6: A signature with missing wheels (signature #96, site BN)

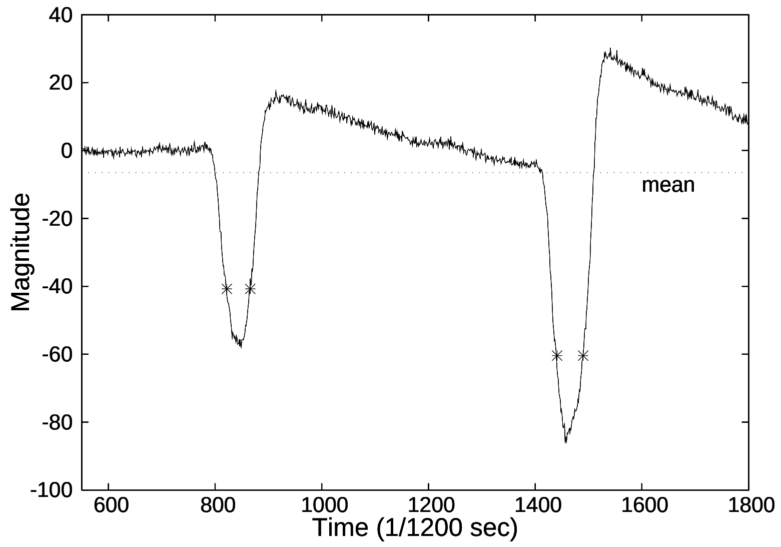


Figure 7: A signature with channelization error (signature #251, site BN)

when channelization errors occur (Section 4.2.1). Ignoring these errors can result in significant inaccuracy in wheelbase estimation. Figure 7 shows a signature in channelization error.

To solve this problem, we take several steps to examine wheel placement for internal consistency. (We explored adjusting the steep slope target S to catch low peaks, but in general one cannot catch all low peaks and be robust to noise.) We count the number of wheels in each wheel well. If we believe a wheel is missing, we identify which wheel is present, or perhaps determine that we cannot tell which wheel we observe. There are four possible wheel placement states: both wheels present, left wheel or right wheel missing, and unable to determine which wheel is missing. There are 16 possible combinations of these cases when we consider detection at the first and last wheel wells.

When both wheels are present we get two wheelbase estimates per loop. If a wheel is missing but we can identify which wheel is present we calculate a single wheelbase estimate with the present wheel. However, when we cannot determine which wheel is present, or if different wheels are present in the front and back wells, we know that our wheelbase estimate will be incorrect. In these cases we report our best estimate along with a lower confidence value in this estimate, potentially allowing multi-sensor error recovery to select a better estimate at another sensor as explained in the next Section 4.2.3.

Finally, we obtain multiple estimates of each feature in a single signature. We can find four wheel-points for each signature in the beginning and ending of the front and back wheel wells. (These are indicated by stars in Figure 5, in addition to the squares and circles.) This gives us four estimates of speed and length. Although these estimates are not completely independent, averaging them provides much less variance than any individual reading. This approach also partially corrects for vehicles that change speed over the sensor. Although reduction in variance does not necessarily translate into improved accuracy, it does imply less susceptibility to noise.

4.2.3 Multi-Sensor Error Recovery

To use multiple sensors to reduce errors we must identify when multiple signatures correspond to the same vehicle and then which classification is most accurate: signature matching and classification preference.

Signature matching: We plan to exploit the constrained topology of a roadway to simplify signature matching. If sensors are placed on a roadway without intersections or exits, then the order of vehicles and signatures is fixed and so signatures can be matched based on timing and ordering. Missing signatures can be inferred by gaps, and mis-segmentations by a very long signature at one sensor followed by a short signature at the next.

Classification preference: Given two matched signatures with different classifications one must choose which classification is more likely to be correct. We are experimenting with two algorithms: quality-best and shortest-best.

The *quality-best* algorithm favors sensors that are able to consider multiple estimates that report consistent values. This approach addresses speed variability, since a larger number of speed and length estimates reduce the impact of variability and allow the sensor to estimate its confidence in the estimated speed and length (more variance indicates less confidence in the estimate). We also adjust the confidence according to the internal consistency of the signature by considering where we believe wheels are placed in a signature. If there are missing wheels and their placements are inconsistent, we reduce the confidence value to half of what we get from the variance.

The *shortest-best* algorithm is much simpler. In our experiment we found some errors were due to mis-segmentation. We can detect mis-segmentation as a long signature at one sensor with two short signatures at the other. Our system is more likely to merge two adjacent signatures than split a long signature, therefore in these cases the shortest-best algorithm selects the two-signature interpretation over the single long signature.

Finally, for analytic purposes, we consider an *oracle* algorithm. The oracle algorithm assumes a perfect classification preference algorithm that always chooses the correct classification if it is present at either sensor. Such an algorithm is impossible to realize in a real system—we can implement it only because we already have ground truth. We present it to provide an upper bound on how well sensor fusion can do.

Current Status: We have implemented classification preference based on sensor quality, and multi-sensor comparison. We have not implemented a complete matching algorithm, since in our data collection experiment sensors were separated by a large distance and an intersection, so for our preliminary analysis we manually associated signatures at two sensors.

5 Multi-Sensor Automatic Configuration

A goal of our system is that it be easily and rapidly deployable. This goal motivated our choice of hardware platforms (small, wireless devices) and sensors (inductive loops that can be simply taped down on the roadway). However, in addition to physically deploying the system, the software requires configuration as well.

Developing and prototyping a self-configuring system has been a significant focus of the Year 1 SRVC effort. This section reviews what aspects of our system must be configured; later in Section 5 we describe the new algorithms we have developed to support configuration of many parameters.

We evaluated the automatic configuration for segmentation thresholds, intra/inter-sensor loop separation distance, network topology and time synchronization. Some others are left as open issues.

Algorithm 2 Automatic segmentation threshold

```
Observe signal of the probe vehicle.
  Record the stream of signal during this period without any segmentation.
  While observing the stream, calculate windowed standard deviation, $E$ (signal energy).
  After the probe vehicle runs over the sensor $N_{actual}$ times, find $Max(E)$.
  Set the temporary thresholds
    $Th_{Hi} = T * Max(E)$
    $Th_{Lo} = 0.5 * Th_{Hi}$
Run the segmentation algorithm with above thresholds on the recorded signal.
  Initialize the threshold adjustment size
    IF $N_{signature} < N_{actual}$ THEN $\Delta = \Delta_{inc}$
    IF $N_{signature} > N_{actual}$ THEN $\Delta = \Delta_{dec}$
  IF $N_{signature} == N_{actual}$ THEN set the thresholds for the classification
    ELSE adjust the thresholds:
      IF $N_{signature} < N_{actual}$ THEN increases thresholds by $\Delta$
      IF $N_{signature} > N_{actual}$ THEN decrease thresholds by $\Delta$
      Update $\Delta = \Delta/2$
  Iterate above steps until matches to actual count.
  If it doesn't match within two iterations, then declare "Failed to find thresholds"
Parameters:
$T$ : Initial threshold factor, 0.25
$N_{signature}$ : Number of signatures
$N_{actual}$ : Actual vehicle count
$\Delta_{inc}$ : initial increment size of threshold $((1 - T) * Max(E))/2$
$\Delta_{dec}$ : initial decrement size of threshold $(T * Max(E))/2$
```

5.1 Segmentation Thresholds

Segmentation is an early step in classification where raw sensor data is split into per-vehicle segments (see Section 4.1.1 for a complete description of the algorithm). Segmentation requires a threshold to separate the active vehicle signal from background noise.

Ideally we could assign a fixed segmentation threshold based on laboratory data. Unfortunately, we found that the threshold used varied based on the version of sensor detect card, and the loop construction (type of wire, number of turns, etc.). While we manually configured it for our preliminary experiments, we do not expect end-users to hand configure it and therefore designed an automatic procedure.

Algorithm 2 shows the algorithm we developed. We use a known number of probe vehicle detections as "ground truth", then iteratively adjust an initial value until we get a matching number of detections from the sensor. The initial threshold factor is decided by the heuristics from our July 2006 rooftop experiment (Section 7.1.2).

For a multi-sensor system, we are still investigating the difference between a Centralized calibration and a Distributed one.

Centralized calibration: This means that only one of the sensor should be calibrated and it will then broadcast the result to its peers, who are obliged to use that very calibration result instead of carrying on their own.

Distributed calibration: Each sensor is calibrated individually.

If different detector cards can be proved as homogeneous enough, then a centralized calibration is a better choice because it will cost less and make the system more auto-configured.

5.2 Intra-Sensor Pair Separation Distance

New to SRVC is an automated approach to determine sensor pair separation distance.

Knowing the exact distance between loops is crucial to get the proper speed and length estimation, since we use a pair of loops to calculate the speed of a vehicle as described in Section 4.1.1. Therefore every

Table 6: Configuration Approaches for Sensor Separation Distance Measurement

approach	Human Intervention	Monetary Cost	Accuracy	Short-term Experiment Suitability	Long-term Suitability
Measuring Tape	High	Low	High	N	N
Map Estimation	low	Low	Medium	Y	Y
GPS Sensors	Medium-	High	High-	N	Y
Vehicle GPS	Medium	Medium	Medium-	Y	Y
Bicycle Wheel Round Count	Medium-	Medium	Medium	Y	N
Bicycle Velocimeter	Medium	Medium	Medium	Maybe	N

pair of loops should be measured exactly or they should be separated by precise distance when deploying the loops on the road. Either way could require a long interruption of traffic. The experiment described in Section 7.1.2 shows that the distance between two loops affects the accuracies of speed estimation and length estimation. Hence, a small error in measuring sensor separation distance could result in consistent and significant error in vehicle classification.

To eliminate this measurement error we calibrate the sensor separation distance using the probe vehicle length. This procedure inverts our regular algorithm: given a known vehicle length and a measured time to cross two loops, we back-compute the physical distance between loops. This computation is possible because we know the exact length of the probe vehicle. In addition, we can use multiple readings of the probe vehicle to correct for measurement error.

Auto-configuration of sensor separation give two benefits: First, it simplifies deployment, since sensor distance is not critical. While loops must be parallel to each other, their separation can be arbitrary, since we measure it and correct for variation in a given deployment. Secondly, when there are multiple pairs of sensors, each may be deployed with different separation distance, yet this variation can be correct for by auto-configuration. Finally, if the system is to operate for long durations (more than a few days), potentially we can easily re-calibrate the sensor separation to account for sensor slippage as many vehicles drive over it. (We have not yet evaluated the rate of sensor movement.)

5.3 Inter-Sensor Separation Distance

SRVC Year 1 has also developed an approach to estimating the distances between pairs of sensors.

We must know the distances between nearby sensors (the three pairs in Figure 2) to estimate vehicle speed (Section 4.1.1, length and type (Section 4.1.2), and to match signatures taken at different sensor pairs (Section 6) for sensor fusion (Section 4.1.3).

Knowing the exact distance between sensor sites is crucial to make our any timestamp based algorithm (Algorithm 4,and 5) work. With the distance and a proper traffic flow speed estimation, we can compute the average travel time for a vehicle going between sensors sites. We discuss in Section 9.2 how time window value selection affects final matching results.

We considered many alternatives to determine inter-pair distances. Table 6 shows the criteria for these alternatives. First of all, configuration should involve minimal human intervention, both in terms of absolute time, and in terms of expertise of the operator—an ideal approach requires nearly no time and can be done by anyone with no training. Secondly, the approach should be inexpensive, not requiring special-purpose, per-sensor hardware. Per-sensor cost is important in situations where many sensors are to be deployed. Thirdly, accuracy is our major concern, discussed by the forgoing paragraph. Some approaches are acceptable for a short term experiment, since either they are relatively accurate or easy to implement.

We put down a brief explanation of the methods we studied in Table 6.

Measuring Tape: Measuring the distance between sensor nodes via tapeline ruler. Although the method should yield the most accurate result, it is impractical in out scenario for the following reason. Common tapeline rulers have a measuring limit around 7m, which means that for a 50m∼100m node separation, operator will have to split road and relay the ruler, causing high human intervention.

Map Estimation: Get a rough location distance measurement from a map. The method has lowest human intervention.

GPS Sensors: We use the absolute GPS position of nodes to compute their distance. The method has the highest hardware cost, since a reasonable GPS accuracy is required. This method could help build a fully automatic system, with GPS sensors interoperable with traffic sensors.

Vehicle GPS: Unlike the last approach, we use a relatively inexpensive Vehicle GPS instead. The method could be integrated with our probe vehicle process as described in Section 5.1.

Bicycle Wheel Round Count: Walk a bicycle between two sensor nodes. By counting (by human or sensor) how many wheel rotations, the distance could be easily computed after multiplying the wheel diameter. However, the method has an intrinsic error for one wheel diameter, since it is hard to measure the rotation angle for the final (incomplete) rotation round.

Bicycle Velocimeter: Riding a bicycle between two sensor nodes. The method takes advantage of a simple physics equation: $distance = travel\_time \times speed$. However, an accurate time estimation and constant speed are both hard to achieve.

After examine the forgoing approached, we believe, for medium-term or even semi-permanent deployment, GPS Sensors and Vehicle GPS might be good choices. If the distance is long enough, then error in merely a map investigation will also be tolerable.

5.4 Network Topology

In a multi-sensor system, we must consider how the sensors are organized. We considered two alternatives: peer-to-peer and master/slave. While the organization does not affect accuracy, it does affect how the software is integrated and it can affect the level of network traffic. We describe and compare these two basic topologies below.

Peer-to-Peer: Each node is equal. They all report back to TMS individually. Since this is a distributed scenario, it should be more fault tolerant, since other nodes can return data even if a single sensor fails. There are two major disadvantages to this approach. First, if each node is equivalent, all nodes must have wide-area communication, a potentially more costly option. In addition, the management of the system can be more complex.

Master/Slave: A node is elected as the central, "master" node, in charge of controlling the other "slave" sensor nodes. After doing their own local detection and computation, all nodes report the data collected to the central node, who then combining the all information and carrying further signature matching and sensor fusion procedure, before publish the final result to TMS. The main advantage of a master/slave configuration is that it is simpler: slave nodes simply relay data to the master, and all sensor-network-wide decisions can be made at the master node. The disadvantage is that the master node can be a single point of failure, and relaying the data to a master can consume more network bandwidth than processing it locally.

Master/Slave scenario raises another challenge–how to elect the central node? We propose a simple algorithm to elect a master when the network starts. On system start, all nodes broadcast some credentials to run for central node simultaneously. Credentials can be the timestamp of its wake-up time or credential transmission time (This needs time synchronization discussed in Section 5.5); it can also be its IP address, mac address or other unique ID. If node credentials are stable (as the above are) and all nodes receive all credentials quickly, then they will all select the same master.

In short term experiment for now, we decide to choose a *mix* of both topology. Either node collects signature of vehicles and computes the final feature vectors independently. A server (could be separated or combined with either one of the node) does sensor fusion upon receiving those feature vectors and decides whether report the result to TMS or just record on local hard disk. This topology can 1) take advantage of processing resource on both nodes; 2) lower network traffic; 3) simplify structure and 4) have a reasonable robustness, since the server now is imposed of less burden.

5.5 Sensor Time Synchronization

Time synchronization across sensor nodes is essential, since we compare the time difference of two signatures from two sites to the estimated travel time and we use the travel time to match detections at different sensors (Algorithm 4).

Table 7: Configuration Approaches for Time Synchronization

approach	Complexity	Fault Tolerance	Network Traffic	Good for Short-term Experiment?	Good for Long-term Deployment?
Central node timestamp incoming data arbitrarily	Medium	Medium	Low	Y	Maybe
Network Time Protocol(NTP)	Low	Medium	High	Y	Maybe

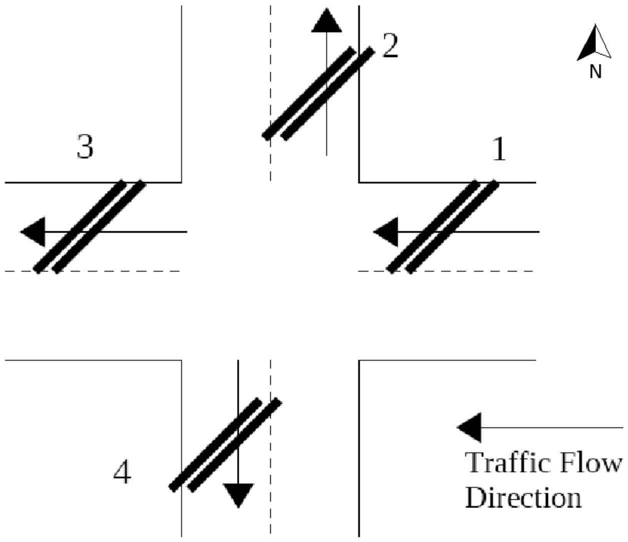


Figure 8: Auto-Position Inference Diagram

Table 7 briefly compares two approaches for time synchronization. If energy consumption is not a major concern, then mere implementing Network Time Protocol (NTP) [28] is enough for the purpose. Otherwise, a more arbitrary approach could be considered.

We choose to let central node timestamp incoming data in our current experiment implementation, since the transmission delay in our network could be ignored and we want to keep network traffic as low as possible.

5.6 Other Kinds of Automatic Configuration

The approaches described above show the principles of automatic configuration needed for the algorithms developed thus far. We are considering exploring how sensor location and relative positions could also be automatically configured.

For example, we could automatically infer which sensors are deployed in which traffic lanes, the direction of traffic flow, and which sensors are the downstream. Take Figure 8 as an example. After initialization and monitoring a certain amount of traffic, the system should be able to realize that (with the help of timestamp) Sensor 1 is the upstream site for 2, 3 and 4. It should further infer, with absolute position information from some GPS sensors, that the traffic across Sensor 1 and 3 is westbound, Sensor 2 northbound and 4 southbound. Finally, it might realize there's a right turn between 1 and 2 but a left turn between 1 and 4.

6 Multi-sensor Signature Matching Algorithm

While SURE-FT tested the premise that multiple sensor values can be integrated to improve accuracy, it did not address the problem of how to identify when readings from different sensors correspond to the same

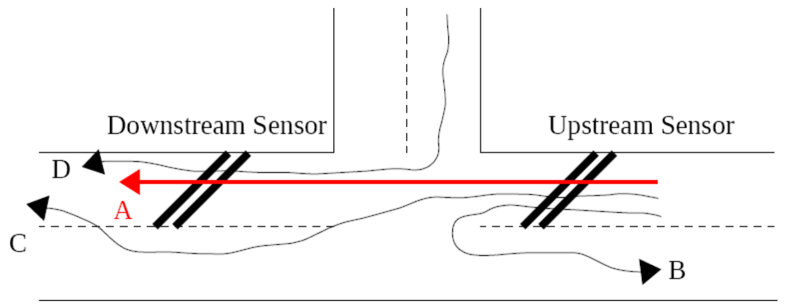


Figure 9: Example of challenges in signature matching. Both sensors should correctly detect vehicles along Route A (in red). If vehicles take route B and C, only the upstream node will detect them. If route D is taken, vice verse.

vehicle. We call this problem *signature matching*; solutions to this problem have been a major focus of SRVC Year 1. The goal of signature matching is to take two streams of vehicle signatures from two nearby sensors and determine which signatures correspond to the same vehicle. Signature matching can be challenging because vehicles move with varying and unpredictable speeds, vehicles may be detected by one sensor and not the other (because of sensor error or missing one sensor because of lane changes or vehicles taking alternate paths), as shown in Figure 9. We expect our signature matching to determine pairs of signatures corresponding to a same vehicle, or an indication that a signature is non-matched. Perfect accuracy is not mandatory, but sensor fusion requires matching with high probability.

We define three general approaches to signature matching: numbering-based, timestamp-based, and full-signature comparison. We describe these approaches and variants below, based on data from our technical report [52].

6.1 Numbering Based

We start with two variants of a simple, order-based algorithm: *Naïve Numbering* (NN), and *Numbering with Resynchronization* (NwR) handling missing signatures.

6.1.1 Naïve Numbering (NN)

With perfect sensors, i^{th} signature detected upstream should match the i^{th} downstream. Therefore in NN, each sensor numbers its signatures, and then we merge the detections sequentially.

NN suffers from the problem that any missing signatures throw off the stream alignment and result in mis-matches. We call this problem an *avalanche*, since one observation error causes many incorrect matches (the left case in Figure 10). In the real world, signatures often are missing, either because of sensor error (perhaps a transient fault), sensor-target interactions (for example, a vehicle partially missing a sensor because it changes lanes), or targets not passing both sensors (for example, vehicles turning or pulling over between two stations). Since such errors are common in the real world, we present NN as the base to build our next algorithm.

6.1.2 Numbering with Resynchronization (NwR)

NwR (pseudocode in Algorithm 3) adapts NN to real-world noise by periodically resynchronizing streams. Like NN, each station numbers each signature it detects, but with NwR, all stations reset their numbering at a coordinated regular interval. This reset solves the avalanche problem, as shown in Figure 10. Suppose vehicle 1, 2, 3 and 4 pass both stations and are detected as U_i (upstream) and D_i (downstream). The mis-detection of D_2 causes following signatures to match incorrectly. NwR can reset to a new numbering span after D_3 and before U_4; U_4 and D_4 could still be correctly matched.

NwR adds one parameter to NN, the duration between resets. We want to reset frequently enough to prevent avalanches, but resetting makes it difficult to match vehicles in transit between stations, so we do not want to reset too frequently. Reset frequency depends on travel time between stations and how many

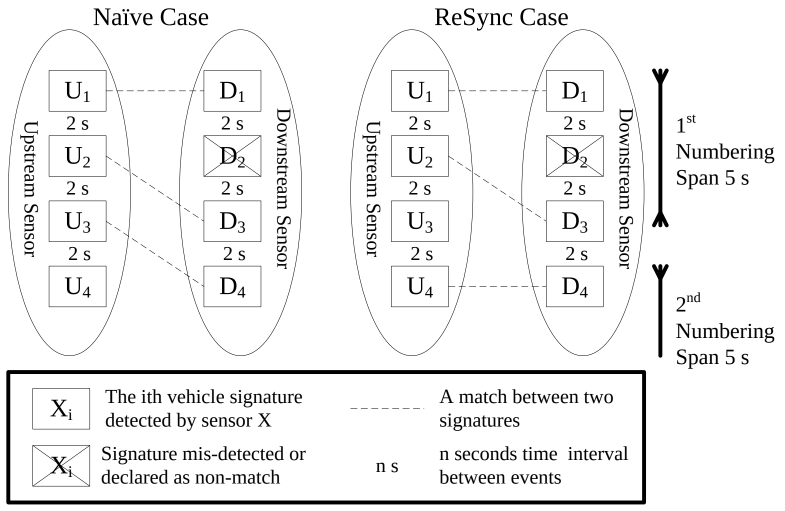

Naïve Case
ReSync Case
Upstream Sensor
Downstream Sensor
U1
U2
U3
U4
D1
D2
D3
D4
2 s
1st Numbering Span 5 s
2nd Numbering Span 5 s
Xi The ith vehicle signature detected by sensor X
A match between two signatures
Xi Signature mis-detected or declared as non-match
n s n seconds time interval between events

Figure 10: Avalanche problem in NN but solved by NwR.

Algorithm 3 *Numbering with Resynchronization* algorithm

```
Require: Dataset $D_{up}$ and $D_{down}$ (signatures are sorted in their time-stamp ascending order) and numbering span length $resync \approx$
    Overall_Detection_Time/Numbering_Span#
Ensure: $M$ (matching set), $N_{up}$ and $N_{down}$ (non-match set on upstream and downstream)
 1: $start_{up} \leftarrow$ Min(time-stamp in $D_{up}$) and $start_{down} \leftarrow$ Min(time-stamp in $D_{down}$)
 2: $numbering\_span \leftarrow 1$ and $index \leftarrow 0$
 3: for $sig_{up}$ in $D_{up}$ do
 4:     if $sig_{up}.timestamp \leq start_{up} + numbering\_span \times resync$ then
 5:         $index \leftarrow index + 1$
 6:     else
 7:         $numbering\_span \leftarrow numbering\_span + 1$
 8:         $index \leftarrow 1$
 9:     end if
10:     $sig_{up}.serialNo \leftarrow index$
11:     $sig_{up}.ns \leftarrow numbering\_span$
12: end for
13: numbering signatures in $D_{down}$ likewise
14: for $sig_{up}$ in $D_{up}$ and $sig_{down}$ in $D_{down}$ do
15:     if $sig_{up}.serialNo = sig_{down}.serialNo$ and $sig_{up}.ns = sig_{down}.ns$ then
16:         $M \leftarrow M \cup \{(sig_{up}, sig_{down})\}$
17:         $D_{up} \leftarrow D_{up} \backslash \{sig_{up}\}$ and $D_{down} \leftarrow D_{down} \backslash \{sig_{down}\}$
18:     end if
19: end for
20: Put remaining signatures from either $D_{up}$ or $D_{down}$ into $N_{up}$ or $N_{down}$ correspondingly
21: return $M, N_{up}$ and $N_{down}$
```

vehicles are detected by only one sensor. The more frequent singletons occur, the shorter the reset interval should be. The reset interval should be in proportion to vehicle travel time. For our deployment, a large number—about one-third—of vehicles are seen by only one sensor, and travel time is about 30 s, so we anticipate a reset interval of $3 \times 30 = 90$ s. We verified this intuition with exhaustive analysis of possible reset intervals, where we found 83 s was our optimal reset interval.

We evaluate NwR in Section 9.1 and find that it provides reasonable correctness (we define *recall* in Section 9.1.1; its recall is 64%, Table 11). However, it is fundamentally difficult to handle vehicles that leave the roadway (for example, by parking) with numbering-based algorithms. We therefore next consider time-based algorithms to handle this case.

6.2 Time-Stamp Based

The next group of algorithms match using the actual detection *times* of vehicles rather than detection order. Stations record a timestamp with each signature, and if we assume travel time between signatures is predictable, we can use differences in these timestamps to match the signatures.

6.2.1 Static Time Window (STW)

STW assumes travel time between sensors is relatively consistent (say, around δ), and so it predicts that an upstream signature at time t corresponds to a downstream signature in $t + \delta \pm v$, where v is the range of variation allowed in travel time. This assumption is true provided vehicles typically travel at consistent average speeds between two sensors [30]. Depending on the window is set, this assumption holds for 90% of vehicles or more as described in Section 9.2.

This algorithm takes advantage of the reality that vehicles in a normal traffic flow tend to maintain a constant speed. Drivers usually observe a 35 mph speed limit in commercial district roadways and 25 mph in local residences in California. Hence a coarse speed range can be easily determined.

Algorithm 4 *Static (Dynamic) Time Window* algorithm

```
Input: $D_{up}$ and $D_{down}$ and time window $[tw_{lo}, tw_{hi}]$
Input: *** a shift value $sv$
Output: $M$, $N_{up}$ and $N_{down}$
    {Note: activate lines with "***" mark in dynamic time window algorithm; ignore them in static.}
 1: for $sig_{up}$ in $D_{up}$ do
 2:     for $sig_{down}$ in $D_{down}$ do
 3:         if $tw_{lo} \le sig_{down}.timestamp - sig_{up}.timestamp \le tw_{hi}$ then
 4:             $M \leftarrow M \cup \{(sig_{up}, sig_{down})\}$
 5:             $D_{up} \leftarrow D_{up} \backslash \{sig_{up}\}$ and $D_{down} \leftarrow D_{down} \backslash \{sig_{down}\}$
 6:             *** Reset $tw_{hi}$ and $tw_{lo}$
 7:             break
 8:         else if $sig_{down}.timestamp - sig_{up}.timestamp > tw_{hi}$ then
 9:             $N_{up} \leftarrow N_{up} \cup \{sig_{up}\}$ and $D_{up} \leftarrow D_{up} \backslash \{sig_{up}\}$
10:             *** both $tw_{hi}$ and $tw_{lo}$ decreased by $sv$
11:             break
12:         else
13:             $N_{down} \leftarrow N_{down} \cup \{sig_{down}\}$ and $D_{down} \leftarrow D_{down} \backslash \{sig_{down}\}$
14:             *** both $tw_{hi}$ and $tw_{lo}$ increased by $sv$
15:         end if
16:     end for
17: end for
18: Put remaining signatures from either $D_{up}$ or $D_{down}$ into $N_{up}$ or $N_{down}$ correspondingly
19: return $M, N_{up}$ and $N_{down}$
```

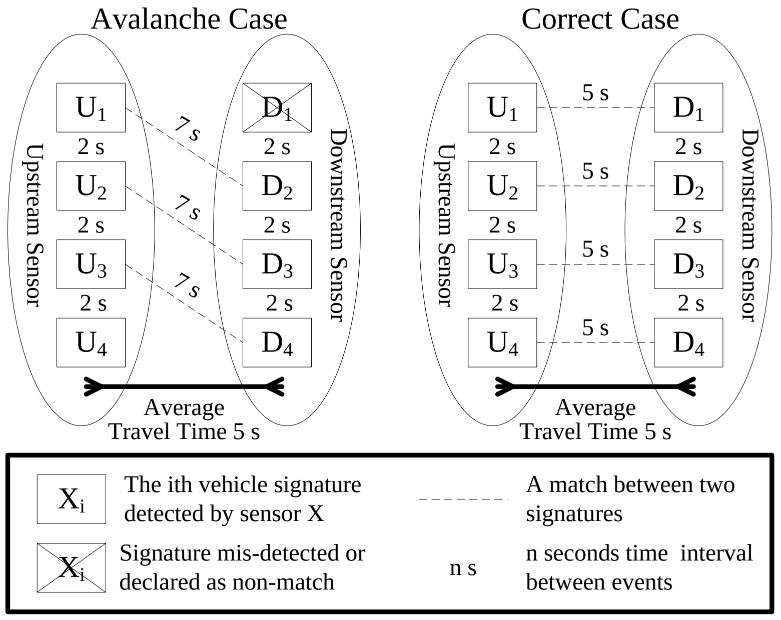
Our STW implementation works as follows (Algorithm 4). The downstream station is responsible for matching; it holds all pending signatures (not yet matched) reported by upstream sensor. When the downstream station detects a new signature, it examines upstream signatures in sequential order. If the timestamp difference between downstream signature and the first buffered upstream signature falls in the time window ($\delta \pm v$), a match is declared and the upstream signature is removed from consideration. If the time difference is less than the smallest possible value, we declare the downstream signature a non-match. If more than the largest, we declare the upstream a non-match. STW is a simple algorithm, well suited to on-line processing. We employ STW in our experimental system (Section 7.3).

However, similar to NN, an incorrect match can throw off future matches if alignment between signatures becomes skewed, as shown in Figure 11. Suppose a vehicle arrives every 2 s, travel time $\delta = 5$ s, and $v = 2$ s. If each vehicle generates a signature at both sensors, all will be correct matched. But if one signature is not recorded (perhaps that vehicle is out of its lane and misses the sensor), all subsequent signatures will be mis-matched, since vehicle separation is within the v window of variation. However, if two vehicles are spaced slightly further apart, STW automatically uses the gap to reset itself and so it is less susceptible than NN to this problem.

A second potential weakness of the algorithm is that the time window δ is fixed. In practice we set the algorithm based on the upstream/downstream sensor distance and typical vehicle travel times; this configuration can easily be automated. However, we next describe two additional algorithms that adapt δ dynamically to account for changing conditions, and we later evaluate choice of parameters for all algorithms in Section 9.2.

6.2.2 Dynamic Time Window (DTW)

STW's fixed time window (δ and v) requires configuraiton. To avoid manual configuration, and to better address the avalanche problem, we next dynamically adjust these values with DTW (Algorithm 4). Since



Avalanche Case
Correct Case
Upstream Sensor
Downstream Sensor
U_1
U_2
U_3
U_4
D_1
D_2
D_3
D_4
2 s
7 s
5 s
Average Travel Time 5 s
X_i
The ith vehicle signature detected by sensor X
A match between two signatures
Signature mis-detected or declared as non-match
n s
n seconds time interval between events

Figure 11: Avalanche problem in STW.

non-matched signatures caused avalanches, we adjust the time window after a non-match with the goal of converging on a good value over time.

Figure 11 suggests adjusting the time window helps. After we declared a non-match on D_1, we decrease δ by 1 s. The reason is that we believe most downstream non-match is mainly caused by vehicles travel too fast, and likewise, over-slow vehicles results in upstream non-match. And speeds of vehicles in a platoon is *not* independent, meaning follow-up ones are likely to have a shorter travel time than $\delta - v$. We shift the time window back to suppress the transient fast traffic flow. While the next signature U_1 will also be incorrect (because the 7 s travel time is outside the window [2,6] s), all further signatures will correctly match.

To control how much the widow moves after a non-match, DTW uses *Shift Value* (*sv*). This new parameter controls the amount of change to the travel-time estimate each non-match. After a successful match we reset δ to the original value. The effect of *sv* is discussed in Section 9.2.

Table 11 shows that DTW improves recall by 2% over STW. However, vehicles that "legitimately" pass a single sensor (for example, parking between the sensors) trigger DTW incorrectly. We therefore next consider the use of additional information to determine accidental missed sensor readings from vehicles that truly trigger only a single sensor.

6.2.3 Wheelbase-Enhanced Time Window (WETW)

Prior algorithms consider only signature timing. We next use *signature features*, such as number of wheels or wheelbase length, to evaluate match correctness. If features are reliable, they can select between multiple potential matches, or rule out incorrect matches.

We choose wheelbase (distance from the front to rear vehicle axle) as the feature for WETW. We already extract wheelbase for classification. WETW starts with the STW algorithm to find a tentative upstream/downstream signature match. However, it then builds on this match by considering the signatures immediately before and after the upstream one. Each signature is tested to see if it falls within the STW time constraints, and also if it approximately matches the downstream wheelbase (plus or minus a *wheelbase window* factor to account for observation error). We then take the first signature that matches both constraints, even if this means undoing a prior match. Our goal here is throw out obviously poor matches. WETW therefore also delays decisions by one signature to allow this wheelbase-triggered re-matching. The details are in Algorithm 5.

Feature-based methods like WETW add an additional dependency: feature extraction from signatures

Algorithm 5 *Wheelbase Enhanced Time Window* algorithm

```
Input: Pre-matched signatures via STW, $M$, $N_{up}$ and $N_{down}$. Wheelbase window and time window.
Output: improved Matching results, $M$, $N_{up}$ and $N_{down}$
 1: repeat
 2:     for $sig_{up}^{i}$ in $N_{up}$ do
 3:         if $sig_{up}^{i-1}$ is matched to $sig_{down}$ then
 4:             if the wheelbase difference between $sig_{up}^{i-1}$ against $sig_{down}$ is outside wheel window and the differences between $sig_{up}^{i}$ and
                $sig_{down}$ fall in both wheelbase and time window then
 5:                 $M \leftarrow M \cup \{(sig_{up}^{i}, sig_{down})\}$ and $M \leftarrow M \backslash \{(sig_{up}^{i-1}, sig_{down})\}$
 6:                 $N_{up} \leftarrow N_{up} \cup \{sig_{up}^{i-1}\}$
 7:             end if
 8:         else if $sig_{up}^{i+1}$ has a match then
 9:             similar process as forgoing
10:         end if
11:     end for
12: until no new matching declared
13: for all signatures in $N_{up}$ and $N_{down}$ do
14:     if the differences of two signatures $N_{up}$ and $N_{down}$ fall in both wheelbase and time window then
15:         transfer these two signatures from $N_{up}$ and $N_{down}$ to $M$
16:     end if
17: end for
18: return $M$, $N_{up}$ and $N_{down}$
```

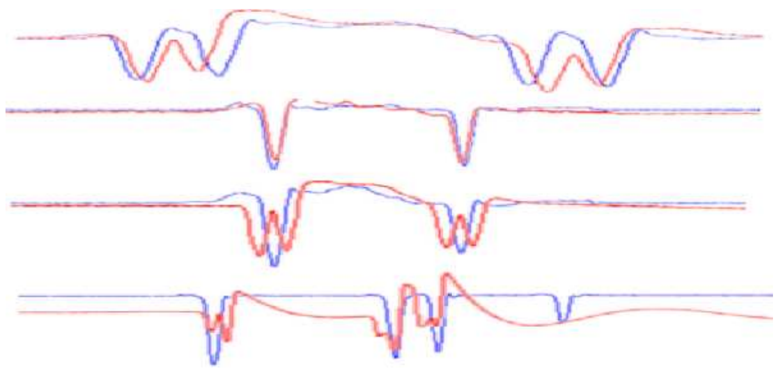


Figure 12: Four pre-scaled raw signatures for comparison. The horizontal axis is time and the vertical is energy.

is not perfect, so incorrect feature extraction actually degrade matching. WETW works best when most vehicles have different wheelbases (say, a mix of cars and trucks). Finally, it considers one possible feature (wheelbase length) in addition to timing, although in principle one could use other or multiple features. We next consider full signatures as a richer feature.

6.2.4 Raw-Enhanced Time Window (RETW)

The better recall of WETW, compared to STW (Table 11), suggests that more information helps matching. To determine if more information *always* improves results, we replace the matching function in WETW, changing it from wheelbase length to full comparison of raw signatures. We call the new algorithm *Raw-Enhanced Time Window.* Raw signatures record the change of loop inductance as the vehicle crosses, sampling at 300 Hz; thus they represent the most complete information about what vehicles pass a sensor. Figure 12 shows four example signature pairs. Red and blue (darker) lines represent different signatures.

One cannot directly compare raw signatures, because slight differences in vehicle speed or signature segmentation result in different length signatures. To correct for this distortion, we use *Dynamic Time Warp* [41] to compute the similarity between raw signatures. This approach has two steps. First, it warps the time axis of one signature iteratively until each data point in this sequence is optimally aligned to a point in the other signature. Second, it evaluates the similarity of the signatures by summing the Euclidean distance of between all point-pairs in the warped signatures. (This evaluation measure is also used to evaluate the quality of warping.) Other than this change of comparison function from wheelbase to time-warped signature, RETW is similar to WETW, using the same constraints on signature sequencing and timing.

Surprisingly, Table 11 shows RETW has slightly poorer recall than WETW, with one fewer correct non-matches of the total 107 correct matches and non-matches.

To evaluate if combining WETW and RETW would yield better results, we compare vehicle-level matching results between these two algorithm. We find that the result are similar; only two matches out of total 138 events are different. Therefore, an oracle algorithm producing a union of the correct matches of these two algorithms would improve by at most 2% in this study. We next evaluate if comparing full, raw signatures helps.

6.3 Full Raw Signature Comparison

Our time-window family of algorithms use some feature to shift signature matches. However, potentially one could compare all signatures against all other signatures, as implemented by Cheung et al. [6]. In Section 6.2.4 we modified the time window algorithm to consider full signature comparisons. Here we remove the temporal constraints of RETW to see if full matching freedom can improve results. However, it turns out that more information does not help RETW do matching.

To test this question, *Full Raw Signature Comparison* compares *all* signatures at the two sensors. This algorithm works by testing each match against all signatures at the other sensor side. We use the dynamic time warp as the comparison function. This comparison declares a *tentative match* as the closest possible score. We then test this tentative match a threshold to see if it is a good match, or instead to declare that signature as a non-match. We determine the threshold by training on known ground truth and using the median distance score of true matching. Thus the main difference against RETW is that full matching always compares all signatures (using maximal information), while RETW compares a time-constrained subset.

We find full matching is much worse than RETW: *none* of the 65 matchable vehicles find its true counterpart. This somewhat surprising result is mainly due to two problems. First, without temporal constraints (like RETW), raw matching relates many signatures that are unreasonably earlier or later. Second, full matching requires a threshold to determine match/non-match, but there is no single fixed threshold that identifies correct matches. When we train with known correct matches, the median distance score among the 65 true matches is four times higher than that of all tentative matches. In otherwords, for real data, incorrectly matched vehicles always look more similar to each other than true matches—too much information can mislead.

We see three causes raw signatures often fail to match. First, environmental noise and measurement error may distort signatures, sometimes causing wheels to be mis-detected. Since distortion is usually independent at the two sites, signatures of some true matches are inherently different from each other. For example, while the top row of Figure 12 shows two complete signatures, in the third row the darker (blue) signature recorded only one wheel, likely because the car was straddling lanes. However, we can sometimes compute a correct wheelbase for partial signatures, even if one wheel is missed. Less information (a partial signature) can still result in a correctly matching feature (wheelbase).

Second, a large number of signatures make accidental mis-identification easy, particularly with many vehicles of similar general type (passenger car, truck, etc.) or make. Table 8 shows that raw signature matching often (about 30% of the signatures) finds the best match as a vehicle of another category, even though a human would detect that clear mistake. With more than 100 potential signatures, accidental matches are increasingly likely. Thus time constraints (in the time-window algorithms) help focus on plausible candidates. Finally, DTW can arbitrarily warp signatures, and perhaps too much freedom makes mis-identification easier. Our time-window algorithm corrects for speed differences, but assumes acceleration and higher order derivatives are zero, perhaps a more reasonable assumption than allowing on-zero higher-order derivatives.

We examine raw signature comparison to get a best possible result by using all available information. In doing so, we ignore the network bandwidth and energy requirements of sending around full signatures. We conclude that, for our sensors, careful chosen features (such as wheelbase) represent vehicles *better* than full information, because feature detection filters out noise.

Other researchers have reported better results comparing full signatures (Cheung et al. report 100% re-identification [6, 7], details in Section 2). We believe they succeed because their test set is smaller (seven vehicles), their sensor spacing closer (several meters away), and their sensor provide informative (three-axis

Table 8: Category-level matching results by full raw signature comparison

105 vehicle passed upstream site	# of whose best match in downstream is a **passenger car**	**SUV**	**truck**
24 passenger cars	15	8	1
64 SUVs	4	57	3
17 trucks	0	15	2

magnetometer). While their results suggest the need for more work, to see if their results generalize to larger datasets, and more distant or different sensors.

Because of these challenges, we conclude that both full signature comparison, both with all signatures and time-limited signatures, is not desirable—too much information hurts more than it helps. Instead, wheelbase or other extracted features can provide better results by effectively filtering out noise, and time constraints help avoid improbable matches.

6.4 Algorithm Discussions

The advantages of algorithms diverge upon matching correctness, parameter sensitivity, complexity, real-timeness and applicability to different monitoring settings. DTW, WETW and RETW should have higher recall than STW. STW might not be able to do well under heavy traffic, because vehicles are lack of temporal separation, while DTW could handle the problem. If the traffic is promiscuous, WETW is sure to utilize wheelbase difference to make better decision. If no intersection amid the road and vehicles keep driving order, NwR could yield satisfactory result. Several parameters have to be embedded into each algorithm, but we want to keep the sensitivity minimized. Section 9.2 briefly concludes the relation between parameter and performance. One merit of all these algorithms, except RETW is low complexity, comparing to raw signature comparison.

In all, from the simulation result in Table 11 as well as our description above, we draw the conclusion that STW is the most appropriate algorithm for our short term experiment, while others could be analyzed in post-facto processing. STW yields a high-enough recall (about 73%) without losing applicability and simplicity. A more comprehensive performance analysis is in Section 9.1.

7 Data Collection

In our previous project, SURE-FT we used several datasets to draw our conclusions. Our primary dataset is a collection of 1500 vehicle signatures taken in August 2004 at USC as part of the SURE-SE project. To this we added two additional, small scale experiments taken on a parking garage rooftop at USC/ISI. We will summarize them briefly in Section 7.1. A brief description of our preliminary Year 1 ISI parking structure rooftop is in Section 7.2. Our Year-2 multi-sensor vehicle classification experiment is given in Section 7.3, drawing on material from our technical report [52].

7.1 Previous Experiments Summary

A brief summary about our previous experiments in SURE [19, 17] is given it this section.

7.1.1 USC August 2004 Experiment

From 7 am to noon, August 6, 2004, we collected traffic data at the USC University Park campus. Working with Steven Hilliard of IST, we collected 1500 detections of vehicles at three locations. Site A and site B, consisting of southbound (BS) and northbound (BN), are both shown on Figure 13, with a deployment separation distance of about 100m. Sensor data was supplemented with human observers and videotape to provide ground truth data. We selected three locations on internal campus streets to get a mix of low- and moderate-speed traffic. We selected a data collection day when construction was underway on campus, allowing us to capture a mix of periodic traffic, including the USC shuttle bus, construction traffic, including

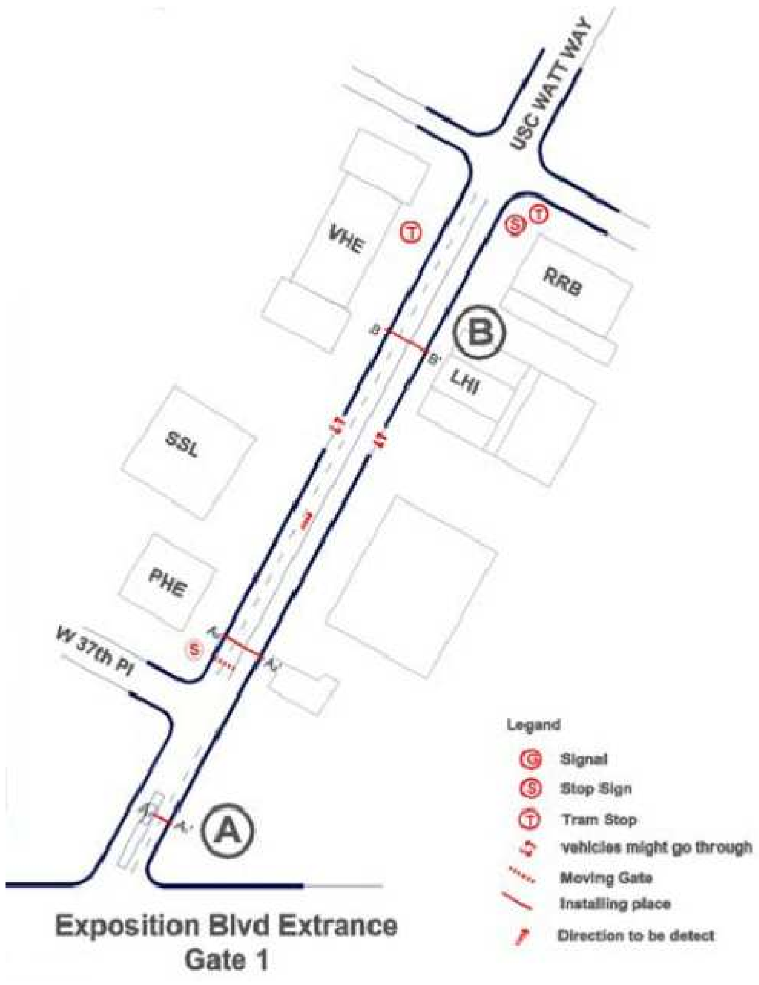


Figure 13: Placement of sensors for data collection.

cement mixers and 18-wheel trucks, and general automobile traffic. In addition to general traffic, we selected two passenger cars and ran them over each sensor 10 or more times to provide a baseline known vehicle to evaluate re-identification and sensor consistency. Details about exact deployment locations are available in a technical report [19].

7.1.2 USC/ISI Rooftop Experiment on 2005 and 2006

To examine the accuracy of a single-sensor regarding the sensor separation and speed of vehicles, we conducted an experiment on the roof of ISI parking structure. We took data on December 7, 2005. Combinations of different loop pairs separations and vehicle speed were tested and the results were analyzed against "ground truth" from radar gun.Data from this experiment is on the 061206 CD-ROM in `TrafficSensor_v1/data/2005_12_07` directory.

On July 27, 2006, both classification and auto-configuration algorithm were tested with a short experiment conducted at the same place. For these experiments we configured the IST card in automatic configuration. (The IST card has a dip-switch to change sensor sensor sensitivity according to attached loops types. We choose the "automatic" mode and let the detector card to determine the proper sensitivity for our custom 22 awg wire.) Each of three vehicles were used as a probe vehicle (as described in Section 5), then collected 8–11 signatures of each vehicle. Results from this experiment are described in Section 3.4. Data from this experiment is on the 061206 CD-ROM in `TrafficSensor_v1/data/2005_07_27` directory.

7.1.3 Data Availability

Complete copies of the data described in this paper are available on request from the authors, with information available on the project website `http://www.isi.edu/ilense/sure/index.html`. This includes both the new experiments done as part of SURE-FT (available on one CD-ROM), and from the USC experiments done as part of SURE-SE (provided as one CD-ROM of data and several DVDs of video ground truth).

The researchers have provided this dataset to several other research groups in the last several years.

7.2 SRVC Year 1 Signature Matching Rooftop Tests

To give our *multi*-sensor system an on-field test, on November 6 and 18, 2008, we carried out 2 small-scaled experiments on the roof of ISI parking structure. The equipment we used are almost the same as our previous experiment, except for an addition of a Linksys wireless router. We installed two sensor sites on parking structure, an Upstream and a Downstream, which were then connected by 802.11 wifi. A master server are initialized separately on Upstream node. After the calibration process, the probe vehicle in Section 5.1, we drove 2 different sedans, a Mercury Cougar V6 and a BMW 325i across both sensor loops along the road for several times to do enough data collection. Two sensors sent vehicle feature vectors back to the server, who does signature matching under *Static Time Window* (Section 6.2.1) algorithm as well as the final sensor fusion. The experiment result is trivial and we will not do a further analysis in this report, since there are only 2 different vehicles and no other distracting traffics along our test field. However, the success of these 2 preliminary experiments proves not only our newly-developed system is working but also an *automatic* multi-sensor classification is totally possible. Our system now needs testing in a more realistic and complicated traffic environment.

7.3 SRVC Year 2 USC Data Collection Experiment

From 8 a.m. to noon, February 19, 2009, we carried out a field test and traffic data are collected at USC campus. Figure 16 briefly shows our plan for the new experiment. Figure 14 and 15 depicts the location of our planned sensor deployment.

The goal of this experiment is to show that a fully auto-configured multi-sensor vehicle classification system could be built. We also plan to verify multi-sensor classification accuracy in a real world scenario, comparing to the result in 2004 USC experiment, where we were using a perfect (manual) signature matching algorithm.

During the nearly 3-hour long field test, we collected about 300 detections of vehicles at upstream and downstream locations on a public road on our campus. We also took videotape of traffic and later manually examined this record to generate ground truth. The collection stations were on one of USC campus internal streets, with two stations 90 m distant, each with two adjacent Blade sensors. Figure 16 shows real deployment—each station has a laptop, an IST-222 detector and two loop tapes. A photo of real deployment is shown in Figure 17. Stations were connected by a wireless router, 8 dB wireless dish adapter and 15 dB high-gain antenna. Although our campus has campus-wide wireless coverage, we deployed our own LAN to mimic the same kind of deployment that would be used on a city street. The downstream site both detected vehicles and did signature matching and sensor fusion from upstream signatures. We used sensor calibration as described previously [34], and each station ran local single-sensor classification, while the master (the downstream node) performed on-line signature matching using STW (Section 6.2.1) and then sensor fusion.

Before we report our results, we describe the traffic and on-line processing. First, we observed a mix of traffic including general automobiles, campus busses, shuttle vans, construction vehicles, delivery and semi-trailer trucks. We observed 33 passenger cars, 86 SUVs and 19 trucks, and a number of carts, motorcycles, and bicycles. Our system automatically discards the signatures of carts, motorcycles, and bicycles from our dataset because our goal is to classify cars. Second, although we did on-line processing in the field, the results reported here have been re-evaluated post-facto. This re-evaluation is necessary because our field experiment was mis-calibrated with incorrect typical vehicle speeds.

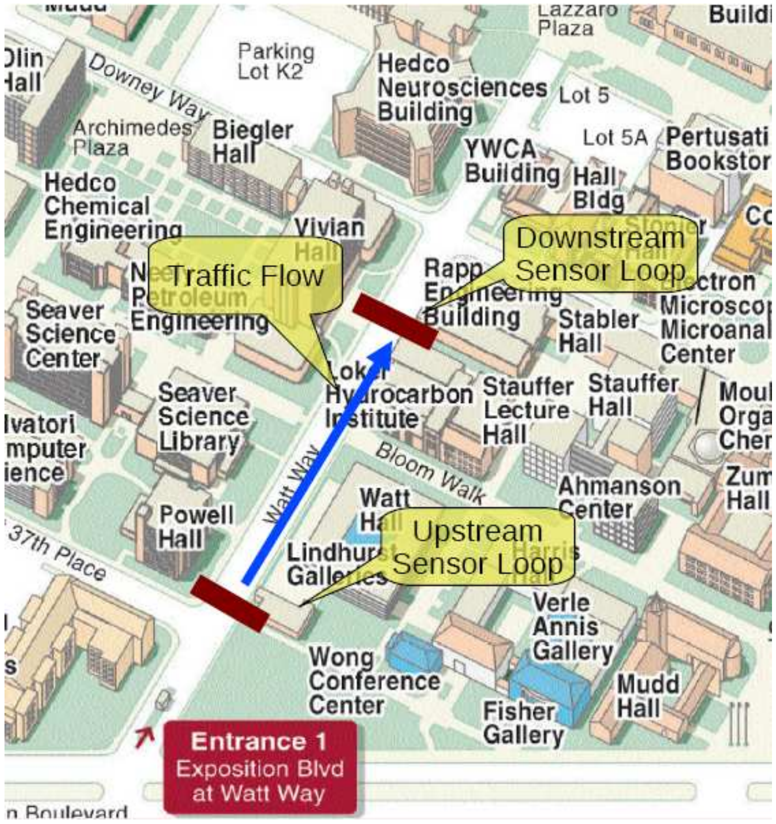


Figure 14: Placement of sensors for Year 2 field test.

8 Simulation Evaluation of Signature Matching

In order to test the consistancy in performance of our signature matching algorithms, introduced in Section 6, we ran a simulation over 2004 USC campus experiment dataset. This simulation provided preliminary analysis of our work before the year 2 experiment (Section 9). A list of algorithms performance is given in Table 9. We define recall, our metric to evaluate matching correctness, later in Section 9.1.1.

For clarification, we will explain the meaning of the parameters in Table 9. For time stamp based algorithms (Static/Dynamic Time Window and Wheelbase Enhanced), "parameter" is the time window size, i.e. the time difference for the same vehicle running over upstream and downstream sensor nodes. For numbering based algorithms, it means the span between two resets of counters for signatures. Finally, it becomes the initial value of time windows.

We ran this simulation to prove our hypothesis that those matching algorithm in Section 6 will successfully match or non-match those signature detected on upstream and down stream sensor nodes. We ran our algorithms over 04 USC experiment dataset and modify the internal parameter of matching algorithms. After that, a comparison was given to discover the performance improvement on correct matching rate. We expected that our algorithm will give out an acceptable matching rate, say 80%. Otherwise, it will be useless for the ultimate classification rate improvement.

As Table 9 shows, some algorithms, with particular internal parameter, are capable of fulfilling the above task. In general, time window based algorithms' performance is good, since they can be deployed in real time. Another observation is that recognizing True Positive still needs better approaches. Averagely, only one out of four non-match cases can be detected correctly by the system. We will discuss individual algorithm performance in following paragraphs.

Static Time Window: The correctness of the algorithms depends a lot on traffic flows variation. For an appropriate time window value, we do achieve good signature matching ($P(correct|parameter = 380, 320) \approx 85\%$). If vehicles do not maintain a relatively constant speed between two sensor nodes, we can not expect

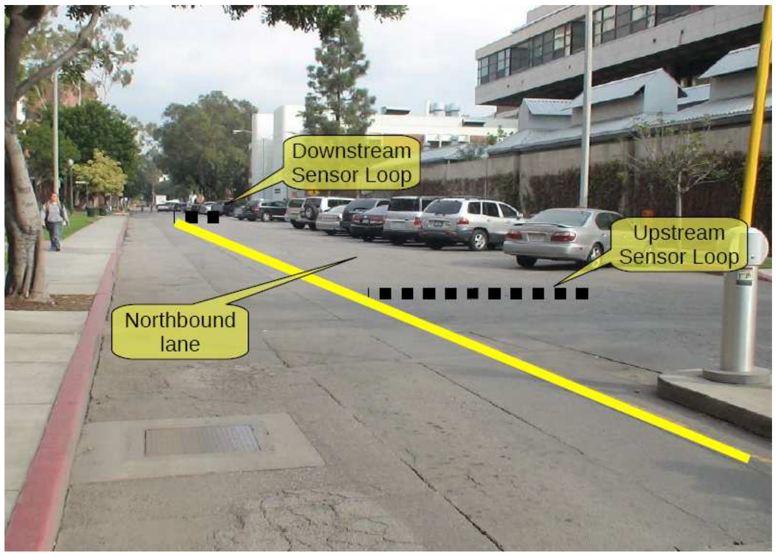


Figure 15: A site survey for our sensor deployment

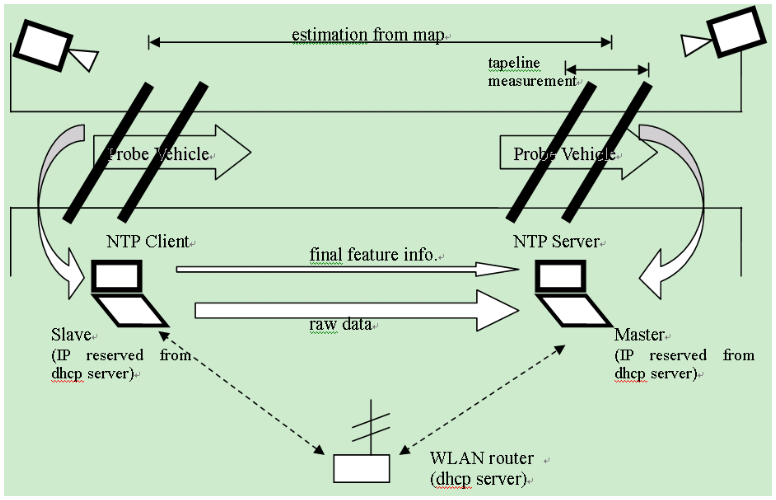


Figure 16: A sketch of our experiment deployment.

an acceptable matching rate. Another flaw of the algorithm is that the time window is manually specified, it somewhat violates our goal of automatic configuration (Section 5. Traffic engineers will have to re-estimate an appropriate time window value for the system before every experiment, because of the divergence of traffic situation, say congested or un-congested or different speed limit.

Dynamic Time Window: The algorithm does not have a significant improvement over Static Time Window as we expected. Another surprise is that tuning Shift Value does not have a large performance boost, as shown in Table 9. The reason for this, from our observation, is that in this experiment, vehicles tended to maintain low speed variation, namely not much acceleration and deceleration between sensor sites.

Wheelbase Enhanced: With the help of wheelbase information, the system does make slightly better matching decision, better than Static Time Window algorithm. We believe if the wheelbase estimation is more accurate, it will offset, to some extent, errors from vehicle speed change.

Numbering with Resynchronization: The algorithm do not yield an acceptable matching result. The intermediate output of our simulation program shows that the signatures for the same vehicle are prone to fall in different, successive, numbering span. With this intrinsic problem, shorter numbering span does not

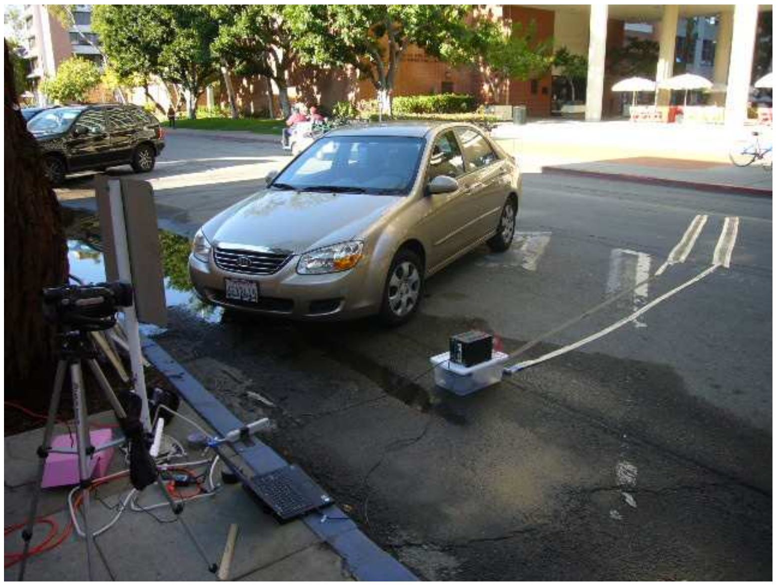


Figure 17: North sensor site in the year 2 deployment.

Table 9: Simulation Result

Algorithm	**Parameter**	**Recall (%)**	**Correct** matches	**Correct** non-matches	Incorrect matches	Incorrect non-matches
Static Time Window	400-310	87.50	34	1	4	4
	380-310	90	35	1	3	1
	370-320	44.23	22	1	4	25
Dynamic Time Window	**Shift Value = 2**					
	380-310	90	35	1	3	1
	380-325	72.09	30	1	5	7
	370-320	44.23	20	1	4	25
	Shift Value = 10					
	380-310	90	35	1	3	1
	380-325	74.42	31	1	4	7
	370-320	39.62	20	1	5	27
Wheelbase Enhanced	400-310	87.5	34	1	4	1
	380-310	92.5	36	1	2	1
	370-320	44.23	22	1	4	25
Numbering with Resynchronization	100	32.08	17	0	7	29
	500	64.29	27	0	8	7
	800	45.10	23	0	14	4

mean a better correct matching rate, since more counter resets occur.

These simulation results provided preliminary results and gave us confidence in carrying out our Year-2 evaluation, descrbed next. Our simulation results are generally consistant with our year 2 evaluation: First, timestamp based algorithms generally have better recall than numbering based algorithms, with appropriate parameters. Second, improvement from shift time window or wheelbase information is limited, considering a trade-off on simplicity, generality and parameter sensitivity. One last thing to note is that most of the vehicles in this dataset are *normal*, and that is why we see few correct non-matches.

Table 10: Event types for matching

Types	Expected Events per Occurrences	Occur-ences	Events
Normal	1 match	65	65
Singleton	1 non-match	46	46
Over-Segmented	1 match	13	13
Pull-Over	2 non-matches	7	14
PONO	1 match	0	0
total events		131	**138**

9 Evaluation of Signature Matching

To evaluate our matching algorithms we collected a 3-hour traffic dataset with our prototype system supplemented by human observers and videotape ground truth data. This section analyzes our year 2 experiment data, compare matching result among and within our algorithms, examine algorithm parameter sensitivities and finally the effect of matching over vehicle classification. This section draws on material from our technical report [52].

9.1 Matching Algorithm Correctness

Although we evaluated STW on-line, to compare all of our signature matching algorithms (Section 6), we replayed the data off-line through each one of the algorithms. Our goal is to maximize matching correctness in the face of real-world noise, and to compare matching algorithm performance and overhead. Our expectation is that exchange of more information (up to full signatures) would enable better matching, but instead we find that real-world noise fundamentally limits the correctness of matching.

9.1.1 Defining Correctness

Before looking at numerical comparisons we must first define our measure of correctness. Within the final output of matching system, there are four major situations: (i) Both sites detect a signature of a vehicle respectively, and the system declare a match on these two signatures, a *True Match*, or M; (ii) The system incorrectly declare a match on two signatures corresponding to two different vehicles, a *False Match*; (iii) One signature of a vehicle is missing at either site, and the system declared a non-match on the detected one, a *True Non-match*, or N; (iv) The system incorrectly declared a non-match for a signature which does have a counterpart detected by the other node, a *False Non-match.*

Case (iii), where signatures are missing from one site, is important because it shows how real-world conditions can violate the assumption that every signature must be matched. In practice, *not* every signature should be matched, for a variety of reasons. Signatures can be missing from either site because of sensor or algorithm error, undesirable vehicle/sensor interaction (for example, if the vehicle is half in the lane), or driver choices that violate our assumptions (for example, a vehicle that stops and parks between our sites). While sensor or algorithm errors can perhaps be corrected with better software or hardware, matching is impossible if vehicles never pass both sites.

We break vehicle detections into five groups (Table 10 shows how many of each we see):

Normal: vehicles drive continuously across two sites at a reasonable speed (say 10 to 40 mph)

Singleton: vehicles only pass one site

Over-Segmented: vehicles have more than two signatures generated on one node at the same time

Pull-Over: vehicles pull over in between the two sites and are overtaken by others. But they pass both sites. $T_{travel} > 200s$

PONO: (Pull-Over, Non-Overtaken) pull-over vehicles where no other vehicles overtake them (the relative order of vehicles maintained)

To evaluate correctness, we must normalize our results by number of true events. An *event* is an oracle-defined true match or true non-match ($M + N$). We determine oracle events by manual analysis of videotape to get accurate oracle results representing ground truth.

Algorithm	**Recall**	**Correct** **matches**	**non-matches**	Incorrect matches	non-matches	Reported signature#	Precision
STW	101 (**73%**)	46	55	23	28	152	66%
DTW	103 (**75%**)	41	62	17	43	163	63%
WETW	108 (**78%**)	51	57	19	24	151	72%
RETW	107 (**78%**)	51	56	19	25	151	71%
NwR	88 (**64%**)	31	57	18	65	171	51%
oracle	138 (100%)	78	60	0	0	138	100%

Table 11: Matching correctness of algorithms

To evaluate our currects, we draw terms from information retrieval [48]. IR defines *recall* as $tp/(tp+fn)$, characterizing how much of the true result is found. In our case, $tp + fn$ is the number of events, as defined above, since for orcale matching, the number of incorrect non-matches is always zero, while true positives represent correct matches and non-matches. The output of our algorithm is therefere evalutated by: $recall = (\widehat{CM} + \widehat{CN})/(M + N)$, where $\widehat{CM}$ represents the number of correct matches output by a matching algorithm, and $\widehat{CN}$ the output of correct non-matches.

We also report precision, to charcaterize how often a matching algorithm's output is incorrect: $precision = (\widehat{CM} + \widehat{CN})/(\widehat{CM} + \widehat{CN} + \widehat{IM} + \widehat{IN})$ where $\widehat{IM}$ are the number of incorrect matches (and $\widehat{IN}$ are incorrect non-matches). In general, we focus on recall to evaluate our correctess, but we also report precision.

9.1.2 Observations

Table 11 shows our evaluation of matching for this experiment. We draw several conclusions from the comparison among all of the algorithms. First, all algorithms are generally good—the poorest algorithm has matching recall above 60%.

Second, time-stamp based algorithms generally yield better correctness than others, with recall above 73%, 10% better than the 64% or lower rates of alternatives.

Considering the different time-stamp based algorithms, we observe that DTW, WETW and RETW provide only slight improvements over STW (a 2% or 5% improvement over STW's 73% recall). We therefore recommend STW as the preferred algorithm overall, because it is much simpler to implement and configure than DTW, WETW and RETW and nearly as accurate.

The three derivatives appear to have no significant improvement over the base time-stamp algorithm (only 3%–4%). If we examine more carefully, the change of actual matching correctness numbers indicates we have achieved our designing goal in Section 6.2.2 and 6.2.3. DTW has better recall and fewer incorrect matches but more incorrect non-matches than STW. WETW successes in correcting its base version's a few incorrect matches into correct matches.

Surprisingly, we find more information does not always help. RETW has slightly lower recall than WETW, meaning comparison of full signatures do not improve on evaluation of signature similarity by extracted wheelbase length.

9.2 Parameter Sensitivity

Each matching algorithm is controlled by several parameters. We here summerized how sensitive algorithm accuracy is to those parameter settings. We focus on time-stamp based algorithms because they are most successful. An ideal algorithm is insensitive to parameter settings, so even if mis-configured it will perform reasonably.

We begin by considering STW, where the time window size and location are the only parameters. Figure 19 shows how STW's matching accuracy varies for all possible window configurations. The x-axis and y-axis are the lower bound and upper bound of time window, while the grayscale value indicates STW's recall for our experimental dataset. The best accuracy (73%) occurs for range [14,44] s (δ = 29 s, v = 15 s). As can be seen in Figure 18, this time window accepts most vehicles, since the window center (δ = 29 s) is

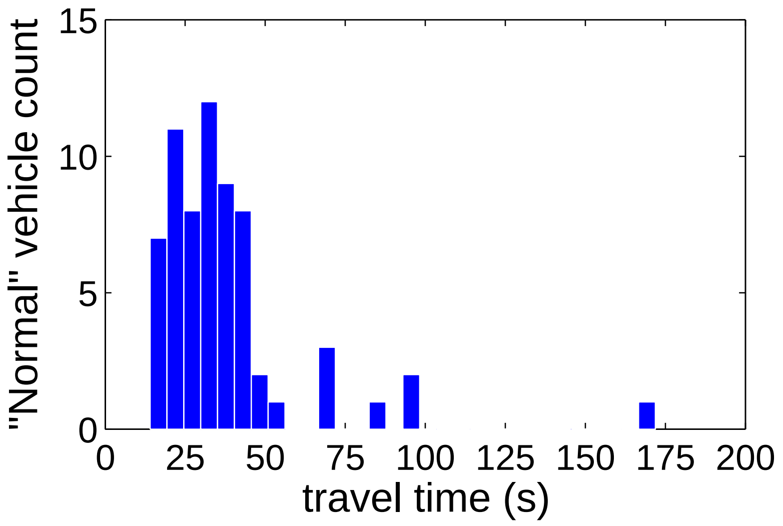


Figure 18: Travel time distribution of the 65 *Normal* vehicles.

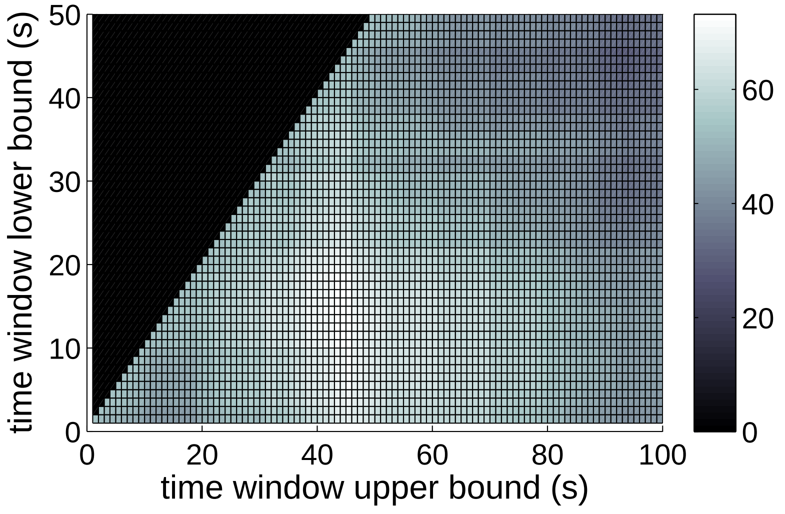


Figure 19: Accuracy % of STW.

near the median vehicle travel time (32 s). However, other outliers, which have a really long travel time is fundamentally difficult for any time-stamp based algorithm.

STW is fairly robust to an imperfectly set window. However if our parameters are off by 50%, we lose about 20% accuracy. And a 20% change of window only causes about 10% loss (accuracy 62%). In all, we draw three conclusions from above observations. First, as we expected, we have to locate the time window center (δ) close to median travel time to get optimal parameters. Second, the time window size should be adjusted according to vehicle speed variance. The reason is that the broader the window (large v), the more incorrect matches and less correct non-matches since it is easier to mistakenly match a singleton vehicle to another one. And the narrower the window (small v), the more incorrect non-matches and less correct matches because a small travel time range of *Normal* vehicles is accepted. Finally, STW works surprisingly well (above 60% accuracy) over a wide parameter ranges — upper/lower bound could various in ranges [30,50]/[10,20] s approximately.

DTW adds an additional parameter, the *Shift Value* (sv). To evaluate sensitivity to different shift values,

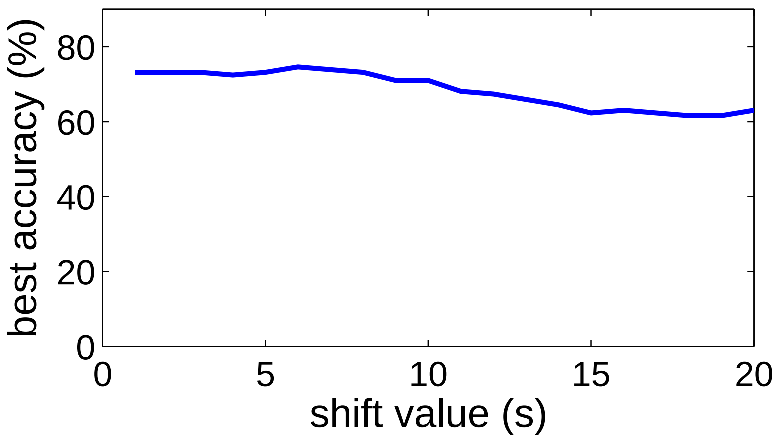


Figure 20: Best accuracy % of DTW with different *Shift Values*.

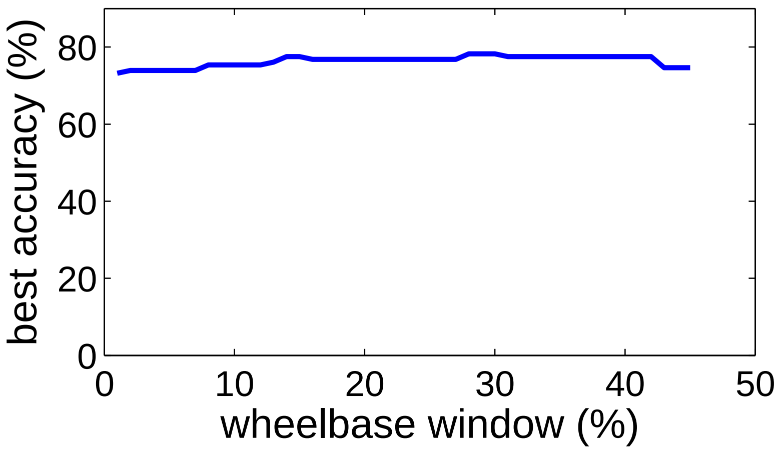


Figure 21: Best accuracy % of WETW with different *Wheelbase Windows*.

we search the entire parameter space (shift value and window bounds). For each sv, we pick the best accuracy over 2-D TW space and show them in Figure 20. We see that DTW is quite insensitive to sv, although accuracy drops when $sv \geq 8$ s and eventually lower than STW. The reason explains this is that traffic is sparse and hence our premise in Section 6.2.2 does not always hold. We assert that in a short platoon, vehicle speeds are not independent but in the experiment, most vehicle travels alone. Shifting time window too much makes it harder to match follow-up normal vehicle.

We also evaluated WETW using the same approach as DTW (with optimal other parameter) and found it is quite insensitive to the wheelbase window. We omit RETW discussion since it performs similarly to WETW.

Reviewing how parameters affects the matching accuracy shows us that our time-stamp based algorithms, in general, are insensitive to internal parameters. STW yields reasonable accuracy over a wide range of time windows. Besides, those parameters are easy to configure under certain rules. For example, time window should center around the estimated common vehicle travel time. The upper/lower bound of time window should be chosen close to (1±50%) of the travel time. We leave the parameter auto-configuration as an open issue.

Table 12: Classification accuracy, multi vs. single

Classification	**Veh.**	**Categories** two	**Categories** three
single sensor:			
upstream alone	105	80 (76%)	68 (65%)
downstream alone	99	79 (80%)	43 (43%)
oracle matching			
oracle fusion:	138	117 (85%)	90 (65%)
quality-best fusion:	138	111 (**80%**)	78 (**57%**)

9.3 Impact of Matching on Classification

The goal of signature matching in our system is to support multi-sensor fusion, or more generally, to synthesize conclusions from detections from multiple sensors. In this section, we study how the matching correctness affects multi-sensor fusion. Our hypothesis is that better matching algorithms result in better multi-sensor classification. However, classification and matching have a non-linear interaction since errors that make matching difficult also make classification difficult, so studying real data is important.

9.3.1 Multi-Sensor Classification Review

Park et al. previously showed that classification can benefit from multi-sensor fusion [34]. Combining readings from multiple sensors can correct some, but not all, classes of errors. For example, although a vehicle may temporarily leave a lane and so be mis-detected by one sensor, it likely returns to its lane later. Park et al. examine the accuracy of several sensor fusion algorithms relative to human observation and show that sensor fusion can allow automatic classification rates exceeding that of human observers. Accuracy depended on how many groups were classified (2 or 3, with more categories having lower accuracy because there is more opportunity to error), and the sensor fusion algorithm. They found the best accuracy was for their *quality-best* fusion algorithm, giving 97% for 2-category and 74% for 3-category. By comparison, human observation had 87% for 2-category and 83% for 3-category, and 100% accuracy is actually impossible because of overlap in the categories themselves.

This prior work, however, assumed a perfect (oracle-based) signature matching algorithm. We next evaluate classification accuracy with a realistic and therefore imperfect matching algorithm.

9.3.2 Evaluation Baseline

Building on prior work [34], we consider two classification tasks: three-category of passenger cars, light trucks (SUVs or pickups trucks), large trucks (FHWA classes 2, 3, and 4–13) and two-category of trucks and non-trucks (FHWA classes 2–3 and 4–13) [12]. Three-category is inherently harder because many light trucks can easily be confused with cars and even humans have difficulty to make a perfect judging (when small SUVs blur into cars) [34]. Our goal here is to evaluate how matching effects results of realistic classification, so we set as our baseline oracle matching, and then compare to realistic matching algorithms. We use quality-best fusion, the best choice from [34]. Each sensor assigns a quality value for each signature it extracted, based on wheel detected, signal strength and other factors. Hence when fused, a matched vehicle could have 2 candidate classifications and we choose the one with higher quality value.

Table 12 summarizes our baseline. For two-category and three-category, the baseline accuracies are 80% and 57% respectively. In general, improvement of oracle matching on two-category classification is insignificant (4% over upstream alone and 0% over downstream alone), However, it partially fixes the poor result from downstream sensor in three-category test (a 14% boost), because the better classifications from upstream suppress downstream ones with lower quality value in most matched cases.

9.3.3 Metrics

With this baseline we now must define how to quantify multi-sensor classification accuracy. Single-sensor classification accuracy is easily defined as the fraction of correctly-classified vehicle number by the total.

Table 13: Reported once and correctly classified once

signatures		**matching**	**classification**
ups.	**downs.**	**correct?**	**result**
U_i	D_i	correct matches	C(i)=F(U_i, D_i)=G(i)
U_i	X	incorrect	C(i)=F(U_i,X)=G(i)
U_x	D_i		C(x)=F(U_x, D_i)
U_i	D_x	incorrect	C(x)=F(U_i, D_x)
X	D_i		C(i)=F(X,D_i)=G(i)
U_i	–	correct non-matches	C(i)=F(U_i, –)=G(i)
U_i	D_x	incorrect matches	C(i)=F(U_i, D_x)=G(i)
–	D_i	correct non-matches	C(i)=F(–, D_i)=G(i)
U_x	D_i	incorrect matches	C(i)=F(U_x, D_i)=G(i)

U_i, D_i:	upstream and downstream signatures generated by vehicle i
X:	"don't care", i.e., non-matches or matches against a fake signature (noise) or a signature of some other vehicle
–:	means non-match
G(i):	the ground truth category of vehicle i is x
C(i):	multi-sensor classification result of vehicle i
F(x,y):	fusion of one or two signatures

Multi-sensor fusion with perfect (oracle) matching can also be defined similarly.

However, just as matching correctness is complicated by duplicate or undercounts (Section 9.1.1), those cases make it difficult to provide a simple accuracy metric for classification with imperfect matching. For example, if two detections of one true vehicle are not matched, and one is correctly classified and the other is not, does these two reports represent (i) two errors (since it was mis-matched and we cannot determine which is correct), (ii) one error and one correct result, or (iii) one correct result (taking the correct classification as overriding the incorrect duplicate)? Or what if a single vehicle was reported twice and classified correctly both times, is this (iv) incorrect, since it is over-reported, or (v) correct, since both reports are consistent? We can define the number of vehicles in each case as V_m^c, where c indicates how many times a true vehicle was correctly classified, and m indicates how many times it was reported.

We therefore define two levels of accuracy: strict and relaxed. *Strict* Accuracy is the most demanding: we require that each vehicle be correctly classified exactly once—conclusions (i) and (iv) above. If we define V_1^1 as the number of vehicles seen and classified exactly once, and V as the set of all true vehicles (events), strict accuracy is: $Accu_{strict} = V_1^1/|V|$.

Relaxed accuracy is relevant if, instead of demanding perfect counts, our goal is to approximate the percentages of each vehicle class. Here we consider overcounts due to incorrect matching to be correct provided both signatures are classified correctly, thus taking cases (ii) and (v) in the examples. If a vehicle is seen twice and classified correctly for twice, we define it as one V_2^2. We further define relaxed accuracy: $Accu_{relaxed} = (V_1^1 + V_2^2)/|V|$.

Although we talk about V_1^1 and V_2^2 here, there are actually a number of specific cases. We enumerate how we handle each case of V_1^1 in Table 13.

9.3.4 Matching Algorithm Effects on Classification

We next consider the effects of matching accuracy on end-to-end classification accuracy. For this evaluation, we compare against the baseline of perfect (oracle) matching, shown in Table 14. Each algorithm uses optimal parameters (as defined in Table 11)). The resulting signatures use quality-best fusion [34] to generate the final classification result.

The comparison proves our hypothesis, confirming that correctness in signature matching has a large, but correlated effect on end-to-end classification. (Here we refer to three-category classification; two-category classification is similar.) First of all, with our algorithms, the strict accuracy can approaches that of the baseline (50% for WETW vs. 57% oracle, with a 7% penalty due to incorrect matching). Categorization

Table 14: Multi-sensor classification accuracy

Matching		**2-cat. accu.** (%)		**3-cat. accu.** (%)	
Algor.	**Recall** (%)	**strict**	relaxed	**strict**	relaxed
STW	73 (-27)	69 (-11)	73 (-7)	49 (-8)	50 (-7)
DTW	75 (-25)	68 (-12)	76 (-4)	47 (-10)	50 (-7)
WETW	78 (-22)	70 (-10)	75 (-5)	50 (-7)	51 (-6)
RETW	78 (-22)	70 (-10)	75 (-5)	50 (-7)	51 (-6)
NwR	64 (-36)	59 (-21)	74 (-6)	39 (-18)	46 (-11)
oracle	100 (0)	80 (0)	–	57 (0)	–

with oracle matching is not perfect because the underlying single and multi-sensor classification methods are imperfect. The addition of realistic matching further lowers classification accuracy because mis-matched signatures can results in signature duplication, omission, or incorrect multi-sensor classification. We further study this correlation in Section 9.3.5.

Second, as expected, accurate matching helps improve classification while poor matching hurts. We find WETW matches signatures most accurately (78% against 64%, the worst case with NwR). Higher matching recall here shows a corresponding improvement in end-to-end classification accuracy, with WETW allowing 50% classification accuracy (vs. 39% worst-case, NwR). WETW's improvement is due to more correct matches (51 of 78 cases, Table 11), while poorer matching algorithms cause duplicated or omitted signatures.

Finally, our multi-sensor classification has moderate improvement over single-sensor in terms of both accuracy and robustness, even when coupled with realistic (imperfect) matching algorithms. We see that the downstream sensor is less accurate than upstream, possibly due to differences in vehicle speeds and channelization at the two sites. However, multi-sensor fusion improves downstream classification accuracy result by 7%, even with imperfect signature matching (WETW), showing that multi-sensor fusion can be more robust to deployment or sensor error.

While Table 14 shows how end-to-end classification accuracy changes due to matching, it doesn't show why the results differ. We look at that question next.

9.3.5 Understanding Correlation between Matching and Classification

We next look more deeply at *why* signature matching and classification accuracy affect each other. Both matching and classification use the same sensor data, so inaccurate data at one sensor (perhaps due to target or environment noise, or deployment differences) can *both* make matching difficult and affect multi-sensor classification accuracy. If the algorithms were completely correlated, then end-to-end accuracy should be the minimum of either algorithm's correctness. If they were strictly uncorrelated, then end-to-end accuracy should be their product.

Table 15 shows there is partial correlation between matching correctness and classification accuracy. We report matching recall for each matching algorithm, and three-category vehicle classification accuracy (with oracle matching), then compare expected accuracies with no and full correlation to experimental results. We find that the end-to-end multi-sensor classification accuracy with our matching algorithms is *always* between what would be predicted by no or full correlation. These experimental results suggest that accuracy of the two algorithms is partially correlated. This correlation shows up in end-to-end accuracy, where uncorrelated WETW would predict a 13% classification penalty (from incorrect matching), but correlation means that experimentally the penalty is only 7%.

To understand what causes these correlations, we next reanalyze the STW case from Table 14: Table 16 shows STW accuracy grouped by correctness in either or both matching and classification. The first case (yes, yes) is both matching and classification are correct, our goal. The second (yes, no) includes vehicles that are correctly matched or non-matched, but where multi-sensor classification gives an incorrect result. Presence of the third category where matching fails but classification succeeds (no, yes) is unexpected, but in these cases multi-sensor fusion selects the correct signature and result to recover. Vehicles in the fourth case (no, yes/double) are correctly classified, but because matching fails there appear to be two vehicles (one

Table 15: The correlation between matching and classification.

Algor.	**Match. recall** (m)	**Class. accu.** (c)	**Correlation none** $m \cdot c$	**full** $min(m, c)$	**experiments**
STW	73%	57%	42%	57%	49%
DTW	75%	57%	43%	57%	47%
WETW	78%	57%	44%	57%	50%
RETW	78%	57%	44%	57%	50%
NwR	64%	57%	36%	57%	39%

Table 16: Matching vs. classification in STW

correct?		**categories**	
matching	**classification**	**two**	**three**
yes	yes	83 (60%)	59 (43%)
yes	no	18 (13%)	42 (30%)
no	yes	12 (9%)	8 (6%)
no	yes/double	6 (4%)	2 (1%)
no	no	19 (14%)	27 (20%)
		138 events (100%)	

at each sensor), so we over-count (the V_2^2 case from Section 9.3.3). In final case (no, no) both matching and classification fail.

We draw three conclusions after comparing matching against end-to-end classification result. First, incorrect matches do not always result in incorrect classifications. In 8 out of 138 (6%) (no, yes) cases, matching fails but classification is correct. in three-category classification. The cases of incorrect matches or non-matches in Table 13 can still result in correct classification when $C(i) = G(i)$. Should matching and classification are completely correlated, these incorrectly matched signatures would never be correctly classified. Second, in the (no, yes/double) case matching fails and we overcount one vehicle twice, at each sensor. This case prompted us to consider strict and relaxed accuracy (Section 9.3.3), although with only 2 cases of 138 (1%), this event is rare. The only exception is with NwR matching, where 65 incorrect non-match (more than other algorithms, Table 11) results in more of these V_2^2 events (7% vs. others 1–3%, Table 14). Finally, we find correct matches do not always result in correct classification, either. Unfortunately, although our STW algorithm did well on 42 out of 138 (30%) (yes, no), our imperfect single-sensor classification and fusion fail to turn them into correct classification. We see opportunity that with better vehicle classification and sensor fusion might help us to achieve a 30% improvement.

These results are based on summarization of our results. We provide full details of all cases considered of Table 16 in Table 17. This tableThe fine-grained detail in accentuates the complexity in evaluating matching and classification in the presence of multiple kinds of error in each.

Overall, these results demonstrate that correlation between these algorithms has significant, quantifiable effects on end-to-end performance. While both algorithms can be studied and improved independently, we conclude that a full evaluation must consider both in the context of real data, and good overall accuracy requires a balance of good algorithms for matching, classification and multi-sensor fusion.

10 Conclusions

This report summarizes the research results of the SRVC project. Our work has verfiied one can effectively use multiple sensors to reduce error rates in vehicle classification, even with real-world signature matching. We have also begun the process of developing an integrated system that can be easily deployed and automatically configured to collect such data.

Table 17: Matching vs. classification in STW (dissection)

multi-sensor	**ups.**	**downs.**	**matching**	**reported**	**correct in multi-**	**# in STW**	
class. result	**sig.**	**sig.**	**correctness**	**veh. class.**	**sensor class.**	**2-cat.**	**3-cat.**
V_1^1	U_i	D_i	correct match	C(i)=F(U_i, D_i)=G(i)	strict; relaxed	37	29

V_1^1	U_i
	U_x
		X
		D_i
			incorrect	C(i)=F(U_i,X)=G(i)
				C(x)=F(U_x, D_i)
						3	2

V_1^1	U_i
	X
		D_x
		D_i
			incorrect	C(x)=F(U_i, D_x)=G(i)
				C(i)=F(X,D_i)=G(i)
						0	0

V_1^1	U_i	–	correct non-match	F(U_i)=G(i)		22	18
V_1^1	U_i	D_x	incorrect match	F(U_i, D_x)=G(i)		6	5
V_1^1	–	D_i	correct non-match	F(–, D_i)=G(i)		24	12
V_1^1	U_x	D_i	incorrect match	F(U_x, D_i)=G(i)		3	1

V_2^2	U_i
	X
		X
		D_i
			incorrect	F(U_i,X)=G(i)
				F(X,D_i)=G(i)
					relaxed	6	2

V_2^1	U_i
	X
		X
		D_i
			incorrect	F(U_i,X)=G(i)
				F(X,D_i)$\neq$G(i)
					none	3	5

V_2^1	U_i
	X
		X
		D_i
			incorrect	F(U_i,X)$\neq$G(i)
				F(X,D_i)=G(i)
						0	0

V_1^0	U_i	D_i	correct match	C(i)=F(U_i, D_i)$\neq$G(i)		9	17

V_1^0	U_i
	U_x
		X
		D_i
			incorrect	C(i)=F(U_i,X)$\neq$G(i)
				C(x)=F(U_x, D_i)
						4	5

V_1^0	U_i
	X
		D_x
		D_i
			incorrect	C(x)=F(U_i, D_x)
				C(i)=F(X,D_i)=G(i)
						0	0

V_1^0	U_i	–	correct non-match	F(U_i,–)$\neq$G(i)		6	10
V_1^0	U_i	D_x	incorrect match	F(U_i, D_x)$\neq$G(i)		1	2
V_1^0	–	D_i	correct non-match	F(–,D_i)$\neq$G(i)		3	15
V_1^0	U_x	D_i	incorrect match	F(U_x, D_i)$\neq$G(i)		0	2

V_2^0	U_i
	X
		X
		D_i
			incorrect	F(U_i,X)$\neq$G(i)
				F(X,D_i)$\neq$G(i)
						1	3

V_0^0	–	–	–	–		10	10
total vehicles						**138**	**138**

11 Acknowledgments

We are grateful for the assistance of Mr. Steven Hilliard, IST Corporation, the Los Angeles City Department of Transportation, and University of Southern California, Transportation Services Division. Several graduate students participated in the field data collection.

Many people assisted in our field experiments. We thank Min Zhu, Liang Han and Hsing-Hau Chen for their volunteering in 2009 field experiment. We thank Padma Haldar, Arnold Diaz, and Dongjin Son for use of their person automobiles for additional tests, and Dongjin Son for assisting in ISI rooftop experiments. We thank Unkyu Park for his help in lab and field tests.

We thank Fabio Silva for his help understanding prior SURE-SE and SURE-FT code.

We thank Andrew Goodney for his input on our raw signature matching algorithm.

References

[1] B. Abreu, L. Botelho, A. Cavallaro, D. Douxchamps, T. Ebrahimi, P. Figueiredo, B. Macq, B. Mory, L. Nunes, J. Orri, M.J. Trigueiros, and A. Violante. Video-based multi-agent traffic surveillance system. In *Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE*, pages 457 –462, 2000.

[2] National Highway Traffic Safety Administration. NHTSA Wheelbase and track width information. `http://www.nhtsa.dot.gov/`.

[3] D. Banister and J. Berechman. *Transportation Investment and Economic Development.* UCL Press, London, UK, 2000.

[4] Adam Baumberg. Reliable feature matching across widely separated views. In *Proc. of IEEE CVPR*, volume 1, pages 774–781, June 2000.

[5] Richard R. Brooks, Parmesh Ramanathan, and A. M. Sayeed. Distributed target tracking and classification in sensor networks. *Proceedings of the IEEE*, 91(8):1163–1171, August 2003. Invited Paper.

[6] Sing Yiu Cheung, Sinem Coleri, Baris Dundar, Sumitra Ganesh, Chin-Woo Tan, and Pravin Varaiya. Traffic measurement and vehicle classification with a single magnetic sensor. In *the Annual Meeting of the Transportation Research Board*. Transportation Research Board, January 2005.

[7] Sing Yiu Cheung, Sinem C. Ergen, and Pravin Varaiya. Traffic surveillance with wireless magnetic sensors. In *the 12th World Congress on Intelligent Transport Systems*, San Francisco, California, USA, November 2005.

[8] Benjamin Coifman. A new algorithm for vehicle reidentification and travel time measurement on freeways. In *Proc. of the Fifth Conference on Applications of Advanced Technologies in Transportation*, pages 167–174, April 1998.

[9] Benjamin Coifman. Vehicle reidentification and travel time measurement in real-time on freeways using the existing loop detector infrastructure. *Transportation Research Record No. 1643*, pages 181–191, 1998.

[10] Intel Stargate Computer. Crossbow Technologies SPB400.

[11] D. J. Dailey. Travel-time estimation using cross-correlation techniques. *Transportation Research Part B: Methodological*, 27(2):97–107, April 1993.

[12] Federal Highway Administration and Office of Highway Policy Information. FHWA vehicle types. web page `http://www.fhwa.dot.gov/policy/ohpi/vehclass.htm` and `http://www.dot.state.oh.us/techservsite/availpro/Traffic_Survey/SchemeF/FHWA_Scheme_F_Report.PDF`, October 2003.

[13] Ford. Ford Vehicle. `http://www.fordvehicles.com/`.

[14] Peter Gordon and Qisheng Pan. Assembling and processing freight shipment data: Developing a gis-based origin- destination matrix for southern california freight flows. Final report, National Center for Metropolitan Transportation Research (METRANS), 2001.

[15] Lin Gu, Dong Jia, Pascal Vicaire, Ting Yan, Liqian Luo, Ajay Tirumala, Qing Cao, Tian He, John A. Stankovic, Tarek Abdelzaher, and Bruce H. Krogh. Lightweight detection and classification for wireless sensornetworks in realistic environments. In *the Third ACM SenSys Conference*, pages 205–217, San Diego, California, USA, November 2005. ACM.

[16] M. Hansen, U A. Dobbins D. Gillen, Huang, and Puvathingal M. The air quality impacts of urban highway expansion: Traffic generation and land use change. Technical Report UCB-ITS-RR-93-5, Institute of Transportation Studies, University of California, Berkeley, CA, USA, 1993.

[17] John Heidemann, Yuri Pradkin, Ramesh Govindan, Christos Papadopoulos, Genevieve Bartlett, and Joseph Bannister. Census and survey of the visible Internet (extended). Technical Report ISI-TR-2008-649b, USC/ISI, February 2008. Updated August, 2008.

[18] John Heidemann, Fabio Silva, and Deborah Estrin. Matching data dissemination algorithms to application requirements. In *the First ACM SenSys Conference*, pages 218–229, Los Angeles, California, USA, November 2003. ACM.

[19] John Heidemann, Fabio Silva, Xi Wang, Genevieve Giuliano, and Mengzhao Hu. SURE-SE: Sensors for unplanned roadway events—simulation and evaluation: Final report. Metrans project 05-08 final report, USC/ISI, May 2005. finalized July 2007; typographic corrections April 2008.

[20] Inductive Signature Technologies Inc. IST sensor model 222.

[21] Inc Inductive Signature Technologies. Ist sensor model 222ps datasheet. `http://www.ist-traffic.com/datasheets/ist222ps.pdf`.

[22] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *the ACM International Conference on Mobile Computing and Networking*, pages 56–67, Boston, MA, USA, August 2000. ACM.

[23] Jaehoon Jeong, Shuo Guo, Tian He, and D. Du. APL: Autonomous passive localization for wireless sensors deployed in road networks. In *Proc. IEEE Infocom*, pages 583–591, April 2008.

[24] David B. Johnson and David A. Maltz. *Dynamic Source Routing in Ad Hoc Wireless Networks*, chapter 5, pages 153–181. Kluwer Academic Publishers, 1996. in Mobile Computing, edited by Tomasz Imielinski and Hank Korth.

[25] S. Kent and R. Atkinson. Security architecture for the internet protocol. RFC 2401, Internet Request For Comments, November 1998.

[26] Karric Kwong, Robert Kavaler, Ram Rajagopal, and Pravin Varaiya. A practical scheme for arterial travel time estimation based on vehicle reidentification using wireless sensors. In *Proc. Trans. Research Board Annual Mtg.*, January 2009.

[27] E. Miller, D. Kriger, and J. Hunt. Research and development program for integrated urban models. In *the Annual Meeting of the Transportation Research Board*, pages 169–176. Transportation Research Board, January 1999.

[28] D.L. Mills. Network time protocol (version 3) specification, implementation and analysis RFC 1305. `http://www.rfc-editor.org/rfc/rfc1305.txt`, March 1992.

[29] L. Mimbela and L. Klein. Summary of vehicle detection and surveillance technologies used in intelligent transportation systems. Technical report, New Mexico State University, 2000.

[30] Victor Muchuruza and Renatus Mussa. Traffic operation and safety analysis of minimum speed limits on Florida rural interstate highways. In *Proc. of the 2005 Mid-Continent Transportation Research Symposium*, August 2005.

[31] Cheol Oh, Stephen G. Ritchie, and Shin-Ting Jeng. Vehicle reidentification using heterogeneous detection systems. In *the 83rd Annual Meeting of the Transportation Research Board*, Washington, DC, USA, January 2004. Transportation Research Board.

[32] Songhwai Oh, Inseok Hwang, Kaushik Roy, and Shankar Sastry. A fully automated distributed multiple-target tracking and identity management algorithm. In *Proc. of AIAA Guidance, Navigation, and Control Conference*, August 2005.

[33] P. V. Pahalawatta, D. Depalov, T. N. Pappas, and A. K. Katsaggelos. Detection, classification, and collaborative tracking of multiple targets using video sensors. In *Proc. ACM/IEEE IPSN*, pages 529–544, April 2003.

[34] Unkyu Park, Fabio Silva, Mengzhao Hou, John Heidemann, Genevive Guiliano, Xi Wang, and Nikhil Prashar. Single- and multi-sensor techniques to improve accuracy of urban vehicle classification. Technical Report ISI-TR-2006-614, USC/ISI, April 2006. submitted for review.

[35] E. Pas. *The Geography of Urban Transportation, second edition, S. Hanson, editor*, chapter The urban transportation planning process. The Guilford Press, 1995.

[36] Charles E. Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. In *the ACM SIGCOMM Conference*, pages 234–244, London, UK, August 1994. ACM.

[37] Karl F. Petty, Peter Bickel, Michael Ostland, John Rice, Frederic Schoenberg, Jiming Jiang, and Ya'acov Ritov. Accurate estimation of travel times from single-loop detectors. *Transportation Research Part A: Policy and Practice*, 32(1):1–17, January 1998.

[38] Joseph Polastre, Jason Hill, and David Culler. Versatile low power media access for wireless sensor networks. In *the Second ACM SenSys Conference*, pages 95–107, Baltimore, MD, USA, November 2004. ACM.

[39] L. Raad and J.J. Lu. Development of a prototype traffic data collection system using infrared beam sensor array. Technical Report TNW-98-05, Transportation Northwest, Seattle, WA, 1998.

[40] Donald B. Reid. An algorithm for tracking multiple targets. *IEEE Trans. Automatic Control*, 24(6):843–854, December 1979.

[41] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. In Alex Waibel and Kai-Fu Lee, editors, *Readings in speech recognition*, pages 159–165. Morgan Kaufmann Publishers Inc., 1990.

[42] Jaewon Shin, Leonidas J. Guibas, and Feng Zhao. A distributed algorithm for managing multi-target identities in wireless ad-hoc sensor networks. In *Proc. ACM/IEEE IPSN*, pages 625–641, April 2003.

[43] Jaspreet Singh, Upamanyu Madhow, Rajesh Kumar, Subhash Suri, and Richard Cagley. Tracking multiple targets using binary proximity sensors. In *Proc. ACM/IEEE IPSN*, pages 529–538, April 2007.

[44] A. Sivakumar and C. Bhat. A fractional split distribution model for statewide commodity flow analysis. Technical report, University of Texas, Austin Department of Civil Engineering, Austin, TX, USA, 2002.

[45] Carlos Sun, Stephen G. Ritchie, and Seri Oh. Inductive classifying artificial network for vehicle type categorization. *Computer-Aided Civil and Infrastructure Engineering*, 18(3):161–172, 2003.

[46] Transportation Research Board. Expanding metropolitan highways. Special Report 245, National Research Council, Washington, DC, USA, 1995.

[47] Transportation Research Board. Evaluation of the congestion mitigation and air quality program. Special Report 256, National Research Council, Washington, DC, USA, 2002.

[48] C. J. van Rijsbergen. *Information Retrieval.* Butterworths, London, UK, 2nd edition, 1979.

[49] M. Wegener. Operational urban models: State of the art. *Journal of the American Planning Association*, 60(1):17–29, 1994.

[50] Wei Ye and John Heidemann. Ultra-low duty cycle MAC with scheduled channel polling. Technical Report ISI-TR-2005-604b, USC/ISI, July 2005.

[51] Wei Ye, John Heidemann, and Deborah Estrin. Medium access control with coordinated, adaptive sleeping for wireless sensor networks. *ACM/IEEE Transactions on Networking*, 12(3):493–506, June 2004. A preprint of this paper was available as ISI-TR-2003-567.

[52] Chengjie Zhang and John Heidemann. Evaluating signature matching in a multi-sensor vehicle classification system (extended). Technical Report ISI-TR-2011-675, USC/ISI, November 2011.

[53] Feng Zhao, Jaewon Shin, and James Reich. Information-driven dynamic sensor collaboration for tracking applications. *IEEE Signal Processing Magazine*, 19(2), March 2002.