Evaluating Signature Matching in a Multi-Sensor Vehicle Classification System (extended) *

ISI Technical Report ISI-TR-2011-675

Chengjie Zhang John Heidemann
USC/Information Sciences Institute

Nov. 2011

Abstract

Many academic sensornet systems consider the problem of tracking targets as they move through a fixed field of sensors. Such applications must relate sensor *detections* to specific *targets*, yet prior work has often ignored this problem, assuming either a single target, sufficient spatial separation that target-to-sensor mapping is clear, or some out-of-band detection-to-target mapping. There has been little study of algorithms to associate detections with targets, and effects of their accuracy on detection results, particularly in environments with dense targets. We explore the question of *detection-to-target* mapping in the context of a *vehicle classification system* for urban roadways, where vehicles pass fixed sensors at varying but frequent rates. We develop several *signature matching* algorithms that relate detections at different sensors to same or different vehicles. We evaluate these algorithms with data taken in a field test of live traffic compared against ground truth obtained through manual analysis of video and the resulting matching recall is over 78%. We investigate the effects of mapping accuracy on length-based vehicle classification. We show that accurate signature matching is critical to multi-sensor algorithms. We compare our matching algorithms against an oracle (perfect information), and find that all matching reduces end-to-end accuracy somewhat, but a poor matching algorithm reduce accuracy by 21%, while our best algorithm reduces it by only 10% in our case study. Finally, we quantify the degree of correlation between matching correctness and classification accuracy.

*This research is supported in part by USC/CSULB METRANS Project 07-04.

1 Introduction

Object tracking is one of the canonical problems for sensor networks. A fixed field of autonomous, inexpensive sensors observes their environment, identifies objects, then compares observations to track objects that move through the field [21, 11]. Object tracking has applications in military security, biology and animal detection and counting, and in workplace sensornet deployments [6],

An essential component of an object tracking algorithm is how the sensornet relates one or more sensor detections to one or more actual targets. However, most current sensornet work pays little attention to this sub-problem. Tests often assume as single target, sufficient spatial separation that target-to-sensor mapping is clear [14, 26, 20], or that some out-of-band source provides detection-to-target mapping (for example, [25]). More specifically, there has been little study of algorithms to associate detections with targets, and the effect of their accuracy on detection results, particularly in environments with dense targets.

In this paper we explore the question of detection-

to-target mapping in the context of a *vehicle classification system* for urban roadways, where vehicles pass fixed sensors at varying rates. We tie detections at different sensors to individual vehicles by *signature matching* algorithms using features of the signatures or signature timing. Our goal is to understand how imperfect detection mapping affects end-to-end accuracy.

Although our results are evaluated in the context of this particular application, the observation that matching affects sensor fusion accuracy applies to other applications where multiple sensors observe multiple targets.

Our work differs from most prior sensornet tracking and systems because they assume an unconstrained environment, sparse target identities with wide spatial separation, and dense sensor deployment [21, 14, 26, 20]. We instead assume a constrained, environment (a public roadway), with dense, poorly separated targets (rapidly moving cars), and sparse sensors. Unlike vehicle re-identification systems [5, 4, 15, 16, 2, 23, 13], we target an urban roadway with easily deployable sensors. More detailed related work is covered in Section 4.

The main contribution of this paper is the design of *signature matching algorithms* (Section 2). We propose several classes of algorithms, evaluating approaches to matching using different features: ordering, timing, target features, and raw target observations. Unsurprisingly, algorithms using little information (such as ordering), are easily misled by missing readings. Surprisingly, full signature comparisons are also easily mislead by overly specific details (Section 2.4). We conclude that a fairly simple *static time window* (STW) algorithm is the best choice, even over algorithms that are more complex or use more information (Section 3.2). We have integrated of signature matching into a multi-sensor vehicle classification system. While prior researchers have considered vehicle re-identification and vehicle-specific applications, to our knowledge we are the first to explore the specific effects of signature matching in a multi-sensor system.

The second contribution of our work is evaluating signature matching as part of a full system on an active roadway. We first evaluate matching by itself (Section 3.2), comparing five algorithms and confirming STW as best considering both correctness and simplicity, correctly matching 73% of the time. A complete system must classify vehicles into different categories (passenger car, SUV, truck, etc.), and so complete system performance depends on both signature matching and sensor fusion. Since those algorithms can have correlated errors, evaluation of a full system on real data is essential to confirm our algorithm choice (Section 3.4). We show that the cost of imperfect matching (with STW) on overall classification accuracy is only 7–11% compared to perfect (oracle) matching (Section 3.4), while a poor matching algorithm can reduce end-to-end classification accuracy by 21%. By testing signature matching in a real-world system, we explore the high degree of correlation between matching and classification errors in real-world data, and we find that the accuracy of end-to-end, multi-sensor classification accuracy with our algorithms is consistent with theoretical predictions of partial correlation (Section 3.4.5).

Finally, our third contribution is to generalize these results to multi-sensor tracking algorithms and quantify them in a real multi-sensor vehicle classification systems. We find that defining metrics to compare algorithms is surprisingly difficult (Section 3.4.3), because observations are not just right or wrong, but also duplicated or omitted. Although similar problems occur in pattern recognition, this analysis has been little explored in sensornets and we expect our metrics are useful to characterize other matching problems. Our numeric results are specific to our case study, but we show the importance of good matching algorithms through comparison of several algorithms against a perfect (oracle). This result suggests future multi-sensor tracking must consider error due to individual sensors, multi-sensor fusion, *and* detection-to-target mapping.

2 Matching Algorithms

We here first formalize our matching problems and then compare algorithms using sequential orders, timestamps, and raw signatures. All can operate online, and in experiments (Section 3.1) we do so, al-

though for simplicity our pseudocode describes post-facto analysis.

2.1 Problem Formalization

The goal of signature matching is to determine when observations at two sensors observe the same of different actual targets. Given two sensors, a *match* is when they observe the same target and a *non-match* as when only one sensor observes a target, perhaps because the target does not intersect the view of one of the sensors. We explore matching in the context of vehicle re-identification on a roadway, where non-matches indicate vehicles that park or turn between sensors.

Signature matching is trivial if ideal sensors generate two perfect event streams and all targets pass both sensors. However, a real world application it can be quite challenging: signatures are missed because vehicles straddle lanes; they can be merged by tailgating vehicles; there may be no matching signature if a vehicle turns between sensors; or signature timing may vary greatly if vehicles change speeds or park. A reliable, real-world matching algorithm must therefore detect matches and also report non-match signatures when no counterpart can be found. Vehicle classification algorithms can then build on it to do multi-sensor fusion [16]).

2.2 Numbering Based

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

2.2.1 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

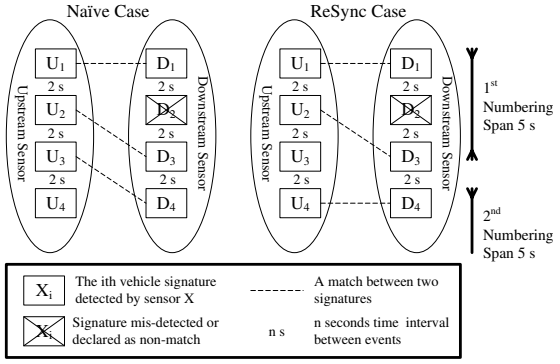

Figure 1: Avalanche problem in NN but solved by NwR.

matches (the left case in Figure 1). 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.

2.2.2 Numbering with Resynchronization (NwR)

NwR (pseudocode in Appendix A) 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 1. 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 fre-

quency depends on travel time between stations and how many 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 3.2 and find that it provides reasonable correctness (we define *recall* in Section 3.2.1; its recall is 64%, Table 3). 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.

2.3 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.

2.3.1 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 [12]. Depending on the window is set, this assumption holds for 90% of vehicles or more as described in Section 3.3.

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 1 *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** $tw_{lo} \leq sig_{down}.timestamp - sig_{up}.timestamp \leq tw_{hi}$ **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 1). 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 3.1).

However, similar to NN, an incorrect match can throw off future matches if alignment between signatures becomes skewed, as shown in Figure 2. 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 sig-

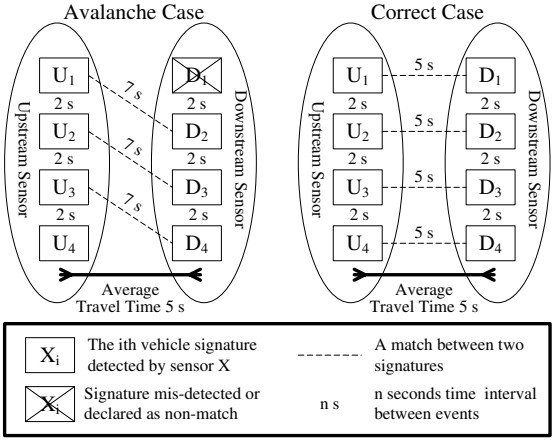

Figure 2: Avalanche problem in STW.

nature 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 3.3.

2.3.2 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 1). Since 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 2 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 3.3.

Table 3 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.

2.3.3 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 rematching. The details are in Algorithm 2.

Algorithm 2 *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}

repeat
  for sig_{up}^{i} **in** N_{up} **do**
    if sig_{up}^{i-1} is matched to sig_{down} **then**
      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**
        $M \leftarrow M \cup \{(sig_{up}^{i}, sig_{down})\}$ and $M \leftarrow M \backslash \{(sig_{up}^{i-1}, sig_{down})\}$
        $N_{up} \leftarrow N_{up} \cup \{sig_{up}^{i-1}\}$
      end if
    else if sig_{up}^{i+1} has a match **then**
      similar process as forgoing
    end if
  end for
until no new matching declared
for all signatures **in** N_{up} and N_{down} **do**
  if the differences of two signatures N_{up} and N_{down} fall in both wheelbase and time window **then**
    transfer these two signatures from N_{up} and N_{down} to M
  end if
end for
return M, N_{up} and N_{down}

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

2.3.4 Raw-Enhanced Time Window (RETW)

The better recall of WETW, compared to STW (Table 3), 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 3 shows four example signature pairs. Red and blue (darker) lines represent different signatures.

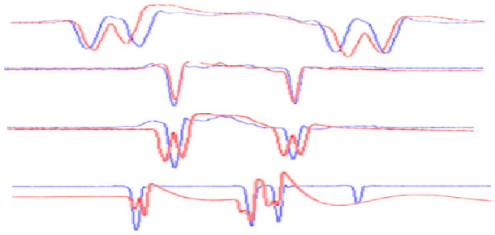

Figure 3: Four pre-scaled raw signature pattern comparisons. The horizontal axis is time and the vertical is energy.

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* [19] 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 3 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.

2.4 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. [2]. In Section 2.3.4 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 3 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 1 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 [2, 3], details in Section 4). We believe they succeed because their test set is smaller (seven vehicles), their sensor spacing closer (several meters

Table 1: 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

away), and their sensor provide informative (three-axis 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.

2.5 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 3.3 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 3 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 3.2.

3 Evaluation

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 describes the details of that field test, compare matching result among and within our algorithms, examine algorithm parameter sensitivities and finally the effect of matching over vehicle classification.

3.1 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. 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 4 shows real deployment—each station has a laptop, an IST-222 detector and two loop tapes. 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

Figure 4: Real deployment.

from upstream signatures. We used sensor calibration as described previously [16], and each station ran local single-sensor classification, while the master (the downstream node) performed on-line signature matching using STW (Section 2.3.1) 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.

3.2 Matching Algorithm Correctness

Although we evaluated STW on-line, to compare all of our signature matching algorithms (Section 2), 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.

3.2.1 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 2 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.

To evaluate our currects, we draw terms from information retrieval [24]. 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.

3.2.2 Observations

Table 3 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 2.3.2 and 2.3.3. 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.

3.3 Parameter Sensitivity

Each matching algorithm is controlled by several parameters. We have studied parameter selection and find that, in general, our time-stamp based algorithms are insensitive to internal parameters. We omit details here due to space constraints; full details are in Appendix B. STW yields reasonable recall over a wide range of time windows. If our parametes are off by 50% from optimal setting in STW, we lose only about 20%, and a 20% change of window only causes about 10% lower recall. DTW and WETW too are insensitive to exact values of sv and wheelbase window. We observe only a 10% change in recall when sv ranges from 1 to 10 s. While parameters are relatively easy to configure, additional autoconfiguration is an area of future work.

3.4 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

Table 2: Event types for matching

Types	**Expected Events per Occurrences**	**Occurences**	**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**

		Correct		Incorrect		Reported	
Algorithm	**Recall**	**matches**	**non-matches**	matches	non-matches	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 3: Matching correctness of algorithms

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.

3.4.1 Multi-sensor Classification Review

Park et al. previously showed that classification can benefit from multi-sensor fusion [16]. 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 misdetected 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.

3.4.2 Evaluation Baseline

Building on prior work [16], 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) [8]. 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) [16]. 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 al-

gorithms. We use quality-best fusion, the best choice from [16]. 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 4 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.

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

However, just as matching correctness is complicated by duplicate or undercounts (Section 3.2.1), 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 5.

3.4.4 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 6. Each algorithm uses optimal parameters (as defined in Table 3)). The resulting signatures use quality-best fusion [16] 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 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 3.4.5.

Second, as expected, accurate matching helps improve classification while poor matching hurts. We find WETW matches signatures most accurately

Table 4: Classification accuracy, multi vs. single

		Categories	
Classification	**Veh.**	**two**	**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%**)

Table 5: 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

Table 6: 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)	–

(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 3), 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 6 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.

3.4.5 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 7 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%.

Table 8: 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%)	

To understand what causes these correlations, we next reanalyze the STW case from Table 6: Table 8 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 at each sensor), so we over-count (the V_2^2 case from Section 3.4.3). In final case (no, no) both matching and classification fail. A more detailed dissection of Table 8 is in Appendix C.

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 5 can still result in correct classification when $C(i) = G(i)$. Should matching and classification are completely correlated,

Table 7: The correlation between matching and classification.

Algor.	Match. recall (m)	Class. accu. (c)	Correlation none $m \cdot c$	Correlation 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%

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 3.4.3), 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 3) results in more of these V_2^2 events (7% vs. others 1–3%, Table 6). 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.

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.

4 Related Work

Our work builds on prior work in target tracking in unconstrained and constrained environments.

4.1 Unconstrained Environments

Much early work in sensornets considered target tracking in unconstrained environments. Early work used dense networks of sensors tracking relatively sparse targets, [21, 18, 14]. Zhao et al. use information theoretic techniques for better vehicle path estimation [26]. Shin et al. consider overlapping targets and use information about distinct targets to clarify the status of targets near each other [20]. As with this prior work, we are concerned with confusion of observations about target near each other (the missegmentation in road traffic [16]); 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 2.

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 [15]. The technique was introduced by Baumberg to cope with the situation that naïve direct correlation coefficient comparison is not enough or even impractical [1]. In Section 2.4, 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.

4.2 Constrained Roadways

Our work builds on prior work in vehicle tracking or re-identification. Unlike most of them, we focus on the effects of signature matching on correctness, not

the application of re-identification for classification or speed estimation.

Coifman proposes a system for freeway deployment [5, 4]. 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 2.2.2), 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 [2, 3], 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 [10]. 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 [13]. 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 [22]. 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 3.3. As with Coifman [5], 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 [9]. Dailey [7] and Petty et al. [17] 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 2.4).

We have previously looked at sensor fusion to improve classification accuracy [16]. 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.

5 Conclusions

This paper explores multi-sensor target tracking, evaluating both signature matching algorithms in a multi-sensor vehicle classification system. Our results show that a simple static-time window algorithm (STW) is both efficient and the most accurate of a range of algorithms, including full raw signature comparison. We show that signature matching has significant effect on the end-to-end accuracy of a multi-sensor classification system. An important effect is the correlation between matching and classification; we quantify that the algorithms are partially correlated. Ultimately, we show that real-world matching algorithms can reach end-to-end classification accuracies within 90% of perfect matching when evaluated in real-world field tests.

Acknowledgments

Thanks Unkyu Park for his help on lab and field test. Thanks Min Zhu, Liang Han and Hsing-Hau Chen for their volunteering in 2009 field experiment. Thanks Andrew Goodney for his input on our raw signature

matching algorithm.

References

[1] Adam Baumberg. Reliable feature matching across widely separated views. In *Proceedings of 2000 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1 of *CVPR '00*, pages 774–781, Hilton Head, SC, USA, June 2000.

[2] 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 *Proceedings of the 84th Annual Meeting of the Transportation Research Board*, pages 173–181, Washington D.C, USA, January 2005. Transportation Research Board.

[3] Sing Yiu Cheung, Sinem C. Ergen, and Pravin Varaiya. Traffic surveillance with wireless magnetic sensors. In *Proceedings of the 12th World Congress on Intelligent Transport Systems*, ITS '05, pages 173–181, San Francisco, CA, USA, November 2005.

[4] 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, Newport Beach, California, April 1998. American Society of Civil Engineers.

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

[6] W. Steven Conner, John Heidemann, Lakshman Krishnamurthy, Xi Wang, and Mark Yarvis. *Wireless Sensor Networks: A Systems Perspective*, chapter Workplace Applications of Sensor Networks, pages 289–307. Artech House, Inc., 2005. Chapter 19, Nirupama Bulusu and Sanjay Jha, editors.

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

[8] Federal Highway Administration and Office of Highway Policy Information. FHWA vehicle types. `http://www.fhwa.dot.gov/policy/ohpi/vehclass.htm`, October 2003.

[9] Jaehoon Jeong, Shuo Guo, Tian He, and D. Du. APL: Autonomous passive localization for wireless sensors deployed in road networks. In *Proceedings of The 27th Conference on Computer Communications*, INFOCOM '08, pages 583–591, Phoenix, AZ, USA, April 2008.

[10] 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 *Proceedings of the 88th Annual Meeting of the Transportation Research Board*, Washington D.C, USA, January 2009. Transportation Research Board.

[11] Juan Liu, Jie Liu, James Reich, Patrick Cheung, and Feng Zhao. Distributed group management for track initiation and maintenance in target localization applications. In *Proceedings of the 2nd International Conference on Information Processing in Sensor Networks*, IPSN '03, pages 113–128, Palo Alto, California, USA, April 2003. Springer-Verlag.

[12] 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*, Ames, Iowa, USA, August 2005.

[13] Cheol Oh, Stephen G. Ritchie, and Shin-Ting Jeng. Anonymous vehicle reidentification using

heterogeneous detection systems. *IEEE Transactions on Intelligent Transportation System*, 8(3):460–469, September 2007.

[14] 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*, San Francisco, CA, USA, August 2005.

[15] P. V. Pahalawatta, D. Depalov, T. N. Pappas, and A. K. Katsaggelos. Detection, classification, and collaborative tracking of multiple targets using video sensors. In *Proceedings of the 2nd International Conference on Information Processing in Sensor Networks*, IPSN '03, pages 529–544, Palo Alto, CA, USA, April 2003. Springer-Verlag.

[16] Unkyu Park, Fabio Silva, Mengzhao Hu, 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.

[17] 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.

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

[19] 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.

[20] Jaewon Shin, Leonidas J. Guibas, and Feng Zhao. A distributed algorithm for managing multi-target identities in wireless ad-hoc sensor networks. In *Proceedings of the 2nd International Conference on Information Processing in Sensor Networks*, IPSN '03, pages 625–641, Palo Alto, CA, USA, April 2003. Springer-Verlag.

[21] Jaspreet Singh, Upamanyu Madhow, Rajesh Kumar, Subhash Suri, and Richard Cagley. Tracking multiple targets using binary proximity sensors. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks*, IPSN '07, pages 529–538, Cambridge, Massachusetts, USA, April 2007. ACM.

[22] Carlos C. Sun, Glenn S. Arr, Ravi P. Ramachandran, and Stephen G. Ritchie. Vehicle reidentification using multidetector fusion. *IEEE Transactions on Intelligent Transportation Systems*, 5(3), September 2004.

[23] Carlos C. 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.

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

[25] Wensheng Zhang, Guohong Cao, and Tom La Porta. Data dissemination with ring-based index for wireless sensor networks. *IEEE Transaction on Mobile Computing*, 6(7):832–847, July 2007.

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

A The Numbering with Resynchronization Algorithm

In Section 2.2.2 we introduced our Numbering with Resynchronization (NwR) algorithm and described it in text. Here we provide additional details and pseudocode for this algorithm.

Algorithm 3 shows psuedocode for NwR using post-facto analysis. The algorithm works as follows: We sort signature streams from upstream and

Algorithm 3 *Numbering with Resynchronization* algorithm

Input: 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#
Output: M (matching set), N_{up} and N_{down} (non-match set on upstream and downstream)

```
start_up ← Min(time-stamp in D_up) and start_down ← Min(time-stamp in D_down)
numbering_span ← 1 and index ← 0
for sig_up in D_up do
    if sig_up.timestamp ≤ start_up + numbering_span × resync then
        index ← index + 1
    else
        numbering_span ← numbering_span + 1
        index ← 1
    end if
    sig_up.serialNo ← index
    sig_up.ns ← numbering_span
end for
numbering signatures in D_down likewise
for sig_up in D_up and sig_down in D_down do
    if sig_up.serialNo = sig_down.serialNo and sig_up.ns = sig_down.ns then
        M ← M ∪ {(sig_up, sig_down)}
        D_up ← D_up \ {sig_up} and D_down ← D_down \ {sig_down}
    end if
end for
Put remaining signatures from either D_up or D_down into N_up or N_down correspondingly
return M, N_up and N_down
```

downstream in timestamp ascending order. Then for each stream, we segment it into multiple span groups based on the length of numbering span. After the segmentation, we assign span number and relative index within its span group for each signature. If two signatures from two streams have the same span number plus intra-span index, we declare a match. And for those cannot find such counterpart, we declare non-matches.

B Parameter Sensitivity

In Section 3.3, we briefly summerized how sensitive the accuracy of our algorithms are to settings of various parameters. Here we present a more detailed evaluation of this issue. 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 6 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 5, this time window accepts most vehicles, since the window center (δ = 29 s) is 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.

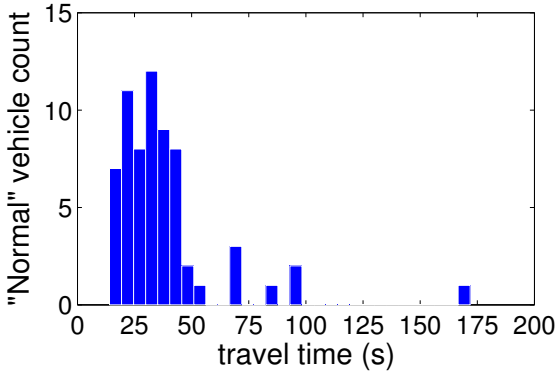

Figure 5: Travel time distribution of the 65 *Normal* vehicles.

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

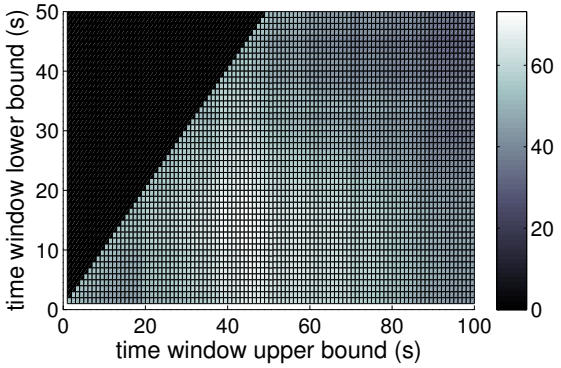

Figure 6: Accuracy % of STW.

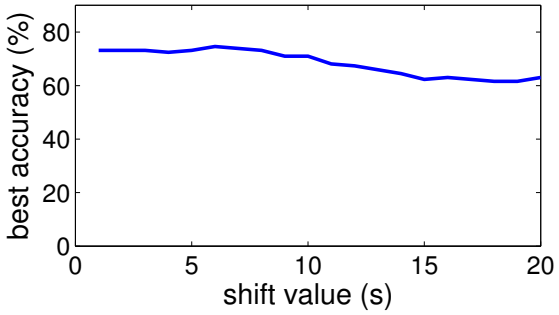

Figure 7: Best accuracy % of DTW with different *Shift Values*.

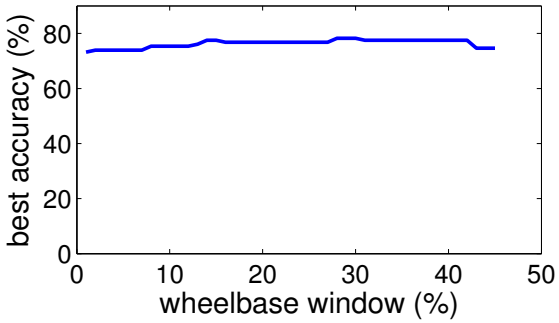

Figure 8: Best accuracy % of WETW with different *Wheelbase Windows*.

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, 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 7. 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 2.3.2 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 autoconfiguration as an open issue.

C Details about the relation between matching and classification

We summarized the correlation between our matching algorithm and classification in Section 3.4.5. That summary omitted the details of the many cases we considered.

Table 9 shows the full details of all cases we cons-

Table 9: Matching vs. classification in STW (dissection)

multi-sensor class. result	**ups. sig.**	**downs. sig.**	**matching correctness**	**reported veh. class.**	**correct in multi-sensor class.**	**# in STW 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)	strict; relaxed	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)	strict; relaxed	0	0
V_1^1	U_i	–	correct non-match	F(U_i)=G(i)	strict; relaxed	22	18
V_1^1	U_i	D_x	incorrect match	F(U_i, D_x)=G(i)	strict; relaxed	6	5
V_1^1	–	D_i	correct non-match	F(–, D_i)=G(i)	strict; relaxed	24	12
V_1^1	U_x	D_i	incorrect match	F(U_x, D_i)=G(i)	strict; relaxed	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)≠G(i)	none	3	5
V_2^1	U_i X	X D_i	incorrect	F(U_i,X)≠G(i) F(X,D_i)=G(i)	none	0	0
V_1^0	U_i	D_i	correct match	C(i)=F(U_i, D_i)≠G(i)	none	9	17
V_1^0	U_i U_x	X D_i	incorrect	C(i)=F(U_i,X)≠G(i) C(x)=F(U_x, D_i)	none	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)	none	0	0
V_1^0	U_i	–	correct non-match	F(U_i,–)≠G(i)	none	6	10
V_1^0	U_i	D_x	incorrect match	F(U_i, D_x)≠G(i)	none	1	2
V_1^0	–	D_i	correct non-match	F(–,D_i)≠G(i)	none	3	15
V_1^0	U_x	D_i	incorrect match	F(U_x, D_i)≠G(i)	none	0	2
V_2^0	U_i X	X D_i	incorrect	F(U_i,X)≠G(i) F(X,D_i)≠G(i)	none	1	3
V_0^0	–	–	–	–	none	10	10
total vehicles						**138**	**138**

diered, and how we classified them. This table supports our prior results and illustrates the complexity in evaluating correctness when faced with two sensors, each of which may omit or duplciate or incorrectly detect a vehicle.