On Map Matching of Wireless Positioning Data: A Selective Look-ahead Approach

Matt Weber[1,*], Ling Liu[1], Kipp Jones[2], Michael J. Covington[1], Lama Nachman[3], and Peter Pesti[1]

[1]Georgia Institute of Technology

[2]Skyhook Wireless

[3]Intel Corporation

ABSTRACT

Wireless Positioning Systems (WPS) are popular alternative localization methods, especially in dense urban areas where GPS has known limitations. Map-matching (MM) has been used as an approach to improve the accuracy of the estimated locations of WiFi Access Points (APs), and thus the accuracy of a wireless positioning system. Large-scale wireless positioning differs from satellite based positioning in at least two aspects: First, wireless positioning systems typically derive the location estimates based on war-driving access point (AP) data. Second, the locations of the AP beacons are not generally known at the same precision as that of the satellite locations. This results in lower accuracy and a lower confidence factor in the use of wireless positioning. This paper presents a fast selective look-ahead map-matching technique, called SLAMM. Existing MM algorithms developed for real-time location tracking of a moving vehicle are ill-suited for matching large collections of war-driving data due to the time complexity. Another unique feature of SLAMM is the map-matching of critical location samples in an AP trace to the road network before matching non-critical samples. Our experiments over a real dataset of 70 million AP samples show that SLAMM is accurate and significantly faster than the traditional MM approaches.

1. INTRODUCTION

Large-scale WPS typically determine the position of a WiFi enabled mobile client based on the geographical locations of its observed wireless APs. War-driving is a popular means of mapping observed WiFi APs to locations via GPS. For example, the WPS solution provided by Skyhook Wireless[1] delivers location estimates based on a user's surrounding wireless APs and cellular towers. Skyhook Wireless employs numerous drivers in different regions of the world to drive around and record the signal strengths of observed APs and cellular towers along with the corresponding GPS readings, a process known as war-driving. The result is roughly 1,400 hours of GPS data uploaded daily.

MM refers to a process of matching location samples to a road network. In this paper, we focus on matching very large datasets of trajectories with location samples collected at once per second. We argue that MM can assist in both the accuracy of the final location estimation and the management of the war-driving data collection process. However, to achieve these enhancements, both MM accuracy and speed are required. Thus, the unique challenge for matching large AP datasets is twofold: First, we need a high quality MM algorithm that can utilize available global knowledge to increase the accuracy of AP location data, while leveraging the speed of incremental matching algorithms.Second, the MM approach should make use of the road network topology to enhance the accuracy of incremental techniques, and must be resilient to both errors in the location measurement and error in the road network.

With these design objectives in mind, we develop a selective look-ahead MM approach, called SLAMM. SLAMM employs three filtering techniques that step by step identify and match critical location samples before matching the rest. In addition, the matched samples are given a road segment based quality estimate unlike any in current literature.

We compare the speed and accuracy of SLAMM to a traditional incremental MM approach from a real AP dataset of 70 million location samples, and show that the SLAMM three-level progressive filtering based MM approach is accurate and significantly faster than traditional approaches.

The remainder of the paper is structured as follows. In section 2 we discuss the unique challenges with matching commercial war-driving data. In section 3 we provide an overview of existing work and review two basic MM approaches. In section 4 we present SLAMM in detail. In section 5 we evaluate SLAMM by comparing it to two basic MM approaches. In section 6 we discuss related work, and finally in section 7 we give our conclusion and discussion for future work.

2. WAR-DRIVING CHALLENGES

The study conducted in this paper uses the war-driving dataset collected by a fleet of drivers for Skyhook Wireless, who systematically drive in tens of thousands of cities and towns worldwide to scan for 802.11 WiFi APs. The data

*To whom correspondence should be addressed. Email: mattweb@gatech.edu

is logged using proprietary scanning software from Skyhook Wireless. WiFi localization errors stem from several orthogonal sources, including the GPS measurement errors during the AP dataset collection via war-driving, the errors due to the subsequent radio propagation method, ranging from simple triangulation of signals [5, 9] to more complex hierarchical Bayesian sensor models [12], and the overall coverage of radio beacons. MM is a popular means for correcting GPS error and has been shown to improve WPS accuracy. [10] In addition, MM can also help to automate war-driver progress management.

With respect to MM, commercial war-driving poses three key challenges that are not uncommon among existing work, map error, GPS measurement error and the large volume of resulting data. Map error can exist as missing, or incorrectly digitized, roads in the network. Measurment error refers to incorrect position measurements and is often at its worst in poor weather and in parts of cities with densely placed buildings. The large volume of data, as mentioned in the introduction, is in the form of about 1400 hours of war-driving data collected daily.

These common problems, however, are greatly exacerbated when combined with the unique challenge introduced by fundamentally different driving patterns exhibited by war-drivers. Namely, to maximize WPS accuracy, drivers must attempt to traverse every road in an assigned area.

Understandably, map error may not be a priority for many existing MM algorithms because modern digital maps are largely accurate, particularly with arterial roads. However, when every road is followed, every map error is exposed. In addition, frequent u-turns are made and privately owned routes are followed (ie: parking lots, drive-throughs) that are not on the map. These occurances have an adverse effect on algorithms that rely on road network topology, as we'll show later.

A similar problem arises with measurement error. While not true everywhere, measurement error is routinely severe in certain parts of larger cities. Ironically, these areas are among those where WPS has the highest potential for use. Because a single drive will often collect data with both high and low measurement error, the unique challenge is to provide a way to identify and seperate the bad data from the good, even if they are not so far apart.

Finally the large volume of data requires all of these problems to be handled with a fast algorithm. Even though we do not have to match in real-time as the samples are collected, the traditionally more accurate global algorithms, discussed later, are either too slow, too reliant on network topology rules, or both.

3. MAP MATCHING

Given a trajectory T and a road network N known to constrain the travel of T, MM is the process of first identifying an arc $A \in N$ on which T is believed to have traveled at time t and then assigning a precise position along A to which $T(t)$ should be matched. A formal definition of MM can be found in [16, 8, 3].

One common way to classify the existing MM algorithms is as either incremental or global. Incremental algorithms attempt to determine the most likely arc, one sample at a time, by following the trajectory. Global algorithms find all of the possible paths that could have been taken by the trajectory and attempt to match the entire trajectory to one of the paths based on some similarity measure. When matching location samples to an arc, two types of information from the map data are generally used: (i) the geometric information based on the shape of the arcs, and (ii) the topology of road network such as how the arcs are connected. [3]

Algorithms that rely soley on geometric data are typically incremental. Many incremental algorithms also utilize the road network topology to constrain or weight the possible arcs based on connectivity to previously matched arcs. For example, [4] uses a recursive "look-ahead" which only considers those arcs that branch out n arcs from the previous match. One problem with this approach is the number possible paths growing rapidly, though on a smaller scale than with global algorithms discussed later. Another problem with this approach is the reliance on the previous match to choose the next candidate arcs. That is, if an arc is chosen incorrectly in the previous step, it is possible that the next set of candidate arcs will not contain the correct arc.

In the case of global algorithms, topology is required to determine all of the possible paths that a trajectory could follow. Different global algorithms vary the ways they make use of topology. For example, one approach [17] weights each arc in a road network based on distance to the trajectory, and then use Dijkstra's shortest path algorithm to determine the best path. Other approaches [2, 4, 6] attempt to find a path in the network, which best fits the entire trajectory. The general problem with these approaches are the high runtimes associated with the curve distance measures and the exponentially growing number of corresponding paths.

Next we describe two basic MM approaches, Distance MM and Look-Ahead MM.

3.1 Distance Map Matching

Distance Map Matching (DMM) is an incremental geometric only matching algorithm which simply matches each sample to the closest arc in the given road network. More specifically, for each sample S_i, the distance to each arc in the road network is measured. The arc that has the smallest distance to S_i is chosen as the selected arc and S_i is then "snapped" to that arc.

Figure 1 illustrates two common problems with this approach. First, as the trajectory crosses an intersection, the samples are closer to the road perpendicular to the direction of travel and are therefore matched incorrectly. We refer to this type of error as cross-track error. Second, as the trajectory comes across parallel roads, either measurement error or map error causes the samples to be closer to the incorrect arc for some period, causing incorrect matches. We call this type of error along-track error [10].

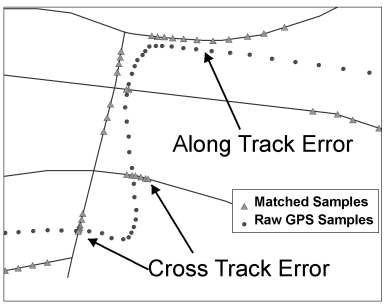

Figure 1: Simple MM problems

Using the topology of the road network is the most im-

mediate way to deal with these types of errors. By forcing the chosen arcs to form a valid path, arcs that would lead nowhere can be ruled out, even if they are the closest. Next we describe one such approach.

3.2 Look-Ahead Map Matching (LAMM)

When using the phrase "Look-Ahead" in terms of MM it is important to make the distinction between arc and sample look-ahead.

Arc look-ahead is a common technique in incremental MM algorithms that uses the road network topology to select future candidate paths by branching out n arcs from the arc that was previously selected for a match. As pointed out in [10], one problem with arc look-ahead is that the number of candidate paths can grow quite large depending on the value set for n.

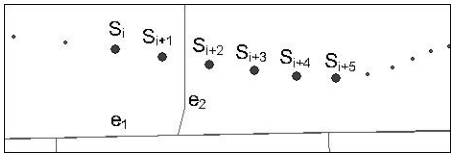

Figure 2: An example of LAMM parameters.

Sample look-ahead, on the other hand, is a less common technique that looks ahead k future samples, from the sample currently being matched, to make a more informed decision about which candidate arc to choose. Figure 2 shows an example of this technique where $k = 5$. In the figure, S_i is the current sample being matched, S_{i+1}, S_{i+2},...,S_{i+5} are the future samples, and all of the arcs in the image can be considered candidate arcs. From the image it is apparent that the samples are following the horizontal path labeled e_1, however S_i is actually closer to e_2. In this example, DMM would suffer from cross-track error by matching S_i to e_2. With sample look-ahead, however, one of many existing line distance measures could be used to correctly match S_i to e_1 using the future samples to define the curve.

The LAMM algorithm proposed in [10] uses both sample and arc look-ahead, matching each sample of a trace to a road network incrementally. Here we briefly describe this algorithm and conclude with an illustrated example.

For brevity, we begin assuming that the procedure has begun and that the previous sample has been matched to arc A_c. The next sample is matched by finding all possible paths that extend from A_c out the arc look-ahead value, matching S_i, S_{i+1},..., S_{i+k} to each path and taking the Hausdorff distance from the original samples to their matched counterparts. The chosen path is the one with the smallest Hausdorff distance and the new chosen arc A_c is selected as the arc in the chosen path that is closest to S_i, as illustrated in figure 3. In the top left image we see a road network with edges A,B,C,D,E,F and a set of ordered samples. A_c is the previously chosen arc, S_{i-3}, S_{i-2}, and S_{i-1} are matched at M_{i-3}, M_{i-2}, and M_{i-1} respectively. This example assumes an arc look-ahead value of two and a sample look-ahead value of six. The top right shows the first potential path, A,C,F. The samples S_i to S_{i+6} have been matched to the closest points along A,C,F illustrated by the points M_i to M_{i+6}. The next step is to take the Hausdorff distance between the original points in S and the corresponding matched point in M. The two bottom images show the same process for edges A,B,D and edges A,B,E. it is apparent from the images that the correct path is A,B,E and the Hausdorff distance would be the lowest for this path as well. After choosing this path, the next chosen arc is the one in the chosen path that is the closest to S_i, which in this case is B.

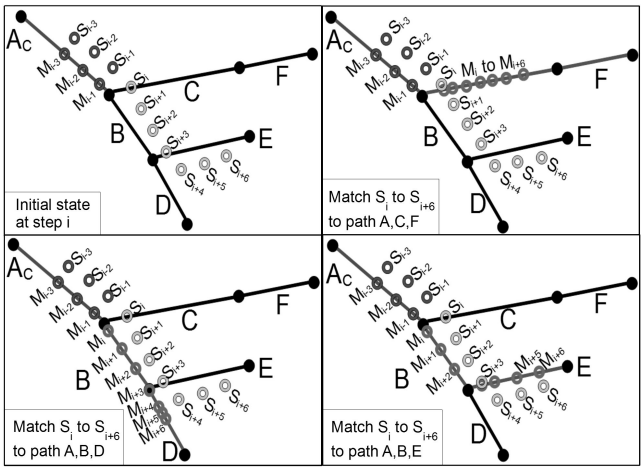

Figure 3: Simple LAMM

In theory, LAMM is an effective and fast way to deal with cross track and along track errors. However, when dealing with real data, sample look-ahead can be expensive and arc look-ahead is sensitive to errors in the road network.

With sample look-ahead, the look-ahead value must be set appropriately to be effective. Ideally the value would be high enough to see samples that would be matched to at least one arc ahead of the previously matched arc. However, the look-ahead value is also the number of times each sample will be considered plus one. For example, with a look-ahead value of three, each sample will be included in an evaluation four times, three times as one of the look-ahead samples and once as the current sample. This obviously has an impact on the runtime.

Using arc look-ahead makes the algorithm highly sensitive to errors in the road network. Simply having one arc missing in the road network can cause several samples to be matched incorrectly before the disconnect is identified and the candidate arcs are reset. This problem may be infrequent for many MM applications, however in the case of war-driving, the drivers are required to traverse every road that they find. This results in many missing roads being followed.

Next we present a MM algorithm that leverages the topology of the road network but is less sensitive to road network errors. Furthermore, our algorithm not only removes the redundancy of sample look-ahead, but can actually match many of the samples without even considering them.

4. SELECTIVE LOOK-AHEAD MM (SLAMM)

We describe the design of our selective look-ahead MM approach in this section, starting with the design principles and then presenting the three progressive filtering techniques.

4.1 Design Principles

Most existing MM algorithms, both incremental and global, are forced to check each sample in the trajectory. In the case of navigation applications, this is due to the real-time constraint, namely the samples must be matched as they arrive

from the GPS device to enable real-time routing algorithms. Another reason that existing MM algorithms process every sample in the trajectory is a low sampling rate that results in a sparse trajectory. In this situation every sample is needed to handle the increased sampling error associated with the low sampling rate. For example, a rate of one sample every 15 seconds might have 4 samples collected for a road that the driver spent a single minute on, where a rate of 1 sample per second would result in 60 samples for the same road. Clearly, neither of these constraints exist in the Skyhook Wireless dataset which is collected at a one second sampling rate.

Many of the fastest and most accurate MM algorithms that exist today have proven their results on datasets with a 15 - 30 second sampling rate. If we were to plug a once per second dataset into one of these algorithms and then a subset of the same dataset with only every 15th sample, one might think the subset would be processed faster, but with diminished accuracy. However, since more time has gone by between each sample, the corresponding sampling error region is larger which results in a larger road network subgraph to consider. So the challenge then becomes effectively choosing a subset of the original trajectory, to achieve higher performance, while preserving the advantage of a high sampling rate.

When we consider the constraints of driving in relation to the MM problem, one thing becomes obvious, that is we cannot change the road we are on without reaching an intersection. In the digital representation of the road network these are sometimes called transition nodes. Since each sample has a temporal component we can deduce that for a given sub-trajectory, if the first and last samples are matched to the same road, and there are no transition nodes within the subset of the road network bounded by the error region of the sub-trajectory (there are no intersections near the sub-trajectory), then all of the sub-trajectory must have been sampled from the same road (or the driver ran off the road). The SLAMM approach is based on the above analysis.

4.2 SLAMM Overview

Conceptually, SLAMM works by breaking a trace into segments of contiguous samples and then matching each segment to an arc in the road network. To accomplish this, each segment must have a one to one correspondence with an arc in the network, that is, each segment represents the time the driver spent on no more than one arc. To create these segments, samples are identified along the trace that will be used as break points, these samples are referred to as critical samples. Figure 4 illustrates this process.

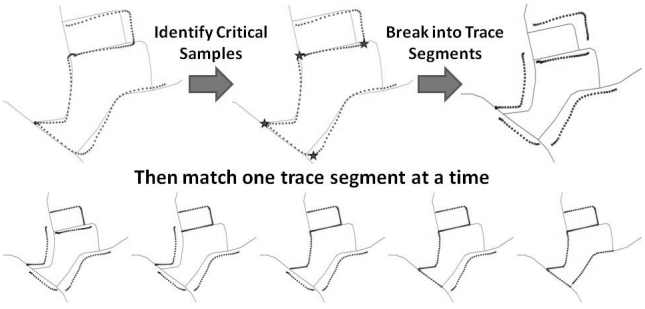

Figure 4: High Level Illustration of SLAMM

In practice, the process is divided into a preprocessing step and three phases, Trace Buffer (TB) based Arc Filtering, Critical Sample selection and matching, and trace segment verification. A detailed explanation for each phase follows.

4.3 Preprocessing Phase

Consider the Skyhook Wireless AP dataset, the trajectory data was collected at a static rate of one sample per second. However, gaps occurred during collection due to satellite signal loss or a driver taking a break. Thus, the SLAMM preprocessing step breaks the daily trajectories into subsets (traces) where these gaps do not occur. A trace with 1001 samples would have taken exactly 1000 seconds. We argue that removing the gaps is a natural way to segment the trajectories. The gaps that exist from satellite signal loss are often preceded by increasingly worse measurement error, so even if the gap was only for a second, it is often an indicator of high measurement error to come. The gaps that exist from drivers taking breaks are clearly good places to start and stop a trace as the driver may travel from one location to another during the gap which would have the same effect as missing roads on the algorithm.

4.4 Trace Buffer based Filtering

In SLAMM development, the first filtering technique consists of the trace buffer generation and the arc filtering. The goal of the trace buffer (TB) is to identify the nodes and arcs in the road network that are candidates for matching a trace. The key challenge is to construct a TB that can minimize the number of nodes and arcs that are the initial parameters to SLAMM without missing any that are relevant.

A TB is a polygonal region that bounds the error for a single trace. Ideally this region should be large enough to intersect all potentially relevant nodes and arcs for matching the trace but small enough to minimize the bounded error region.

Theoretically, the TB can be described as the Minkowski sum of a line interpolating the samples of the trace and a disc with a given radius. The Minkowski sum of two sets $S_1 \subset \mathbb{R}^2$ and $S_2 \subset \mathbb{R}^2$, denoted by $S_1 \oplus S_2$, is defined as

$$S_1 \oplus S_2 := \{p + q : p \in S_1, q \in S_2\}, \tag{4.1}$$

where $p + q$ denotes the vector sum of the vectors p and q.

Specifically for the TB case, we denote S_1 as a set of line segments and S_2 as a disc, and visualize the Minkowski sum as the union of copies of S_2 as its center moves along S_1. The road network database is then queried to retrieve only those arcs that intersect the TB. Figure 5 illustrates the overall process. The results of this query are then passed onto the next phase of SLAMM.

4.5 Critical Sample Selection and Matching

In general, the next phase will break the trace up into contiguous segments. The goal is for each segment to have a one to one correspondence with the arc to which it will ultimately be matched. By segmenting first and then matching we, are able to match the entire segment without considering all of the samples therein, for most cases. We start this process by identifying the critical samples, on which the trace will be segmented. Once we have defined the critical samples and segmented, we next match the beginning and end of each segment to an arc. This is done by first establishing a set of candidate arcs and then weeding them out through

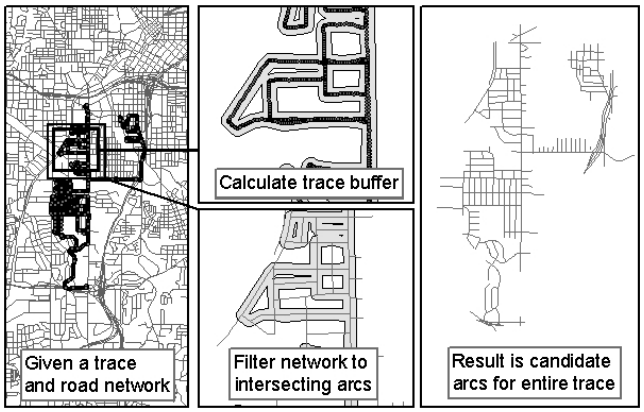

Figure 5: TB Filtering

a process later referred to as ellipse matching. Finally we pass the matched samples to the last phase.

Concretely, this phase consists of three steps, critical sample selection, candidate arc selection and finally matching the critical samples to the candidate arcs.

4.5.1 Critical Sample Selection

Critical samples are those samples along the trace that represent a transition from one road segment to another. Figure 6 illustrates the critical samples near the transition nodes. The intuition behind the finding and matching of

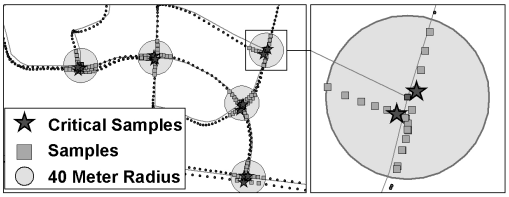

Figure 6: Find the samples closest to each transition node

critical samples first is the following. Given that each sample has a time stamp, we can deduce that for a given sub trajectory, if the first and the last samples are matched to the two end nodes (transition nodes) of the same road, and there are no transition nodes within the subset of the road network bounded by the error region of the sub trajectory, then all the sub trajectory must have been sampled from the same road. Ultimately the critical sample selection will have great impact on how many samples need to be considered to make a match, thus influencing the runtime performance of SLAMM. We develop a two stage process to identify the critical samples, first we identify the node passes for each transition node then we find the critical sample for each node pass.

The Node Pass. The first step is to define a set of node passes for each node. Simply put, a node pass represents a vehicle crossing an intersection once. A node can have multiple node passes because a drive can pass through a transition node multiple times during the war-driving process. Figure 10 shows two examples of the node passes of the transition node. The transition node in the left figure has multiple corresponding node passes while the node on the right figure has only one node pass.

We define a node pass as a contiguous sub trace S of a

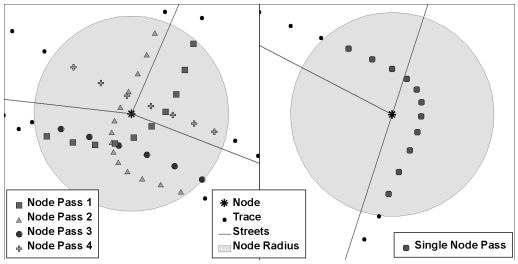

Figure 10: Node Pass Examples

trace T, which is within a radius r from the given node P. They are contiguous in that no two adjacent samples in a list of samples sorted on time, are more than the sampling rate ΔT apart.

Let the trace T be a set of location samples. S is said to be a contiguous sub trace of T if the following conditions are met:

1. $\forall\ S_i,\ S_{i+1} \in S,\ S_i,\ S_{i+1} \in T$
2. $\forall\ S_i, S_{i+1} \in S,\ S_{i+1}$ - $S_i = \Delta T$

Now we define a node pass as follows. Let P denote a transition node, r is the radius of node P, and S denote a contiguous sub trace of a trace T. We say that the subset S is a node pass of P with respect to radius r if and only if for any S_i in S, d(S_i,P) <= r.

The Critical Sample. Once the node passes have been defined, the next step is to assign a critical sample for each node pass. The critical sample for a given node pass is simply the sample that is closest to the corresponding node.

Let S be a node pass and P be the corresponding transition node. The critical sample S_c is defined as

$S_c = \min_{S_i \in S}(d(S_i, P))$

The Algorithm. In our implementation, the trace data structure stores an r-tree internally to allow for fast queries on the samples within that trace. When critical samples are to be determined for a given node, first the trace is queried for the samples that are within the given radius from the node. Then the returned samples are sorted on time to ensure they are in chronological order. This sorted list of samples is then broken up into separate contiguous lists or node passes as described above. Finally, for each node pass, the Euclidean distance from each sample to the given node is measured and the closest is stored in a temporary list of critical samples, specific to the given node, to be processed later. The pseudo-code is shown in algorithm 1.

Algorithm 1 findCriticalSamples(T,r,p)

Input: Trace T, radius r, Node p
Output: Critical Sample List

```
1: query T's RTree for samples S ∈ T where dist(S_i, p) <= r
2: sort S on time
3: break S into sublists of contiguous samples (split S where gaps in time exist)
4: for all contiguous sublist s do
5:     add sample in s closest to p to criticalSampleList
6: end for
7: return criticalSampleList
```

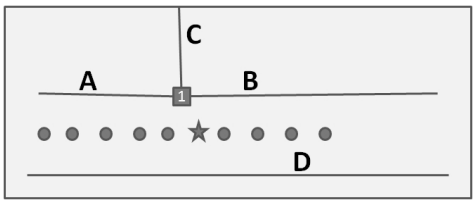

Figure 7: Samples followed A,B or D

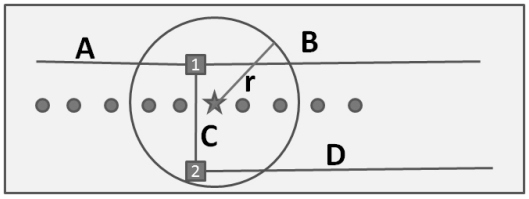

Figure 8: Example candidate arc selection

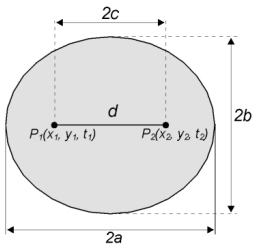

Figure 9: Error Ellipse

Once all of the critical samples have been identified SLAMM moves on to the matching step.

4.5.2 Candidate Arc Selection

The second step is to select a set of candidate arcs for the critical samples. As we'll see, each critical sample will have its own set of candidate arcs. The overall goal of candidate arc selection, at this stage, is to select all arcs that could have possibly been involved in the potential arc transition represented by the given critical sample.

One might think that simply using the arcs adjacent to the node that created the critical sample would be sufficient, however, there is no guarantee that the critical sample actually represents a transition on the node that created it. The node could have simply been near the arc from which the critical sample was recorded. Figure 7 shows an example of this. The trace segment could have followed arcs A and B via node 1 or it could have followed arc D. Since D is not adjacent to node 1 simply selecting the adjacent nodes could potentially leave out the correct arc. For this reason, we use a critical sample buffer based approach.

With critical sample buffer candidate arc selection we select those arcs that are within some radius r from the critical sample as candidates. This is illustrated in figure 8. In the example we see that arcs A,B,C and D are within r from the critical sample and are therefore the candidate arcs.

Clearly the choice for the value of r is important. Too small and it may not include the actual arc and of course an unreasonably large value would select too many. However, since the candidate arcs will be filtered later, too large is better than too small. In our implementation we use the value used for the TB width. This ensures that the arcs adjacent to the node that created the critical sample are included.

4.5.3 Ellipse Match

Ellipse matching is the final step of phase two. Its goal is to further filter the candidate arcs. Given a critical sample and the corresponding candidate arcs, the ellipse match will follow the trace in both directions, starting from the critical sample, weeding out candidate arcs as it goes along until, ideally, only one arc is left for each direction. Figure 11 illustrates these steps.

The error ellipse, shown in figure 9, is a common technique for bounding sampling error proposed by [14]. By sampling error, we are referring to the uncertainty of the location of a moving object in between position samples. Concretely, an ellipse is defined by two consecutive position samples, P_1 and P_2, in the time interval $[t_1, t_2]$ and a given maximum speed S_m that the object can travel. The eccentricity 2c is the Euclidean distance between P_1 and P_2. The length of the major axis 2a is calculated as $S_m(\Delta t)$, the maximum distance the object can travel, and the minor axis 2b is calculated as $\sqrt{(2a)^2 - (2c)^2}$. The resulting ellipse is a region that bounds all of the possible paths that could have been followed between P_1 and P_2. The advantage of this approach is that it guarantees a bounded sampling error. The disadvantage is that it does not take measurement error into account. Though this may not matter in the case of low sampling rates, datasets with higher sampling rates may risk not overtaking the measurement error. That is, with a high sampling rate, the boundary of the region created by the simple error ellipse may actually be closer to the samples than the measurement error estimate. To address this, [15] proposed the thickened error ellipse. This is a simple error ellipse and the Minkowski sum of the measurement error.

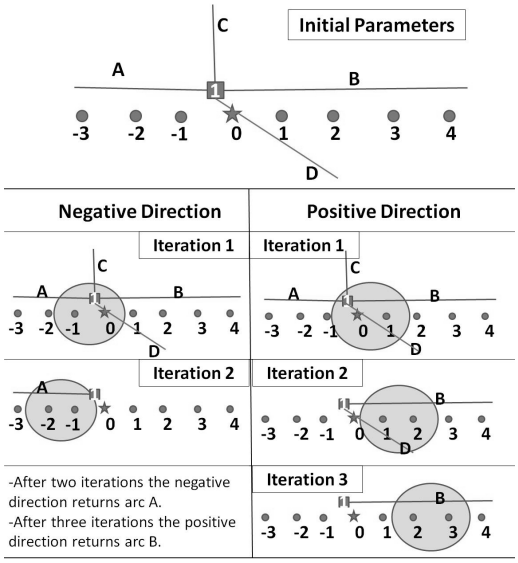

Figure 11: Ellipse Filter

Ellipse Match Algorithm. As mentioned, each of the critical samples will be a starting point for this step of the algorithm which proceeds as follows. For each direction along the trace, an error ellipse is generated from the critical sample S_t to the next sample in that direction. So for the positive direction, S_t and S_{t+1} are used as parameters to the error ellipse. Then each arc in the candidate arc list is checked to intersect the ellipse. If an arc does not intersect the ellipse, that arc is removed from the list. This proceeds with S_{t+1} and S_{t+2} ... At each iteration, for each direction, the following three cases are considered:

1. Only one arc is left in the candidate arc list. In this case the remaining arc is returned as the chosen arc.

2. More than one arc is left in the resulting ellipse. In this case a node collision check is made. A list of nodes is passed into the ellipse match which contains all of the nodes in the graph obtained from the TB phase. Each time this case is reached, the nodes are checked to see if any intersect the ellipse.

   - If there is a node collision: the distance from each sample considered so far to each arc remaining in the candidate arc list is summed and the arc with the shortest distance is returned as the selected arc.
   - If there is not a node collision: continue with ellipse for next iteration.

3. Zero arcs are left in the candidate arc list, restart direction with a thickened ellipse.

Figure 12 shows an example of the first two cases. On the left the ellipses were generated until another node was hit because two arcs intersected each ellipse along that portion of the trace. The right shows the best case where only one arc intersected the ellipses after only a few samples, in each direction.

In case three, the simple error ellipse does not intersect any arc resulting in no arc being chosen. This is often due to the ellipse simply not being large enough to bound all of the error introduced by the components of a given step. In this case the process is repeated from the beginning using a thickened ellipse instead of the simple ellipse. As discussed earlier, the simple error ellipse only bounds sampling error and does not take measurement error into account. By thickening the ellipse we are simply adding the measurement error into the bound.

Until now, the goal of the filters has been to ensure that no potential candidate arcs are missed. In the ellipse match filtering, our goal changes to resolve obvious matches quickly. Since much of the data is near the correct path, a tighter bound can effectively test this. At a high sampling rate, like ours, the sampling error, and thus the bound, tend to be small. Those instances where this test fails, due to high sources of error, (ie: measurement, map,...) the final phase is designed to catch by checking the consistency of the matching decisions made in this step.

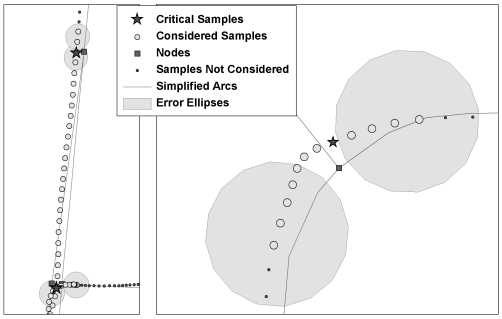

Figure 12: Best and worst case of the node pass step

Here we present the complete algorithm. For each critical sample:

We begin the ellipse based match. For each direction:

1. First we generate an ellipse using the critical sample and the next sample in the current direction. So for clarity we'll say that if S_i is the current sample being considered then i = 0 for the critical sample and we start by generating an error ellipse with S_i and S_{i+1} (or S_{i-1} for the negative direction).

2. Then we iterate over the candidate arc list checking whether each arc intersects the current error ellipse. If the arc does not then it is removed from the list.

3. Next we perform our case check, checking number of arcs in candidate arc list

   (a) Case 1: There is exactly one, that arc is returned as the chosen arc for this direction of the current critical sample.

   (b) Case 2: There are more than one, the distance of S_i to each remaining arc is added to a running sum for each arc.

   (c) Case 3: No arc remains, then the process is restarted with the thickened ellipse. In the infrequent case that no arcs remain and the ellipses have already been thickened, no arc is chosen and the choice is left to the final step explained later.

4. Then we check for transition nodes that may intersect the ellipse. In our implementation all of the nodes that remain, after the TB filtering step, are stored in an r-tree for fast query.

   (a) Since this is the first ellipse, any nodes that intersect are excluded from future checks for this direction of the current critical sample. This is because any node intersecting the first ellipse is likely the node from which the driver transitioned.

   (b) If this is not the first ellipse and a node is intersected that was not excluded by the first ellipse, then the arc with the lowest summed distance to the considered samples is returned as the chosen arc.

4.6 Trace Segment Verification

The final phase consists of analyzing the ellipse match results to make the trace segment assignments, finalizing the chosen arc and applying distance-based quality estimates to each trace segment match.

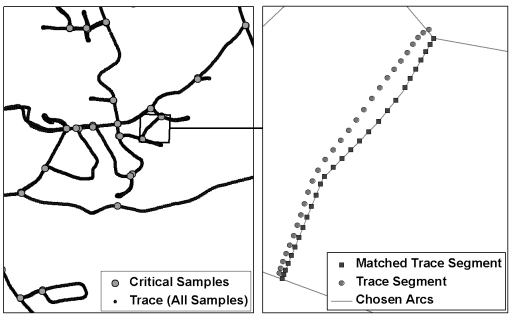

Figure 13: Example Trace Segment

4.6.1 Trace Segment

We define those samples that temporally fall between critical samples as trace segments. Ideally, a trace segment will be a contiguous set of samples that have been collected on

the same road. Intuitively a trace segment is all of the samples collected from the time the driver began traveling on a road to the time the driver left that road. For example, in Figure 13, the left image shows a portion of a trace with the corresponding critical samples, the right image shows a single trace segment and the corresponding matched samples.

4.6.2 Final Arc Selection

Up until this point we have discussed critical samples as independent transitions, not necessarily related to one another. Here it is important to recognize their relationship as endpoints to trace segments. That is, the samples in the positive direction of one critical sample would be the beginning of a trace segment. The samples in the negative direction of the next critical sample would be the end of the same trace segment. As we have discussed, each direction of each critical sample has been evaluated, which means we have an initial arc selection for the beginning and end of each trace segment. Our task now is to determine if these selections are consistent, and if they are not, to make a final decision as to which arc each trace segment will be matched.

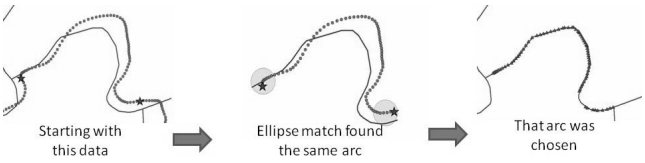

Figure 14: Ideal case in final arc selection

To accomplish this our algorithm iterates over the critical samples as follows. For each critical sample C_i, if the positive direction C_{i_pos} chooses the same arc as the negative direction of the next critical sample C_{i+1_neg} then we appoint this as the chosen arc A_c for the trace segment. Figure 14 illustrates this scenario. This is the ideal case and often means that we have made our match without considering all of the samples in the trace segment. In the less frequent case that C_{i_pos} and C_{i+1_neg} don't agree, we employ our TB concept but localized to the given trace segment. Specifically, we generate a TB for the trace segment and check for any arcs that are contained within. This will itself result in three simple cases

1. Only one arc is contained: In which case it is assigned as A_c
2. There are multiple arcs contained: In this case the trace segment is matched to each, and the Hausdorff distance is taken from the raw trace segment to each matched trace segment. A_c is then assigned as the arc that produced the smallest distance. This scenario is illustrated in figure 15.
3. There are no arcs contained: In this rare case we choose A_c from the arcs that were originally assigned to C_{i_pos} and C_{i+1_neg} in the same manner as in the previous case.

As may have been noticed, the first and last trace segments are only associated with one critical sample each. For the first, the arc chosen for C_{1_pos} is set as A_c, and similarly, C_{n_neg} for the last. In either case, if no arc was selected in the ellipse match phase, the localized TB is applied to the corresponding trace segment, as described above.

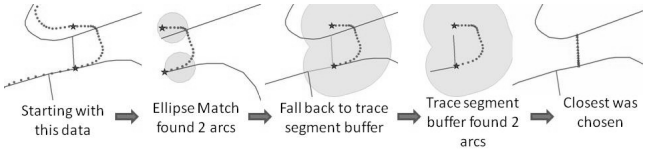

Figure 15: Trace segment buffer fallback

4.6.3 Quality Estimate

The final step in SLAMM is to physically snap the samples of each trace segment to its chosen arc. The Vertex Hausdorff distance of the original samples to the snapped samples is then measured and assigned to the trace segment.

The Vertex Hausdorff distance is a fairly straight forward version of the Hausdorff distance. In this implementation, the distance from each matched sample to every other original sample is measured and the minimum distance is stored, the Hausdorff distance is the maximum of these minimum distances. This distance is then the measure of the match. Once this is complete, a quality threshold can be applied to further evaluate the overall matches. This is explained in greater detail in the evaluation.

5. EVALUATION

This section reports the experimental evaluation of the SLAMM approach over a sample of data from the real world dataset. We conduct three sets of experiments. The first set compares the speed of SLAMM, LAMM, and DMM. The second set will compare the accuracy of the algorithms. Finally, we evaluate the effectiveness of our trace segment quality estimate. The system was implemented in Java using JTS topology suite version 1.10. All experiments were performed on a laptop with 2.4Ghz Intel Centrino Duo processor with 1GB of ram allocated to the JVM.

5.1 Running Time

To compare the speed of the three algorithms we measure the run time of each after the the road network has been queried. For DMM and LAMM, we also compare their performance with and without the benefits of the TB. In the case of no TB, the road network within 10 KM of the given trace was used. In the tests that did use the TB, a width of 60 M was used for all algorithms. SLAMM was designed and implemented specifically for use with the TB so its performance was not measured without it.

The dataset consists of $\approx$ 4.3 million samples broken into 1,147 traces that were collected between January 2006 and April 2008. The size of the traces range from 1,000 to 22,000 samples each. To select the data, seven driver id's were picked at random and their corresponding traces were the one's queried.

Figure 16 shows the results of the experiments where DMM and LAMM did not have the benefit of the TB. As expected SLAMM outperforms LAMM and DMM significantly. This test is mainly to show how effective the TB based filtering can be to any algorithm. This becomes clear with our next test, the results of which are shown in figure 17. Note that the SLAMM values do not change between the two graphs. Surprisingly, SLAMM actually outperformed DMM in four of the seven datasets. The dataset where

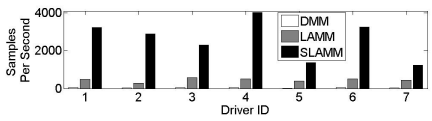

Figure 16: Speed without TB

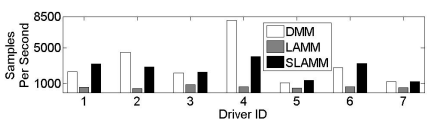

Figure 17: Speed with TB

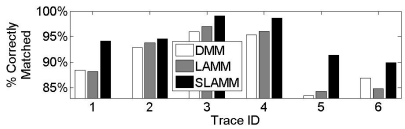

Figure 18: Accuracy

SLAMM did worse can be explained by the larger number of small traces. Overall, DMM was slightly faster with smaller traces but much slower than SLAMM with larger traces. As expected, SLAMM significantly outperformed LAMM on all traces.

5.2 Accuracy

As with most MM evaluations, accuracy is difficult to evaluate. This is because the only true way to measure the accuracy of a MM result is to know which road the driver was actually on.

To measure the accuracy, we hand-matched six traces picked at random from the ≈ 4.3 million samples used in our evaluation. A total of 34,543 samples were hand-matched, of those, 2,343 did not have a road in the road network. The samples without a corresponding road were left out of the accuracy comparison. Figure 18 shows the percent of the samples that were matched correctly for each trace with SLAMM, LAMM and DMM. SLAMM proves more accurate than LAMM and DMM for all six traces. In traces one and six, DMM actually has better results than LAMM. After inspecting the traces, this appears to be caused by LAMM's sensitivity to missing arcs.

5.3 Error Identification

As described in Section 4.6.3, the quality estimate of a trace segment, defines the reliability of the samples on that trace segment. Given a reliability threshold, the verification step identifies trace segments that exceed that threshold. The idea of this evaluation is to determine how well the verification phase identified the errors at different values for the threshold. Table 1 shows the results, at thresholds of 20, 40, 60 and 80 meters. Percent of error identified is the percentage of mis-matched samples that fell on a trace segment greater than or equal to the given threshold. Similarly, percent of missing road identified, is the percentage of the samples with no corresponding road in the graph. Percent redundant is the percentage of all of the samples that fell on a trace segment greater than or equal to the given threshold, regardless of the error classification. This would represent how many samples would have to be analyzed in a subsequent procedure. Ideally, a threshold would be selected that minimizes the percent redundant while maximizing the identified errors.

Application of Error Identification Feature.

We conclude the evaluation with a sample application that leverages the error identification features. At this point, we assume that SLAMM has finished processing the data and now all of the arcs, that had matching samples, have been linked to the respective trace segments. The trace segment to arc relationship has stored the aforementioned quality estimate. Our simple application, illustrated in figure 19, generates a grid, the boundary of which contains the data to be analyzed. Each grid cell is assigned the mean quality metric for all of the arcs that it intersects. As an example, if a cell intersects three arcs, and each of the three arcs had two trace segments matched, then the cell's value would be the average of the six quality estimate values. In figure 19, the grid and arcs are shown in the top right. The top left image zooms into a cell with a low value and, as one would expect, the samples appear to have low GPS measurement error. The lower images zoom into an area with much higher values and we see that high measurement error has driven the values up. In addition to measurement error, this system can also identify errors in the road network, such as missing arcs, and even occasions when a driver cuts through a parking lot or pulls into a drive-through for a snack. All of these sources of error are quite frequent with large war-driving datasets and can easily be identified with SLAMM's error identification features.

Metric \ Threshold (M)	20	40	60	80
% errors identified	94.38	65.29	52.81	42.11
% missing road identified	100	79.47	56	48.89
% redundant	43.23	19	11.61	9.13

Table 1: Error Identification Results

6. RELATED WORK

Ochieng, Quddus and Noland [13] provide a good overview of various MM algorithms, while Wenk, Salas, and Pfoser [15] provide an incremental MM algorithm that effectively trades accuracy for speed of computation. There are several examples of systems that use global or look-ahead knowledge [10, 11]. These approaches provide the basis on which our research is built with a focus on providing an efficient MM algorithm with high accuracy for large scale wireless positioning systems. In [7], Chawathe proposed a sample look-ahead approach with an incremental algorithm. The fundamental difference between Chawathe's approach and ours is that our trace segments are defined by the surrounding nodes in a global manner while Chawathe segments a trajectory incrementally using a predifined interval. Our SLAMM approach can be viewed as a careful hybrid of global MM and incremental MM. The unique feature of our approach is the three level selective filtering based curve-to-curve fitting algorithm, which improves the speed of MM with the selection of critical samples and the error-bounded trace buffers, while enhancing the accuracy and reliability estimation with selective look-ahead of both samples and matching arcs. By using a three-level progressive filtering, SLAMM is able to significantly speed up the execution time compared to LAMM [10] while maintaining accurate results.

7. CONCLUSION

In this paper we have presented SLAMM, a selective look-ahead MM approach, with three unique features. First, it employs the trace buffer filtering technique to remove those road network segments that are irrelevant to the given location datasets in terms of MM. Second, it promotes selective

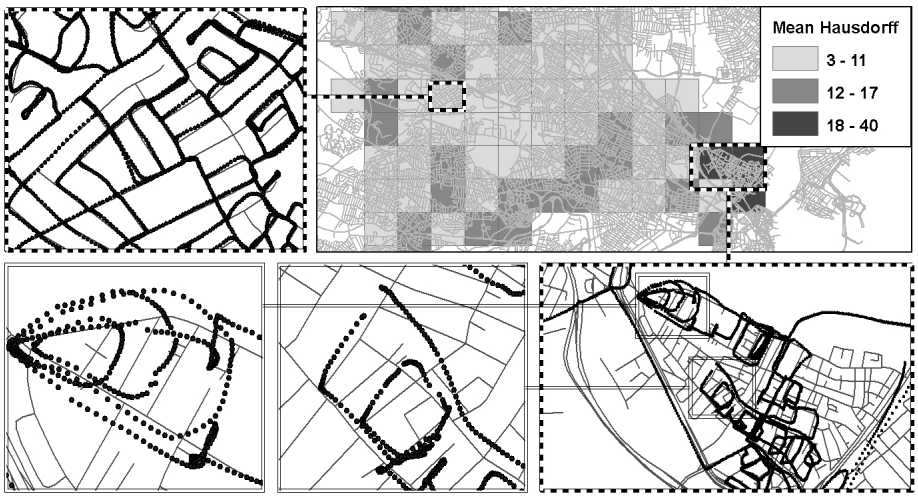

Figure 19: Sample application of error identification features

look-ahead of both samples and relevant arcs in conjunction with error bounding techniques to identify and match critical location samples to the road network before matching the rest. Third but not the last, it develops verification based matching technique that can match non-critical portions of the AP location datasets by identifying and isolating potentially bad segments. We compare the speed and accuracy of SLAMM to the existing look-ahead MM algorithm from a real dataset of 70 million location samples, and show that the SLAMM three-level progressive filtering based MM approach is accurate and significantly faster than traditional MM approaches.

8. ACKNOWLEDGMENTS

This work is partially supported by grants from NSF CISE NetSE program, CyberTrust program, and a grant from Intel research council. Our special thanks are due to Skyhook Wireless for their invaluable datasets and the discussions with Farshid Alizadeh-Shabdiz, the Chief Scientist for Skyhook Wireless.

9. REFERENCES

[1] Skyhook wireless. http://skyhookwireless.com, 2009.

[2] H. Alt, A. Efrat, G. Rote, and C. Wenk. Matching planar maps. *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 589–598, 2003.

[3] D. Bernstein and A. Kornhauser. An introduction to map matching for personal navigation assistants. *New Jersey TIDE Center*, 1996.

[4] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. *Proceedings of the 31st international conference on Very large data bases*, 2005.

[5] Simon Byers and Dave Kormann. 802.11b access point mapping. *Commun. ACM*, 46(5):41–46, 2003.

[6] H. Cao and O. Wolfson. Nonmaterialized motion information in transport networks. In *In Proceedings of Int. Conf. on Database Theory (ICDT).*, 2005.

[7] S. S. Chawathe. Segment-Based map matching. In *2007 IEEE Intelligent Vehicles Symposium*, pages 1190–1197, 2007.

[8] J. S. Greenfeld. Matching GPS observations to locations on a digital map. *81th Annual Meeting of the Transportation Research Board*, 2002.

[9] J. Hightower, A. LaMarca, and I. E. Smith. Practical lessons from place lab. *IEEE Pervasive Computing*, 5(3):32–39, 2006.

[10] Kipp Jones, Ling Liu, and Farshid Alizadeh-Shabdiz. Improving wireless positioning with look-ahead Map-Matching. In *MobiQuitous*, pages 1–8, 2007.

[11] W. Kim, G. I. Jee, and J. G. Lee. Efficient use of digital road map in various positioning for ITS. *Position Location and Navigation Symposium, IEEE 2000*, pages 170–176, 2000.

[12] J. Letchner, D. Fox, and A. LaMarca. Large-scale localization from wireless signal strength. In *Proceedings of the National Conference on Artificial Intelligence*, 2005.

[13] W. Y. Ochieng, M. A. Quddus, and R. B. Noland. Positioning algorithms for transport telematics applications. *Journal of Geospatial Engineering*, 6(2):10, 2004.

[14] Dieter Pfoser and Christian S. Jensen. Capturing the uncertainty of Moving-Object representations. In *Proceedings of the 6th International Symposium on Advances in Spatial Databases*, pages 111–132. Springer-Verlag, 1999.

[15] C. Wenk, R. Salas, and D. Pfoser. Addressing the need for Map-Matching speed: Localizing global Curve-Matching algorithms. *Proceedings of the 18th International Conference on Scientific and Statistical Database Management*, 2006.

[16] C. E. White, D. Bernstein, and A. L. Kornhauser. Some map matching algorithms for personal navigation assistants. *Transportation Research Part C: Emerging Technologies*, 8:91–108, 2000.

[17] H. Yin and O. Wolfson. A weight-based map matching method in moving objects databases. *Proceedings of Int. Conf. on Scientific and Statistical Database Management.*, 2004.