Probabilistic Vehicular Trace Reconstruction Based on RF-Visual Data Fusion

Saif Al-Kuwari[1,2] and Stephen D. Wolthusen[1,3]

[1] Information Security Group, Department of Mathematics, Royal Holloway, University of London, Egham Hill, Egham TW20 0EX, United Kingdom
[2] Information Technology Center, Department of Information and Research, Ministry of Foreign Affairs, P.O. Box 22711, Doha, Qatar
[3] Norwegian Information Security Laboratory, Gjøvik University College, P.O. Box 191, N-2802 Gjøvik, Norway

Abstract. Geolocation information is not only crucial in conventional crime investigation, but also increasingly important for digital forensics as it allows for the logical fusion of digital evidence that is often fragmented across disparate mobile assets. This, in turn, often requires the reconstruction of mobility patterns. However, real-time surveillance is often difficult and costly to conduct, especially in criminal scenarios where such process needs to take place clandestinely. In this paper, we consider a vehicular tracking scenario and we propose an offline post hoc vehicular trace reconstruction mechanism that can accurately reconstruct vehicular mobility traces of a target entity by fusing the corresponding available visual and radio-frequency surveillance data. The algorithm provides a probabilistic treatment to the problem of incomplete data by means of Bayesian inference. In particular, we realize that it is very likely that a reconstructed route of a target entity will contain gaps (due to missing trace data), so we try to probabilistically fill these gaps. This allows law enforcement agents to conduct off-line tracking while characterizing the quality of available evidence.

Keywords: Tracking, Trace, Bayesian, Scene Reconstruction, Vehicular, Fusion.

1 Introduction

Wireless mobile devices such as mobile phones and laptops can provide useful geolocation information. However, while such data enables novel ways of *tracking* mobile entities, collecting it from multiple sources and logically correlate it often creates new challenges for digital forensics.

Generally, tracking can either be online or offline. Online tracking involves observing the movement of a target entity in real time and is usually reactive (and adaptive) according to the target's behaviors. Offline tracking, on the other hand, entails the extraction of the target's movement traces from raw tracking data to be analyzed. However, offline tracking, in most cases, encounters a problematic issue that we call *tracking gaps*. These gaps represent missing tracking data over particular areas/periods where tracking of the target was not possible, usually due to tracking resources constrains. In this case, it is important to try filling these gaps by probabilistically guessing the routes

B. De Decker and I. Schaumüller-Bichl (Eds.): CMS 2010, LNCS 6109, pp. 16–27, 2010.

the target would have most likely taken between the end points of the gaps. For a gap, these end points are called the *Ingress*, which is the point at which tracking of the target was lost marking the beginning of a gap, and the *Egress*, which is when tracking later resumed marking the end of that gap. While such scenarios are certainly important to consider for tracking individuals, we instead discuss vehicular tracking since this is the most common means of transportation. Vehicular tracking data can be collected by explicitly tracking the target vehicle, or by observing data from the existing traffic infrastructure (e.g. CCTV) that is not originally meant to be involved in a tracking process. Either way, we assume that this data was collected passively since this is required by most law enforcement applications. Such data, and beside containing tracking gaps, will certainly exhibit different types of measurement errors, thus we develop a probabilistic method based on basic Bayesian inference to reconstruct the target trace, noting that this method can be adapted for an online mobility prediction too, which is a natural extension to an offline reconstruction, but this is beyond the scope of this paper.

This paper is organized as follows: in section 2 we briefly review some related work. In section 3 we discuss the feasibility of fusing RF and visual traces to minimize the number of tracking gaps in a target's trace. The trace reconstruction algorithm is proposed in section 4, followed by a discussion about its accuracy in section 5. In section 6 a few simulation results are presented, and finally, the paper concludes in section 7.

2 Related Work

Incomplete (or missing) data is a common problem in many applications. Expectation-Maximization and Data Augmentation methods (along with their improved variants) are among the most popular statistical treatments for the missing data problem [1]. Similarly, scene reconstruction (which usually deals with missing data) has been a highly active area of research over the past few decades, especially for forensics and crime investigation purposes. In general, most of the research in this area involves reconstructing scenes from images. For example, in [2] Calbi *et al.* proposed a set of computer-vision-based algorithms that are able to reconstruct a 3D scene of multiple moving objects. However, this system depends on pre-installed camera entities surveilling the area to be reconstructed. Moreover, in [3], Conaire *et al.* discussed how to fuse image-based and RF-based localization to improve the overall accuracy of the process. The authors tested their mechanism in a museum environment where visitors are equipped with devices containing a portable camera. The device can take photos for its current location and compare them with a database of images to identify its current location. This information is then fused with an estimation of signal strength from several wireless networks around the museum; again, signal strength histogram was create for various locations in the museum and stored in a database to be compared with the measurements.

Although most of these algorithms were proposed to track individuals, we are explicitly concerned with vehicular tracking. Examples of such works is presented in [4] by Brakatsoulas *et al.* who discuss the feasibility of reconstructing the movement patterns of objects by observing their GPS tracking information. Their algorithm adopts the so-called *map matching* technique where the tracking information of the objects (which are vehicles in this case) are analyzed and applied to a road-map. This paper adapts

a somewhat similar approach to map-matching but for the purpose of reconstructing a full tracking trace of a particular entity, not quite concerned about accurate localization since it suffices to know that a target vehicle has been in a particular roadway to draw conclusions about how its traces can be reconstructed.

3 Trace Fusion

Vehicular traces are records containing movement information of vehicles over a particular area and during a specific period of time. These traces can be collected by various tools, like GPS and Radar. However, data from other tools like CCTV (Closed Circuit Television), which are not originally tailored for tracking, can be used to improve the existing tracking traces. In this paper, we assume that we have two sets of tracking data, one collected by traditional tracking processes (see [5] for a survey on active tracking, and [9] for a work on passive vehicular tracking) and another retrieved from CCTV cameras that happen to be surveilling the tracking scene – we will call the first data *RF (Radio Frequency) tracks*, and the second *visual tracks*. The aim of this preparatory phase is to fuse RF and visual tracks of a target. Note, however, that this is only an auxiliary phase and is not necessarily required to proceed with the reconstruction algorithm, that is, we assume that both RF and visual tracks exhibit tracking gaps, but when fused, we hope that the number of these gaps is minimized.

Interpreting RF tracking data is straightforward, as a minimum, each record consists of time, vehicle ID and location, and we aim to simplify visual tracks (represented by images taken from CCTV cameras) to be consistent with the RF tracks which in turn will simplify the fusion process. Assuming prior knowledge of the fixed locations of the CCTV cameras and their somewhat narrow recording angular distance, we can estimate the location of a detected target to be the location of the detecting camera. One possible detecting method is by the plate number of the vehicles and detect the target once his plate number is observed, this can be done by capturing live images and compare them to a database of images containing the required plate number. Methods like SURF [6], perform such image-based detection by identifying interest points in the images. The main difference between RF and visual tracks, though, is that RF tracks represent continuous movements of the vehicle for a period of time (represented by a set of chronological tracking records), while the visual tracks represent fixed locations of the target where it was detected.

In this fusion process, we further aim to construct a trace of the target containing gaps only between intersections. However, real tracking information may not satisfy this requirement and can possible loose track of the target half way through roadways. In such cases, the available tracks for these roadways (with incomplete traces) may be: (a) visual only, (b) RF only, or (c) both RF and visual. In all these situations, we run a prediction algorithm to estimate the time it took the target to reach the next intersection and thereby filling the whole roadway. However, situation (a) provides very little information for this algorithm to work, so such roadways are ignored and marked as *semi-incomplete* which basically indicates that the target was observed in this roadway (this information may prove useful in the gap-filling algorithm; see section 4). In situations (b) and (c), and for simplify the description, we assume that there are only

single RF and/or single visual tracks per roadway, noting that the algorithm can easily be extended to consider multiple RF/visual tracks. The prediction algorithm is based on estimating the speed of the target over the period covered by the available tracking information which will then be used to predict the time it will take the target to reach the next intersection given the remaining distance of the corresponding roadway. In situation (c), if the visual track is within the RF track range, it provides no extra information (essentially becoming situation (b)) and the visual tracks can be ignored, but if the RF and visual tracks are apart, we extend the range of the RF to include the visual track. Usually, the longer the available tracking range, the better prediction we can expect. After connecting RF and visual tracks, the result is a tracking range and the predicted time to the next intersection can be calculated as follows:

$$t_i = \frac{(d(R)_1 - d(R)_0)(n - (d(R)_0 + d(R)_1))}{t(R)_1 - t(R)_0} \tag{1}$$

where: $d(R)_0$ and $d(R)_1$ denote the beginning and the end of the tracking range, respectively, and $t(R)_0$ and $t(R)_1$ denote the times at which the tracking range started and ended, respectively. If $d(R)_0 \neq d_{I,i}$ (where $d_{I,i}$ is the beginning of the roadway), then this means that the tracking range didn't start at the beginning of the roadway, so we need to do a *backward* time prediction (connecting the beginning of the range with the beginning of the roadway) as well as a *forward* time prediction (connecting the end of the range with the end of the roadway), which can easily be done in a similar manner. Furthermore, if $d(R)_1 = d_{E,i}$ (where $d_{E,i}$ is the end of the roadway), then only backward prediction is required. In any case, once we know the average speed of the target (which we can easily be calculate from $d(R)_0$, $d(R)_1$, $t(R)_0$ and $t(R)_1$), then given a distance, we can modify equation 1 to compute the time it would take the target to drive that distance, regardless of whether it is forward or backward.

4 Trace Reconstruction

Even after RF-visual fusion, the target's tracking records will still most likely include tracking gaps. In this section, we propose a 2-phase algorithm to probabilistically fill such gaps. To simplify the discussion, we assume that the underlying layout of the tracking scene resembles a Manhattan grid (see figure 1), in phase 1, the two end points (Ingress and Egress) of a gap along with all possible routes between them are identified. Then in phase 2, and based on the target's available tracking traces, the driving behaviors of the target is analyzed and used to fill the gaps by selecting the connecting routes that the target would most likely have taken through these gaps.

4.1 Phase 1: Routes Identification

In this phase, the end points, P_{I,G_i} and P_{E,G_i} (the Ingress and Egress, respectively), of a gap, G_i (where $i = 1, 2, .., n$ for tracking trace with n gaps), along with the possible routes between P_{I,G_i} and P_{E,G_i} are identified assuming that P_{I,G_i} and P_{E,G_i} correspond to intersections. However, it may be computationally expensive (or even infeasible) to identify all the possible routes between P_{I,G_i} and P_{E,G_i} when having a

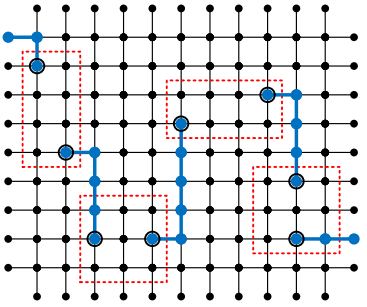

Fig. 1. Sample Scenario

large tracking area, regardless of the size of the gap. Thus, we restrict the area under consideration by setting boundaries around the tracking gap, this bounded area is called *search area* and we expect that this area will cover the most probable routes a target would take through the gap. Figure 1 visually illustrates a sample tracking trace of a target with four tracking gaps. Since we assumed that the tracking area is a Manhattan grid, the search area will be of a rectangular/square shape containing both P_{I,G_i} and P_{E,G_i}. We consider two situations for the locations of the P_{I,G_i} and P_{E,G_i} (we will refer to intersections as vertices and roadways as edges):

- Aligned Ingress/Egress: if P_{I,G_i} and P_{E,G_i} are either aligned horizontally or vertically, they form a straight line, but it is naive to assume that the target used that route to travel from P_{I,G_i} to P_{E,G_i}. Hence, we widen the search area to include the routes through the intersections above and the below (or at the right and left of) P_{I,G_i} and P_{E,G_i}. The search area is then bounded by the rectangle/square whose vertices are: $P_{I,G_i} - 1, P_{I,G_i} + 1, P_{E,G_i} + 1, P_{E,G_i} - 1$ where $+1$ and -1 imply the above and below (or right and left) intersections relative to P_{I,G_i} and P_{I,G_i}, respectively. Finally, all these vertices are marked.
- Diagonal Ingress/Egress: if P_{I,G_i} and P_{E,G_i} are neither aligned horizontally nor vertically, they form a diagonal line and the search area is bounded by the rectangle/square whose vertices are: $P_{I,G_i}, P_{I,G_i} + 1, P_{E,G_i}, P_{E,G_i} - 1$ or $P_{I,G_i} - 1, P_{I,G_i}, P_{E,G_i} + 1, P_{E,G_i}$, depending on the positions of the Ingress and Egress relative to each other. Finally, all these vertices are marked.

Once the four vertices bounding the search area are identified (and marked), they are connected. All the vertices that these connection lines pass through are then marked too indicating that they are part of the bounded search area; we call all marked vertices *search vertices*. We then propose an algorithm, called Bounded Route Counter (BRC), to find all the routes between P_{I,G_i} and P_{E,G_i} that are within the search area. The BRC algorithm is resembles a flooding (or broadcast) algorithm [7]. In the conventional flooding algorithms, the goal is to deliver a message to all nodes within a particular area by configuring nodes to forward every message it receives to all other nodes expect the node it received the message from. Similarly, BRC uses a message-like broadcast mechanism to discover the routes between P_{I,G_i} and P_{E,G_i}. When BRC

algorithm is first executed at P_{I,G_i}, it generates as many messages as there are exit points attached to P_{I,G_i}, each message represents a separate flow. These flows then duplicate or triplicate at every intersection they reach as long as it is a search vertex. This process continues until all flows are terminated. A flow terminates when it reaches: (1) a non-search vertex, (2) the P_{I,G_i} or (3) the P_{E,G_i}. When a flow terminates it raise a special tag with a value of 1, if the flow reached P_{E,G_i}, 0 otherwise. If the termination value was 1, the corresponding flow sends a message back to P_{I,G_i} reporting its traversed path. Algorithm 1 illustrates the BRC algorithm.

It is easy to see that the algorithm will both terminate and find all possible routes from P_{I,G_i} toP_{E,G_i} (that are within the search area). Initially, the algorithm is executed at P_{I,G_i} where it has four possible directions to send the flows through (though, following the flooding algorithm rules, it can't send a flow backward). At least one flow will hit a search vertex and will further propagate since at least one direction out the P_{I,G_i} leads to a search vertex. Also, since a flow can't terminate as long as it is propagating through search vertices and that all vertices will propagate a received flow out all their possible directions, it is guaranteed that all search vertices will be visited and only flows that terminate at the P_{E,G_i} will return to the P_{I,G_i}.

Algorithm 1. Bounded Route Counter Algorithm

```
P_{I,G_i} ← Ingress {identify Ingress}
P_{E,G_i} ← Egress {identify Egress}
Links ← array {which routes connect P_{I,G_i} and P_{E,G_i}}
exitPoints ← no. of exit points attacked to a vertex, usually 4
markSearchVertices(P_{I,G_i}, P_{E,G_i})
flood(P_{I,G_i}) {begin the flooding process}
label Loop
for i = 1 to i = exitPoints − 1 do
    if NonSearchVertexReached = True then
        continue
    else if IngressReached= True then
        continue
    else if EgressReached = true then
        addRoute(Links, i) {add route to the Links array}
        continue
    else if SearchVertexReached = True then
        Tag {tag to becomes a search vertex}
        flood(i)
        goto Loop
    end if
end for
```

We note that due to the visual-RF fusion process, we expect that the size of the search areas is minimized, and so adopting more sophisticated algorithms, such as branch-and-bound, will probably just slightly enhanced the efficiency of the whole route identification process at the cost of unnecessary overall complication.

4.2 Phase 2: Routes Analysis and Selection

At this stage, all routes between P_{I,G_i} and P_{E,G_i} are identified; we will denote the routes as R^i_j, where $i = 1, 2, ..., n$ and $j = 1, 2, ..., f(G_i)$, for the j^{th} route of the i^{th} gap. Also, each R^i_j consists of $f(R^i_j)$ roadways, where $f(R^i_j) = 1, 2, ...$ The function f can be thought of as an *overloading* function which behaves differently depending on its input, that is, f returns the number of routes through a gap when given that gap (e.g. $f(G_i)$), or the number of roadways forming a route when given that route (e.g. $f(R^i_j)$).

In this phase we analyze every route identified in phase 1 individually and estimate the time the target would most likely spend when traveling between P_{I,G_i} and P_{E,G_i} if he use each route, and compare it to the actual time difference between P_{I,G_i} and P_{E,G_i} as obtained from the original incomplete tracking traces. These analyses are based on a basic Bayesian inference by first studying the driving behaviors of the target and then assign probabilities for each possible route through the gap. By investigating the target's driving behavior, we essentially try to devise a mobility model to identify patterns in the target's movement, which is then used in the route selection algorithm. However, prior to the route selection algorithm, we first check whether any of the routes contains *semi-incomplete* roadways. Recall from section 3 that a semi-incomplete roadway is a roadway in which the target was observed at but couldn't be included in the RF-visual fusion process. If a semi-incomplete roadway falls between P_{I,G_i} and P_{E,G_i}, and since we know that the target was indeed in that roadway, we can safely ignore any route not passing through that roadway, which minimizes the number of routes under consideration. Once the possible routes are identified, the algorithm proceeds in 5 steps:

Step 1. After identifying the routes and the roadways each route is composed of, we search the available traces of the target for roadways with similar lengths as those in the routes between P_{I,G_i} and P_{E,G_i}. Then, we calculate the mean (average) of the times the target spent driving those roadways and assign the result to the corresponding roadways of the gap's routes. In this step, we aim to find the mean and variance of the target's driving time for each route through the gap. Suppose we have a route R^i_j that consists of $f(R^i_j)$ roadways, then its mean is:

$$\mu_{R^i_j} = \frac{1}{f(R^i_j)} \sum_{x=1}^{f(R^i_j)} \left(\frac{1}{S(x)} \sum_{y=1}^{S(x)} t_{E,y} - t_{I,y} \right) + \omega_x \tag{2}$$

where $S(x)$ is the number of roadways from the available target's traces with the same length as roadway x, ($x = 1, 2, ..., f(R^i_j)$), and ω is the delay factor which may be different for different roadway, see section 5 for a discussion about how to model this parameter. Since it is easy to extract information about roadways with available tracking information, it is possible to find when the target entered and existed each one of these roadways ($t_{I,y}, t_{E,y}$, respectively, for roadway y). Next, we calculate the corresponding variance of every route as follows:

$$\sigma^2_{R^i_j} = \frac{1}{f(R^i_j)} \sum_{x=1}^{f(R^i_j)} \left(\left(\frac{1}{S(x)} \sum_{y=1}^{S(x)} t_{E,y} - t_{I,y} \right) + \omega_x - \mu_{R^i_j} \right)^2 \tag{3}$$

We then use Bayes' theorem to calculate the probabilities that the target took each route between the Ingress and the Egress of a gap, and select the route with the highest probability as the connecting route. We base our probability calculations on the time difference between the Ingress and the Egress of the gap $t_{G_i} = t_{E,G_i} - t_{I,G_i}$ as obtained from the original trace, that is, for every route, we calculate the probability that the target took that route given the time we obtain by observing the times at P_{I,G_i} and P_{E,G_i}:

$$P(R_j^i|t_{G_i}) = \frac{P(t_{G_i}|R_j^i)P(R_j^i)}{P(t_{G_i})} \tag{4}$$

where: $P(t_{G_i}|R_j^i)$ is the conditional probability that given the target took the route R_j^i, he spent t_{G_i} driving it; which is calculated for every route (with a fixed t_{G_i}). $P(R_j^i)$ is the prior probability that a target will take the route R_j^i, and $P(t_{G_i})$ is the marginal probability. Steps 2 to 4 below show how these probabilities are calculated.

Step 2. Conditional probabilities: the probability that the target spent the time t_{G_i} while driving from $P_{I,i}$ to $P_{E,i}$ through gap G_i given that he took route R_j^i is:

$$P(t_{G_i}|R_j^i) = \frac{1}{\sqrt{2\pi\sigma_{R_j^i}^2}} \exp\left[-\frac{(t_{G_i} - \mu_{R_j^i})^2}{2\sigma_{R_j^i}^2}\right] \tag{5}$$

This probability is calculated using Gaussian probability density function assuming that the underlying process is Gaussian [8], that is, the driving behavior of the target will most likely follow a Gaussian distribution or can be estimated as Gaussian. In fact, it is easy to see that this process is Gaussian because typical driving behavior is to speed at the middle of a roadway and slow down at the beginning/end of that roadway, while driving with an average speed elsewhere.

Step 3. Prior probability: since we don't have enough information to model the target's preferences, the probability that the target selects a particular route (prior probability) through a gap is uniform for all available routes, and can be calculated as follows:

$$P(R_j^i) = \frac{1}{\sum_{x=1}^{f(G_i)}} \tag{6}$$

where $f(G_i)$ is the total number of routes through the gap G_i. Although we assumed that taking any route is equally likely, in practice and in some situations, some routes are more likely to be taken by the target than others. This may be due to traffic flow condition for example, see section 5 for a discussion about such factors. Another way to model this can be done by adopting a mobility prediction algorithm, such as that presented in [9] which predicts the direction a target vehicle would most likely take out an intersection by observing its current lane as it is approaching that intersection. However, carrying out such algorithms is difficult for offline tracking.

Step 4. Marginal probability: the marginal probability $P(t_{G_i})$ is the sum of the conditional (step 2) and prior (step 3) probabilities of all the routes and is calculated as

follows (the marginal probability acts as a normalizing constant in the sense that it makes sure that all the Bayesian probabilities of the routes will sum up to 1):

$$P(t_{G_i}) = \sum_{j=1}^{f(G_i)} P(t_{G_i}|R_j^i)P(R_j^i) \tag{7}$$

Step 5. Finally, and using equation 4, we can now calculate the Bayesian probabilities for each route and select the route with the highest probability as the most likely route the target would have taken through the corresponding gap.

5 Estimation Accuracy

The accuracy of our reconstruction algorithm is influenced by a number of factors, mainly concerning the accuracy of the tracking data collection. Beside the conventional RF measurement errors, it's very likely that the tracking traces are collected by several entities, so unless these entities are tightly synchronized, there will be timing errors among the recorded traces. Traffic-wise, the traffic delay factor ω (which models the various delays sources) influences the accuracy of the algorithm, and is explicitly used in equations 2 and 3. Examples of factors influencing ω are:

Roadways lengths: Routes are most likely composed of several roadways that are usually of different lengths. However, while the accumulation of the lengths of the roadways that form a route is representative to the distance between the Ingress and the Egress through that route, the speed of the movement through this route is affected by the acceleration and deceleration during the journey and around the intersections connecting the route's roadways. Hence, the speed of the target should be estimated for the individual roadway as oppose to the speed over the whole route.

Roadways speed limits: Every roadway restricts the speed of vehicles to a specific speed limit threshold. Knowledge of these limits is useful for estimating the maximum time threshold during which a vehicle can pass the corresponding roadway.

Traffic management type: Traffic delay highly depends on the traffic management type. For example, it is very likely that an intersection managed by stop signs will experience longer delays than another managed by traffic lights. However, care should be taken when considering traffic lights because the delay due to traffic lights depends on the state of the traffic light upon arrival (i.e. vehicles reaching the intersection while the traffic light indicates green will most likely experience much less delay than otherwise).

Points of Interest (POI): Another very effective traffic factor is the existence of points of interests. These points represents locations that are frequently visited by people, like banks and grocery stores, and hence represent locations of common interest among people. Intuitively, the existence of such points along a roadway will very likely increase the traffic density at that roadway and, consequently, the traffic delay. We note that information about POI can be extracted from a few specialized maps (like Google maps) and integrated into the weighting assignment of this phase.

Traffic density and flow: If available, knowledge of vehicular density (number of vehicles per km) and flow (number of vehicles crossing a point per hour) is very useful. Such information can either be statistically estimated from real traces, or probabilistically inferred based on location and time. For example, a particular area may be increasingly crowded during a particular period of time of the day, like an intersection leading to offices which may be crowded only at early morning and late afternoon.

Clearly, obtaining the information above to model ω is extremely difficult, so in section 6, we propose a method to model ω without having to access extra information about the traffic, that is, we estimate the traffic flow of the concerned roadways by referring to the original tracking traces (before extracting the target's trace from it). However, we also note that beside the explicit modeling of ω, we also implicitly accounted for ω (at least partially) when we calculated the average driving time in step 1 which already includes implicit traffic delays.

6 Simulation and Validation

Due to the difficulty of obtaining real vehicular traces to validate our probabilistic trace reconstruction algorithm, we used a vehicular mobility simulator to generate artificial vehicular traces for different scenarios. In this simulation, we used VanetMobiSim simulator [10] to generate vehicular mobility traces based on IDM-LC (IDM with Lane Changes) mobility model [11] which is an extensions to the IDM (Intelligent Driver Motion) mobility model [12]. These traces are then fed into NS-2 simulator [13] to generate the movement of the entities (i.e. vehicles) and a trace database. Arbitrarily appointing one of the simulated entities as a target, we manually (and randomly) created gaps in that target's mobility traces after extracting it from the original trace database. We then executed our trace reconstruction algorithm to probabilistically choose the most likely routes to connect those gaps and compare the selected routes with the routes the target actually took. Since it is difficult to model the delay factor ω without having access to real traffic traces, we modeled ω by observing the node density on the roadways that the gaps' routes are composed of. In particular, we referred to the original mobility traces for all the simulated nodes (before extracting the target's trace) and looked at traces taken for any node that happen to be passing through any of the roadways that are part of the gap's routes, then estimated the average node density of these roadways to assign values for their corresponding ω (clearly, the higher the node density, the larger the value of ω). This is easy to do in practice too since the traces from which the target trace was extracted usually contain other information about other entities that we can use to model the delay factors, which then can be used in equations 2 and 3. The delay factor ω is basically a time delay assigned to individual roadway and is different for different roadways. Figure 2 illustrates our simulation results for scenarios with different node density over a 1000 m^2 area consisting of roadways with different lengths –every simulation was run for 100 seconds. In each scenario, we manually created a gap (with 4 possible routes between the Ingress

and the Egress) and run the algorithm to select the most probable route. The figure shows the probabilities of each of the four possible routes at the manually created gap in each scenario. The results shown in figure 2 indicates that our reconstruction algorithm along with our method of modeling ω by averaging the traffic flow of roadways works very well in scenarios with lighter node densities where the probabilities of the routes vary drastically and it is easy to see which route is the most probable, but as node density increases, the probabilities become closer. Although we argue that this way of modeling ω is, in most cases, efficient since it gives a good estimation of other traffic delay factors affecting the roadways without actually having to model them individually, further modeling may be required in the more cluttered scenarios. However, there are always some routes that can be easily ruled out (like route 4 in all scenarios) usually because they introduce much longer delay compared to the time the target trace was missing over a gap.

We believe that if these algorithms were applied to real traces, the results will be more accurate since most of the simulation-based mobility models (which we used) don't always maintain a tightly consistent driving behavior for every entity, so modeling the driving behavior of the target based on his history trace is slightly less accurate in this case. In real life scenarios, on the other hand, every driver has a unique driving behavior, in fact, recent work [14] even showed that the driving behavior of individuals would make a reasonable biometric measure.

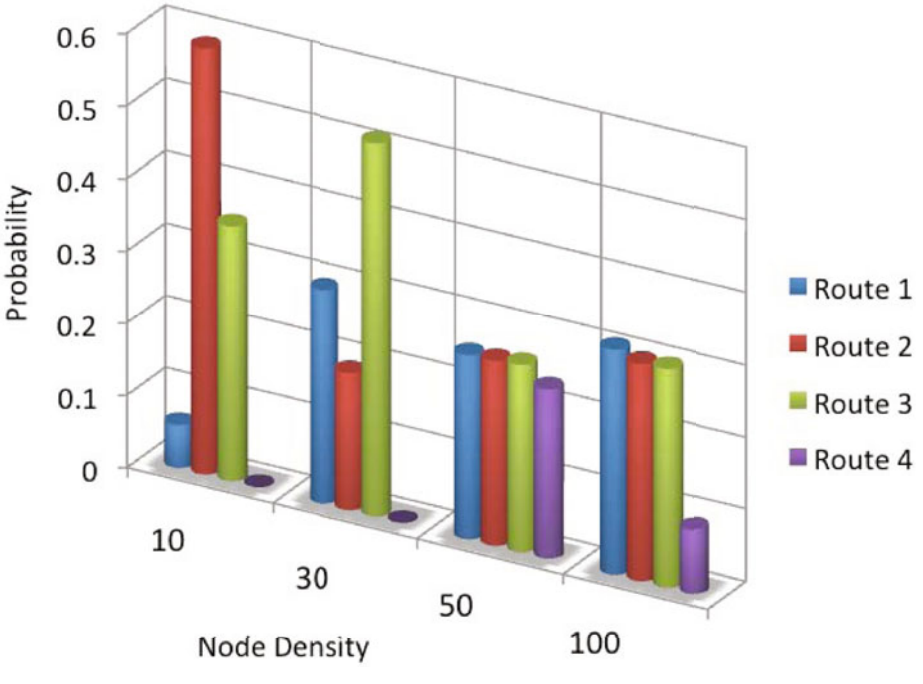

Fig. 2. Simulation Results

7 Conclusion

In this paper, we presented an algorithm for the offline reconstruction of vehicular traces of a target entity from fused RF-visual traces of that entity, but we assume that these traces exhibit occasional missing data, usually due to lack of surveillance resources. Our algorithm reconstructs the target's mobility traces by first identifying the locations of missing data and treats them as gaps. All the possible routes connecting these gaps are then identified. Based on a basic Bayesian inference approach and the driving behaviors of the target (obtained from the available traces), these routes are analyzed and the most probable one is selected. This process is repeated for all the gaps until the full (probabilistic) route of the target is reconstructed. We have validated the algorithm using simulations. Future work will, however, include validation using real vehicular traces, which allows capturing driving behavior more accurately.

References

1. Tan, M., Tian, G.L., Ng, K.W.: Bayesian Missing Data Problems: EM, Data Agumentation and Noniterative Computation. CRC Press, Boca Raton (2009)
2. Calbi, A., Marcenaro, L., Regazzoni, C.: Dynamic Scene Reconstruction for 3D Virtual Guidance. In: Gabrys, B., Howlett, R.J., Jain, L.C. (eds.) KES 2006. LNCS (LNAI), vol. 4252, pp. 179–186. Springer, Heidelberg (2006)
3. Conaire, C.O., Fogarty, K., Brennan, C., O'Connor, N.E.: User Localisation using Visual Sensing and RF Signal Strength. In: ImageSese (2008)
4. Barakatsoulas, S., Pfoser, D., Salas, R., Wenk, C.: On Map-Matching Vehicle Tracking Data. In: VLDB '05: Proceedings of the 31st international conference on Very large data bases, VLDB Endowment , pp. 853–864 (2005)
5. Al-Kuwari, S., Wolthusen, S.: A Survey of Forensic Localization and Tracking Mechanisms in Short-Range and Cellular Networks. In: Goel, S. (ed.) 1st International Conference on Digital Forensics & Cyber crime (ICDF2C), vol. 31, pp. 19–32 (2009)
6. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: Speeded up robust features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006)
7. Tanenbaum, A.: Computer Networks. Pearson Education Ltd., London (2003)
8. Miyajima, C., Nishiwaki, Y., Ozawa, K., Wakita, T., Itou, K., Takeda, K., Itakura, F.: Driver Modeling Based on Driving Behavior and Its Evaluation in Driver Idenfication. Proeedings of the IEEE 95(2), 427–437 (2007)
9. Al-Kuwari, S., Wolthusen, S.: Forensics Tracking and Mobility Prediction in Vehicular Networks. In: Sixth Annual IFIP WG11.9 International Conference on Digital Forensics (2010)
10. Harri, J., Fiore, M., Fethi, F., Bonnet, C.: VanetMobiSim: Generating Realistic Mobility Patterns for VANETs. In: Proc. of the 3rd ACM International Workshop on Vehicular Ad Hoc Networks (VANET'06), Los Angeles, USA (2006)
11. Fiore, M., Harri, J., Filali, F., Bonnet, C.: Vehicular Mobility Simulation for VANETs. In: ANSS '07: Proceedings of the 40th Annual Simulation Symposium, pp. 301–309. IEEE Computer Society, Los Alamitos (2007)
12. Treiber, M., Hennecke, A., Helbing, D.: Congested Traffic States in Empirical Observations and Microscopic Simulations. Physical Review E 62(2), 1805–18024 (2000)
13. Information Sciences Institute (NS-2), `http://www.isi.edu/nsnam/ns`
14. Wahab, A., Quek, C., Tan, C.K., Takeda, K.: Driving Profile Modeling and Recognition Based on Soft Computing Approach. IEEE Transactions on Neural Networks 20(4), 563–582 (2009)