Data-Driven Vehicle Trajectory Prediction

Philip Pecher
Georgia Institute of Technology
765 Ferst Drive, NW
Atlanta, GA 30332, USA
philip161@gmail.com

Michael Hunter
Georgia Institute of Technology
Mason Building, 790 Atlantic Drive
Atlanta, GA 30332, USA
michael.hunter@ce.gatech.edu

Richard Fujimoto
Georgia Institute of Technology
266 Ferst Drive
Atlanta, GA 30332, USA
fujimoto@cc.gatech.edu

ABSTRACT

Vehicle trajectory or route prediction is useful in online, data-driven transportation simulation to predict future traffic patterns and congestion, among other uses. The various approaches to route prediction have varying degrees of data required to predict future vehicle trajectories. Three approaches to vehicle trajectory prediction, along with extensions, are examined to assess their accuracy on an urban road network. These include an approach based on the intuition that drivers attempt to reduce their travel time, an approach based on neural networks, and an approach based on Markov models. The T-Drive trajectory data set consisting of GPS trajectories of over ten thousand taxicabs and including 15 million data points in Beijing, China is used for this evaluation. These comparisons illustrate that using trajectory data from other vehicles can substantially improve the accuracy of forward trajectory prediction in the T-Drive data set. These results highlight the benefit of exploiting dynamic data to improve the accuracy of transportation simulation predictions.

CCS Concepts

• **Computing methodologies~Supervised learning**

Keywords

DDDAS; Destination prediction; Path prediction; Route prediction; Trajectory prediction; Markov processes; Pattern recognition; Prediction methods.

1. INTRODUCTION

Route or trajectory prediction algorithms attempt to predict the path that a vehicle will follow in the future assuming its current position is known, but the vehicle's final destination is unknown. These algorithms assume other information is available such as the trajectory taken by the vehicle thus far and/or the routes taken by other vehicles in its vicinity or derived from historical information. As discussed later, several algorithms have been developed to predict future trajectories.

Trajectory and destination prediction finds applications in many areas. For example, in military applications focusing on aerial attack of stationary and unknown ground target [2] and to determine mobility patterns of cell phone users for desirable antenna handoffs for moving vehicles [4].

SIGSIM-PADS '16, May 15-18, 2016, Banff, AB, Canada
 ISBN 978-1-4503-3742-7/16/05…$15.00
DOI: http://dx.doi.org/10.1145/2901378.2901407

It has been suggested that advertising signs can be targeted for nearby locations based on trajectories of approaching travelers [14]. Similarly, mobile apps may suggest restaurants, hotels, or other services based on the driver's expected trajectory [11]. Law enforcement could make use of these models as decision support tools to track the movement of individuals by iteratively sensing the location of a target vehicle, estimating its future location, and redeploying mobile sensors to locations where it is likely to reside in the future [6]. The fuel efficiency of hybrid vehicles can be improved if the system knows the planned route of the driver in advance; automated generation of this information eliminates the need for the driver to explicitly specify this information ([10][5]). This is especially true for routine trips, where the accuracy of personalized trajectory prediction schemes can be particularly good ([13][3][15][19]). Further, the need for such capability is seen in that it has been observed that in practice, in only approximately 1% of all trips does the user specify his/her intended destination ([12]).

Here, we are concerned with the use of trajectory prediction algorithms in the context of dynamic data-driven applications systems (DDDAS) [21]. DDDAS systems involve the use of a control loop that iteratively (1) senses the current state of the system, (2) predicts future system states, and (3) relocates or reconfigures monitoring devices or deploys changes in the operational system to optimize its behavior along one or more dimensions. One use of trajectory prediction is to predict routes to be used by faster-than-real-time microscopic transportation simulation. Such simulations may predict temporal and spatial properties of congestion likely to arise in the future in order to provide travelers with useful information for decision-making, e.g., to determine if an alternate route should be chosen. Today, more drivers are taking advantage of applications that use real-time data to inform them of traffic information such as existing congestion or obstructions (e.g., the mobile app Waze). Accurate simulation results may advise these users of suggested routes and departure times that reduce congestion [14] or reduce travel time globally or individually; we note that these objectives may conflict, and do not necessarily result in identical policies [18].

Similarly, online simulations are often used by transportation engineers to manage the transportation network itself through means such as varying traffic signal timing, ramp metering, or providing travelers with information or recommendations to plan their travel activities. Transportation engineers routinely use microscopic transportation simulation in order to gain insights in traffic, e.g., to determine desirable traffic control policies or logistical considerations of construction projects under a variety of scenarios ([7][20]). Desired output statistics for these models may include the level of congestion and travel times on roads of interest.

Among other factors, the accuracy of these simulation results depends on the validity of the simulation model data that they use.

Specifically, these simulators depend on knowledge of what routes will likely be taken by vehicles in the transportation network. Data-driven transportation simulations require dynamically collected data both to capture the current state of the system as well as to predict future system states. An important question concerns what data and how much data needs to be collected to make reliable predictions. This paper specifically examines the value of utilizing past vehicle trajectory information to provide information for routes likely to be taken by vehicles in the future. Trajectory predictions allow the simulators to use more sophisticated vehicle entity behavior than independent turn probabilities at each intersection.

Sources of dynamic traffic data are numerous. The government uses toll collection transponders, embedded roadway loop detectors [14], automatic license plate recognition, satellite imagery etc., and companies use information collected from drivers who carry smartphones (e.g., Google Maps) and other technologies. Technologies such as Bluetooth and WiFi detectors offer the ability to obtain partial trajectory information of individual vehicles in real-time in an operational transportation network.

Our hypothesis is that dynamic data significantly improves route prediction models. In the evaluation of the prediction models, our aim is to highlight the performance improvement that dynamic data-driven models yield over those that do not utilize such data. The predictive power of dynamic data in urban computing is seen in other areas as well. For example, [24] proposes a cloud-based scheme to predict the travel time of a driver along candidate routes using dynamic data: traffic conditions, time of day, weather, driver behavior, etc. As a result, this system allows GPS navigation systems to find the fastest route for a user, outperforming models that do not utilize this type of dynamic data.

This paper compares methods to predict a driver's future trajectory given the observation of the partial trajectory traversed by the vehicle thus far. For the vehicle whose travel is being predicted we do not assume that we have historical travel information, i.e., common destinations used by that driver in the past. Reliance on such data is problematic due to privacy concerns. This assumption allows the models to be easily applied to different vehicles in a DDDAS application, but will offer lower accuracy than personalized models since the preferred routine destinations of a particular driver are not known. Work by other authors have been used to predict a vehicle's future destination, irrespective of the future route taken to reach it. For example, Krumm's destination prediction model, described later, is based on efficient routes [11]. A modified form of Krumm's model is evaluated in this study.

In this paper, we discuss different methods to predict the trajectory (or route) of a driver on an urban road network and compare their accuracy utilizing the T-Drive trajectory data set. This data set consists of GPS trajectories of over ten thousand taxicabs (15 million data points) in Beijing, China. The performance of the models is measured by how often the next zone (a 1.25 km x 1.25 km square) visited by the vehicle falls within a predicted zone. Our results demonstrate that using route data collected for other drivers substantially improves the accuracy of forward trajectory prediction in the T-Drive data set.

While our primary goal is to show the value of trajectory prediction to drive on-line data-driven simulations, our contributions extend further. First, several proposed trajectory prediction algorithms are compared using a common dataset. Second, this paper examines the benefit of utilizing dynamic data from other vehicle trajectories in trajectory prediction algorithms. Third, several extensions to previously published prediction algorithms are developed and evaluated.

The next section reviews the destination prediction literature. This is followed by a description of the models examined in this study. The performance of the models using the T-Drive data set is presented, followed by a discussion of future work.

2. RELATED WORK

Early work in destination prediction was developed by Krumm [11]. This work utilizes a Bayesian model that uses the immediate past trajectory taken by a vehicle to predict the vehicle's intended destination. An underlying assumption used in this work is that drivers utilize efficient routes in order to reach their intended destination.

An efficient Markov model for destination and trajectory prediction is presented in [16]. When a prediction is requested, a data structure is traversed that holds the partial trajectory observed for the vehicle thus far. This data structure subsequently provides the empirical distribution of the forward trajectory.

Trajectory prediction using artificial neural networks is described in [14]. The previous five locations visited by the target vehicle are used to estimate the future trajectory by training a feed-forward artificial neural network with two hidden layers consisting of 500 neurons each. These three approaches – Krumm's approach, Markov models, and neural networks are discussed in greater detail later.

In 2008, a Carnage Mellon University group published the PROCAB model (Probabilistic Reasoning from Observed Context-Aware Behavior) [25]. This model uses the current context (e.g., accidents or congestion) and the user's preferences (e.g., fuel efficiency or safety) in order to predict his/her actions, rather than just focus on previous actions for prediction. The model maps actions into a Markov Decision Process (MDP), where intersections are encoded as states and road segments are encoded as transitions. State transitions are associated with a cost, namely the sum-product of road segment features and cost weights obtained from collected training data. The model assumes that drivers attempt to minimize the cost to reach their destination. For destination prediction, it has been reported that PROCAB outperformed Krumm's Predestination algorithm for the first half of the trip. The PROCAB implementation used in [25] takes into account specific road features such as the number of lanes and speed limit. A further challenge in the context of the work discussed here is we assume the destination is unknown; one of the inputs used by [25] for trajectory prediction is the actual destination of the target vehicle.

In [8] the authors suggest using a personalization profile of smartphone users for localized searches. User activities and on-device sensors are queried to build a context profile, including demographical features and previous user activities. The weights of this information along with an environmental profile (weather, temperature, etc.) are trained via an artificial neural network. These profiles are used to rank personalized queries (e.g., local businesses). One could envision utilizing the same model to predict destinations or routes, similar to PROCAB.

In [1] the authors use Krumm's destination prediction model based on efficient routes, but truncate the destination sample space to locations reachable within 30 minutes from the trip's

starting location, citing a study [9] that concluded that most driving trips end before that amount of time.

In [19] a hidden Markov model (HMM) is used to estimate a particular driver's destination and route, by using his/her previous trajectories along with driving time. Because personal identification is not specified in the data files of the T-Drive trajectory sample, we apply our Markov model (see Section 3.3) without discrimination towards any particular driver. That is, we train our model using the trajectories of many different drivers, rather than rely on individualized routes.

In [17], historical trajectories are converted into polygons by adding a 10m radius around the observed path. This data is then stored in a database. As soon as a prediction is requested, a polyline is formed through the requesting driver's observed GPS points and the database is checked for any intersecting polygon. If there are multiple results, further filters are applied to obtain the prediction. For example, match of the driver id would rank that particular polygon higher than one from another driver.

Sub-Trajectory Synthesis (SubSyn) [22] addresses the data sparsity problem, as well as privacy concerns, by synthetically generating trajectories from existing routes. These existing routes are decomposed into two-pair nodes and a 1st order Markov Model is used to generate the new trajectories.

Some of the approaches described above are not well suited for use of the T-Drive data set for evaluation. In some cases, necessary information such as source-destination information for trips are not known. Inference of this information from taxi trajectories is not straightforward. Further, the T-Drive sample data is not at a suitable temporal granularity for effective evaluation of some methods; for example, speed data cannot be easily derived.

3. PREDICTION MODELS

The three prediction models compared in this study are described next. These include Krumm's model along with several extensions developed during the course of this study, a model based on artificial neural networks, and a Markov model. Some of these approaches use trajectory information collected from other vehicles. An approach to store this trajectory information, termed Past and Future Trees, is also briefly discussed.

3.1 Krumm's Destination Prediction Algorithm Based on Efficient Routes

Krumm developed a *destination* prediction model based on the intuition that drivers reduce their minimum remaining travel time to their destination as the trip time increases [11], that is, drivers tend to the shortest path to a destination rather than more circuitous routes. Figure 1 illustrates this intuition. According to data collected in the Seattle area. In [11] Seattle was divided into cells measuring 1 km x 1 km. For the study, it was observed that when drivers transitioned between cells they reduced the potential minimum travel time to their destination 62.5% of the time. Krumm utilizes this driver tendency to seek efficient paths to estimate the probability that a given location (i.e. cell) is the driver's destination given the current partial path (i.e. the partial route of the traveler to their current location). That is, suppose a driver has now traversed a partial trajectory P and we wish to determine the estimated probability of the driver's destination being a cell c_i. Each probability $Pr(P|c_i)$ is computed by a product of $\{p, (1 - p)\}$, where $p = 0.625$ and the number of factors is equal to the partial trajectory length (there is a surjective mapping from the cells of the partial trajectory to $\{p, (1 - p)\}$). A factor of p is selected when the partial trajectory cell brings the vehicle closer to the candidate destination c_i and $(1-p)$ otherwise. All of these candidate $Pr(P|c_i)$'s are then multiplied with the prior probability $Pr(c_i)$, if the cell bias is known, and normalized, so that their sum equals 1 for a valid probability mass of the $Pr(c_i|P)$'s.

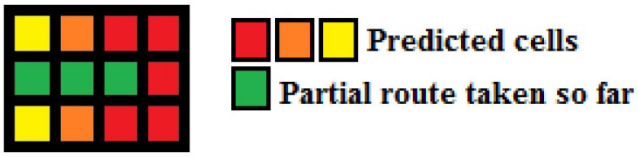

Figure 1: The red cells are assumed to be more likely because they are consistent (in efficiency) with the partial route taken so far (rectilinear distance).

In algorithm format, the prediction algorithm operates as follows:

```
1 For each candidate cell c_i
2  p_c = 1
3  For each cell c in partial route P
4    if c is closer to c_i than any previous cell in P
5     p_c = p_c * 0.65
6    else
7     p_c = p_c * 0.35
8  P(c_i) = p_c
9 Normalize all P(c_i) to get a valid probability mass
```

In the above algorithmic description, $Pr(P|c_i)$ is abbreviated as p_c and the cell bias is assumed to be unknown. If the cell bias is known, the probability estimate on Line 8 can be scaled accordingly.

In Section 4.1, we modify this algorithm to use it for *trajectory* prediction, or more precisely, for the *next transition* in the trajectory.

3.2 Artificial Neural Networks

Artificial neural networks (ANN's) map input values to output values and are often used to approximate complex functions (e.g., recognizing handwritten digits). For trajectory prediction, the input layer may encode some number of previously visited locations and the output layer may represent one or more projected future locations. ANN's consist of nodes called neurons (as they are inspired by the brain) and edges that connect them. In a feed-forward ANN (see Figure 2), information flows from the input layer to the output layer, via one or more hidden layers. The nodes in any layer are fully connected to the nodes of their preceding layer. Edges have weights associated with them that determine the strength of the signal sent by the preceding neuron. Typically, the product of the weight and the signal is then received by the neuron to which that the edge points. The signal sent by a neuron is determined by its *activation function*. Sigmoid functions are commonly used in feed-forward ANN's. One of them is the logistic function, $(1+exp(-x))^{-1}$, where x is the total input signal to the receiving neuron. For most values of x, the sigmoid function evaluates to a real number either close to 0 or close to 1. A popular, supervised learning algorithm for training feed-forward ANN's (i.e., determining the edge weights) is the backpropagation algorithm. Usually, the initial weights are randomized after which they are modified to minimize the error via gradient descent.

Since ANN's are often used in prediction to map relevant factors to an estimated output and training data is available, it seems

reasonable to assume that we can utilize them for trajectory prediction as well. Just as in the other trajectory prediction models, the partial trajectory of the target vehicle can be used as the input and the output layer can encode a future location of the vehicle. The design of the ANN, including the encoding of the input & output neurons and the architecture of the hidden layers is not trivial. [14] outlines three neuron encoding schemes and opts for the third one in the experiment: (i) A single neuron encodes the location, (ii) each possible location value is associated with a dedicated neuron, and (iii) a neuron encodes a bit in the binary representation. The learning parameters must also be selected carefully, in order to avoid oscillations, high errors, and/or under-training. [14] evaluates the trajectory prediction accuracy of a feed-forward artificial neural network (Figure 2) with sigmoid activation functions on artificially generated data. The input layer consists of the five previously visited locations in the trajectory where each location has multiple input neurons representing the integral location id in binary form. Two hidden layers of 500 neurons are used. Using a training set of 2000, the authors attain a success rate of almost 99% and 64% in their synthetically generated datasets, where the former quantity is tested on a dataset described as an easy data set with very little stochasticity while the latter quantity is tested on a dataset described to contain a considerable amount of random decisions.

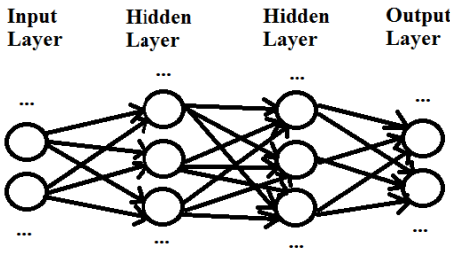

Figure 2: The edges have weights associated with them that set the strength of the value being sent. The circles are neurons that typically send values close to 0 or close to 1.

In order to reduce the training time to an amount comparable to the other tested models, we reduce the hidden layer count to just one and the hidden layer size to just seven neurons. With our fixed learning parameters, ANN's with more hidden layers and/or larger hidden layer sizes performed worse in our profiling tests.

3.3 Markov Models and Trajectory Storage

Pecher et al. [16] proposes a data structure that serves as the fast backbone of an arbitrary-order Markov model. This data structure can be used for a simple trajectory prediction algorithm if it is used to store historical trajectories. If queried order is sufficiently high and/or the data is sufficiently sparse, the future trajectory distribution is obtained from the maximal order for which data is available. A tolerance for minimum sample size can be set by the user. The emphasis on speed makes this implementation especially useful for embedded applications and/or simulation engines where realizations have to be drawn quickly. Historical route data is stored in Past and Future Trees and the data are queried on demand. The Past Tree of a location is queried with the partial trajectory traversed by the target vehicle up to that location. A Future Tree that contains the future trajectory distribution is then returned by the Past Tree. If the user wishes to get an estimate of the location of the target vehicle at some defined time in the future, the model invokes a microscopic traffic simulation. If storage is more important than access time, a hashtable implementation may be used instead (see Figure 3).

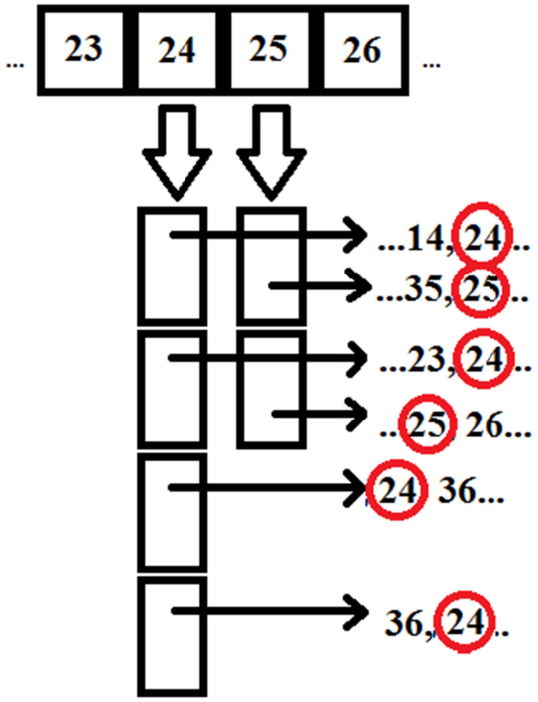

Figure 3: All trajectories that contain the cell 24 are being pointed to by the column 24. Empirical distributions of any Markov order can be obtained quickly by traversing this column.

Suppose the target vehicle is at cell 24, a prediction for the next cell is requested, and the partial trajectory so far is supplied. The 24th column can be accessed in constant time and its elements can be traversed comparing the elements with the partial trajectory observed so far. An empirical distribution can now be constructed by collecting a sufficient number of partial-trajectory matches along this column.

The models discussed thus far – in their current implementation – statically predict the future route with some spatial discretization, but do not specify point estimates for the time needed to reach the points along that route and do not return the projected location at a specific future time. If, rather than the next cell, the projected location of the target vehicle at Δt time units into the future is requested, one can construct a decision tree, T, that enumerates all possible paths in a breath-first fashion. Leaf nodes are tagged with the expected travel time to the intersections represented by those nodes until they exceed Δt. Once all leaf nodes exceed Δt, all locations expected to be reached at Δt time units into the future are tagged with the likelihoods stored in the Past Trees (or with those returned by the models of Section 3.1 and 3.2), and normalized to yield a valid probability mass. For example, if Past Trees are used and all the relevant leaf nodes of T have been enumerated and flagged, the Past Tree column of last observed location of the target vehicle is retrieved (Figure 3), a suitable number of matches of the partial trajectory observed thus far is queried from this column, and used to build an empirical distribution of future locations. Each path from the root node of T to a flagged node can now be used to retrieve the estimated probability from the empirical distribution. All these estimated

probabilities are then divided by their sum in order to scale them to a valid probability mass.

3.4 Generating Routes from Probability Maps

The prediction models described above generate probability maps indicating the likelihood the vehicle will reside at particular locations in the future. It is straightforward to use the probability maps to generate future routes. These models can also be used to simulate entity routing in microscopic traffic simulation. For example, if an ANN estimates the next-transition probabilities, one can input a random number into the inverse c.d.f. of the empirical distribution, and generate a next location. This next location, along with the previously visited locations, can be used as an input to the ANN and this process can be repeated. In the same fashion, the probability map returned by Krumm's prediction model based on efficient routes can be used to generate a cell. Lastly, Past Trees can be traversed quickly to construct an empirical distribution function of the next transition and random numbers can be used to obtain a sample of the next decision. In all these cases, a destination should be realized at some point of the simulated trip.

4. EXPERIMENTAL EVALUATION

The data set used for evaluating the accuracy of the models is the T-Drive sample dataset published by Microsoft Research ([24][23]). It consists of 15 million GPS points collected from over 10,000 taxicabs from February 2 to February 8, 2008, in Beijing. The sampling interval between two consecutive GPS coordinates is variable with a mean of 177 seconds (average distance: 623m), but can be as long as 600 seconds. It is important to note that the dataset consists of tour-based data and not origin-destination pairs. In other words, all the origin-destination pairs of sub-trajectories (taxi trips) are hidden in the tours of each taxicab file. The taximeter data of the T-Drive sample data set is not published (as of January, 2016). In our experimental evaluation, we discretize the road network into a two dimensional grid.

From the T-Drive Sample Dataset, we focus on an area that is approximately 100km × 100km: longitude from 116.0 to 117.1 (inclusive) and latitude from 39.6 to 40.5 (inclusive), in decimal degrees. This area was selected by isolating grid cells that have over 100,000 data points at the resolution of 0.1 decimal degrees in both dimensions (approx. 10km × 10km). This region is overlaid with a 96-wide and 80-high cell grid, where each cell maps to a 1.25km × 1.25km area. The relatively coarse grid size was chosen because of the relatively long median sampling interval in the dataset. Finer resolutions would require one to speculate on the path taken by the taxicab between successive reported locations. This is also the reason, in general, why the turn ratio at intersections cannot be determined with certainty in this data sample. To get consistent results, we use the same size for all cells; using variable cell sizes (e.g., determined by the variable sample intervals) could result in an arbitrary grid schema. Taxicab files are only considered if they fall within the above mentioned region. Taxicabs frequently visit certain, popular hubs to drop-off or pick-up passengers. In order to avoid utilizing these "artificial" trips we use a simple heuristic of ignoring the top 5% of the most popular cells, which are more likely to be hub locations, from the results.

We use the first 80% of filtered data points to train the models, and the last 20% of filtered data points to test the models. All interior cells in the grid contain eight neighbors (see Figure 4).

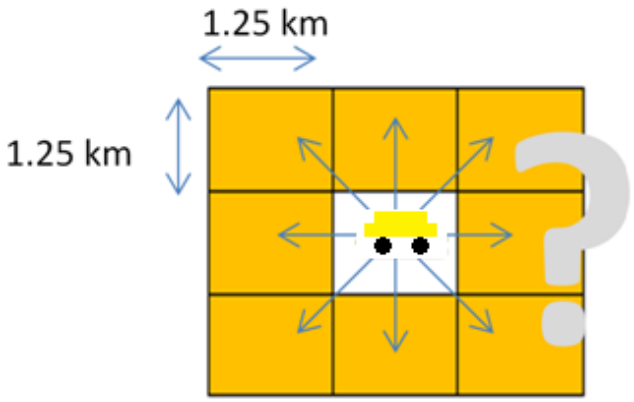

Figure 4: The next zone visited by the taxicab is uncertain, but is restricted to eight possible zones.

We consider the partial trajectory of the taxicab in question in order to predict the next cell visited. To evaluate the accuracy of the models, we assume that we can probe i grid cells, ranging from one to eight, for the presence of the vehicle. For each model, let $A(i)$ denote the measured cumulative probability of capturing the vehicle by greedily selecting the i most probable cells as estimated by the respective model. The objective is to achieve a high probability of detection for small i. In this section, if a model *m2* is described to be *x*% *better* or *improved* than model *m1 (*for a specific number of observed grid cells i), x is derived from the distance $A(i)_{m2} - A(i)_{m1}$ - not from a ratio of the two terms.

The test system uses an Intel i7 6700 CPU, 32GB of DDR4-2133 main memory, and a Samsung 840 Evo 1TB solid state disk. Neuroph v2.92 is used for the artificial neural networks in Section 4.3.

4.1 The Efficient Route Model and Extensions

We modify the efficient route algorithm from Section 3.1:

```
1 For each candidate cell c_i
2  p_c = 1
3  For each cell c in partial route P
4    If c is closer to c_i than the previous cell in P
5     p_c = p_c * 0.75
6    else
7     p_c = p_c * 0.25
8  P(c_i) = p_c*Markov(P(P.length-2), P(P.length-1))
9 Normalize all P(c_i) to get a valid probability mass
```

We use a less strict condition, on Line 4, to determine whether the vehicle opted for efficient transitions, in order to achieve better accuracy of the prediction of the next transition in the T-Drive sample. In most applications, the sample space of the destination is orders of magnitude greater than the immediate trajectory, so a more discriminatory algorithm is likely to perform better for the prediction of the destination. We also tune the p parameter (Line 5 and 7), denoting the fraction of time a transition is efficient, with respect to the T-Drive sample training set. Lastly, rather than use just the static cell popularity or no adjustment at all, the candidate $Pr(P|c_i)$'s are scaled by the 2nd order Markov estimates on Line 8. *Markov(start, end)* uses the subsequence of P, which starts with cell *start* (inclusive) and ends with cell *end* (inclusive), as arguments, and returns the corresponding Markov probability estimate. The indexing into P is assumed to be zero-based. We will refer to the condition on Line 4 with the term *isCloser* in the subsequent discussion.

For the *modified* Krumm route efficiency model without any historical data for the prior, we noticed very minor changes in the prediction accuracy for low i, as the efficiency likelihood parameter p is varied from 0.55 to 0.95 (using 0.1 increments). Among these, the best results were obtained with p=0.75. In Figure 5, the dark blue plot visualizes the performance, $A(i)$, of this model. This plot, as well as all other plots in Section 4, is derived from evaluating the test set - the last 20% of the data points.

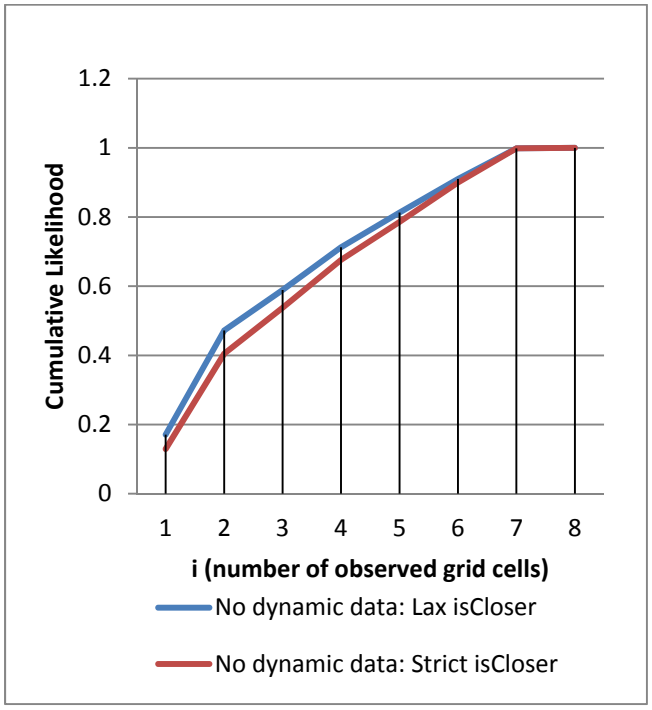

Figure 5: Cumulative likelihood for recapturing the target vehicle at the next cell transition using the path efficiency models (Krumm) with a p=0.75 parameter.

In the Seattle dataset, used in the original study, a value of p equal to 0.65 offered the best results. However, due to the different road network (Seattle vs. Bejing), the different driving population (Microsoft employees vs. cab drivers), slightly different grid size (1.25 km vs. 1.25 km), different objective function (distance to destination vs. next transition) or simply random noise (due to hidden factors) we found that setting p equal to 0.75 performed slightly better in this context. We experimented with dynamically varying the strength of the p parameter for the observed partial trajectory, using higher p parameter values for more current cell transitions than in older transitions. This results in slightly worse performance than a constant p=0.75 for all transitions. Thus, for all subsequent variations of the Krumm route efficiency model, we use p=0.75. We also note that truncating the partial route traversed by the taxicab to two or three cells did not yield any benefits. In other words, using the entire partial trajectory of the taxicab yielded superior results than just considering a limited horizon. Lastly, when deciding if a cell transition is efficient, different distance functions can be used. When we opted for Euclidian distance, rather than rectilinear distance, we obtained slightly worse results. As a result of these two considerations, partial trajectories for the efficient route models will not be truncated and rectilinear distances will be used.

One important algorithm modification that led to the previously mentioned results was the replacement of the efficiency random variable. In the original model, a transition is considered efficient if it brings the vehicle closer to the candidate cell relative to *any* other previously visited cell of the partial trajectory traversed thus far. Instead, we consider a transition efficient if it brings the vehicle closer to the candidate cell relative to the *previously visited* cell only. The original random variable results in slightly worse reacquisition likelihoods as the red-colored plot in Figure 5 illustrates. To be clear, the term *strict isCloser* exclusively refers to the conditional expression in Line 4 of the first algorithm listed in Section 3.1.

The original efficiency standard is stricter, as the comparison is relative to *all* previously visited cells. The sample space of potential destinations is much larger than just the next transition and if the efficiency standard is strict, a large area of potential destinations can be virtually eliminated (with very small assigned probabilities). Our objective is to predict the immediate trajectory, rather than the destination, a likely reason for the better performance of the lax conditional (the conditional on Line 4 of the algorithm listed at the beginning of this section). For all subsequent variations of the Krumm route efficiency model, we only consider a cell transition efficient if it brings the vehicle closer to the candidate cell compared to the previously visited cell only.

4.2 Markov Models

To evaluate the Markov models, we use the Past Tree implementation [16]. We do not set a sample size limit; in other words, we use relevant observations from the entire training set to populate the Past Trees. In our discussion, the term *order-i* refers to using the previous i locations of the target vehicle to make the next prediction. The results of Figure 6 are obtained from the pure order 1, 2, and 3 Markov models:

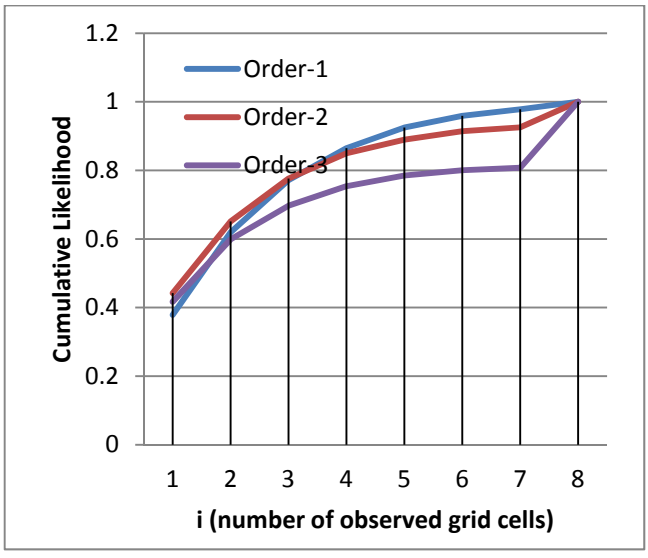

Figure 6: Cumulative likelihood for recapturing the target vehicle at the next cell transition using 1st, 2nd, and 3rd-order Markov models (Past Tree implementation).

Due to over-fitting or data sparsity, the order-3 model performs worse than the order-2 and order-1 model. The order-2 Markov model performs slightly better than the order-1 variant for i=1,2,3 because directional information can be inferred from the previous two destinations. However, there is a slight drop off in this advantage as i increases and the implementation of the order-1

model is easier. Below i=6, every Markov model outperforms the models that do not have historical data (the blue and red plots in Figure 5) and the i=1 accuracy is over twice as high in the Markov model results. Even though the models without historical data directly infer the immediate future trajectory from the *behavior* (route to current location) *of the target vehicle itself*, the data of *other vehicles* that engaged in the same behavior offers better point estimates on average.

Figure 7 shows the results of Markov models that attempt to use estimates from the next lower orders, after a candidate cell could not be found in the higher order model (due to lack of data). In other words, the estimate is extracted from the highest order model that has the candidate location available. If there is still a miss of the candidate cell in the order-1 model, the static cell popularity is used. There is little benefit of this cascading fallback mechanism at order-1, compared to the plain order-1 model, as the data available at order-1 is quite abundant (i.e., rare fallbacks to static cell popularities). However, the order-2 fallback estimates are on average 1% better from i=1 to i=3 and 4% better from i=5 to i=7, compared the plain order-2 model. The order-2 fallback model performs slightly better than the order-3 fallback model.

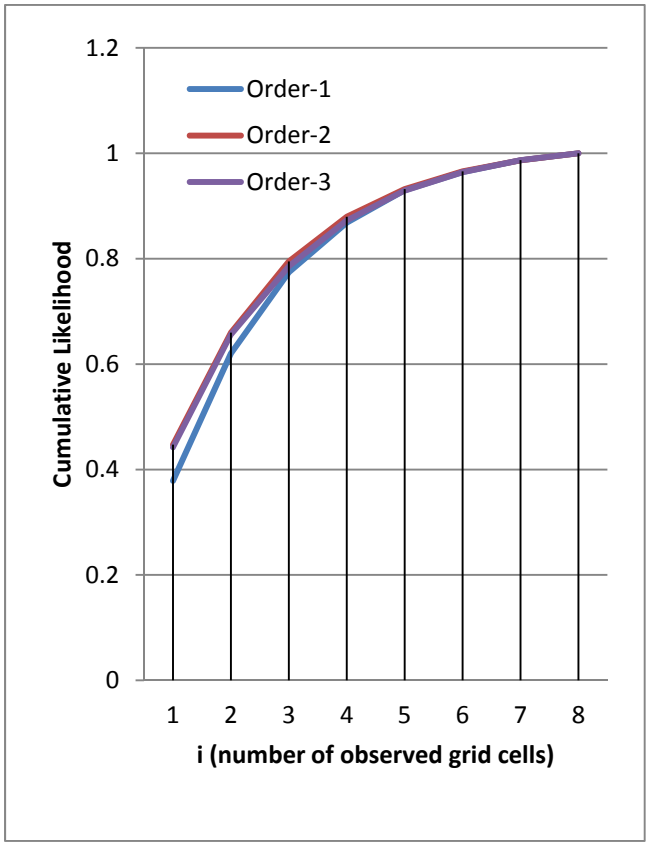

Figure 7: Cumulative likelihood for recapturing the target vehicle at the next cell transition using 1st, 2nd, and 3rd-order Markov models with a cascading fallback mechanism towards lower orders.

If we define the prior (i.e., the cell bias $Pr(c_i)$) in the Krumm route efficiency model as the static cell popularity (how much has each cell been visited by a vehicle in the training dataset), rather than just a uniform constant (in the case where no vehicle density data is available), and then normalize these probabilities for only the eight possible cells, we obtain a significantly improved (relative to the basic route efficiency model) distribution function, as can be seen in the dark blue-colored plot of Figure 8.

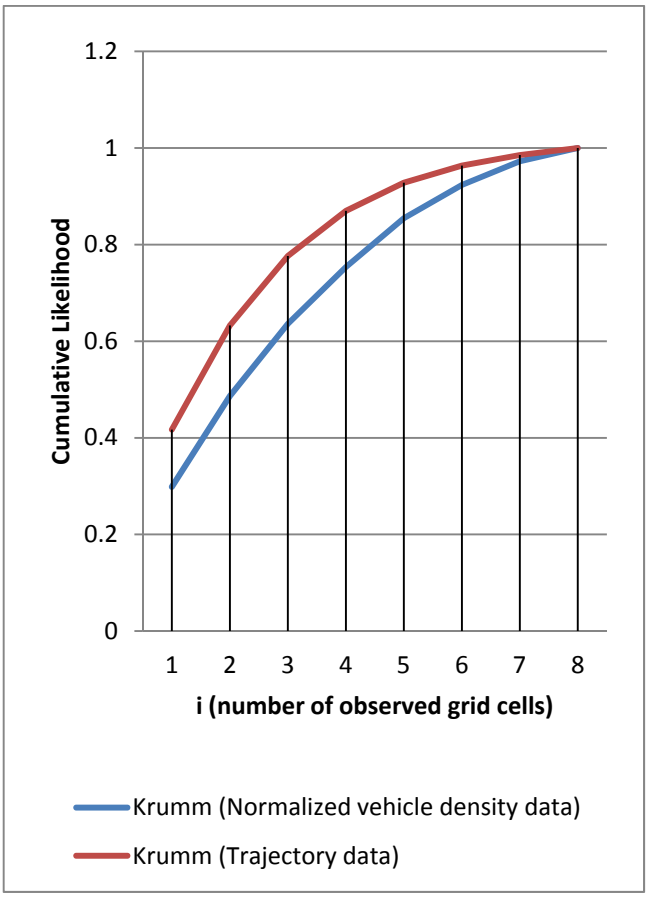

Figure 8: Cumulative likelihood for recapturing the target vehicle at the next cell transition using the modified Krumm models and dynamic data.

When we replace the prior (or the cell bias) in the Krumm route efficiency model with order-2 fallback probability estimates, we obtain reacquisition likelihoods that are improved further (the red-colored plot in Figure 8).

These results illustrate the power of using dynamic data in this particular application. Compared to the original route efficiency model, the probability of correctly predicting the target vehicle's location more than doubles at i=1. This advantage is crucial if there are several trials (reacquisition attempts), translating into savings in tracking resources and/or the more likely successful tracking of the target vehicle.

4.3 Feed-Forward Artificial Neural Network

For the feed-forward artificial neural network results, we use a single hidden layer with seven neurons in order to reach a training time comparable to the other models. Our profiling experiments showed that adding significantly more or less than seven neurons to the hidden layer and adding more than one hidden layer worsened the results, fixing the learning parameters described shortly. Omitting the hidden layer completely worsened the results. The activation functions are hyperbolic tangent and the edge weights are learned with the backpropagation algorithm. Because we opt for binary encodings of the input (of the five previously visited nodes, as in the setup in [14]) and our cells are

identified with 12 bits, we have 60 (=5×12) input neurons in total. The first six binary values of a binary cell identifier encode the row index, and the last six values encode the column index. The output consists of eight neurons, denoting the prediction of the next cell visited (direction relative to the current cell of the vehicle; see Figure 4), a higher output value being interpreted as a higher likelihood of the next cell visited matching with the cell associated with the given output neuron. Given a step size of 0.2 (based on our results, just slightly better than 0.8), a maximum error of 0.1 (based on our results, as good as 0.01), at most five learning cycles (better than lower values and just as good as most higher values), and including the 5% most popular cells, we obtain the results depicted in Figure 9. The value(s) in parentheses, after the abbreviation *ANN*, refer to the number of neurons in the hidden layer(s).

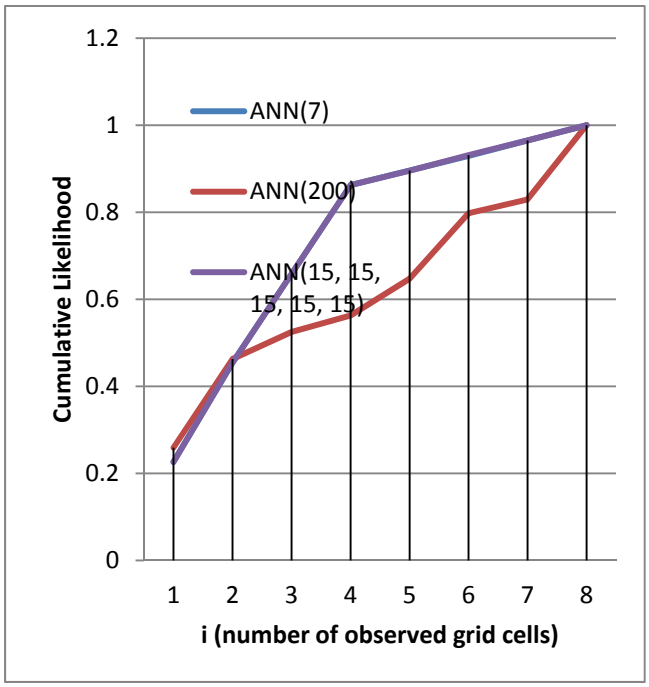

Figure 9: Cumulative likelihood for recapturing the target vehicle at the next cell transition using a feed-forward artificial neural network with one hidden layer, consisting of seven nodes.

The learning parameters are fixed for each ANN model. The performance of ANN(7) and ANN(15,15,15,15,15) is virtually the same. The ANN(7) results are superior to random guessing and the models that are only aware of the partial trajectory traversed by the target vehicle. It took 25.7 sec and 2.3 sec of CPU time to train and test ANN(7), respectively. One possible explanation of the change at i=4 is that most of the variation in the output is captured by the most significant bit of the most recent node's row and column index.

5. COMPARISON OF THE MODELS

Figure 10 shows the best performing members of the discussed model families. The blue plot shows the performance of the ANN with one hidden layer of seven neurons

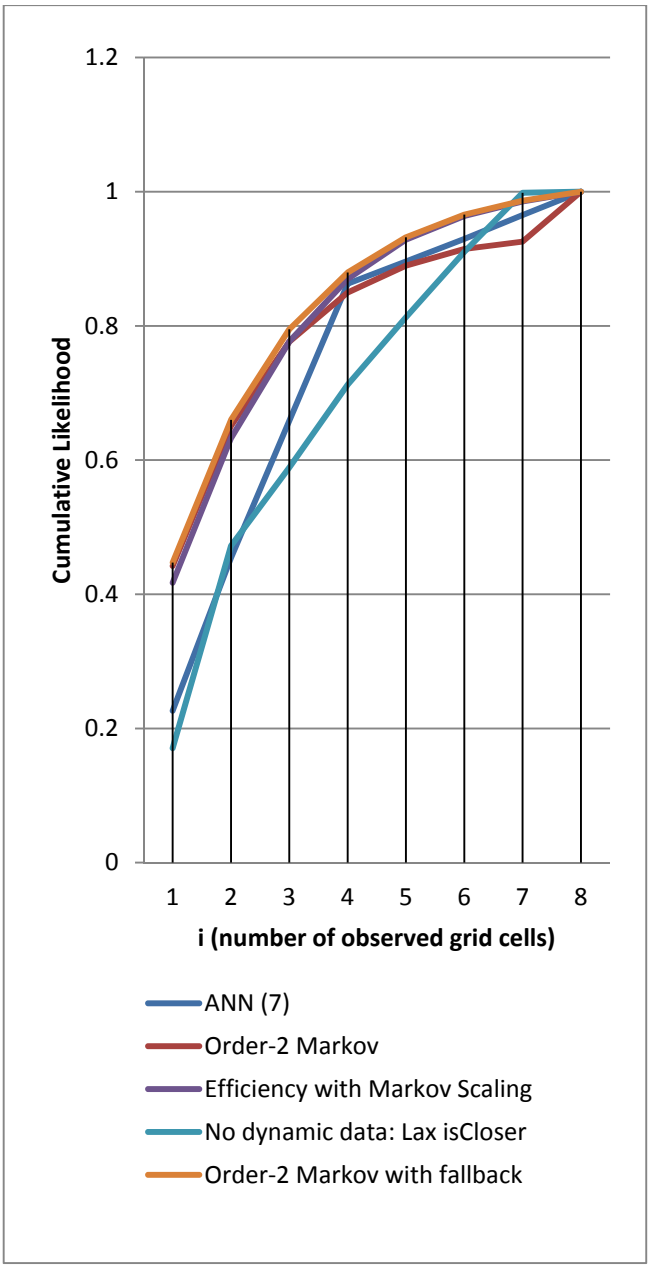

Figure 10: Cumulative likelihood for recapturing the target vehicle at the next cell transition using the best-performing submodels.

The order-2 Markov model with the fallback mechanism outperforms all other models for i<7. The preference for efficiency appears to already be implicit in the data. We have also evaluated a tournament predictor (not shown) that uses the estimates from either the hybrid model (efficiency scaled by order-2 fallback estimates) or the order-2 fallback model, depending on how well each model has performed in the current context (the trip progress and specific last two cells visited). For i<5, the tournament predictor performs worse than just the order-2 fallback model. While the pure efficiency prediction does not take into account specific features of the Beijing road network or other dynamic factors (cyan), it is – in its modified form (purple) - able to leverage the information that is implicit in the historical order-2 Markov estimates. The modified Krumm model, using the order-2

Markov fallback model in the cell bias term, performs slightly worse than just the pure order-2 Markov model at i=1, perhaps due to the power law nature of popular locations and thus the strong preference of drivers to transition to certain cells. With higher i, however, the skewness of the transition distribution reduces and drivers are likely to still be in transit, when efficiency is still very much relevant. Further investigation is needed to determine appropriate hyper parameters, architectures, and training heuristics to train deep neural networks for trajectory prediction. Our simple ANN architecture is able to outperform the Order-2 Markov model (without fallback) only after i=3, but is dominated by the efficiency-Markov hybrid model across all i.

6. CONCLUSION AND FUTURE WORK

This paper examines different route prediction models on the real-world T-Drive trajectory data sample collected in Beijing, China. All of the models tested can be executed reasonably quickly, even on mobile devices and/or simulation engines. For models making use of large data-sets, queries may be performed in the cloud. Some of the models are provided with the current trajectory of the driver, up to just before the tested zone, while others are provided with a relatively large training set of previously observed trajectories. In all cases where the model is permitted to use historical route data, rather than just the observed partial route, we observe substantial improvement in the forward-trajectory prediction accuracy on the T-Drive trajectory data sample - especially the order-2 Markov models and the Krumm-Markov hybrid model. We would like to emphasize the value of exploiting dynamic data, in accordance with the DDDAS paradigm, for the vehicle tracking problem (see [6]). It becomes especially important to improve the single trial reacquisition probability, when the scheme is repeated for a tracked vehicle. In any case, the economies of scale of the data acquisition costs should be compared to the tracking equipment acquisition and maintenance costs combined with the utility of recapturing the target vehicles. The value of successful trajectory prediction is observed in many commercial and non-commercial applications and is likely to grow as the penetration of smartphones and augmented reality devices increases. The models may also be used to generate accurate trajectories for entities in microscopic traffic simulation.

In the future, we plan to look at more factors and blended models on potentially different conceptual models; for example, roads being modeled rather than zones by randomly selecting among possible subsections in trajectories whose sampling interval may be relatively long. We also plan to investigate the impact in the accuracy of simulation output statistics, when these prediction models are used instead of the traditional inputs. Rather than just predict the forward trajectory statically, we envision predicting the forward trajectory by incorporating time.

7. ACKNOWLEDGMENTS

Funding for this project was provided by NSF Grant 1462503.

8. REFERENCES

[1] Amini, S., Brush, A.J., Krumm, J., Teevan, J., and Karlson, A. 2012. Trajectory-aware mobile search. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12). ACM, New York, NY, USA, 2561-2564.

[2] Andrew, S. J., Prazenica, R. J., and Jeffcoat, D. E. 2008. Optimal and feedback path planning for cooperative attack. Journal of Guidance, Control, and Dynamics 31, no. 6 (2008): 1708-1715.

[3] Ashbrook, D. and Starner, T. 2003. Using GPS To Learn Significant Locations and Predict Movement Across Multiple Users. Personal and Ubiquitous Computing, 2003. 7(5): p. 275-286.

[4] Cheng, C., Jain, R., and Berg, E.v.d. 2003. Location Prediction Algorithms for Mobile Wireless Systems, in Wireless Internet Handbook: Technologies, Standards, and Applications. 2003, CRC Press: Boca Raton, FL, USA. p. 245-263.

[5] Deguchi, Y., et al. 2004. Hev Charge/Discharge Control System Based on Navigation Information. In: SAE Convergence International Congress & Exposition on Transportation Electronics, Detroit, Michigan USA (2004)

[6] Fujimoto, R., Guin, A., Hunter, M., Park, H., Kannan, R., Kanitkar, G., Milholen, M., Neal, S., and Pecher, P. 2014. A Dynamic Data Driven Application System for Vehicle Tracking. International Conference on Computational Science, Dynamic Data Driven Application Systems Workshop, June 2014.

[7] Fujimoto, R., Hunter, M., Sirichoke, J., Palekar, M., Kim, H-K., and Suh, W. 2007. Ad Hoc Distributed Simulations. In Principles of Advanced and Distributed Simulation.

[8] Gui, F., Adjouadi, M. and Rishe, N. 2009. A contextualized and personalized approach for mobile search. Advanced Information Networking and Applications Workshops, 2009, 966–971

[9] Hu, P.S. and Reuscher, T.R. 2001. Summary of travel trends: 2001 national household transportation survey. Oak Ridge National Laboratory Technical Report ORNL/TM

[10] Johannesson, L., Asbogard, M., and Egardt, B. 2007. Assessing the potential of predictive control for hybrid vehicle powertrains using stochastic dynamic programming. IEEE Transactions on Intelligent Transportation Systems, 8(1):71–83, March 2007.

[11] Krumm, J. 2006. Real time destination prediction based on efficient routes. No. 2006-01-0811. SAE Technical Paper, 2006.

[12] Krumm, J. 2009. Where will they turn: Predicting turn proportions at intersections. Personal and Ubiquitous Computing, 2009

[13] Laasonen, K. 2005. Route prediction from cellular data. In Workshop on Context-Awareness for Proactive Systems (CAPS), Helsinki, Finland, vol. 1617. 2005.

[14] Mikluščák, T., Gregor, M., and Janota, A. 2012. Using Neural Networks for Route and Destination Prediction in Intelligent Transport Systems. In *Telematics in the Transport Environment*, pp. 380-387. Springer Berlin Heidelberg, 2012.

[15] Patterson, D.J., et al. 2004. Opportunity Knocks: A System to Provide Cognitive Assistance with Transportation Services. In UbiComp 2004: Ubiquitous Computing. Nottingham, UK: Springer.

[16] Pecher, P., Hunter, M. and Fujimoto, R. 2014. Past and future trees: structures for predicting vehicle trajectories in real-time. In *Simulation Conference (WSC), 2014 Winter* (pp. 2884-2895). IEEE.

[17] Persad-Maharaj, N. et al. 2008. Real-Time Travel Path Prediction Using GPS-Enabled Mobile Phones. Proc. 15th

World Congress on Intelligent Transportation Systems, ITS America, 2008.

[18] Roughgarden, T., and Tardos, E. 2002. How bad is selfish routing?. Journal of the ACM (JACM) 49, no. 2 (2002): 236-259.

[19] Simmons, R., Browning, B., Zhang, Y., and Sadekar, V. 2006. Learning to predict driver route and destination intent. Proc. Intelligent Transportation Systems Conference, pages 127–132, 2006.

[20] Suh, W., Hunter, M. P., and Fujimoto, R. 2014. Ad hoc distributed simulation for transportation system monitoring and near-term Prediction. Simulation Modeling Practice and Theory 41 (2014): 1-14.

[21] Voronkov, A., and Darema, F. 2004. Dynamic Data Driven Applications Systems: A New Paradigm for Application Simulations and Measurements. In International Conference on Computational Science. 662-669.

[22] Xue, A., Zhang R., Zheng, Y., Xie, X., Huang, J., and Xu, Z. 2013. Destination Prediction by Sub-Trajectory Synthesis and Privacy Protection Against Such Prediction. IEEE International Conference on Data Engineering (2013).

[23] Yuan, J., Zheng, Y., Zhang, C., Xie, W., Xie, X., Sun, G., and Huang, Y. 2010. T-drive: driving directions based on taxi trajectories. In Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS '10, pages 99{108, New York, NY, USA, 2010. ACM.

[24] Yuan, J., Zheng, Y., Xie, X., and Sun, G. 2011. Driving with knowledge from the physical world. In The 17th ACM SIGKDD international conference on Knowledge Discovery and Data mining, KDD '11, New York, NY, USA, 2011. ACM.

[25] Ziebart, B. D., Maas, A. L., Dey, A. K., and Bagnell, J. A. 2008. Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior. In Proceedings of the 10th international conference on Ubiquitous computing, pp. 322-331. ACM, 2008.