Proceedings of the 2014 Winter Simulation Conference
A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, eds.

PAST AND FUTURE TREES: STRUCTURES FOR PREDICTING VEHICLE TRAJECTORIES IN REAL-TIME

Philip Pecher

School of Industrial and Systems Engineering
Georgia Institute of Technology
765 Ferst Drive, NW
Atlanta, GA 30332, USA

Michael Hunter

School of Civil and Environmental Engineering
Georgia Institute of Technology
Mason Building, 790 Atlantic Drive
Atlanta, GA 30332, USA

Richard Fujimoto

School of Computational Science and Engineering
Georgia Institute of Technology
266 Ferst Drive
Atlanta, GA 30332, USA

ABSTRACT

We propose a data structure that stores previously observed vehicle paths in a given area in order to predict the forward trajectory of an observed vehicle at any stage. Incomplete vehicle trajectories are conditioned against in a *Past Tree*, to predict future trajectories in another tree structure - a *Future Tree*. Many use cases in transportation simulation benefit from higher validity by considering historical paths in determining how to route vehicle entities. Instead of assigning static and independent turn probabilities at intersections, the storage and retrieval of historical path information can give a more accurate picture of future traffic trends and enhance the capabilities of real-time simulations to, say, inform mobile phone users of expected traffic jams along certain segments, direct the search efforts of law enforcement personnel, or allow more effective synchronization of traffic signals.

1 INTRODUCTION

Computer simulation of road networks and transportation entities is a valuable tool for managing traffic and designing new construction projects. Some transportation models are small in scale and make many simplifying assumptions; others can be very sophisticated and computationally demanding. Many dynamic factors are present in the real world that are difficult to model and predict. For example, say, a particular event takes place that attracts one specific demographic; a simulation model can be made more accurate if it takes into account recently observed trajectories, rather than static turn frequencies at decision nodes. One way to view this information is via decision trees holding relative frequencies at each branching point. Observed trajectories can be stored in a tree data structure (Past Tree) where nodes represent past trajectories up to the given decision point and hold references to other trees (Future Trees) that store observed frequencies of the remaining trajectory. If the target entity is currently in an area where trajectories have been observed in the past, population path similarity can be used to predict where a vehicle will turn or to generate realistic trajectories. For reasons that will become obvious in Section 2.1, we will subsequently refer to *Past and Future Trees* as simply *Past Trees*. We will also sometimes use the expression *Method of Past Trees* (or simply, *Past Trees* in the charts of Section 3) to specifically refer to the querying operation on Past Trees that outputs location estimates (see Section 2.2).

Perhaps the most interesting use of Past Trees is found in the arena of dynamic data-driven application systems - DDDAS (see Voronkov and Darema 2004). Modeling individual vehicles accurately creates a variety of applications. The analysis of emergent, macro-scale phenomena in real-time data-driven types of simulations can be of great interest to various stakeholders. To optimize systems for efficiency, for example, it is possible to model traffic congestion (Fujimoto et al. 2007) and evacuation models (Chaturvedi et al. 2006) with a high degree of validity. The data centers of web search providers have clustered equipment based on the expected workload in different domains (Marin et al. 2013). Past Trees can model consumer preferences and assist in parameter tuning (Ye et al. 2008) and bottleneck analysis of computer networks. Past Trees may also be used in online simulations to determine an accident from cell phone data (Madey, Szabo, and Barabási 2006; Madey et al. 2007) or in law enforcement applications (Fujimoto et al. 2014).

1.1 Problem Description

Consider a directed road network *G(V, E)* where vertices represent intersections and arcs represent road segments. Suppose a transportation entity, such as an automobile, E_T, has been observed at position s_0 on arc e_0 at time t_0 (s_0 is a real scalar in [0, 1] to represent the position *within* the directed arc e_0). Our objective is to estimate the location of E_T at time $t_0 + \Delta t$.

It is helpful, but not necessary, to assume that before t_0, we have observed some partial trajectory of E_T. As an application within the context of dynamic data-driven simulations, we discuss in Fujimoto et al. (2014) an on-line framework for vehicle tracking that may use the Past Trees to iteratively simulate future locations of E_T. The estimated locations are then published to mobile sensors (or a crowd-sourced participant set) which can, in turn, respond with the realized location.

1.2 Related Work

Destination estimation is an active field of research. The constraint on the models selected here, however, is that they must be at a conceptual scope that can be mapped directly to a general transportation simulation model. Prediction models for a highly domain-specific area - like handoffs of a mobile phone from antenna-to-antenna (Cheng, Jain, and Van Den Berg 2003), or where the target is at an unknown position, but stationary (Sinclair, Prazenica, and Jeffcoat 2008), for example - are inappropriate within general simulation frameworks. Those that extrapolate from direct vehicular motions (e.g., dead reckoning models; see Cai, Lee, and Chen 1999) are too granular and impose an unrealistic behavior on a given vehicle entity. We also ignore models that depend on the identification of a *particular* vehicle and its historical data (Laasonen 2005).

In Section 1.3, we summarize a prediction model based on efficient routes (Krumm 2006). The model predicts based on the idea that drivers tend to make choices that are fairly efficient when attempting to reach their destination. Unlike Past Trees, Krumm's prediction model does not use any historical trajectory data. Nevertheless, we will use it as a benchmark in Section 3.

1.3 Krumm's Prediction Model

John Krumm developed a static prediction model that assigns higher likelihoods to areas that are consistent with the path taken thus far. Thus, inefficient potential forward trajectories are given less weight than those that are consistent with efficient behavior. It is important to note that this model does not utilize previously recorded trajectories, outside the partial trajectory of the vehicle currently being observed. Krumm assigns each cell, c_i, in a region an estimated probability, based on the observed path, S, taken thus far, using Bayes' Law:

$$p(c_i|S) = \frac{p(S|c_i) \times p(c_i)}{\sum_{j=1}^{N_c} p(S|c_i) \times p(c_i)}.$$

Krumm has empirically determined that the probability of an efficient cell transition is 0.625 in a particular experiment in Seattle, WA. For a particular cell c_i in the grid, each cell of the observed trajectory is multiplied by 0.625 if it brings the target closer to c_i, and by 1-0.625 if not. This is then multiplied by an overall cell bias term $p(c_i)$ and normalized.

2 PAST AND FUTURE TREES

2.1 Conceptual Structure and Trajectory Insertions

In order to make proper use of Past and Future Trees, a number of population trajectories must be sampled first. In the following conceptual description of the structures, we assume that this data has been collected already and has been inserted into the structures. A discussion on trajectory insertion (see Figure 1) follows this description. Querying Past Trees to make predictions (i.e., the Method of Past Trees) is discussed in the next section.

Every decision point in a region (e.g., the road on the western side on a particular intersection) is associated with an instance of a Past Tree and each node in a Past Tree links to an instance of a Future Tree (see Figure 1). The vertices of a Future Tree contain traversal counts as well as destination counts. If a partial trajectory, $T_p=(e_0, e_1,...,e_f)$, of a vehicle up to a decision point has been observed, the corresponding Past Tree is traversed in the reverse order of T_p (the root corresponds to e_f). Traversing the Past Tree in this fashion is equivalent to conditioning against the observed trajectory. This means that if no partial trajectory up to e_f, had been observed, we would simply access the Future Tree associated with the Past Tree's root node. If the currently observed partial trajectory does not exist in the tree, the traversal only goes down to the node that does not have the requested child. Once the corresponding Future Tree is discovered, it is traversed to yield the measure of interest. Some of these metrics are discussed in the next two subsections (2.2 and 2.3). For example, if one wishes to construct an estimated destination probability map, a breadth-first search for positive destination counts can construct a frequency distribution, leading to the desired map.

In Figure 1 below, we assume no previous trajectories have been observed (i.e., this is the first trajectory being observed). Further, this trajectory will cause updates on every Past Tree adjacent to the realized path. However, for illustrative purposes, we will only focus on the updates to the Past Tree corresponding to the western side of Intersection 3, $\text{PastTree}_{3,W}$. A vehicle has approached Intersection 3 by first traversing Intersection 1 and then Intersection 2. This partial trajectory is inserted into $\text{PastTree}_{3,W}$ (the solid black arcs). After that point, the remaining path is inserted in the Future Trees of all vertices in the path up to Intersection 3 (the lightly dotted arcs), so that any future read request on the Past Tree may access this particular observation for any partial trajectory that follows the same pattern up to the current decision point (irrespective of the length). The Future Tree's traversal counts are incremented if the vehicle traverses the given intersection and the destination counts are incremented if the vehicle stops in the given intersection's proximity (i.e., terminal point along the trajectory). As mentioned before, we assume this is the first trajectory being observed. Had the Past Tree already been partially populated, it could have been the case that no additional arcs would have had to be inserted. However, the counts would still need to be incremented.

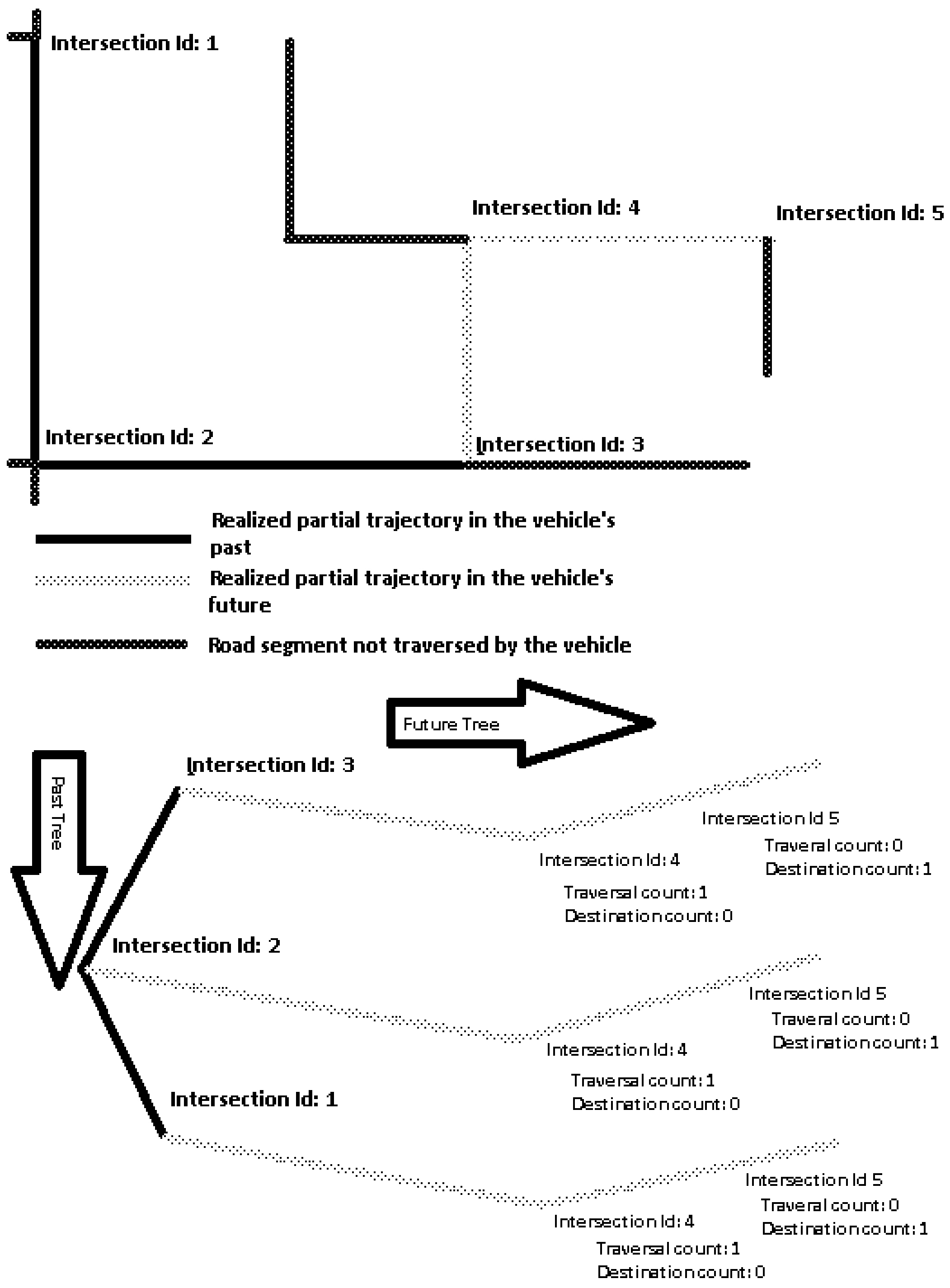

Figure 1: Past & Future Tree visualization.

2.2 The Method of Past Trees: Predicting Trajectories and Destinations

Once a Past Tree for a given decision point is populated, it can be used to make predictions for a vehicle approaching the given decision point. If the vehicle's trajectory up to the point has been observed, the given Past Tree can be queried to retrieve the corresponding Future Tree. Predictions regarding any future decision point can be made by considering the corresponding nodes at that depth of the Future Tree. It should be noted that this yields the nth decision into the future and, therefore, has limited use: These points could be asynchronous in the expected arrival time (e.g., the example travel times to the Future Tree nodes in Figure 2). A prediction for *Δt* time units into the future requires locations to be tagged with point estimates of the travel time it takes to reach the respective points. A general search algorithm can then return the set of feasibly reachable locations (the intersection of the Future Tree and the shaded line cut in Figure 2). The frequency distribution of the traversal counts in this set can then be used to build an estimated probability mass on the map. The following pseudocode (for a parallel shared-memory machine), applied on the Future Tree of an observed vehicle, yields this prediction (a *probability map*):

```
Input: Future Tree (this instance), Δt
1 i:=0
2 root.travelTime := 0
3 while (there is at least one non-leaf vertex of travelTime < Δt at depth $D_i$) {
4   Par for all ( non-leaf vertices u ∈ $D_i$ with u.travelTime < Δt ) {
5      Par for all w that are adjacent to u {
6         <edgeTravelTime, positionAtΔt> := simulateTravel((u,w,Δt))
7         w.travelTime := u.travelTime +edgeTravelTime
8         if (w.travelTime > Δt) {
9            w.late := true
10           frequencyMap.populateCountAtPosition(positionAtΔt, u, w)
11        }
12     }
13  }
14  sync()
15  i := i+1 //master process only
16 }
17 probabilityMap := frequencyMap.transformToProbabilities()
18 return probabilityMap
```

In general, this depth-synchronous implementation is not efficient, even if the underlying work-scheduler balances optimally: A time-parallel simulation (Kiesling and Luthi 2005) may simultaneously evaluate delays across road segments on differing depths and, therefore, offers the flexibility to saturate all the available processors with work. Nevertheless, we focus on this implementation for ease of exposition. The procedure expands all simple paths from the root node (Lines 3-4) until all the terminal vertices are tagged with a travel time greater than Δt; at this point, the potential locations of the vehicle are found. Pursuant to this goal, vertices are tagged with a point estimate of the travel time to reach the respective intersections by invoking the transportation simulation, *simulateTravel*, on Line 6; this subroutine returns the travel time on the road segment from vertex u to vertex w. Unless this subroutine is implemented in a trivial fashion (for example, by simply retrieving a point estimate from historical data and not considering live data), the execution time of this call will dominate the total runtime of the program. (The time used by this bottleneck primarily depends on the granularity of the simulation model; a computationally efficient microscopic traffic simulation model can be found in Nagel and Schreckenberg 1992.) If the cumulative travel time up to w is greater than Δt, simulateTravel will also return the predicted position of the vehicle at Δt (positionAtΔt). At this point, the observed traversal count is queried from the corresponding Future Tree (from vertex w is the vehicle is closer to w than to u; otherwise the traversal count from u is used). This count is inserted into a map representing the frequency distribution at Δt (Line 10). After this map is populated, it is converted to a probability mass on the map and returned (Lines 17-18).

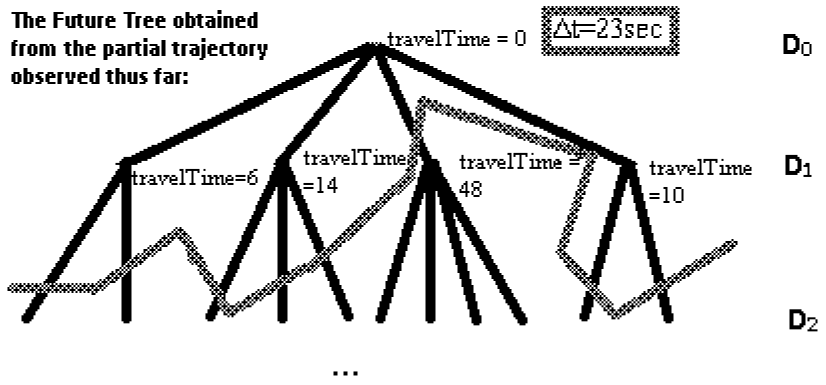

Figure 2: The Future Tree is used to retrieve the potential locations of the vehicle Δt into the future. In this case, we identify all vertices that are reached from the current location within a time span of 23 seconds. The shaded line illustrates the possible locations at that time.

If the question is what *destination* the vehicle is likely to approach, the destination counts described in Section 2.1 are fetched with the help of any tree search procedure (e.g., breadth-first search). The non-zero *destination count* of *every* node in the future tree is inserted into the appropriate location in the frequency map, which is subsequently transformed into a probability map and returned.

2.3 Generating Simulation Trajectories

Suppose we have constructed a subgraph that represents some road-connected intersections and now we wish to generate and route vehicle entities within. The following pseudocode illustrates these steps at a high level:

```
Input: Future Tree (assume code is applied to this instance), vehicle entity e
1 totalCount := 0
2 for all ( vertices u ∈ $D_1$ ) {
3  totalCount := totalCount + u.traversalCount
4  traversalCountBoundaryList.add(totalCount)
5 }
6 $U_0$ := realization from a discrete uniform between 0 and totalCount
7 toNode := computeToNode( traversalCountBoundaryList, $U_0$ )
8 e.setNextNode( toNode )
9 sampleSpaceCardinality := toNode.destinationCount + toNode.traversalCount
10 e.stopAtNextNode := Bernoulli(toNode.destinationCount / sampleSpaceCardinality)
11 return e
```

Vehicle instances will still be created as usual (e.g., from a non-homogeneous Poisson process). Once the entity is in the model and reaches a decision point, the simulation engine will query the Past Tree corresponding to a decision point for the trajectory taken up to that point. This, then yields a reference to the corresponding Future Tree. At Depth 1 of the tree, the observed frequency counts of all sibling nodes are read and inserted into a list, *traversalCountBoundaryList* (Lines 2-5). A random number (Line 6) is then transformed to a decision from this empirical frequency distribution via its c.d.f. (Line 7) and assigns it as the next location of the vehicle entity (Line 8). However, the vehicle entity may stop at that point. To account for this, we first realize the next intersection (Lines 1-8) and then condition against it (Lines 9-10). Both the destination and the traversal counts are known, so whether the vehicle stops can be inferred from a Bernoulli realization with success probability equal to the destination count divided by the sum of the destination count and the traversal count of the realized vertex (Line 10).

2.4 Space-Efficient Storage

The naive storage mechanism (i.e., a tree for each decision point) has much redundancy, but offers fast access to the future likelihoods (in fact, linear time in the trajectory length taken thus far). One observed trajectory will cause insertions into every tree along its path. This redundancy causes much wastage of storage. More precisely, suppose the length of the typical trajectory is *n* and the total number of observed trajectories is *T*. This vector of size O(n) is pushed into *n* Past Trees, but in each Past Tree it is replicated *n* times because it is recorded for each partial trajectory up to the point represented by the tree. The total storage requirement in this representation is thus $O(T \times n^3)$.

A sensible approach would be to cut off the tree at a certain depth - one should expect diminishing gains in accuracy when going down the Past Tree at significant depths (this is obviously not true in all practical situations). The results in Section 3 truncate the Past Trees at Depth 3.

A better approach, reducing the storage requirement from $O(T \times n^3)$ to $O(T \times n)$, is to use an array indexed by directed intersection identifiers (this is with almost no loss of generality as a different identification scheme can be coupled with a hashing rule with no performance loss in the average case). Each cell in the array will hold a reference to the head of a linked list. Such a linked list will simply contain pointers to trajectories containing the corresponding directed intersection identifiers (see Figure 3). Every time a trajectory is observed, all linked lists that contain a directed intersection of that trajectory will be augmented by one more reference. Thus, the storage requirement and the total insertion costs are limited to $O(T \times n)$. The potential for a highly skewed length distribution (on the collection of trajectories on directed intersection identifiers) motivates the use of lists, rather than arrays. Once a prediction is requested (when a vehicle has entered a particular directed intersection), a significant number, k, of previously observed trajectories are queried: The corresponding header can be accessed in constant time (random access with a specific directed intersection id) which allows for the extraction of the k trajectories of interest. Building a corresponding tree (by parsing and inserting the k trajectories) takes linear time in the size of the trajectories if partial trajectories outside of the currently observed one are omitted, leading to a total cost of $O(k \times n)$.

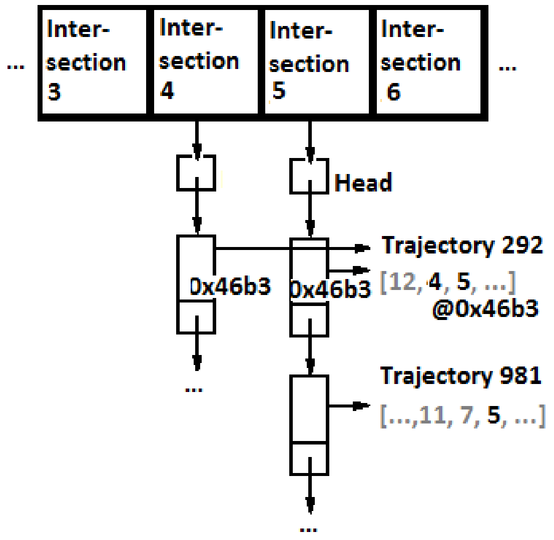

Figure 3: Illustration of the space-efficient storage scheme. Trajectory 292 is stored (only once) at location 0x46b3. Since it contains Intersection 4 and 5, the lists of these two intersections contain pointers to the trajectory's memory location.

3 EXPERIMENTAL EVALUATION

To evaluate the accuracy of Past Trees, separate experiments were performed on randomly generated input as well as real-world data. Both experiments are run on cellular automata (CA) and generate two different error plots each (lower quantities are more desirable). In both cases, the directed graph representation introduced in Section 1.1 can be mapped in one-to-one correspondence with the CA if the appropriate cell transition rules are added.

The Method of Past Trees is compared to the method described in Section 1.3 (Krumm 2006), which does not require previously collected trajectories. Whether the data acquisition and storage costs for Past Trees are acceptable depends on the added value from the increased accuracy for a given application. These costs scale with the desired sample size and the size of the target area.

3.1 Synthetic Experiment: Setup

A computational experiment was performed to sample the degree of accuracy provided by Past Trees. Here, the input (i.e., the trajectories of the drivers) is generated synthetically.

The model is based on a 20-cell-by-20-cell cellular automaton; this discretized plane serves as a conceptual model of a real-world region with pervasive road network coverage. 200 trajectories are sampled as follows: The origin and destination are first sampled by the pseudo-random selection of two cells. In this selection, some cells are more likely to be realized than others because cell popularity is assigned - at the beginning of the simulation - from a probability distribution that mimics the population distribution of US cities. Each cell is assigned a probability by first sampling the lognormal distribution with $\mu=7.28$ and $\sigma=1.75$ (which is a good fit for the population distribution of the United States - see Eeckhout 2004), evaluating the density at those realizations, and then normalizing these values into a probability mass. Further, positive correlation is forced upon the popularities of neighboring cells ($\rho=0.3$): Less (more) popular cells are more likely to be near less (more) popular cells.

After the origin and destination are determined, the cells in between them are generated. At each intermediate cell, the vehicle will transition to a cell that is in accordance with the most efficient route with probability 0.625, and to one that is not with probability 1-0.625; we call this behavior *the desire for efficiency*. Each cell also has a statically assigned 10% chance of being an entry point that leads into a predetermined tunnel of neighboring cells, *a preferred path*. The length of a preferred path is geometrically distributed with a mean of five cells. A cell that leads into a preferred path is privileged to temporarily override the desire for efficiency until the vehicle exits the preferred path. Preferred paths (and their entry points) are randomly generated at the beginning of the simulation experiment and don't change during the execution. In this scenario, preferred paths have a mean length of five cells. The rationale of including preferred paths is to model unexplained trends within the population. Of the 200 simulated trajectories, 80% are first used to populate the trees; the remaining 20% are used to predict destinations and compute errors. The depth of Past Trees is limited to three cells.

3.1.1 Synthetic Experiment: Results

Figure 4 shows the mean accuracy as a function of the fraction of the trip completed (ensemble averages across the 0.2*200=40 trajectories). At each transition of any trajectory in the test set, the error model queries the Past and Future Trees of the current cell for all potential destinations of the vehicle. It then computes the rectilinear distance from the vehicle's current position to the center of the estimated probability mass (the predicted destination) as well as the rectilinear distance from the vehicle's current position to the actual destination. The error in Figure 4 is the absolute value of the difference of these two values.

Across all the simulated trajectories, for each cell transition, an error is computed and inserted into the bin corresponding to the tenths precision bin of the trip completion fraction. Then, for every completed trip fraction bin, the average error is selected. 0.8*200 = 160 trajectories were used to populate the trees.

As the vehicle entity approaches its destination, more data can be used because the partial trajectory up to the given cell is available: In the case of Krumm, it points towards efficient behavior, and in the case of Past Trees it points towards previously observed trajectories that are similar. This explains the steady improvement to a trip completion of 0.7. The increase of the plot at the end of the trip is a result of the vehicle approaching its destination relatively quickly and thus decreasing the first term mentioned above (distance to actual destination), while the second term (distance to predicted destination) does not keep up.

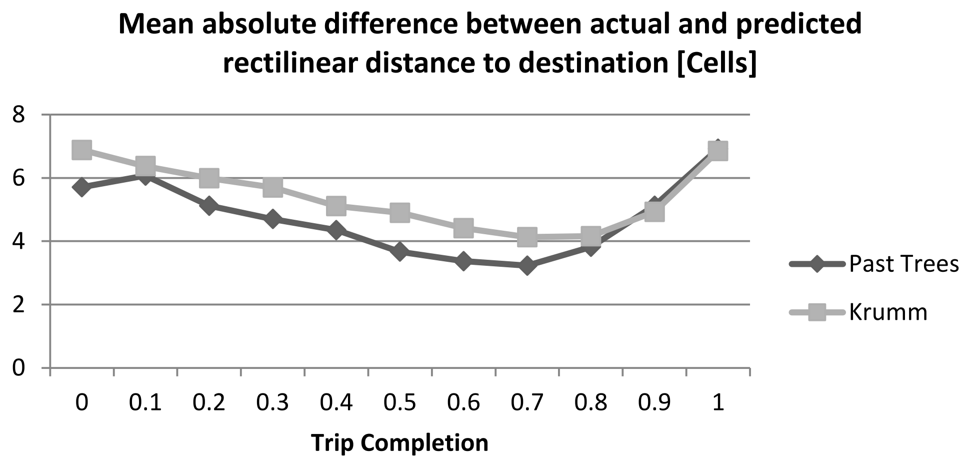

Figure 4: Error plots of the mean absolute difference between actual and predicted rectilinear distance to destination.

Figure 5 simply shows the rectilinear distance between the predicted and actual destination. This quantity hovers around 10 cells for both prediction models (with a slight improvement towards the end of the trip). This is reasonably close in a 20-by-20 grid; no miracles are to be expected in a scenario where the trajectories of vehicles *almost* exhibit the patterns of 2D random walks with cell transitions that are independent of one another.

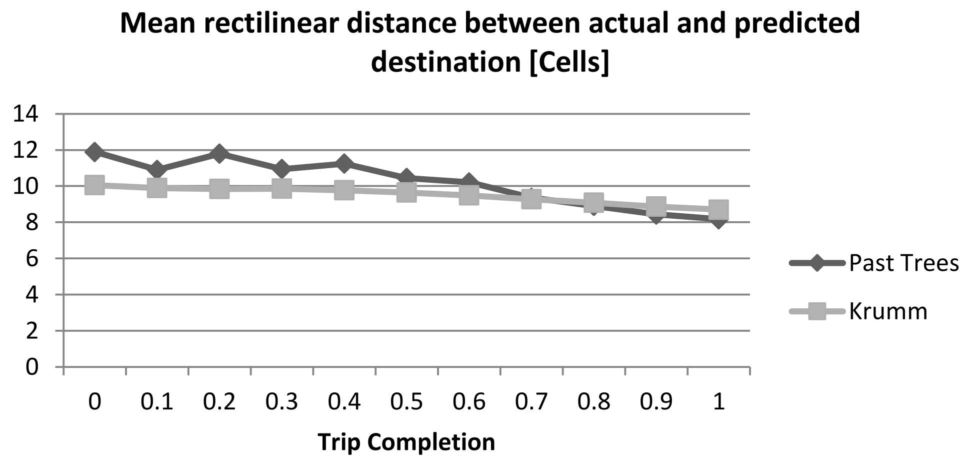

Figure 5: Error plots of the mean rectilinear distance between actual and predicted destination.

3.2 Real-World Traces: Setup and Results

A deterministic simulation was performed using the detailed NGSIM Peachtree Street dataset, collected between 4p and 4:15p on November 8th, 2006 in Atlanta, GA (NGSIM Community Home 2014).

The road segment was modeled as a 30-by-3 cellular automaton and the first 80% of the 461 observed trajectories were used to populate the trees (the remaining 20% were used to test the model).

The following plot illustrates the error computed in the same fashion as illustrated in Section 3.1.2 for Figure 4. It should be noted here that Krumm's model is not well-suited for an almost one-dimensional environment as the estimated destination probabilities will just be uniform across the entire road. Because the sample size is small, the first 80% of each trajectory use superimposed Past Trees for each cell.

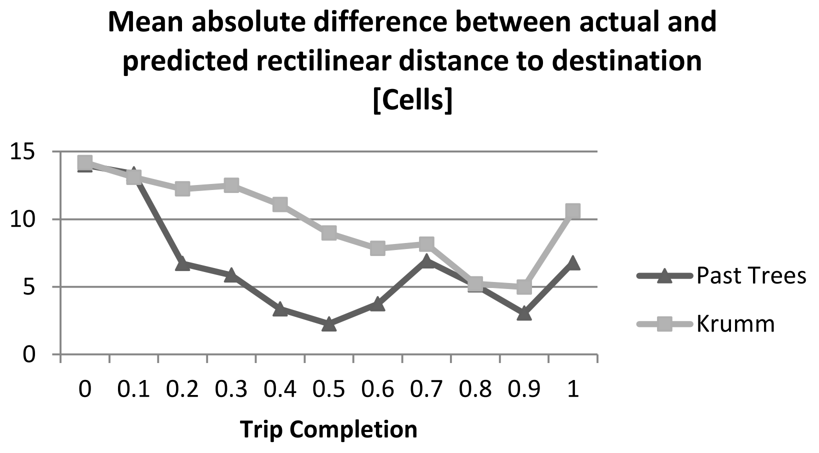

Figure 6: Error plots of the mean absolute difference between actual and predicted rectilinear distance to destination.

Figure 7 presents errors in the same way as Figure 5. For the Past Tree errors, the increased values towards the end of a trip are the result of time not being considered in the models: As the vehicle entity is in one of the final cells, the model assumes that the partial trajectory up to the current point is indifferent from one that is near the center of the trajectory, which leads to an estimated cell much farther away. This may be avoided by not only incrementing destination counts, but also noting the progress along the trajectory during the data collection phase.

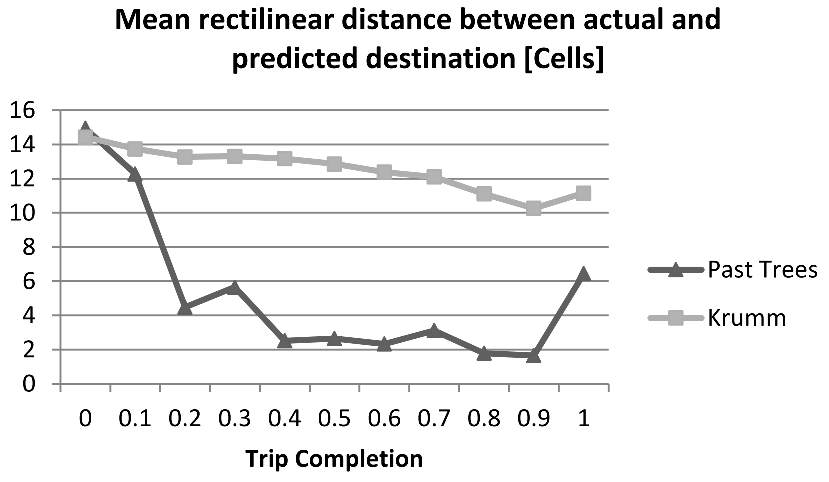

Figure 7: Error plots of the mean rectilinear distance between actual and predicted destination.

4 CONCLUSION AND FUTURE WORK

A structure to store trajectories along road segments was introduced. Past Trees can be used to either generate trajectories for simulation entities or to predict likely destination of a vehicle whose path is being observed. The experimental results show that the structure gives a solid prediction framework. Modeling vehicular movement is clearly of great economic value as driver behavior and traffic management can be made more efficient.

In the future, we plan to investigate enhancements to Past Trees, such as conditioning against more factors as well as weighing more recently observed trajectories more heavily; applying statistical regression and exponential smoothing to the Method of Past Trees has the potential to improve accuracy. Although it was mentioned above that every decision point has one Past Tree associated with it, it can be very useful to have multiple trees to account for factors such as seasonality (the same tree can be used if it is 'colored') or other factors like congestion level. Traffic exhibits seasonal characteristics; for example, drivers might opt for a particular pathway during rush hour in order to avoid congestion in other areas.

It seems reasonable that more recent collections are most heavily correlated with present and near-future trajectories. For example, a popular sports game may tilt historical behavior into a particular direction. If paths are proactively collected in a particular area, exponential smoothing can adapt to recent trends.

ACKNOWLEDGMENTS

Funding for this project was provided by AFOSR Grant FA9550-13-1-0100.

REFERENCES

Cai, W., F. Lee, and L. Chen. 1999. "An auto-adaptive dead reckoning algorithm for distributed interactive simulation." In *Proceedings of the thirteenth workshop on Parallel and distributed simulation.* 82-89. IEEE Computer Society.

Chaturvedi, A., A. Mellema, S. Filatyev, and J. Gore. 2006. "DDDAS for Fire and Agent Evacuation Modeling of the Rhode Island Nightclub Fire." In *Computational Science – ICCS 2006*, edited by V. N. Alexandrov, G. D. v. Albada, P. M. A. Sloot, and J. Dongarra, 433–439. Berlin: Springer Berlin Heidelberg.

Cheng, C., R. Jain, E. Van Den Berg. 2003. *Location Prediction Algorithms for Mobile Wireless Systems.* Wireless Internet Handbook: Technologies, Standards, and Applications.

Eeckhout, J.. 2004. "Gibrat's Law for (All) Cities." *The American Economic Review*, Vol. 94, No. 5.

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

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

Kiesling, T., and J. Luthi. 2005. "Towards Time-Parallel Road Traffic Simulation." In *Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation* (PADS '05). IEEE Computer Society, Washington, DC, USA, 7-15.

Krumm, J.. 2006. "Real Time Destination Prediction Based On Efficient Routes." SAE Technical Paper.

Laasonen, K.. 2005. "Route Prediction from Cellular Data." In *Proceedings of the workshop on Context Awareness for Proactive Systems*, pp. 147-158.

Madey, G. R.., G. Szabo, and A.-L. Barabási. 2006. "WIPER: The Integrated Wireless Phone Based Emergency Response System." In *Proceedings of the 2006 International Conference on Computational Science*. 417–424.

Madey, G. R., A.-L. Barabási, N. V. Chawla, M. Gonzalez, D. Hachen, B. Lantz, Pawling. A, Schoenharl. T., Szabo. G., Wang. P.,and P. Yan. 2007. "Enhanced Situational Awareness: Application of DDDAS Concepts to Emergency and Disaster Management." In *Proceedings of the 2007 International Conference on Computational Science*. 1090–1097.

Marin, M., V. Gil-Costa, C. Bonacic, and R. Solar. 2013. "Approximate Parallel Simulation of Web Search Engines." In *Proceedings of the 2013 ACM SIGSIM conference on Principles of advanced discrete simulation*, pp. 189-200. ACM,.

Nagel, Kai, and M. Schreckenberg. 1992. "A cellular automaton model for freeway traffic." *Journal de physique I* 2. No. 12 . 2221-2229.
NGSIM Community. 2014. NGSIM Community Home. Accessed May 13th. http://ngsim-community.org.
Sinclair, A. J., R. J. Prazenica, and D. E. Jeffcoat. 2008. "Optimal and Feedback Path Planning for Cooperative Attack." *Journal of Guidance, Control, and Dynamics*, Vol. 31, No. 6. 1708-1715.
Voronkov, A., and F. Darema. 2004. "Dynamic Data Driven Applications Systems: A New Paradigm for Application Simulations and Measurements." In *International Conference on Computational Science*. 662-669.
Ye, T., H. Kaur, S. Kalyanaraman, and M. Yuksel. 2008. "Large-scale Network Parameter Configuration using an On-line Simulation Framework." *IEEE/ACM Transactions on Networking*. Vol. 16. 777–790.

AUTHOR BIOGRAPHIES

PHILIP PECHER is a PhD student in the School of Computational Science and Engineering at the Georgia Institute of Technology. He received his B.S. in Industrial and Systems Engineering from the same university. His email address is philip161@gmail.com.

MICHAEL HUNTER is an Associate Professor at the School of Civil and Environmental Engineering at the Georgia Institute of Technology. He received his Ph.D. in Civil Engineering from The University of Texas at Austin. His email address is michael.hunter@ce.gatech.edu.

RICHARD FUJIMOTO is Regents' Professor in the School of Computational Science and Engineering at the Georgia Institute of Technology. He received a Ph.D. in Computer Science and Electrical Engineering from the University of California-Berkeley. His email address is fujimoto@cc.gatech.edu.