Using a Visualization Based Simulator to Test Location Inference Techniques on Movement Patterns

Jon Froehlich and **Jonas Klink**

Department of Computer Science
University of Washington
Seattle, WA 98195
{jfroehli, jklink}@cs.washington.edu

Abstract

The evolution of GPS into the civilian world ushered in a new area of high fidelity, location-sensitive consumer electronics. Current, pervasive technologies such as the cell-phone and PDA provide a rich platform for location-aware applications and many can be purchased with GPS capabilities. Following this trend, recent research has investigated how movement patterns can be learned and future actions can be predicted. In this paper, we describe a novel simulation tool, called *Simulöc,* which provides a visualization framework for testing location-based inferencing algorithms. *Simulöc* combines data about a person's known schedule, bus activity, current location, and past movement history to learn movement patterns and predict future destinations. The visualizer uses global positioning data along with satellite imagery to display simulated user movement and bus activity. This tool is then used to evaluate the performance of three known artificial intelligence inferencing techniques: a naïve first-order Markov model, a first-order Markov model with a uniform prior, and a naïve Markov model with time dependency.

Introduction

At a superficial level, movement often appears to be random – objects move about with seemingly little or no pattern. This feeling of randomness, however, is a matter of perspective - it can be reduced through a combination of contextual knowledge and historical precedence. How is it, for example, that for no obvious reason our friend's cat decides to jump down from the kitchen window sill and run under the living room table whenever we arrive for a visit? Without historical precedence, one might infer that my cat is simply skittish. However, with a certain amount of prior observation, a keen eye might begin to notice the following pattern: The cat, as it so happens, is particularly fond of greeting people; however, she also tends to be quite cautious. Upon seeing someone reach the front door, the cat runs to her shielded observation point under the table (which provides a perfect lookout). Given this knowledge, it seems reasonable to claim that if the cat is on the window sill, her likely next location will be under the living room table -- particularly if visitors are expected.

Learning movement patterns and making predictions like the one above is a particularly rich field in artificial intelligence, as it includes many AI mainstays such as learning and inferencing, pattern recognition, and decision theory. Mid to large scale location-inferencing research is often difficult to quickly test and evaluate as it requires the researcher to move about a locale. This makes location-related AI research a rather fertile ground for simulation.

Our goal in this initial research was to, first, construct a visual simulation environment for testing multiple location-based inferencing techniques and second, use this environment to evaluate the performance of known inferencing algorithms on location prediction and movement behavior. With the outcome of our tests, we hope to pursue our original vision of using location inferencing and movement pattern detection to build a mobile application that supplies the user with contextually useful data, such as the proximity and location of bus stops that serve the user's bus line given the user's current location.

This system would work by aggregating data from a user tracked via GPS (Global Positioning System), with real-time bus data gathered from MyBus.org[1]. The MyBus.org service provides constantly refreshed information on the predicted arrival times (delayed, early, or on time) of any bus on any Seattle bus line. The algorithm used is described in Cathey and Dailey 2003. As we're not actually building a contextually aware mobile application in this project, but rather simulating a world to test the viability of one, the bus data and GPS data are automatically generated by the simulator.

The problems specifically addressed in this simulation world project are: first, can we predict movement and thereby possible destinations of our user. If so, how can we present them with contextually useful information on the

[1] http://www.mybus.org

current status of the nearby bus net to the user. *Contextually useful* and *nearby* are determined by the learned user movement pattern, combined with information on preferred bus routes (the bus routes the user takes most often) and proximity to bus stops in which suitable buses will reach at a convenient time for the user.

Data gathering and technology

Ideally, one would like to implement and test the above described tool in and for the real world. However, accurately gathering sufficient data from the amount of users we need for constructing our application would require much more time than given to this project. Therefore, we made some simplifying arrangements and assumptions, to make data gathering and testing run swiftly.

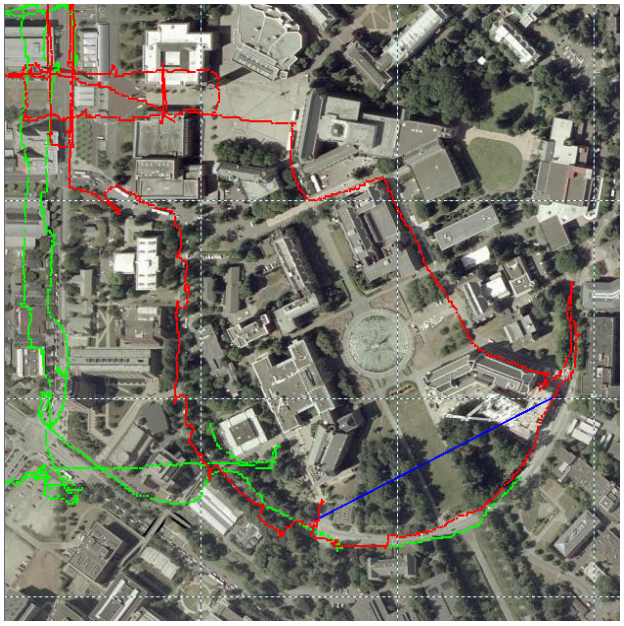


Figure 1. Visualization of real data collected with the GeoLogger and displayed using *Simulöc*

Collecting GPS data

For the setting of this project, the need for accurate GPS data, spanning over the entire campus was essential. Also, mappings between specific locations (e.g. bus stops) and their GPS coordinates were also needed. Systems for collecting and predicting these data are presented in several other contemporary research papers (Ashbrook and Starner 2003; and Liao *et al.* 2004), but for the scale and simplicity of this project, we chose to do a direct mapping by creating databases holding the needed information.

TerraServer. To collect sufficient data of high quality, we used two different sources, both with their own advantages (as well as limitations). The first one, TerraServer[2], is an online database operated by the Microsoft Corporation, providing search options through archived satellite images. The TerraServer-USA Website (which was used in this project) is one of the world's largest online databases, providing free public access to a vast data store of maps and aerial photographs of the United States.

TerraServer is also said to be designed to work with commonly available computer systems and Web browsers over slow speed communications links. Although those restrictions were not an issue in the setting of this project, it provides a valuable opportunity for running mobile applications with TerraServer as well.

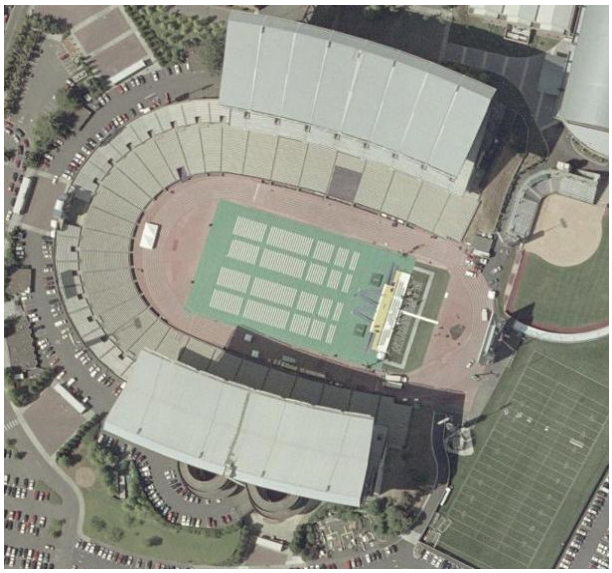


Figure 2. TerraServer high resolution satellite image, showing Husky Stadium.

The high fidelity of the satellite photography available through TerraServer provides a convincing argument for the viability of integrating satellite imagery into current map systems. The resolution of the picture shown in Figure 2 is set to one meter per pixel. The maximum available resolution TerraServer is four times that (0.250 meters per pixel).

The search option is not the only way of retrieving information from TerraServer, but image retrieval and GPS coordinate mapping can also be done via easy web services provided by Microsoft. An extension of our product to an online version is therefore feasible, but in our current version we have included support for generic satellite images ourselves.

[2] http://terraserver.microsoft.com/

Figure 3. Displaying TerraServer satellite images in *Simulöc*

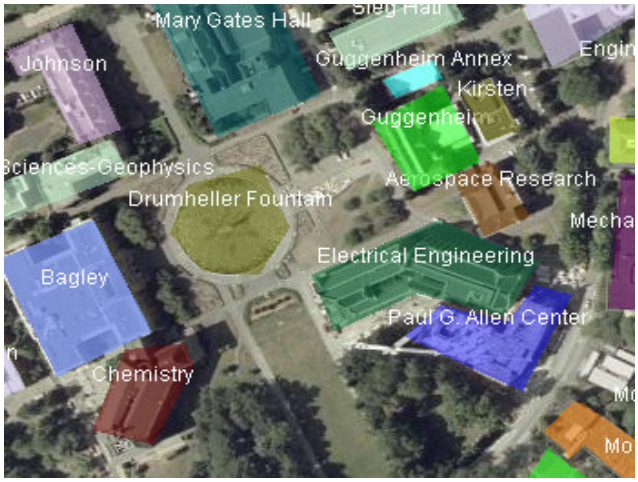


Figure 4. Displaying TerraServer satellite images (with labels and buildings highlighted) in *Simulöc*

GeoLogger. Though we used TerraServer to map building GPS coordinates, a tool for making more precise measurements was needed. For tracking down the exact GPS location of all bus stops included (27 in total), we used the wearable GPS-tool of GeoLogger[3]. The GeoLogger consists of an antenna and a logging unit, powered by either a rechargeable or an internal 9V battery. The flexibility of this tool provides high freedom of movement, and the only additional information needed for establishing connection between the exact location and its GPS coordinates, is a manual recording of the time at the wanted location (e.g. at a bus stop). Since the logger is continuously recording data (with a one-second frequency), these recorded points also provided us with an exact walk over campus. This data could then be loaded up into *Simulöc* to verify both the implementation of our tool as well as the GeoLogger. For coordinates not available to us with the GeoLogger, we used TerraServer data.

Simulöc Visualizer

Simulöc combines data about a person's known schedule, bus activity, current location, and past movement history to learn movement patterns and predict future destinations. The visualizer uses global positioning data along with satellite imagery to display simulated user movement and bus activity. This simulation system grew over the course of its development to be a rapid experimental environment, which allows for the implementation, testing and visualization of many different AI methods for location context-awareness and computations on movement patterns.

[3] http://www.geostats.com/products.php

Data gathering and visualization

Combining our two GPS sources, the tool for data visualization provides an accurate and descriptive way of presenting the different data used. The interface provides the user with the opportunity of displaying any satellite image. By naming the image according to our name principles (let the name consist of the four GPS coordinate pairs for the corners of the image, e.g. TL(-122.31281,47.65694), BL(-122.31291,47.64974), TR(-122.29949,47.65686), BR(-122.29959,47.64966)1.jpg). This way, the application is flexible enough to handle any image, while still providing the correct GPS coordinates for every point on the map, using our mouse-over information display tool. The satellite images of the University of Washington campus used in this setting are actual images, collected from the TerraServer database.

Also presented on the visual map are our own bounding boxes for all buildings on campus. These bounding boxes can individually be switched on and off and the highlighting scheme changed, to make the visual freedom as high as possible and configurable for every user and project setting. Also, the names of buildings and bus stops are printed, to make the visualization of the simulation and inference process easily available to any user.

Simulation

One of the most important parts in an artificial setting like ours is that the simulation closely matches real scenarios. If not, the confidence put to the test results cannot be very high, when used as an early evaluation for any real-world application.

For accurate modeling of the real world, we conducted several interviews with students, to correctly get their average weekly schedules; containing information about

when they get to and from campus, what bus routes they ride, and the weekly schedules (on a five day basis). Information gathered from these interviews was used to structure our database of several schedule files, with the capability of adding an infinite number of individuals

Busstop and busline data is gathered and parsed from the Seattle area bus transportation website (http://transit.metrokc.org). This information is then used to accurately simulate the arrival and departures of buses through campus.

The person activity schedule files are read by the simulator, and their corresponding movement patterns are displayed on the visualization screen. The addition of uncertainty, using randomness, is done in two ways. Firstly, randomness is used in this setting to illustrate the fact that people do not always strictly follow their schedules. Secondly, a person might (with a certain probability) change his mind in midpath, and update his goal to be another location. To make the simulation more realistic, the new target goal must be within 500 ft. of the person's current location. Even though these rules for modeling accurate behavior are very simple, the outcome can be viewed as most realistic.

Learning movement patterns

One of the most critical features of any location agent is the ability to keep track of the current state; such is especially true in an environment such as ours. The world we operate in contains an uncertainty as to where people will be heading next, which is connected to the learning problem of which bus stop and bus line will be accessed next. At best, these constraints on future information available only allow the agent to obtain a probabilistic assessment of the current state (and which are the plausible next ones).

Movement patterns and uncertainty

The inference process is simplified by the fact that our problem is set in a highly scheduled environment (on campus), so we implemented the possibility of random actions amongst the modeled people. Walking is most often not random by itself, but rather goal-directed. Introducing the fact of instant changes of goal for the walk (e.g. the sudden realization of you left your umbrella in a building you visited this morning), however adds an element of randomness and uncertainty to our setting. Also, the fact that users might be at different locations at given times each week (due to the instant decisions of changing movement pattern for a period of time), the introduction of the bus system further adds to the uncertainty. For dynamic worlds, such as the campus environment, we need a model that can handle this uncertainty in future observations.

Markov processes

To simplify the inference process, it is common to make the assumption of the world being looked upon as a series of snapshots, or *time slices*, wherein each time slice is treated as a time-state in the current world. Each state contains a number of stochastic variables, of which some are directly observable (such as our current location) and some are partially or totally unobservable (such as the next movement action in a random system). The observability of each variable can change over time (a fact that is often ignored for simplicity).

However, there are some additional problems with this model. Due to the fact that the number of variables are unbounded (since they exist for every time slice, in an infinite time space), we have conditional probabilistic dependency on a number of parents that is very hard to deal with (infinitely many). The solution to this is two-fold, and includes the concepts of *stationary processes* and the *Markov assumption*. These assumptions simplify our model to only depend on laws that do not change over time and to only depend on the directly previous state, respectively. This type of Markov process is called a *first-order* Markov process (Markov Chain), and is written as

$$P(X_t|X_{0:t-1}) = P(X_t|_{t-1})$$

The general Markov process is a system that can be in one of several (numbered) states, and can pass from one state to another each time step according to fixed probabilities. If a Markov system is in state i, there is a fixed probability, p_{ij}, of it going into state j the next time step, and p_{ij} is called a transition probability.

A Markov system can be illustrated by means of a state transition diagram, which is a diagram showing all the states and transition probabilities (see Figure 5) below (notice that transitions with zero probability have no arc).

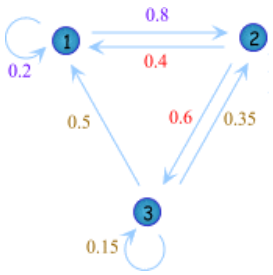


Figure 5. Transistion diagram

The same diagram can also be modeled by the matrix P, whose ij[th] entry is p_{ij}, Matrix P is called the transition matrix associated with the system. The entries in each row add up to 1. Thus, for instance, the above transition diagram would be modeled in its matrix form as in Figure 6 below.

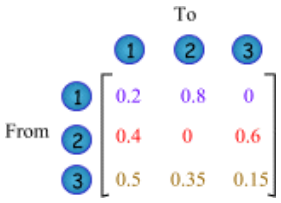


Figure 6. Transition matrix

A simple example. Let us now illustrate the use of Markov processes with a simple and informal example. A Markov model could look at a long sequence of rainy and sunny days, and analyze the likelihood that one kind of weather gets followed by another kind. Let's say it was found that 25% of the time, a rainy day was followed by a sunny day, and 75% of the time, rain was followed by more rain. Let's say we found out additionally, that sunny days were followed 50% of the time by rain, and 50% by sun. Given this analysis, we could generate a new sequence of statistically similar weather by following these steps:

1. Start with today's weather.
2. Given today's weather, choose a random number to pick tomorrow's weather.
3. Make tomorrow's weather "today's weather" and iterate, starting each iteration at step 2.

What we would get when the above algorithm is applied to the probabilities stated above, is a sequence of days like: Sunny, Sunny, Rainy, Rainy, Rainy, Rainy, Sunny, Rainy, Rainy, Sunny, Sunny,… and so on. In other words, the "output chain" of the Markov reasoning would statistically reflect the transition probabilities derived from weather we observed. Such a Markov Chain, while similar to the source in the small, is often nonsensical in the large (which is why it is a terrible way to predict weather). That is, the overall shape of the generated material will bear little formal resemblance to the overall shape of the source, but taken a few events at a time, this simple model seems to work.

Higher Order and Hidden Markov Models

Whether the application of a first-order Markov process is appropriate to a problem setting depends on the domain itself. The loss of accuracy (through the assumption of the most previous state containing all needed information) can be remedied by increasing the order of the Markov model (that is, adding more parents into the dependence of the past). Increased accuracy can also be achieved by adding more state variables; depicting the fact that the process depends on additional variables, previously unaccounted for.

Hidden Markov Model (HMM). In practice, there is a need for yet another kind of Markov Model, called Hidden Markov Model. The HMM illustrates the fact that the actual states of the world is not directly observable, but only indirectly do we interact with the states through our observations. Some (or all) of the states are therefore considered “hidden” from the external observer.

There are a number of variations on HMM problems, e.g.

1. The number of states and transition probabilities are known
   a. Given data, find the optimum (most likely) position of the change points.
   b. How precisely should the points be stated?
2. The number of states is known, but the transition probabilities are not.
   a. Estimate the transition probabilities.
   b. What accuracy is appropriate for the estimates?
3. The number of states, and the architecture, are unknown.
   a. Find an HMM which models the data "well".
   b. The simplest model has one state, the most complex model has one state per data value; almost certainly neither extreme is justified. Quantifying model complexity is therefore a crucial issue.

Movement patterns and Markov Models

For the specific setting we are facing in this paper, there are some arguments concerning which model is needed for accurately learn and predict the behavior in our system. As stated earlier, this decision depends highly on the domain the prediction is taking place in.

Turning our attention towards the real setting of the problem we are addressing, we see that it contains many challenges for the AI used. To be able to correctly infer movement patterns and make predictions on behavior based upon those, we need a model that addresses all the problems of the real world. The challenge here though, lies in the fact that the real world is not perfect: with noisy data, randomness, and other factors in play. In the span of this mini project, we have had no opportunity to collect real data over a long period of time, and also no “real” persons to follow during the same period. A simulator was therefore constructed for the sake of this project, and it allowed us to effectively control our world and running a sizeable amount of experiments, without losing connection to the real world. Fully accounting for all difficulties of the real world is a challenging problem, as the use of a three-level dynamic Bayesian network in (Liao *et al.* 2004) shows.

Our goal throughout the course of this project has been to model the real world as accurately as possible in our simulator, but still some limiting assumptions has have been made, thereby affecting our choice of process model. Firstly, since we are moving in a simulated world, there is no real noise on the data. In the real world, using the GPS system, there are of course several ways noise could be introduced into the data. If, for instance, the signal will beis lost for a while (traveling in a tunnel, or entering a building), only low credibility can be assigned to the data point directly before and directly following the loss of the signal. Also, as with all electronic devices, the reception of the logger used is somewhat affected by magnetic fields in the surrounding world.

In our simulated world, we are still using collected real GPS data points, but with the simplification that they are static, not dynamically generated (although our system is fully capable to handle dynamic generation of data points as well). This simplification reduces the amount of uncertainty we have to take into account. Still, the randomness existing through the user's sometimes instant change of goals, forces us to use process for probabilistic learning and reasoning.

When turning to the question on of whether or not or problem setting qualifies as an instance of the Hidden Markov problem domain, let us reason about what information we have available. What we need, to correctly predict the movement patterns, is to visit every location possible (including bus stops and buildings) and the transition probability of taking the transition from a given location (state) to the next. As we have done extensive mappings all over campus (and also all locations in GPS coordinates are available), the information for the first part of the above problem formulation is certainly satisfied. Unfortunately, we are missing the transition probabilities, with which the user will choose which state to move to from the current location. Therefore, our problem setting falls under point 2, on the listing on the previous page; the transition probabilities need to be estimated.

Are the states of the Markov Model for the setting discussed in this project then really hidden? Are they not all known, and fully accessible by the GPS coordinates or the location name? All of the previous reasoning is true, but as pointed out by (François *et al.*), any given state can be on several paths through the world, and the states are therefore in practice hidden (although they can be deduced form observation sequences).

Implementing Markov Models

As our framework is designed to handle and visualize any inference algorithm, we started out by running several tests on the more basic Markov Models. The variants implemented are both derived from the simple first-order form, where dependence on parent nodes other than the directly previous one, is assumed to be non-existent. These models are very well suited for quick implementation, but can still generate some interesting results and find possible flaws in our design.

The implementation is, as hinted suggested earlier, pretty fairly straight-forward, and relies on two simple formulas derived from a more general one. The goal of evaluation, using a Markov Model, is to estimate the probabilities of seeing a certain output sequence. In the first-order case, the probability of arriving in a state only depends on the most recent state before. The formula below sums up these observations, and it's states using the fact that the probabilities of seeing a certain sequence as time goes from 1 to T (left side), is the probability of seeing the initial state, $p(s_1)$, multiplied by the product of the conditional probabilities of ending up in s_t given s_{t-1} (a transition probability).

$$p(\{s_t\}_1^T) = p(s_1)\prod_{t=2}^{T} p(s_t \mid s_{t-1})$$

Assuming that we have the probability of s_1 being our initial state; $p(s_1)$, we only need to model the conditional probabilities to the far right. We used two different methods for establishing these probabilities, to then be able to evaluate the probability of seeing the entire sequence. The easiest way of doing this is to just perform a count; a count which gives us the number of times a given output B is observed together with its input A, divided by the number of times we observed A. The formula for this is given below.

$$p(B \mid A) = \frac{\sum_{\textbf{states}} occurrences\ of\ AB}{\sum_{\textbf{states}} occurrences\ of\ A}$$

The formula above illustrates what is called the Naïve first-order Markov Model. To do better, we will include the addition of a 1 in the numerator, and the addition of the number of symbols available in the system, N_{symbols}, to the denominator. By doing this, we add the notion of a uniform prior over the transition probabilities. The enhanced formula will therefore be as presented below.

$$p(B \mid A) = \frac{1 + \sum_{\textbf{states}} \#AB}{N_{\textbf{symbols}} + \sum_{\textbf{states}} \#A}$$

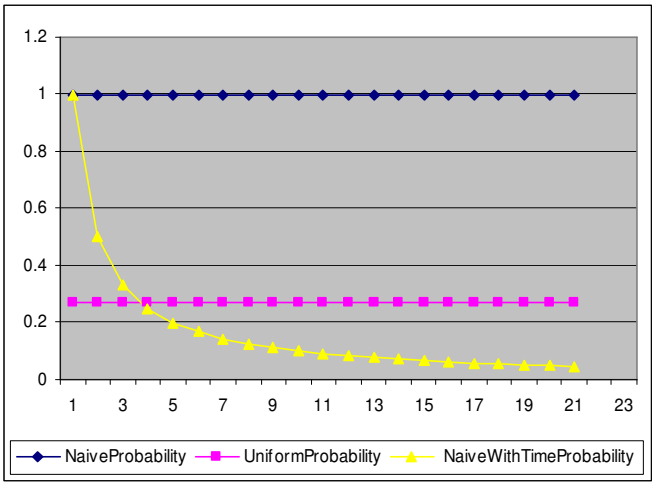
Testing and results

The extensive work on the visualization part proved to be highly interesting, and the testing of our implemented framework for inference proved to be equally so.

The settings for our experiments were as follows. Having implemented the two first-order Markov Models above into our framework, there were some interesting tests to run; for proving the accuracy of our system, but also to evaluate the effect of time and randomness on the inference process.

For testing purposes, we ran three different Markov Models on the test data; the first-order naïve model, the first-order model with uniform prior and also the naïve one with a time dependency (that is, what minute in the week the action was taken was added as a parent).

For the first setup of tests, we ran the models on a simplified schedule file, which held the same day schedule for every day of the week. Also, randomness was excluded from the system. By creating the simplest available setting for the models, we wanted to first establish that the models performed accurate predictions. Indeed, all three models got a perfect score of predicting where the person would move next. These results are verified in the graph below, where all three models can be seen converging towards the result that gave them, individually, the optimal probability setting for doing perfect inference.

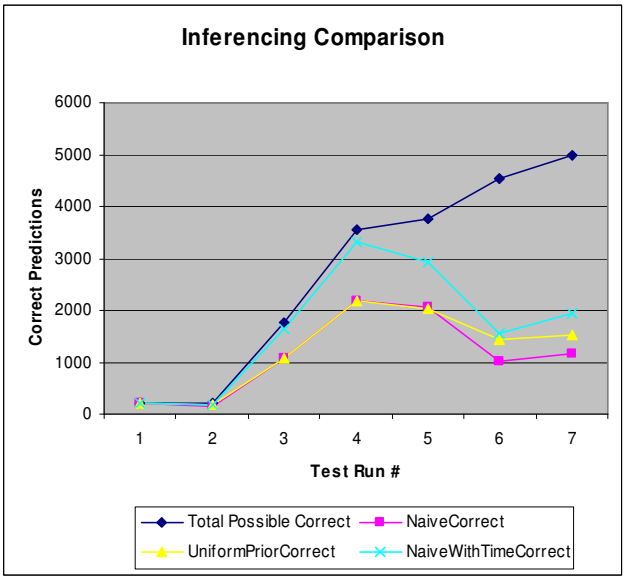


In the second setting, we introduced the models to a more complex schedule, but still without random goal changes in the simulated world. This setting was then used in running all three models, over a simulated one and a two year time period. The distinction between the simpler and the more complex models now became very clear. The importance of including the additional information of departure time (in the naïve model) also became clear. The two first-order models both dropped to an accuracy of 61% correctly inferred transitions. The time dependant model however remained on a high accuracy of 92%, while being tested on about 3500 path predictions in the two year setting.

Test Run Description	Total Possible Correct Pred.	Correct Pred. by Naïve Model	Correct Pred. by Naïve w/ Uniform Prior	Correct Pred. by Naïve w/ Time Model
1. Uniform daily activity schedule; no randomness in simulation; ran for one simulated month. Used to verify correctness of learners.	199	199 (100%)	199 (100%)	199 (100%)
2. Uniform daily activity schedule; 5% goal randomness 0.01% in-route goal randomness; ran for one simulated month	212	140 (66.0%)	175 (82.5%)	166 (78.3%)
3. 5-day daily activity schedule; no randomness; ran for one simulated year	1776	1082 (60.9%)	1078 (60.7%)	1640 (92.3%)
4. 5-day daily activity schedule; no randomness; run for two simulated years	3551	2181 (61.1%)	2176 (61.2%)	3310 (93.2%)
5. 5-day daily activity schedule; 5% goal randomness 0.01% in-route goal randomness; ran for two simulated years	3749	2066 (55.1%)	2044 (54.5%)	2936 (78.3%)
6. 5-day daily activity schedule; 30% goal randomness 0.01% in-route goal randomness; ran for two simulated years	4542	1012 (22.28%)	1429 (31.46%)	1540 (33.9%)
7. 5-day daily activity schedule; 5% goal randomness 0.5% in-route goal randomness; ran for two simulated years	4988	1157 (23.9%)	1537 (30.81%)	1926 (38.6%)

For our two measures of randomness: changing a goal randomly from a given start location and turning randomly in midroute, we introduced small values of 5% and 0.01%, respectively. This setting was now evaluated for all three Markovs, over a simulated one month period and then over two years. For the shorter time period, the uniform prior model dominated the other models, with 82% accuracy, compared to 66% and 78% for the naïve and the naïve with time dependency, respectively. Over the longer time period of two years, the time dependent one widely surpassed the less complex ones, with 93% compared to 61% for both. These results can be explained by the fact that over a short time period, the time dependent one have not seen enough examples of randomness, and will score less over a longer time period.

However, when either one of the random values was increased by much (to 30% and 0.05%, independently), the performance of all three models drops sharply to 22% for the naïve model, 30% for the model with uniform prior and, on average, 36% for the naïve with time dependency.



Clearly, the results achieved through this testing (which was set up more as a verification of the visualization and inference framework), were not overly surprising. All three models were able to perfectly classify a repeated day schedule with no randomness, even in a short simulation time period of 1 simulated month. As the complexity of the schedules increased, the time-dependent first-order Markovian left the other two models far behind. This pattern was even more obvious when a small amount of randomness was included in the setting, and a longer time period given to learn the transition probabilities. Concluding, however, that all three systems did poorly when randomness was brought up to a more significant level, shows us the need for stronger models in such an unpredictable environment.

Future work

Tis section introduces the reader to some of the concepts that can be added to extend and further evaluate the performance of *Simulöc Visualizer*.

To add an extra measure of reality to the simulation, there are a few things that could be extended. Given more time, another natural feature to introduce would be that of *routing*. Routing in an environment such as the one *Simulöc* is working in basically means avoiding obstacles (at the simplest level). To extend the visual accuracy, the fact that the people walking around should be avoiding larger obstacles is of course obvious. An additional benefit of this extension is also an improvement of the walking time elapsed; making it more realistic.

A limitation of the GPS system is its inability to record data information at all times. When losing the signal, e.g. when entering a building or tunnel, no data can be recorded. Also, data recorded just prior to and directly after entering and exiting the building cannot be fully trusted. These are just limitations and sources of error directly associated with user location and movement, and a full discussion on common errors is given in Arpin 2003. Though GPS systems are widely promoted as the ultimate navigation and vehicle-tracking tool, GPS systems are in fact fragile, prone to error, easily disabled, and best suited for navigation purposes only. Since a satellite is orbiting at a 20.000 km distance from the objects it is currently tracking, errors due to receiver noise, atmospheric issues and orbital miscalculations can and do commonly happen.

To overcome the somewhat limited trustworthiness of the GPS system, an effort to introduce a compliment system to our current one must be made. The fact that the current simulation takes place on the University of Washington campus, introduces some interesting possibilities. Available on the UW campus, the *Place Lab* project provides an alternative way of collecting data for accurate positioning. *Place Lab* is a software alternative, providing device positioning for location-enhanced computing applications. The software provides positioning both indoors and out, which would make it suitable for an application like the *Simulöc*.

The *Place Lab* approach is to allow commodity hardware clients like notebooks, PDAs and cell phones to locate themselves by listening for radio beacons such as 802.11 access points, GSM cell phone towers, and fixed Bluetooth devices around us in the environment. These beacons all have at least semi-unique IDs, for example, a MAC address. Clients compute their own location by hearing one or more IDs, looking up the associated beacons' positions in a locally cached map, and estimating their own position by triangulation.

These usual approaches to solve the problem of lost signal when moving indoors, is by assigning it a state, say BUILDING (Liao *et al.* 2004), and treat this as a location. A more intelligent approach would be to make use of the *Place Lab* system for continuously tracking the user inside the building. This approach presents us with several advantages; 1) We have a compliment for verifying GPS data outside. 2) We have an opportunity to treat a BUILDING not only as a whole location, but also divide it further into sublocations. This way, users such as the cognitively challenged can use the continuous benefit of any location-aware application, both outdoors and indoors. 3) We can make optimal use of the highest quality data

available, and will not suffer from the loss of accurate data in moving from inside to outside and vice versa.

Another point we want to stress as important future work, is the implementation and testing of additional inference techniques in our system. The framework created in *Simulöc* makes the efficient implementation and testing easy for any possible inference model. While we only implemented a few simple inference models, more advanced models exist and can be implemented and plugged in for simulation and visualization.

In the current version of *Simulöc*, the user will be provided with an aid as to their navigation on campus. Buses are available in the simulation, but not included as an option for randomly changed goals. Given more time, we would like to extend the current functionality to include the bus stops in the simulation. This will give us the added functionality of being able to make recommendations for which bus stop to go to and which bus to take, given the Mybus.org times and the preferred user bus lines. These preferences can also be learned, using an inference model of the same type as for the persons' schedules.

The last two issues we have implemented, but want to extend, is the use of larger maps and more actual people to provide test results. In the current version of *Simulöc*, maps can easily be interchanged (as described earlier), but providing the user with an easy scrolling opportunity between satellite images displayed on the interface, would be optimal. Even though extensive interviewing was conducted on schedules, we would like to provide the user of *Simulöc* with a larger schedule database, with classification for testing persons of different behavior and movement pattern in the application.

Related work

The field of context-awareness in software agents, combined with the task of localization and tracking, has been a focus of interest in quite a few papers in recent years. The value of a localization aid, for everybody as well as those with special needs, is a topic found interesting by many (in academic environments, as well as industry). We present here a brief summary of some of the more important works; in terms of the impact it had on our project, but also the value they hold for future research and development in the field.

The aspect of aid of the kind of products discussed in this paper, comes to a most important use in the case of helping cognitively disabled (Patterson *et al.* 2002). As an example of such individuals, the world's ever-increasing number of Alzheimer's patients provides a target group, where context-aware agents are certainly useful. The paper of (Patterson *et al.* 2002) stresses the importance of active agent-intervention in times of challenge or confusion for the patient, and has its focus on from the background identifying occurring abnormal (and potentially harmful and confusing) behavior. The balance of missing potentially dangerous activities, while not disturbing the user with unwanted warnings, is a task that requires solid inference from learning, and even more so while predicting future behavior as we do. Combining the positive aspects of awareness with that of prediction accurately could enhance the aid for cognitively disabled greatly.

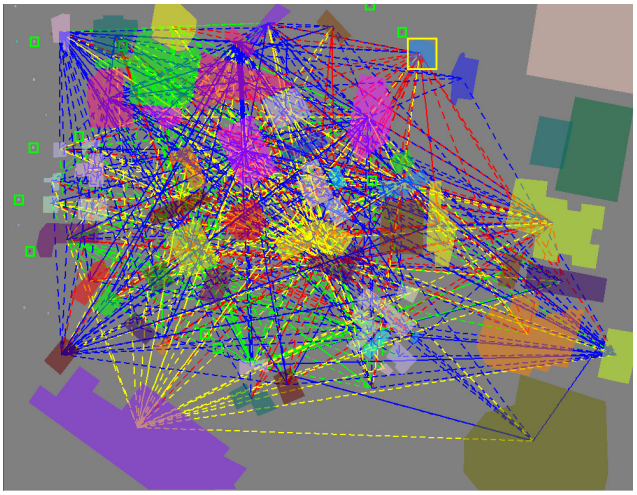


Figure 7. *Simulöc* visualization showing movement patterns after two simulated months with four people

Many techniques have been suggested to accomplish the task of tracking and learning a person's behavior, using his/her location at given times. The abundant source of information from GPS is used in a similar way to ours in (Liao *et al.* 2004). The setting is that of a hierarchical Markov model for learning the movement pattern of the user, and applying the filtered GPS data to an underlying graph of roads, intersections and transfer points. The system provides astonishingly accurate predictions of behavior (98% correct after training on 30 days of everyday movement). While the importance of such a system for future research is great, (Liao *et al.* 2004) also presents a way of detecting user errors, by tracking two models; one based on learning and the other one on potential, but probabilistically less likely, actions. When the user takes an "unexpected" action, the likelihood of the potential model is increased. While our model also includes leaning under the presence of seemingly random behavior, the tracker system suggested can in a more developed form hold great impact on a system such as ours.

Turning to other applications of the context-aware product, the paper of (Starner and Ashbrook 2004), describes its usefulness in both single-user and multi-user contexts (but focuses on the latter). In moving from single-user applications, (such as e.g. reminders of various tasks and

transmission optimization), to the multi-user domain, the potential use for location-based learners is huge. In joint interaction with both environment and other users, a vision on a society of greater social interaction and mutual agreeability is close at hand. The paper introduces an interesting notion of using the movement patterns to learn *places*, defined as all geographical points where the user stays a certain amount of time, and then using additional data for clustering correlated *places* into *locations* and *sublocations*. This way, no information is needed before (as our predefined buildings), and the system makes efficient use of all data available. While spending much time on this building of the inference base, this idea can be applied to previously unknown environments, and combined with the movement data of other (maybe local) users, to give a heuristic for the prediction algorithm in our system. Also, the movement pattern of others learned in our system, could be used as a solving base for problems in a highly scheduled environment (like ours on campus), such as quick adaptation to drastic schedule changes from semester to semester.

The use of context-awareness in a mobile setting using cell phones is discussed in François *et al.* 2004. The use of HMMs in a similar way to ours is here the basis of an application for allowing for smooth handovers of mobile hosts (MHs) between access points (APs). The Hidden Markov Model is used for modeling and learning the predicted path a user is on. Thereby, earlier notification to neighboring APs can be sent out, notifying that a user will be accessing them soon. This adds another layer to the context-awareness, by allowing for a conditional use and transformation of information between internal states, rather than being output to an external viewer. For a system like ours, such an idea could be used in the sense of internal "communication" between locations, with the intention of e.g. preparing the potentially following location for the arrival of one or many individuals.

Conclusion

In conclusion, we've shown how the *Simulöc* simulation system can be used to investigate various location-based inferencing models. *Simulöc,* as a visualization system, is particularly useful for researchers to observe differences in their programmed inference models (as the predicted locations as well as the actual locations are drawn on the screen). We used three different Markov models to validate *Simulöc* framework, one of which – the naïve Markov model with time dependency – was shown to work better than the others.

Contributions

1. Jon Froehlich: Designed and wrote *Simulöc* including the tests to evaluate our three Markov inference models. Assisted with the paper (wrote abstract, edited paper).

2. Jonas Klink: Primary author on most of the paper. Helped design Markov model algorithms for predicting location. Created Markov model diagrams and equations.

External Code

All code was written from scratch for this project except for:

- The Vector2D class which was previously written by Jonas and then updated by Jon.
- The four Polygon2D classes which are open sourced by John Reekie at the University of California.

Acknowledgements

We wish to acknowledge the contributions of Lin Liao and Donald Patterson of the University of Washington to our efforts. Without their help in providing valuable advice and means for collecting essential data, this project would never have been possible to realize. We would also like to acknowledge Kevin Wampler for his assistance with vector and geometry math.

References

Starner, T.; and Ashbrook, D. eds. 2003. *Using GPS to Learn Significant Locations and Predict Movement Across Multiple Users.* Retrieved December 12th, 2004, from: http://www.cc.gatech.edu/ccg/publications/persubi2003.pdf

François, J-M.; Leduc, G.; and Martin, S. eds. 2004. *Learning movement patterns in mobile networks: a generic method.* Retrieved December 12th, 2004, from: http://research.ac.upc.es/EW2004/papers/55.pdf

Liao, L.; Fox, D.; and Kautz, H.; eds. 2004. *Learning and Inferring Transportation Routines.* Retrieved December 12th, 2004, from: http://www.cs.washington.edu/ai/Mobile_Robotics/postscripts/gps-aaai-04.pdf

Patterson, D.; Liao, L.; Gajos, K.; *(et al.)* eds. 2004. *Opportunity Knocks: a System to Provide Cognitive Assistance with Transportation Services.* Retrieved December 12th, 2004, from: http://www.cs.washington.edu/homes/djp3/AI/AssistedCog

nition/publications/UBICOMP/2004/UBICOMP2004Patterson.pdf
In *Proceedings of the Sixth International Conference on Ubiquitous Computing (UBICOMP '04)*

Patterson, D.; Etzioni, O., Fox, D.; and Kautz, H.; eds. 2002. *Intelligent Ubiquitous Computing to Support Alzheimer's Patients: Enabling the Cognitively Disabled.* Retrieved December 12th, 2004, from: http://www.cs.washington.edu/homes/djp3/AI/AssistedCognition/publications/UBICOMP/acubi.pdf
In *Proceedings of the First International Workshop on Ubiquitous Computing for Cognitive Aids (UBICOG '02)*

Rabiner, L.R. eds. 1989. *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*. Retrieved December 12th, 2004, from: http://ieeexplore.ieee.org/iel5/5/698/00018626.pdf
In *Proceedings of the IEEE, 77(2):257-282 (IEEE '89)*

Cathey, F.W.; and Dailey, D.J. eds. 2003. *A prescription for transit arrival/departure prediction using automatic vehicle location data*.
Retrieved December 12th, 2004, from: http://www.its.washington.edu/pubs/trc2003.pdf
In *Transportation Research Part C 11 (2003) 241–264*

Arpin, C. eds. 2003. *Global Positioning System (GPS) Errors*
& Limitations for Vehicle Tracking
Retrieved December 12th, 2004, from: http://www.boomerangtracking.com/en/pdf/whitepaper.pdf

LaMarca, A.; Chawathe, Y.; *(et al.)* eds. 2004. *PlaceLab: Device Positioning Using Radio Beacons in the Wild*
Retrieved December 12th, 2004, from: http://placelab.org/publications/pubs/IRS-TR-04-016.pdf

Hinton, G. eds. 2004 *Hidden Markov Models*
Retrieved December 12th, 2004, from: http://www.cs.toronto.edu/~hinton/csc321/

Russell, S.; and Norvig, P. 2003. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition.