# Probabilistic Models of Driver Behavior

**Nikunj C. Oza**[*]
Computer Science Division
University of California
Berkeley, CA 94720-1776, USA
oza@cs.berkeley.edu

**Abstract**

The BATMobile [1] is one approach to the implementation of a system capable of driving autonomously in normal highway traffic. As with human drivers, the BATMobile would be a superior driver if it were able to make predictions about the actions of other drivers, effectively giving it the ability to respond before such actions even occur. The ability to predict driver behavior would also be helpful in the creation of accurate models of traffic behavior used in freeway design and analysis. In this report, we discuss the creation of dynamic probabilistic networks (DPNs) that constitute models of driver behavior. Specifically, we discuss three model structures we have created, the data used to learn the models, and how well the models predict driver behavior. We compare the three models to observe the benefits of modeling hidden state and using deterministic variables. We also analyze the models to see what aspects of driver behavior the models have learned.

## 1 Introduction

The BATMobile [1] is one approach to the implementation of a system capable of driving autonomously in normal highway traffic. It uses dynamic probabilistic networks (DPNs) to model the dynamic and partially-observable world of driving. Such a network is to be used by the BATMobile to maintain the current belief state, where the state variables that constitute the complete state may include (for example) the locations and velocities of other cars (actual and observed values), the intentions of other drivers, and the orientation of the road. Of course, many of these variables are unobservable some or all of the time.

The main aim of this research is to derive models of the factors affecting the intentions of other drivers, such as speeding up, slowing down, and changing lanes. Creating models of driver behavior will allow an autonomous driver such as the BATMobile to anticipate potentially dangerous situations, such as another car cutting in front of it. It could also warn human drivers of dangerous situations as part of a driver assistance module. Detailed driver models are also essential in the creation of accurate models of traffic behavior used in freeway design and analysis. These also constitute models of driver preferences and behavior which can be used to assess driver competence [2].

Creating our probabilistic driver models involves identifying the variables essential to predicting human driver behavior, using a learning algorithm [3] to automatically construct a DPN model from data, and assessing how well the model predicts drivers' actions. The data comes from either a driving simulator [1] or from videotapes. From the videotapes the locations of various cars, their speeds, and other information are extracted automatically using computer vision algorithms [4]. In the Transportation community, the type of model we are constructing is considered to be a microscopic model because it models the behavior of individual vehicles. This is different from macroscopic models which describe general traffic characteristics such as density and flow [8] [16].

In this report, we first briefly describe microscopic traffic models. In particular, we explain the motivations for developing microscopic models and survey the types of microscopic models that have been postulated and, in some cases, validated[1] This previous work motivates our decision to create a DPN model. We

[*]Supported by CalTrans (PATH MOU 130) and the NSF Graduate Research Training Program in Spatial Cognition.
[1]In this paper, we will limit ourselves to discussing models of highway traffic.

then describe dynamic probabilistic networks (DPNs) and the tools (driving simulator and DPN learning program) that we use to create our probabilistic microscopic model. We then explain the traffic scenarios that our simulator generated from which we gathered the data for learning the models. We then discuss three DPN models we have learned so far: the first model does not have any hidden state nodes, the second one has some hidden state nodes, and the third one also has some deterministic nodes. We first demonstrate that the model learning is working correctly by learning the three DPN models using data generated from the same DPN model structures with parameters set according to our intuition. We then discuss the results of learning the three models using data from our driving simulator. We explore what behaviors each of these models has learned, analyze the hidden state in the last two models to understand what they represent, and compare how well the three have learned. We end with conclusions and future work.

## 2 Microscopic Models

Traffic engineers generally want to maximize the flow of traffic—the number of vehicles passing a given point per unit time—on roads and highways. Using traffic lights, metering lights, stop signs, speed limit signs, and others, traffic engineers can attempt to control the density (number of cars per unit length) and average velocity of vehicles in order to yield maximum flow. This is a difficult problem—an engineer cannot simply attempt to maximize or minimize the density or velocity, as either of these strategies would be disastrous. The Fundamental Diagram of Traffic Flow shown in Figure 1 and the velocity-density relationship depicted in Figure 2 show the relationships between density, average velocity, and flow. These show that traffic engineers must balance average velocity and density against one another. Reducing the rate at which cars enter the highway too much (near point $(0, 0)$ in Figure 1) would allow the cars that enter to go at their desired speeds, but would yield excessively low density and, therefore, less than the maximum possible flow and possibly excessive wait times and bottlenecks at the entrance ramps and upstream of them. Allowing too many cars to enter the road too quickly would lead to excessive density (near the point $(\rho_{max}, 0)$ in Figure 1) causing drivers to reduce their speed because of uncomfortably low following distances. As density increases further, the velocity and flow get very close to zero—a condition of "bumper-to-bumper" traffic. In summary, we cannot control flow simply by inserting cars at the maximum flow rate. Even assuming we knew what the maximum flow (also called the "capacity") is, the flow does not directly correspond to a rate at which we release cars.

One may wonder why we do not simply control the flow directly by inserting cars at the rate corresponding to the maximum flow (called the "capacity"). The answers are that we cannot know the capacity of a highway without collecting data and, even if we do know what the capacity is, we only control how many additional cars enter the freeway; we do not have control over the number of cars already on the highway and their flow rate.

One way to approach the problem of maximizing flow is to observe a highway for a time long enough to see a wide variety of traffic conditions and record the density, flow, and average velocity during this period. The plot of the data may have a shape similar to the Fundamental Diagram in Figure 1, and then one can find the peak flow as well as the density and average velocity at that peak flow. This is essentially a macroscopic model because it models traffic "in the large" without being concerned with the behavior of individual vehicles.

Another way to approach the problem is to model the behavior of individual drivers and use that model to figure out the maximum flow and the conditions under which it is achieved. This yields the usual advantages of having a model over relying directly on the data collected: a better understanding of the process—in this case, why drivers act the way they do—and the ability to use prediction to calculate results that do not have exactly corresponding data points—in this case, making predictions for traffic conditions that have not been observed. Of course, one disadvantage of using a model in general is that one assumes that a particular model is the correct one or close enough to the correct one that any predictions of the model are very likely to be correct. In this paper, we only explore microscopic models—models of individual drivers—because one of our goals stated above is to understand why drivers act the way they do in order to give our autonomous vehicle the ability to predict the behavior of other vehicles.

Below, we discuss and compare several microscopic models.

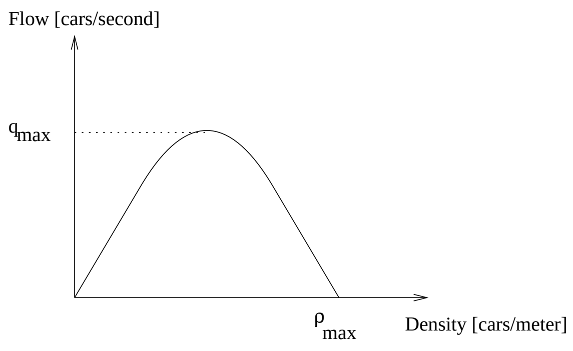


Figure 1: Fundamental Diagram of Traffic Flow (from [8])

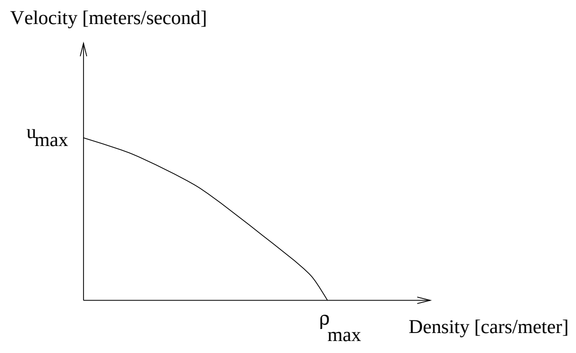


Figure 2: Velocity-Density Relationship (from [8])

## 2.1 Traditional Car-Following Models

One type of model proposed by Lighthill and Whitham [14] and independently Richards [15] is the famous LWR model which postulates that the velocity of a car depends only on the density of cars:

$$u = u(\rho),$$

where $u$ is the velocity and $\rho$ is the density. This is the relationship depicted in Figure 2 above. We also know that flow equals density multiplied by average velocity [8] which, in light of the previous equation, can be written as

$$q = \rho u(\rho).$$

For a given highway, we would like to find the particular relationship ($u(\rho)$) between the velocity and density so that we can find the density and velocity that maximize the flow. This can be done using a car-following model. Here we postulate that an individual car's actions depend only on the car in front. Assume that our car's acceleration is proportional to the relative velocity of the car in front with respect to us, yielding the following linear ordinary differential equation:

$$\frac{d^2y(t)}{dt^2} = \lambda(\frac{dy_f(t)}{dt} - \frac{dy(t)}{dt}),$$

where $y(t)$ is the position along the highway of our car at time t, $y_f(t)$ is the position along the highway of the vehicle in front, and $\lambda$ is a sensitivity parameter that is presumably positive. This model assumes that our car reacts instantaneously to the vehicle in front. As this is unlikely to be true, we should include a time delay $T$:

$$\frac{d^2y(t+T)}{dt^2} = \lambda(\frac{dy_f(t)}{dt} - \frac{dy(t)}{dt}).$$

There is still one missing factor. Our acceleration is likely to depend on the distance to the car in front. In particular, our reactions to the actions of the car in front will be influenced less by the vehicle in front as its distance increases–specifically, our accelerations will tend to be closer to zero for longer following distances. We can model this by changing the sensitivity parameter to

$$\lambda = \frac{c}{y_f(t) - y(t)},$$

yielding the following revised car-following model:

$$\frac{d^2y(t+T)}{dt^2} = c(\frac{\frac{dy_f(t)}{dt} - \frac{dy(t)}{dt}}{y_f(t) - y(t)}).$$

Traffic researchers have proposed several car-following models, most of which are variations on this nonlinear differential equation. From the above model, we can figure out the maximum flow and the density and average velocity that yield the maximum flow as follows. Integrating the above equation yields

$$\frac{dy(t+T)}{dt} = c * ln|y_f(t) - y(t)| + d_f.$$

Let us assume that we are in a steady-state situation in which the distances between vehicles are the same ($d$) so that we can assume that $y_f(t) - y(t) = 1/\rho$ and $\frac{dy(t)}{dt} = \frac{dy(t+T)}{dt}$. We then get

$$\frac{dy(t)}{dt} = -c * ln|\rho| + d.$$

Recall, from the Fundamental Diagram (Figure 1), that at $\rho = \rho_{max}$, the velocity is 0. This initial condition enables us to find $d$ and transform the above equation to

$$u = \frac{dy(t)}{dt} = -c * ln(\frac{\rho}{\rho_{max}}).$$

From this, we get

$$q = \rho * u(\rho) = -c\rho * ln(\frac{\rho}{\rho_{max}}).$$

The maximum flow occurs when $\frac{dq}{d\rho} = 0$ which happens at $\rho = \rho_{max}/e$, in which case $u = c$, so the constant $c$ is the desired average velocity.

So a traffic engineer can calculate the constant $c$ using a least-squares fit to data collected in the field so that the above microscopic model fits real data as well as possible and then adjust various lights and signs in an effort to get the density and velocity as close as possible to the values that yield maximum flow[2]. Unfortunately, the world is not as simple as this. The Fundamental Diagram on which this scheme is based has been shown to be quite inaccurate around the peak flow [19]—this is the regime that traffic engineers are most interested in, because it is the most difficult scenario in which to drive. We also made a number of highly questionable assumptions in the above derivation. We assumed that traffic was in a steady-state situation in which the distances between cars were basically the same, cars only respond to the behavior of the vehicle in front, and the velocity depends only on the density. It is not reasonable for traffic that is nearly at peak flow to be considered steady-state traffic. In the uncongested regime cars have sufficient space between them to allow the cars to act virtually independently, so they are likely to travel at their preferred speed and not change lanes often, which may be close to steady-state traffic. In the congested regime, the cars have so little space between them that they cannot move much or change lanes, so this traffic may be acceptably close to steady-state. The region around peak flow corresponds to a density level at which cars regularly pass each other and cut each other off, which is not a steady-state traffic condition. The car-following model, as its name implies, also assumes that cars merely follow the vehicle in front and do not change lanes. Again, this assumption breaks down under the conditions that we are most interested in: near-peak flow. Additionally, because these car-following models are deterministic, they are unable to capture the wide variation in behaviors of individual drivers–not only do different drivers behave very differently, but individual drivers behave differently at different times even under the same circumstances.

## 2.2 More Recent Microscopic Models

Some microscopic models use if-then rules or decision trees that explicitly test for certain conditions and choose actions on that basis [17] [18]. One example of such a system is the decision tree shown in Figure 6. This decision tree is used to control the vehicles in the driving simulator that generated the traffic scenarios for this paper. This tree and the systems described in [17] and [18] first assess whether the current situation is a normal or emergency situation based on the distance to and the relative velocity with respect to the vehicle in front. In an emergency situation, safety is the only important goal: our car brakes very hard and

[2] Surprisingly, very little experimental data has been collected and car-following model fitting done [16].

steers to avoid the car in front without regard to other goals such as making progress or reaching an exit. In a normal situation, if the vehicle in front is beyond a certain distance, then our car travels at its desired speed and works on achieving other goals such as entering the desired lane if necessary. If the car in front is an intermediate distance away then our car may change lanes if our exit is not coming up shortly, traffic in one of the adjacent lanes is moving faster, and it is safe to change into the faster lane.

The greatest difficulties for models like this are that they are hand-constructed and it is unclear how to set up a mechanism for learning in these models. Also, there are various thresholds, such as desired following distance and following distance beyond which we do not worry about the car in front, that are used to choose actions. The actions chosen when values are on either side of the thresholds need to be similar enough that one does not get unstable oscillations. As one undesirable example, for following distances of less than 10 meters, one may perform panic braking and for following distances beyond 10 meters, one may accelerate to the desired speed. This may lead to oscillations between accelerating and panic braking. Additionally, these thresholds themselves need to change with traffic conditions in order to model vehicle behavior realistically.

There has been some work on creating probabilistic microscopic models [6] [11]. The work in [6] builds on the work in [18]. Specifically, the earlier work in [18], in choosing an action, assumes that surrounding vehicles continue in their current lanes at their current speeds—this is done largely for computational reasons. This may be reasonable for very low time scales, but as the time scale increases, uncertainty plays a much more crucial role. In [6], the authors use an extended Kalman filter to model the behavior of other vehicles and account for sensor noise. However, this work does not incorporate percepts over time to assess unobservable aspects of the current state (such as driver intentions) and the models have not been validated. The work described in [11] creates one Hidden Markov Model (HMM) for each of several predetermined actions such as changing lanes or turning at an intersection. These models have been validated by having human subjects drive in a simulated world. At various times while driving, the subject is told to perform some action such as change lanes or stop at the next intersection. The control data (steering wheel angle, steering wheel turning velocity, vehicle velocity, and vehicle acceleration) corresponding to each action are recorded and used to train the HMM corresponding to that action. Because separate models are learned for each of several predetermined actions and because data not corresponding to the predetermined actions are not used in training, there is limited scope for learning novel or higher-level actions.

This report discusses dynamic probabilistic networks that mitigate many of the drawbacks of the above models.

## 3 Probabilistic Networks

Probabilistic networks are directed acyclic graphs in which each node represents a random variable and each arc represents a belief that the node at the tail has a direct causal influence on the node at the head [3] [9]. The topology of the network implies numerous conditional independence relationships among the random variables. Specifically, every node is conditionally independent of its non-descendants given its parents. Associated with each node is a conditional probability table (CPT) that provides conditional probabilities of the node's possible states given each possible state of its parents (or the prior probabilities if the node has no parents). Probabilistic networks offer a mathematically sound basis for making inferences under uncertainty. The conditional probability tables provide a natural way to represent uncertain events. The conditional independence relationships implied by the topology of the network may allow the joint probability distribution to be specified with exponentially fewer probability values than in the full joint probability distribution. For example, suppose we have the belief network depicted in Figure 3 (borrowed from [9]). This network depicts a joint probability distribution over five binary-valued variables. The full joint distribution requires $2^5 - 1 = 31$ independent parameters (the 32nd parameter is not independent because all 32 probabilities must sum to one). We can see the reduction in the number of parameters that the conditional independence relationships afford as follows. First, note that any probability over the five variables can be written as follows[4]:

[3]Strictly speaking, this is true of a particular type of probabilistic network called a causal network; however, all of the networks we will use in this paper are causal.

[4]In the equation below, we chose an ordering corresponding to a topological sort of the graph in Figure 3. This is necessary in order to readily take advantage of the conditional independence relationships implied by the network

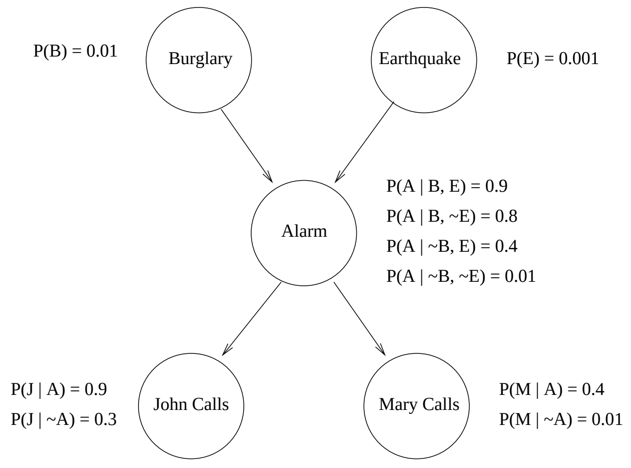


Figure 3: A generic probabilistic network. The ovals denote random variables. The arcs represent conditional dependence relationships. The probabilities given next to each node are the probabilities of that node being true given the different possible values of the parent nodes.

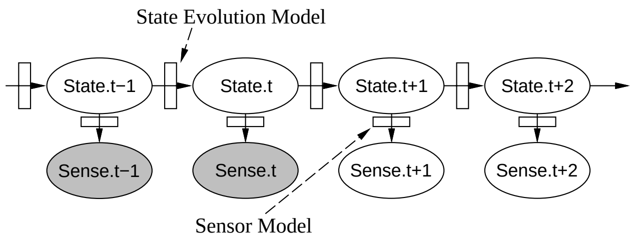


Figure 4: A generic dynamic probabilistic network. The ovals denote sets of state nodes or sensor nodes. The arcs going from one slice to the next form the state evolution model, and the arcs going into the sensor nodes form the sensor model. The shaded ovals denote the observable data available when predicting the state at time t+1.

$$\begin{aligned}P(B=b,E=e,A=a,J=j,M=m) = {} & P(B=b)P(E=e|B=b)P(A=a|B=b,E=e)\\ & P(J=j|B=b,E=e,A=a)P(M=m|B=b,E=e,A=a,J=j).\end{aligned}$$

We can simplify the above expression significantly by noting the conditional independence relationships among the variables. For example, variables B and E do not have arcs between them or common parents, so they are unconditionally independent. This means that $P(E=e|B=b) = P(E=e)$. As a more complicated example, J only has an arc coming from variable A. This means that the influence on J of all the other variables occurs via A, i.e., J is conditionally independent of the other variables given A, so we can rewrite $P(J=j|B=b,E=e,A=a)$ as $P(J=j|A=a)$. Using the existing conditional independence relationships, we can simplify the previous equation to

$$\begin{aligned}&P(B=b,E=e,A=a,J=j,M=m) =\\ &\quad P(B=b)P(E=e)P(A=a|B=b,E=e)P(J=j|A=a)P(M=m|A=a).\end{aligned}$$

All the necessary parameters are given in Figure 3, which shows that we only need to keep 10 parameters rather than the 31 that would have to be stored as part of a full joint distribution. In general, if we have $n$ binary-valued random variables $X_1,...,X_n$,[5] the full joint distribution has $2^n-1$ parameters, whereas the corresponding belief network requires us to store $n2^k$ parameters assuming each node has $k$ parents in the belief network. If the belief network is sparsely connected, $k << n$, which would mean a great reduction in the number of probabilities stored. As we showed above, when specific values are observed for some of the nodes in a probabilistic network, posterior probability distributions can be computed for any of the other nodes. A variety of inference algorithms [9] exist for this purpose.

Dynamic probabilistic networks (DPNs) allow for reasoning in domains where variables take on different values over time [10]. Figure 4 shows the general structure of a DPN (in a real network, there would be numerous “State” and “Sense” nodes). Typically, observations are taken at regular time slices with some appropriate time interval separating the time slices, and a given network structure is replicated for each slice. DPNs model their domains as partially observable Markov processes, so nodes not only are connected to

[5]We assume that the variables are binary-valued only for expositional simplicity.

nodes within the same time slice, but may also have connections leading to the next time slice. The Markov property states that

$$P(State_{t+1}|State_t, State_{t-1}, \ldots) = P(State_{t+1}|State_t).$$

That is, the future is independent of the past given the present. As long as the state nodes represent everything that is necessary to predict the next state, one does not need to keep information from the past.

The CPTs for the set of arcs proceeding from one time slice to the next constitute the state evolution model, which represents how the network believes the state evolves over time. The CPTs for the set of arcs going to nodes whose values are typically observed at each time slice constitute the sensor model, which quantifies the likelihood of making a set of observations given the actual state of the system.

In our work, we need to learn DPNs from data. We provide the structure and assume that only the CPTs need to be learned. As explained above, many aspects of driver behavior are not directly observable. Additionally, we cannot always expect to get values for normally-observable variables because of limited sensor ranges and other data collection problems. Therefore, we are learning DPNs for the known-structure-hidden-variable case. We would like to find CPT entries such that the resulting DPN models our traffic scenarios accurately. We use the Expectation-Maximization (EM) algorithm to find the CPTs that give the maximum likelihood of the test traffic scenario data [3].

## 4 Tools

As part of the Bayesian Automated Taxi (BAT) project [1], we have created a 2-D physical simulator that allows us to create highway networks, vehicle controllers, and probability distributions on various parameters such as the physical sizes of the cars, desired velocities, and desired lane changing rates. For this report, we created an interesting traffic scenario by first generating slow vehicles in the three right-hand lanes out of the four lanes of the highway and then generating faster vehicles following them. Therefore, in order to proceed at their desired speed, the faster vehicles had to change lanes to the left—ultimately into the left lane—so that they could pass the slow traffic in front, or they had to weave through the slower vehicles to pass them. Faster vehicles were continually generated, so there was a significant amount of lane changing and the left lane occasionally became a bottleneck as faster cars in all three of the right lanes wanted to get into the left lane to pass the slow cars. After passing; however, the cars faced uncongested traffic. Therefore, this scenario has a reasonably good range of traffic flow from uncongested to near peak flow, making it an interesting test for our behavior learning. We discuss the scenario and the characteristics of the simulator vehicles that we modeled in more detail in the next section.

We also created virtual cameras that simulate traffic surveillance cameras and the noisy data that they typically return. It is from these virtual cameras that we got the data that we used to learn our driver models. A diagram of the main display of our simulator with a snapshot of the scenario and our virtual cameras is in Figure 5. For each DPN we created, we modified the simulator to return the observable data in the form required by the belief network learning and inference system BAY-SIM [7].

We learned the conditional probability tables (CPTs) of the DPNs using the simulator data by instantiating 30 slices of the network—where each slice represents one reported camera observation of a vehicle and each training example consists of a sequence of 30 reported camera observations of the same vehicle—and learning this as a static network using the EM algorithm with the caveat that the CPTs corresponding to connections between nodes in different time slices remain the same across all time slices.

## 5 Traffic Scenarios and Vehicles

As we discussed in the previous section, the traffic scenario had one slow vehicle (going at 4-6m/s) in all but the left lane out of the four lanes on the highway. About ten seconds after the slow vehicles were generated, faster vehicles were generated. The faster vehicles had to either change to the left lane or weave through the slower vehicles in order to travel at their desired speeds.

The vehicles have three major components to them: the sensing mechanism, the decision mechanism, and the action mechanism. Each vehicle performs one decision cycle—consisting of sensing, deciding, and acting—every 0.1 seconds. The vehicles sense the following information without any sensor noise:

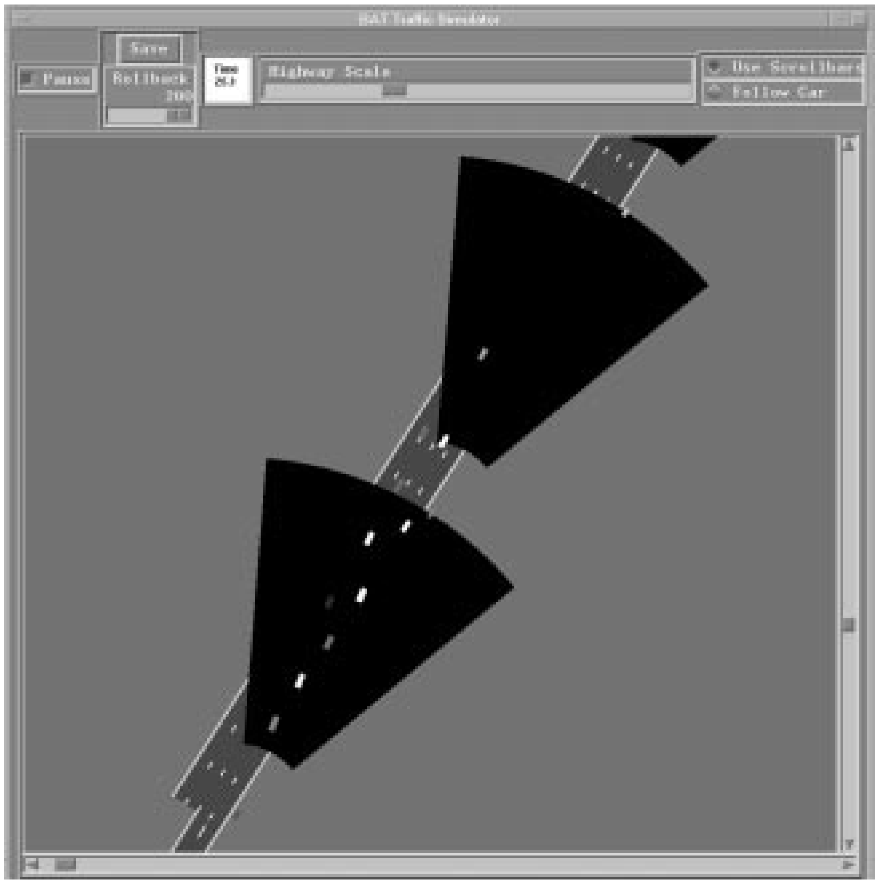


Figure 5: Simulator Environment with Virtual Cameras. The black regions constitute the cameras' fields of view–if a car is in this region, then the camera reports the car's lateral and longitudinal position and velocity.

- Data on the car itself
  - Speed
  - Acceleration
  - Position–lateral and longitudinal along the current highway segment.
  - Heading with respect to straight ahead down the lane.
- Cars up to 100 meters away
  - Relative position
  - Relative speed
- Highway characteristics
  - Curvature
  - Lane width

The vehicles use the sensor readings to update their state. The vehicle's state consists of the latest sensor readings, and the following:

- The current driving state—one of the following:
  - Following the current lane
  - Currently changing lanes
  - Wanting to change lanes

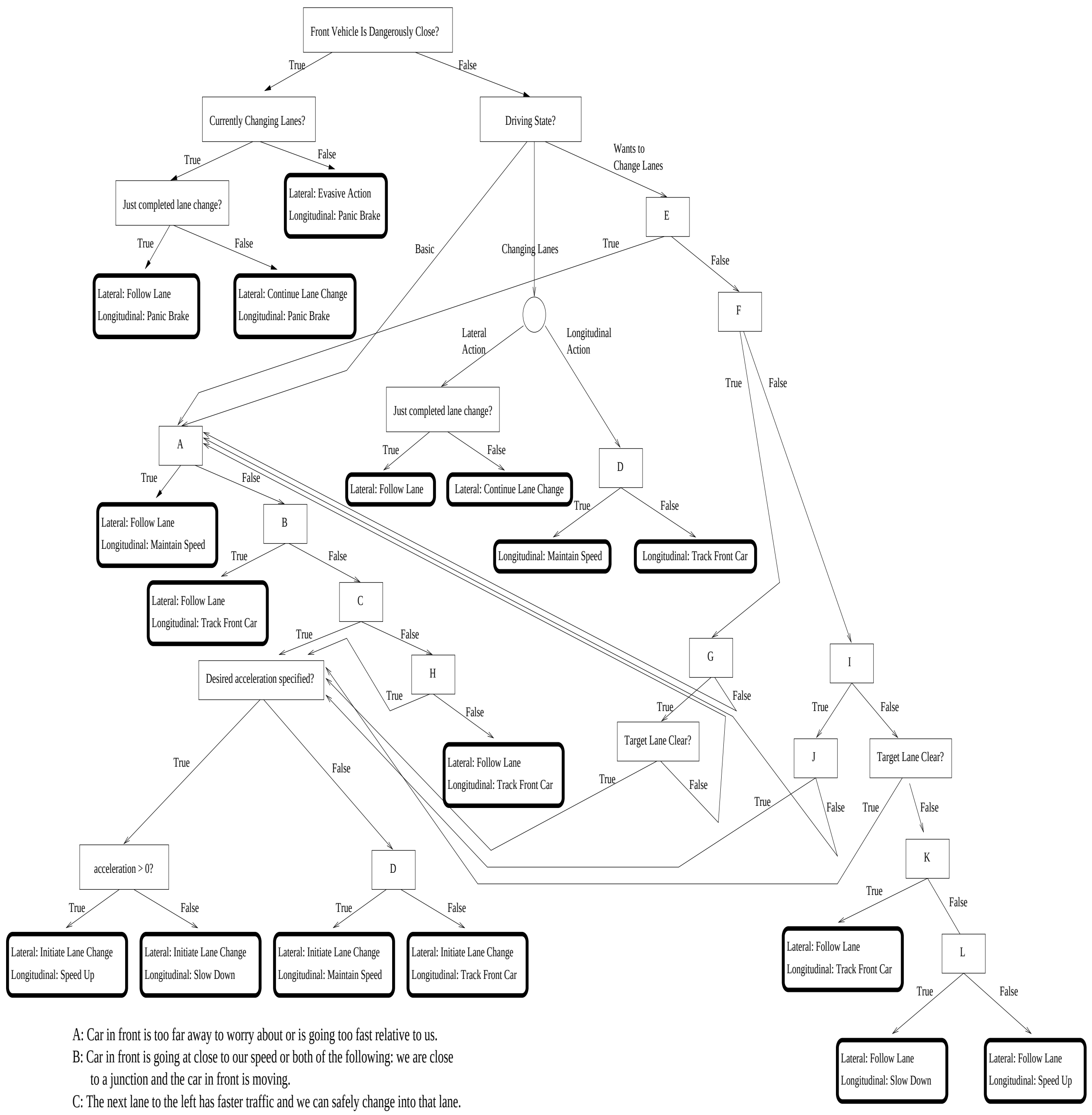
- The current lateral action—one of the following:
  - Initiating lane change
  - Changing lanes
  - Following the current lane
  - Taking evasive action
  - Aborting lane change
- The current longitudinal action—one of the following:
  - Maintaining the current speed
  - Increasing speed
  - Decreasing speed
  - Performing panic braking
- Target speed
- Planned route
- Target lane
- Minimum required following distance
- Several stochastic parameters—distributions are indicated in Table I.
  - random-change-time: Number of seconds after a previous lane change that the car may change lanes purely out of boredom.[6]
  - reaction-time: Time after the car decides to take an action that it actually begins to perform that action.
  - front-danger-dist: Additional slack distance allowed for the car in front in case of noise or miscalculations.
  - front-speed-diff-tolerance: Maximum relative speed (this car's speed minus the front car's speed) allowed.
  - follow-dist: The minimum desired following distance.
  - target-speed: Preferred cruising speed.

The distributions of the stochastic parameters are listed in Table I below[7]. These are sampled once from the distributions listed in the table and set at the time the vehicle is created. To make it easier to test our learning, we kept the vehicle behaviors simpler by not resampling from the distribution during the vehicle's existence.

**Table I : Stochastic Parameters Used by the Vehicles**

| Stochastic Parameter | Distribution |
|---|---|
| random-change-time[sec] | N(5.0,4.0) |
| reaction-time[m] | N(0.1,0.01) |
| front-danger-dist[m] | N(2.0,0.0625) |
| front-speed-diff-tolerance[m/s] | N(2.0,0.25) |
| follow-dist[m] | N(25.0,100.0) |
| target-speed[m/s] | $\forall x \in 10, 12, 14, 15, 16, P(X = x) = 0.2$ |

The vehicle's state information is used to make a driving decision at each time step according to the decision tree given in Figure 6. Each thick-lined box in the figure corresponds to a particular lateral and longitudinal action to take. The remaining boxes are conditionals. Some of the decisions are made based



Front Vehicle Is Dangerously Close?
True
False
Currently Changing Lanes?
Driving State?
True
False
Wants to Change Lanes
Just completed lane change?
Lateral: Evasive Action
Longitudinal: Panic Brake
E
True
False
Basic
Changing Lanes
True
False
Lateral: Follow Lane
Longitudinal: Panic Brake
Lateral: Continue Lane Change
Longitudinal: Panic Brake
F
Lateral Action
Longitudinal Action
True
False
Just completed lane change?
A
True
False
D
True
False
Lateral: Follow Lane
Lateral: Continue Lane Change
Lateral: Follow Lane
Longitudinal: Maintain Speed
B
True
False
Longitudinal: Maintain Speed
Longitudinal: Track Front Car
True
False
Lateral: Follow Lane
Longitudinal: Track Front Car
C
True
False
G
I
H
Desired acceleration specified?
True
False
True
False
True
False
Target Lane Clear?
J
Target Lane Clear?
True
False
Lateral: Follow Lane
Longitudinal: Track Front Car
True
False
True
False
True
False
acceleration > 0?
D
K
True
False
True
False
True
False
Lateral: Initiate Lane Change
Longitudinal: Speed Up
Lateral: Initiate Lane Change
Longitudinal: Slow Down
Lateral: Initiate Lane Change
Longitudinal: Maintain Speed
Lateral: Initiate Lane Change
Longitudinal: Track Front Car
Lateral: Follow Lane
Longitudinal: Track Front Car
L
True
False
Lateral: Follow Lane
Longitudinal: Slow Down
Lateral: Follow Lane
Longitudinal: Speed Up

A: Car in front is too far away to worry about or is going too fast relative to us.

B: Car in front is going at close to our speed or both of the following: we are close to a junction and the car in front is moving.

C: The next lane to the left has faster traffic and we can safely change into that lane.

D: There is no car in front of us or it is too far away to worry about.

E: We are in the target lane.

F: We are in an exit lane.

G: We are far away from a junction or we are in an exit lane.

H: The next lane to the right has faster traffic and we can safely change into that lane.

I: We are far from a junction or both of the following: we are a medium distance from a junction and we are within two lanes of the target lane.

J: Our target lane has faster traffic and we can safely change into that lane.

K: The distance to the car in front is less than the sum of the minimum allowable distance plus some slack.

L: (priority = f((distance from the target lane)/(distance to junction))) is high or (we are moving and (our distance in back is too small relative to the distance in front or the car in front is too close or there are more than 10 cars in the current lane within sight)) or our speed is greater than the target speed plus some slack.

Figure 6: The vehicles in our driving simulator use this decision tree to decide on the next high-level action.

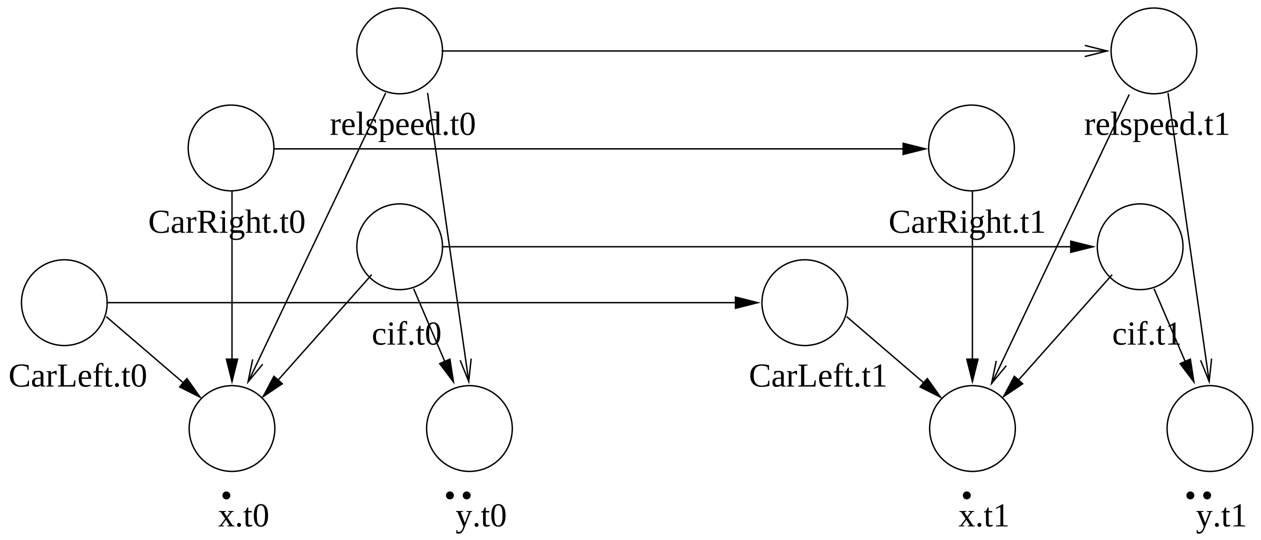


Figure 7: Network with no Hidden State

on the stochastic parameters; therefore, different vehicles will often perform different actions in the same situation.

The vehicle's current decision and last steering wheel position, throttle position, and brake position, as well as the speed, heading, curvature of the road, and several other parameters are used to decide on the current steering wheel position, throttle position, and brake position according to a proportional plus derivative control law briefly described in [13].

It should be noted that, even though for this paper we have very detailed knowledge of the traffic scenario and the way that the vehicles behave, our dynamic belief network models have not been constructed specifically to model this scenario. That is, the belief network structures were created based only on the author's knowledge of driving. We used our knowledge of the traffic scenario only to confirm that the DPNs learned correctly. Since the belief network structures were constructed independently of the scenario, there is every reason to believe that we can learn the behavior of vehicles in any realistic traffic scenario using our method, and we are currently working on confirming this.

## 6 The DPN Model Structures

### 6.1 Model Without Hidden State

The first DPN that we created (first two slices only) is shown in Figure 7. All the DPNs that we use have only discrete variables. The nodes, their meanings, and possible values are as follows (see Figure 8).

**CarLeft:** Is the next lane to the left clear for lane changing?

**Possible Values:** True, False

**CarRight:** Is the next lane to the right clear for lane changing?

**Possible Values:** True, False

**cif:** Size of gap (meters) to the vehicle in front (if any)?

**Possible Values:** $(0,2], (2,5], (5,10], (10,25], (25,50], >50$

**relspeed:** What is our relative speed (meters/second) with respect to the vehicle in front (if any)[8]?

[6] There are regular lane changes that occur for better reasons, of course.

[7] All parameters that have a normal distribution are set to zero if their sampled value is negative.

[8] For 'cif' and 'relspeed,' if the vehicle in front cannot be seen, then no value is reported and the value is considered missing when learning the models.

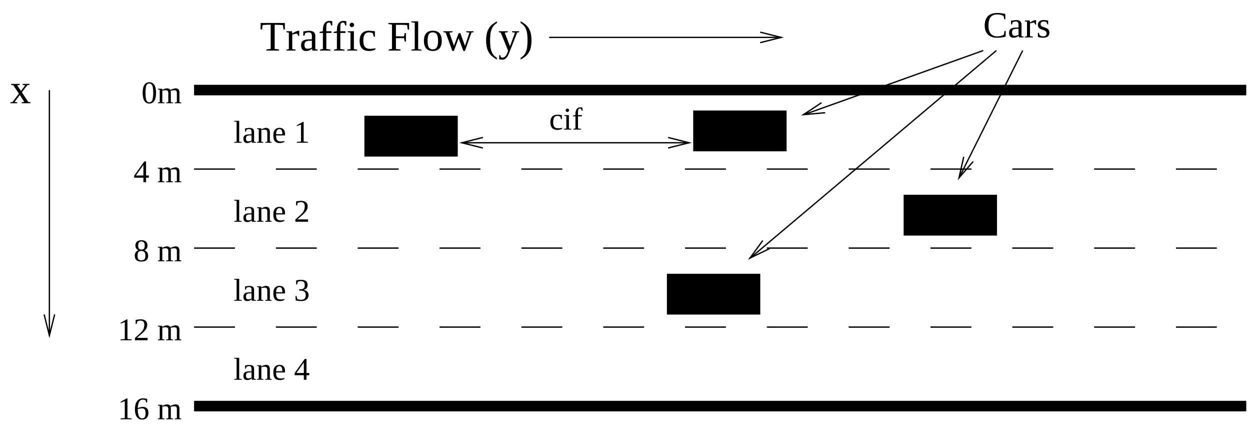


Figure 8: Diagram Illustrating Traffic Variables

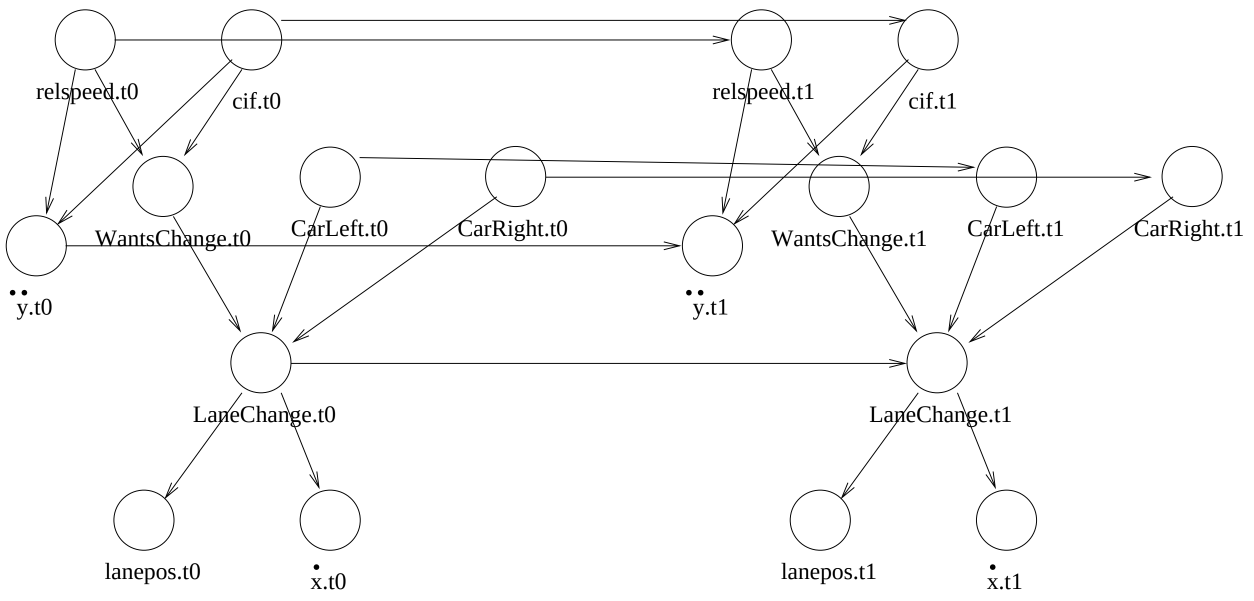


Figure 9: Network with Hidden State

**Possible Values:** $< 0, [0, 1), [1, 2.5), [2.5, 5), > 5$

$\dot{x}$**:** Lateral Velocity (meters/second) (positive velocity is toward the right).

**Possible Values:** $< -1, [-1, -0.5), [-0.5, -0.25),$
$[-0.25, 0.25], (0.25, 0.5], (0.5, 1], > 1$

$\ddot{y}$**:** Longitudinal Acceleration (meters/second/second)

**Possible Values:** $< -0.25, [-0.25, 0.25], > 0.25$

Of course, not all of these variables are observable at all times. Because we obtain our data using cameras mounted on the side of the highway, there are occasionally cars for which the cameras cannot see the car in front either due to occlusion or because the car in front is outside the camera's field of view.

## 6.2 Model With Hidden State

The second DPN that we created is shown in Figure 9. The nodes, their meanings, and possible values are the same as for the previous network but with two additions.

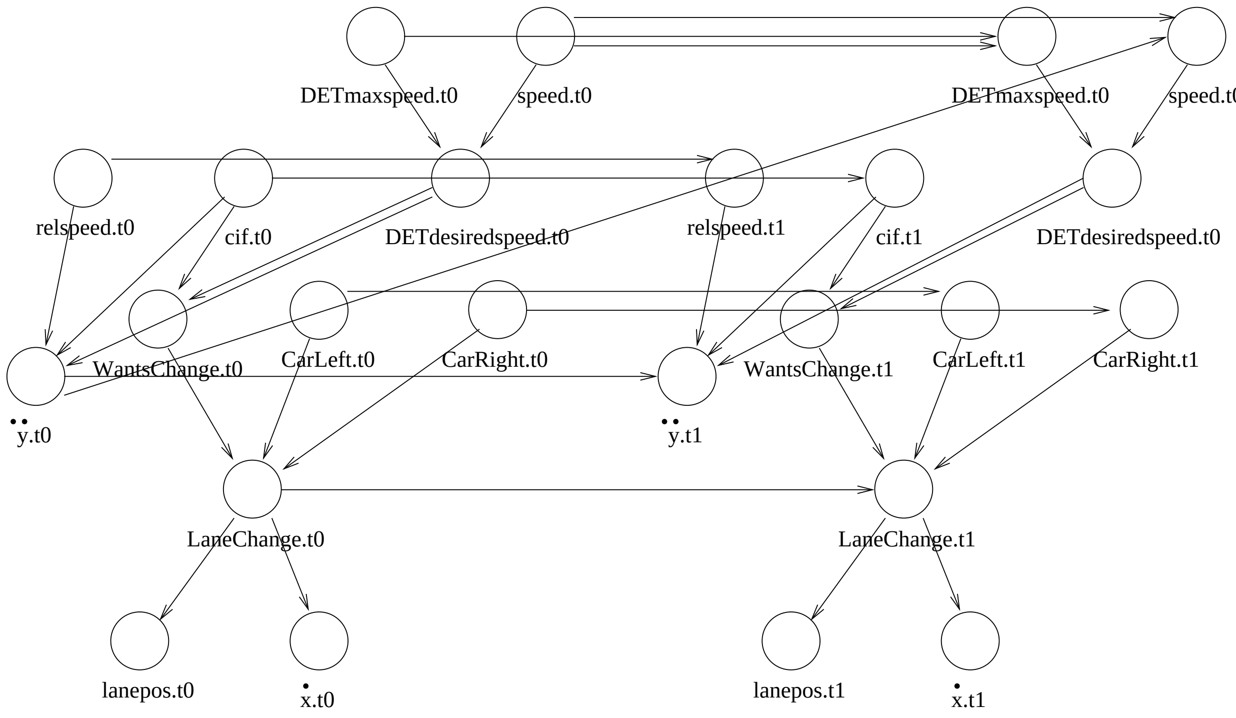


Figure 10: Network with Hidden State and Deterministic Nodes

**WantsChange** Unobservable

  **Possible Values** True, False

**LaneChange** Unobservable

  **Possible Values** StartLeft, FinishLeft, StartRight, FinishRight, Stay

The node "WantsChange" is intended to represent whether or not the driver wants to change lanes. We expect that a driver is more likely to want to change lanes as the distance to the car in front decreases and the relative speed increases. For this reason, we have set up this node so that it is directly dependent on the distance and relative speed and set its initial CPT according to the relationship just described. The node "LaneChange" is supposed to represent whether the driver is actually changing lanes or not and if he is changing lanes, what the state of that maneuver is, i.e., in which direction is he changing and is he in the early ("StartLeft" or "StartRight") or late ("FinishLeft" or "FinishRight") stages of changing. For example, the early stage of a lane change may be the part where the car is still in its original lane but is moving toward the new lane, while in the late stage, the car may be in the target lane and in the process of centering itself in the new lane. The "LaneChange" node is set up to be dependent on whether the driver wants to change lanes and whether the adjacent lanes are clear for lane changing. Of course, because the values of these nodes represent driver intentions that are never observable, we cannot force a particular real-world semantics on them. What we hope is that, after learning, we can examine the CPTs of both the hidden nodes and their children and find real-world interpretations of the hidden nodes that are consistent with the CPTs. This is exactly what we do in section 7.2 for the "WantsChange" and "LaneChange" nodes.

## 6.3 Model With Hidden State and Deterministic Nodes

The third network that we created is shown in Figure 10. The nodes, their meanings, and possible values are the same as for the previous network with three additions.

**speed** Current speed of our vehicle (meters/second).

  **Possible Values** 0-5, 5-10, 10-15, 15-20, 20-25, 25-30, $> 30$

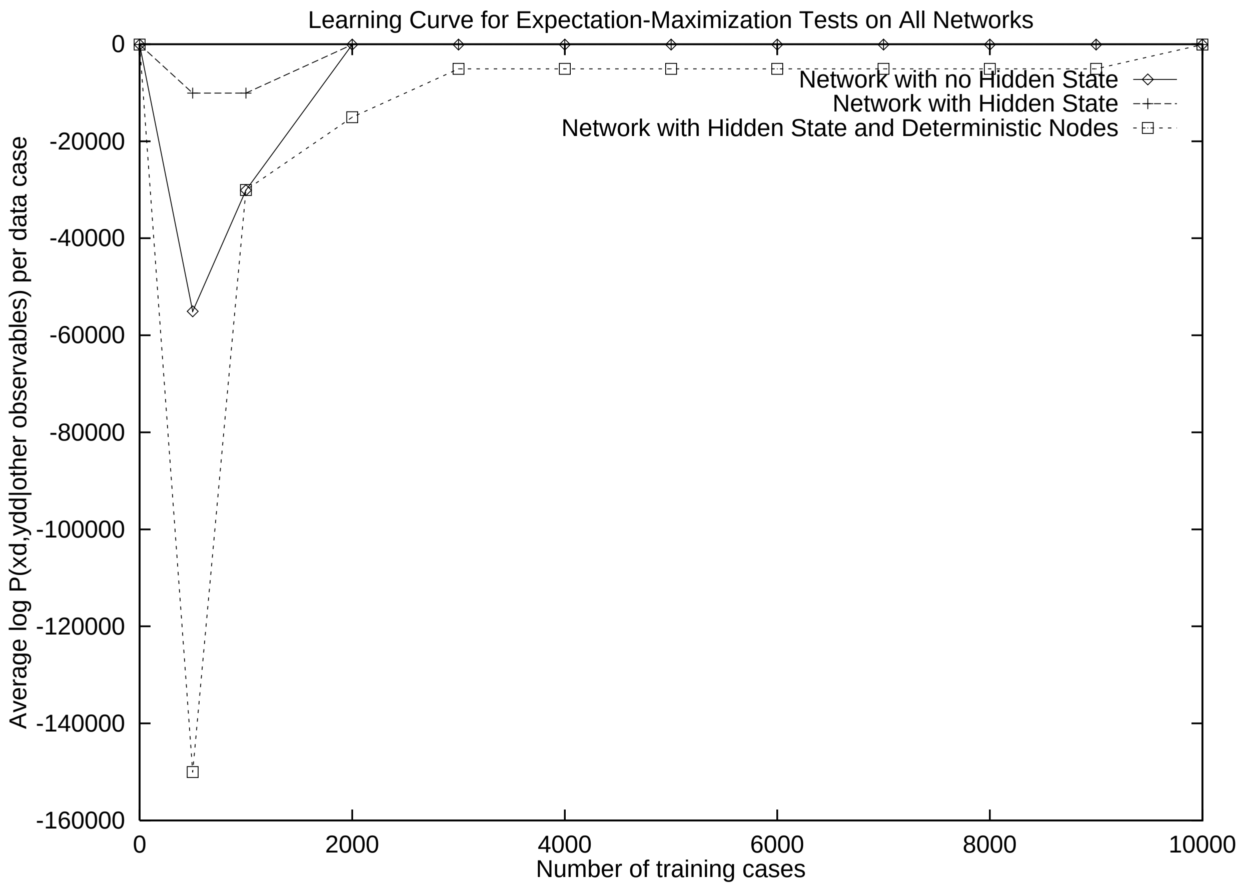


Figure 11: Learning Curve for Testing Expectation-Maximization

**DETmaxspeed** Calculated as the maximum speed of our vehicle in the observations seen so far.

**Possible Values** 0-5, 5-10, 10-15, 15-20, 20-25, 25-30, $> 30$

**DETdesiredspeed** Has car achieved its desired speed (assumed to be its maximum speed reached in previous observations)?

**Possible Values** Traveling at less than its desired speed, traveling approximately at its desired speed, or traveling at more than its desired speed.

The deterministic nodes "DETmaxspeed" and "DETdesiredspeed" are nodes that have conditional probabilities of exactly 1.0 and 0.0. These nodes, including the deterministic nodes, were added because we realized that whether a vehicle wants to change lanes does not merely depend on the distance to and relative speed with respect to the car in front. In particular, a vehicle may slow down so that its relative speed with respect to a slower vehicle in front is nearly zero; however, the vehicle would still want to change lanes because it is going at less than its desired speed. The definitions of the two deterministic nodes make it clear that they can be calculated deterministically from the nodes that they depend on.

## 7 Controlled Experiments

Before learning and testing our DPNs using driving simulator data, we did some experiments to test that the implemented expectation-maximization algorithm converges for the DPNs that we have created. To do this, we created initial CPTs based on our own beliefs about how people drive and used these to generate training and test cases, where each training case and test case is a sequence of 30 observations generated by unrolling the DPN into a 30-slice belief network and simulating it. We then tested the ability to predict the test cases based on learning various numbers of the training cases. The results of these tests are shown in Figure 11. For every learning curve in this paper, the y-coordinate gives the average log-probability over all the test cases of $\dot{x}$ and $\ddot{y}$ given the other observables. That is, define $\dot{x}_{(i,j)}$ and $\ddot{y}_{(i,j)}$ to be the observations of $\dot{x}$ and $\ddot{y}$ in the $i$th time slice and the $j$th test case. Define $A_j = \{\dot{x}_{(1,j)}, \ddot{y}_{(1,j)}, \dot{x}_{(2,j)}, \ddot{y}_{(2,j)}, \ldots, \dot{x}_{(30,j)}, \ddot{y}_{(30,j)}\}$, $B_j$ to be all the variables in the $j$th test case in the unrolled network that are not hidden including those in $A_j$, and $T$ to be the entire set of test cases. Then for each point $(m, L)$ in the learning curve where $m$ is the number of training examples used to learn the network, $L$ is

$$L = \frac{\log P(A|B-A)}{|T|} = \frac{\log \prod_{i=1}^{|T|} P(A_i|B_i - A_i)}{|T|} = \frac{\log \prod_{i=1}^{|T|} \frac{P(B_i)}{P(B_i - A_i)}}{|T|} \tag{1}$$

Note that EM attempts to maximize $P(B)$ [12], so it is entirely possible that our measure, $P(A|B-A)$, decreases with more training examples even though EM is behaving correctly. This happens once when learning the network with deterministic nodes, as we will see in Section 7.3.

We use the metric $P(A|B-A)$ rather than the usual $P(B)$ because the three networks have different sets of observables (i.e., different "B" sets), therefore, using the convention of returning the log-probability of the observable data is too heavily biased toward networks with fewer observables. Additionally, the variables in $A$ constitute the actions of the cars, which are what we want to predict. From now on, when we refer to the log-probability of the test cases, we always mean $\log P(A|B-A)$. In all the learning curves, for every point $(0, L)$, $L$ is the average log-probability per test case given the initial setting of the CPTs. The remaining points $(m, L)$ indicate that after learning the first $m$ training examples using Expectation-Maximization to find the maximum likelihood setting of the CPTs, $L$ is the average log-probability per test case. This means that, to demonstrate convergence, we should reach an average log-probability close to $L$. This is not apparent from Figure 11; however, table II below gives the initial and final average log-likelihoods, which do make it apparent.

**Table II : Average log-likelihoods per test case**

| Network Type | L for points (0,L) | L for points (10000,L) |
|---|---|---|
| No Hidden State | -80.3908 | -80.4159 |
| With Hidden State | -63.0694 | -67.0736 |
| With Hidden State and Deterministic Nodes | -63.3747 | -65.0796 |

## 8 Real Results

In this section, we discuss the results of learning and testing using data from the 2-D driving simulator created as part of the BAT project [1]. The scenario used is described in section 4.

### 8.1 Judging the Quality of our Learning

Below, we give learning curves for the three networks we are considering in this paper. They give the average log-likelihood per test case ($logP(\dot{x}, \ddot{y}|$other observables$)$) after learning different numbers of training cases. We need some standard by which to judge the learning curves. To that end, in this section, we discuss three probabilities we have calculated that give us such a standard.

#### 8.1.1 Ideal Predictions

We wanted to determine the probability that the actions that turn out to be chosen actually were chosen. That is, we wanted to calculate the probability that the action that was chosen in a given situation would actually be chosen again if the same situation presented itself. This is basically the probability that the actual model (the decision tree shown in Figure 6 plus the stochastic parameters and noise from the cameras used to measure the appropriate quantities) chooses the action that was actually chosen. To that end, we modified the simulator so that each time a vehicle made a decision, it resampled the values of its stochastic parameters from their distributions 1000 times, recalculated its decision ($\dot{x}$ and $\ddot{y}$) for each of those samples, and added the appropriate camera noise. This gave us a probability distribution of possible actions that the decision tree could have chosen at that moment. Based on this, we calculated the probability that the initially chosen action was taken. For each car we calculated the probability of sequences of 30 consecutive actions so that the results can be compared with the results of our DPN learning which learned sequences of 30 actions.

The average log-sequence probability achieved across all the vehicles was around -9.20. We do not expect realistic models to perform better than this because this probability is calculated using the model that generates the vehicle behaviors and has full access to the driver's state which is ordinarily hidden. It also uses the fact the vehicles can sense anything within their sensor range of 100 meters (in our DPN learning, we have to rely on simulated cameras and their fixed fields of view). We are currently testing DPN models with more variables to determine how much closer we can get to this ideal.

#### 8.1.2 The "No-Hidden State" Model

The DPN models that we present in this paper predict sequences of $(\dot{x}, \ddot{y})$ in the particular discretizations that we chose, therefore we decided to calculate an additional statistic to see how well our models perform. We kept track of the number of times that each pair $(\dot{x}, \ddot{y})$ was chosen, and using that calculated

$$P(\{\dot{x}_{(1,j)}, \ddot{y}_{(1,j)}, \dot{x}_{(2,j)}, \ddot{y}_{(2,j)}, \ldots, \dot{x}_{(30,j)}, \ddot{y}_{(30,j)}\},$$

which is simply the probability that the chosen sequence of pairs $(\dot{x}, \ddot{y})$ would be chosen again. The average log-sequence probability across all vehicles was approximately -42.08. We would expect all of our models to perform better than this because this represents predicting each action independently without any knowledge of the drivers' reasoning processes. Our DPN models should at least be able to capture some correlations between subsequent pairs $(\dot{x}, \ddot{y})$ to yield some improvement.

#### 8.1.3 The "Do-Nothing" Model

The action of "doing nothing" (maintaining the current speed and remaining in the current lane), is often the most commonly performed action and has a certain default quality to it. To that end, we would like for our models to out-perform a model that simply predicts that a car will maintain its current speed and remain in its current lane.

We modified the simulator again so that it records the probability that the car has $\dot{x} \in [-0.25, 0.25]$ and $\ddot{y} \in [-0.25, 0.25]$, which, in our DPN models, represents the action of continuing straight ahead and maintaining the current speed. The average log-sequence probability we obtained was around -81.47. As expected, all of our models perform significantly better than this.

#### 8.1.4 The "Same-Choice" Model

To observe the benefit we get by accounting for the correlations between actions, we calculate the probability that $\dot{x}$ and $\ddot{y}$ are the same as they were in the previous time step. We again calculated the average log-sequence probability, which is around -37.5. Our Network without Hidden State performed only slightly worse than this in spite of the fact that the DPN often had missing values for variables "cif" and "relspeed" because the car in front left the camera's field of view. Of course, our models with hidden variables performed substantially better.

#### 8.1.5 The Random-Choice Model

We would certainly want for our models to perform better than chance. To calculate the chance probability, we note that there are seven possible values of $\dot{x}$ and three possible values of $\ddot{y}$, giving 21 possible choices per time slice. Because there are 30 time slices, there are $21^{30}$ possible sequences from which our DPNs choose. On average, choosing a sequence at random should yield the correct answer with log-probability:

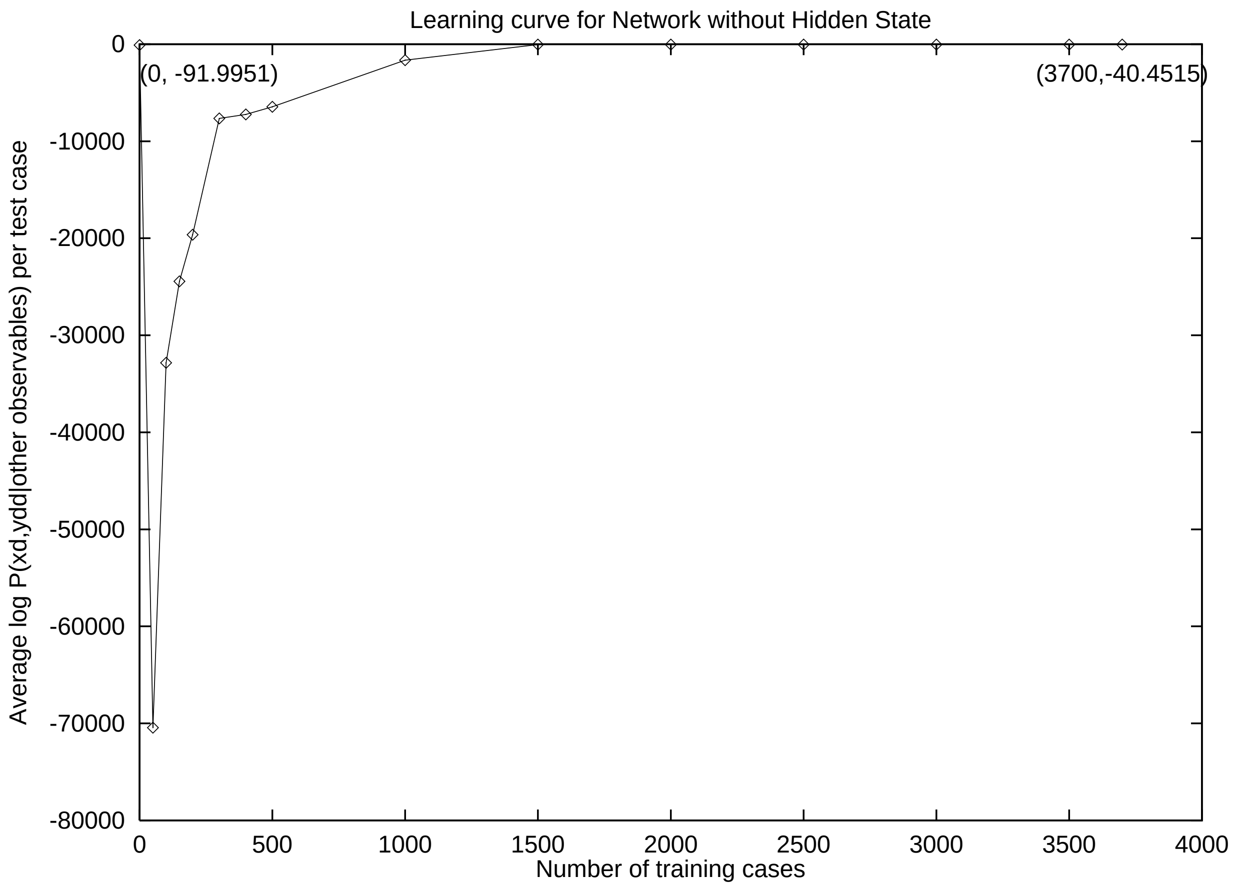


Figure 12: Learning Curve for Network With No Hidden State

$$log \frac{1}{21^{30}} \approx -91.3357 \tag{2}$$

We would like for the average log-probability to end up higher than this chance probability in all cases.

## 8.2 Network With No Hidden State

The learning curve for the network with no hidden state (the network shown in Figure 7) is given in Figure 12. The meanings of the points are as described in Section 6. Note that the average log-likelihood reached a peak of -40.4515. Recall that this is better than the do-nothing model and is comparable to the same-choice model, both of which are described earlier in this section.

We can analyze what this network has learned in more detail by examining the conditional probability tables. One example is the node for lateral velocity. According to the network shown in Figure 7, the lateral velocity directly depends on the distance to the vehicle in front, the relative speed with respect to the vehicle in front, and whether the adjacent lanes are clear for lane changing. Let us examine the case where the car in front is from 25 to 50 meters ahead and the speed of the vehicle is less than that of the car in front, which happen to be the most likely observations for variables "cif" and "relspeed," respectively. Figure 13 shows the lateral velocity distribution for the case where there are cars immediately to the left and right, which precludes the possibility of a lane change. The distribution indicates the correct behavior, which is to continue straight ahead (the car would slow down as necessary depending on the relative velocity with respect to the car in front).

Figure 14 gives the distribution for the case where there is a car on the left but no car on the right, and Figure 15 gives the distribution for the case where there is no car on the left but there is a car on the right. As indicated earlier, the traffic scenario that we used to learn these driver models has vehicles that

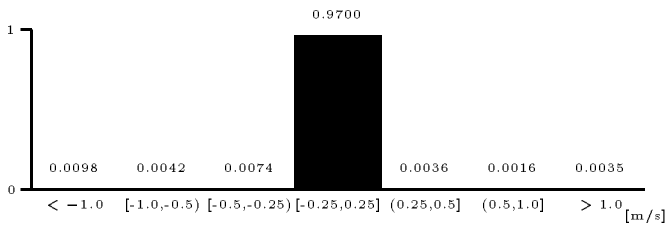


Figure 13: $P(\dot{x}|CarLeft = True, CarRight = True, cif = 25 - 50m, relspeed < 0)$

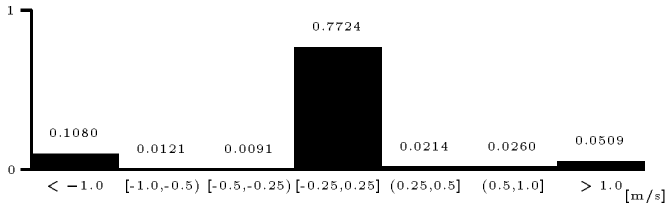


Figure 14: $P(\dot{x}|CarLeft = True, CarRight = False, cif = 25 - 50m, relspeed < 0)$

are heavily biased toward changing lanes to the left rather than the right. For the situations where there is a car on the left but no car on the right, the cases where $\dot{x} < -1.0$ correspond to completion of a left lane change, for example from lane 3 to lane 2, which places the vehicle next to another vehicle in lane 1 (an example of a car in such a situation is the car in lane 3 in Figure 8). During the completion of that lane change, negative values of $\dot{x}$ and the presence of a car to the left are reported. The mirror image case that occurs is $\dot{x} > 1.0$ when there is no car on the left but there is a car on the right. The problem here is that this DPN does not have sufficient variables to distinguish between the start and end of a lane change. In subsequent sections, we add variables to the network with the aim of separating lane changing into stages. These augmented networks confirm the above explanations.

Figure 16 gives the distribution for the case where there are no cars on either side. In this case, the cars we have observed are clearly biased toward changing lanes to the left rather than to the right. Indeed, according to the decision tree in Figure 6, if a vehicle is changing lanes only to pass a slower vehicle (that is, not with the goal of moving to its target lane) and it has a choice of which lane to change to, it will always choose to change to the left.

## 8.3 Network With Hidden State

The learning curve for the network with hidden state is given in Figure 17. The meanings of the points are as described in the previous section describing the controlled experiments.

We learned and tested this network on exactly the same scenarios as we did the network described earlier. Note that there is a marked improvement in how well we were able to predict the test data compared to the previous one (an average log-probability of -32.2737 instead of -40.4515). As we did for the previous network, we can analyze what this network has learned in more detail by examining the conditional probability tables. We first examine the CPTs for the hidden node "WantsChange," which is influenced by "relspeed" and "cif." Figure 18 displays these CPTs. From these CPTs, it is clear that the desire to change lanes, as reflected in $P(WantsChange = TRUE|relspeed, cif)$, decreases as the distance to the car in front increases and increases as the relative speed with respect to the car in front increases. This is exactly what one would expect. Note also that the largest drops in this probability occur when increasing the following distance

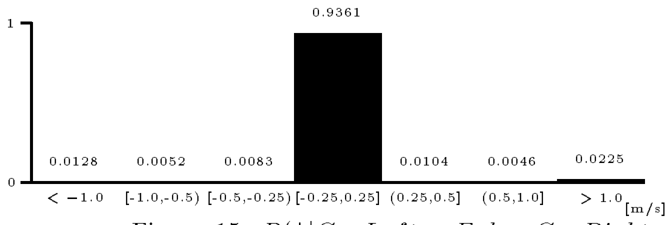


Figure 15: $P(\dot{x}|CarLeft = False, CarRight = True, cif = 25 - 50m, relspeed < 0)$

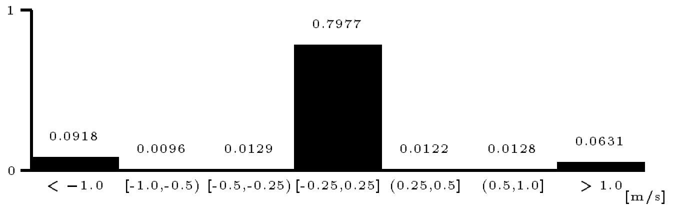


Figure 16: $P(\dot{x}|CarLeft = False, CarRight = False, cif = 25 - 50m, relspeed < 0)$

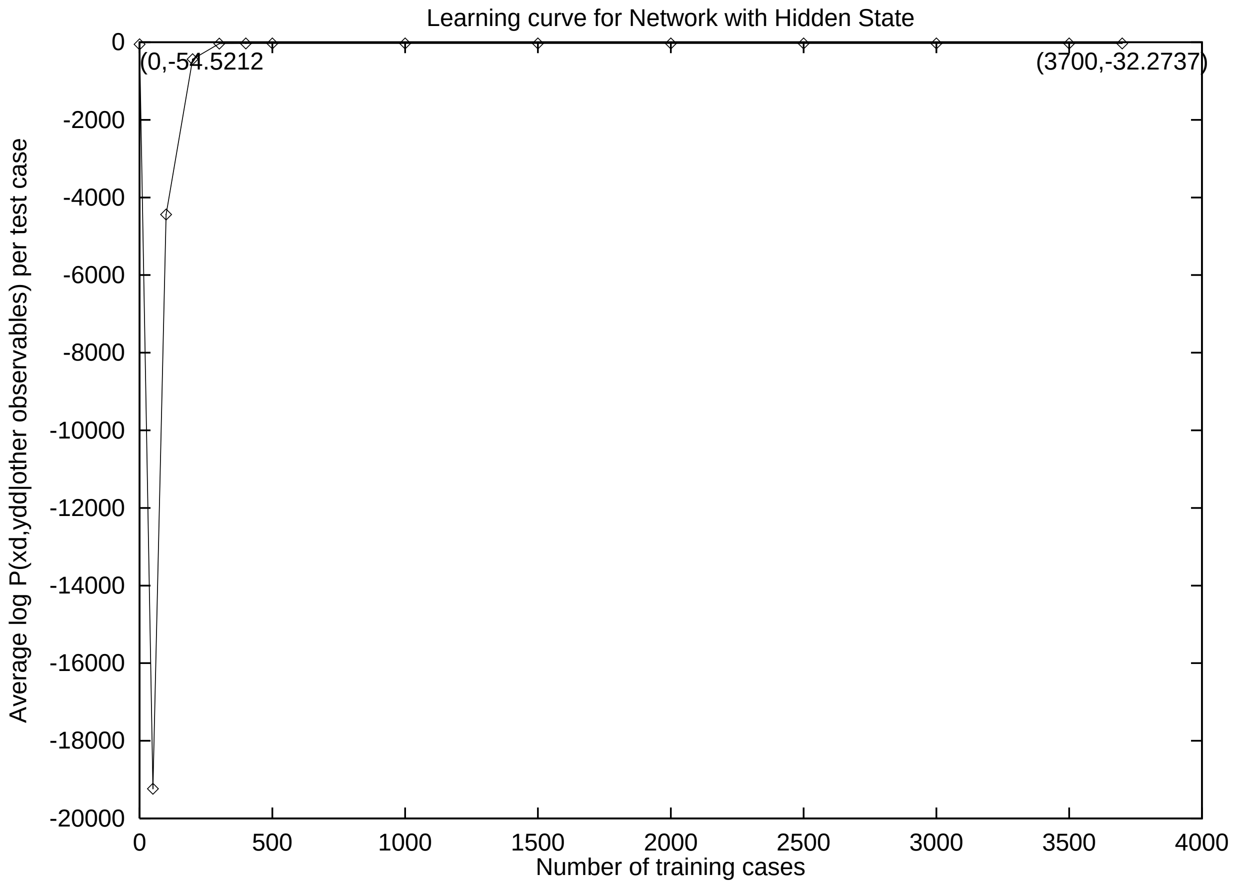


Figure 17: Learning Curve for Network With Hidden State

from 10-25m to 25-50m or 25-50m to more than 50m. This is consistent with the stochastic parameter for preferred following distance, which is normally distributed with mean 25 meters, and the following distance beyond which the vehicles ignore the car in front, which turns out to be 40 meters.

Now, we can examine the hidden node "LaneChange." As discussed earlier, "LaneChange" has five different values. We intended for the values "StartLeft" and "StartRight" to represent the beginning stage of changing lanes, "FinishLeft" and "FinishRight" to represent the completion stage of changing lanes, and "Stay" to represent the state of remaining in the current lane. We can now examine what definitions of these five values were constructed by our learning algorithm by examining the CPTs for "$\dot{x}$" and "lanepos," the two nodes that are influenced by "LaneChange."

The CPTs for $\dot{x}$ are given in Figure 19, Figure 20, and Figure 21. These correctly indicate that cars tend to go toward the left while changing to the left, tend to go toward the right while changing to the right, and tend to go straight along the road while not changing lanes.

The CPTs for "lanepos" are given in Figure 22. This combined with the CPTs for $\dot{x}$ give us different values of "LaneChange" that are well-defined. The definition of LaneChange=Stay is the most straightforward: this corresponds to just staying in the current lane and continuing straight ahead. LaneChange=StartLeft

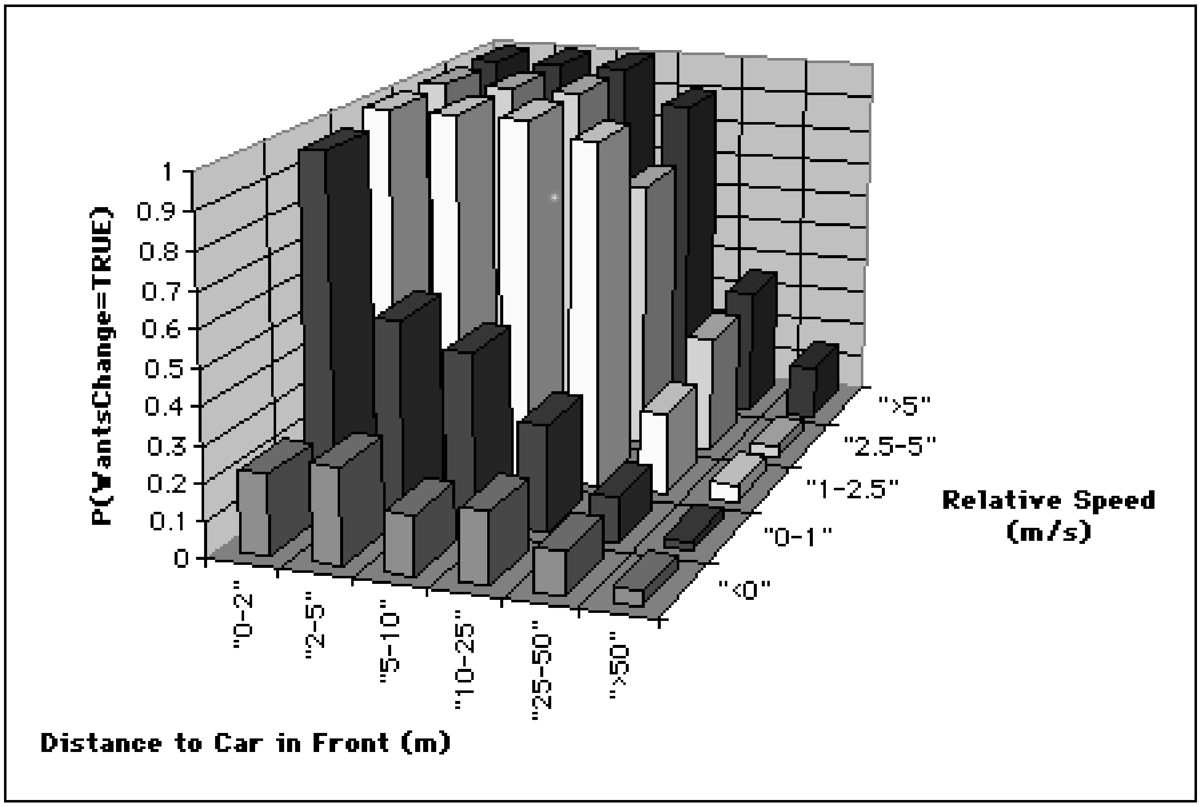


Figure 18: P(WantsChange=TRUE | relative speed, distance to front vehicle)

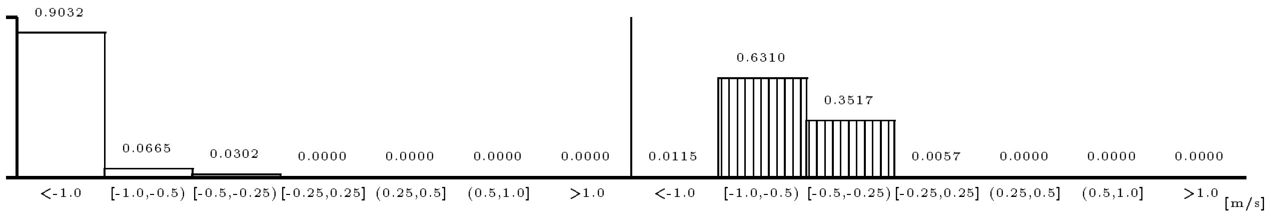
corresponds to changing lanes to the left at the maximum rate possible. Cars typically remain in this state until a short time after crossing the lane marker into the target lane. LaneChange=FinishLeft corresponds to initiating the process of centering the car in the target lane. This is why for LaneChange=FinishLeft, lanepos is skewed away from the left side of the lane and $\dot{x}$ is not as strongly skewed here as it was for LaneChange=StartLeft. Once the car is centered in the lane and is heading in the direction of the lane, LaneChange returns back to the state "Stay." The CPTs for "LaneChange" are very large so they are not presented here; however, they clearly indicate the progression of LaneChange through either the sequence of states Stay→StartLeft→FinishLeft→Stay or Stay→StartRight→FinishRight→Stay with the transitions between states consistent with their definitions.

The form of the CPTs for right lane changes and left lane changes are similar in the basic form of the distributions but have very different parameters. The distribution for $\dot{x}$ is different between left and right lane changes—the distributions are different between StartLeft and FinishLeft, but are quite similar between StartRight and FinishRight. This is quite reasonable given that right lane changes are generally done while traveling at a lower speed than left lane changes, so to complete the lane change in the same amount of time, the vehicles steer more sharply for right lane changes.

## 8.4 Network With Hidden State and Deterministic Nodes

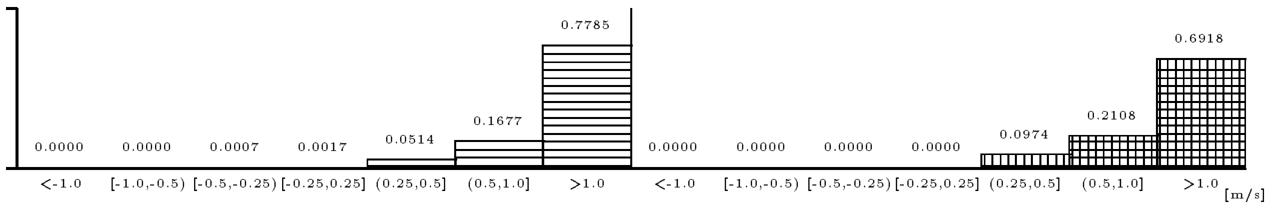
The learning curve for the network with hidden state and deterministic nodes is given in Figure 23. The meanings of the points are as described in the previous section describing the controlled experiments.

We learned and tested this network on exactly the same scenarios as we did the previous two networks. Note that there is a marked improvement in how well we were able to predict the test data compared to the previous two (an average log-probability of -28.4515 compared to -32.2737 and -40.4515). Note that this network predicts better than the routine "do nothing" model. As in the previous network, we examined the CPTs for the nodes "WantsChange" and "LaneChange." "WantsChange" is now influenced



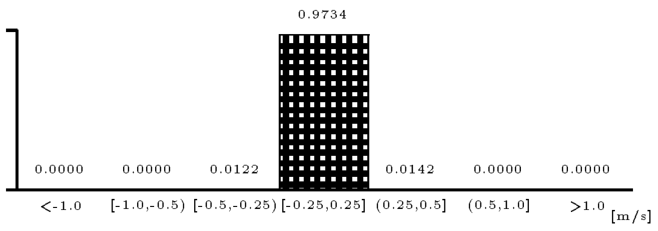
□ LaneChange = StartLeft
▥ LaneChange = FinishLeft

Figure 19: Probability of $\dot{x}$



▤ LaneChange = StartRight
▦ LaneChange = FinishRight

Figure 20: Probability of $\dot{x}$



▩ LaneChange = Stay

Figure 21: Probability of $\dot{x}$

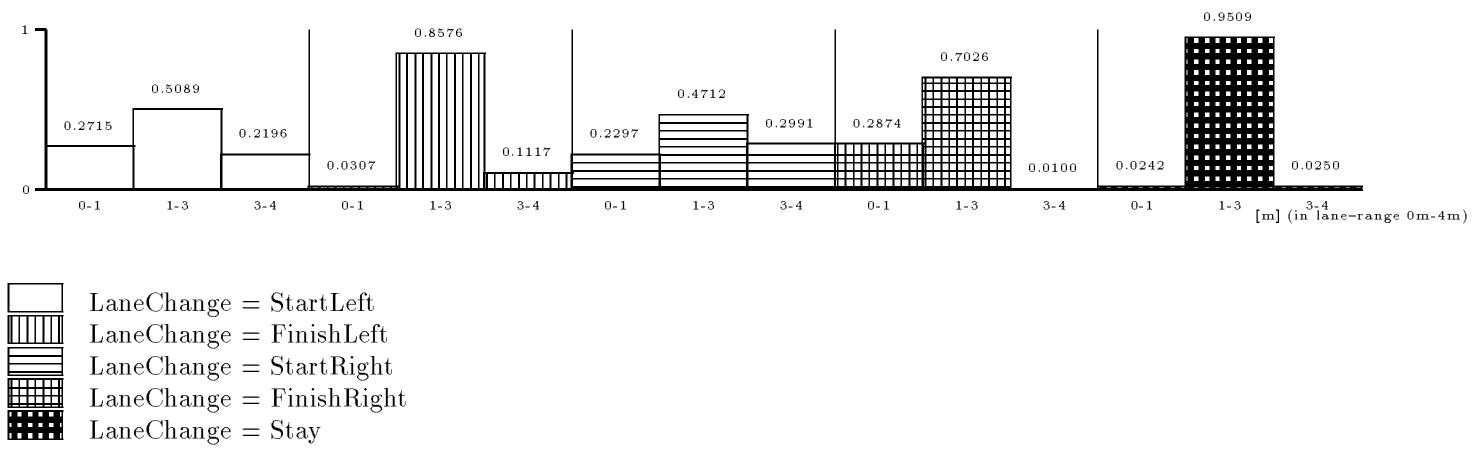


Figure 22: Probability of lanepos

by "DETdesiredspeed" (are we traveling below, at, or above the desired speed) and "cif" (distance to the vehicle in front). The CPT for "WantsChange" is revealed in Figure 24. As before, P(WantsChange=TRUE | desired speed, distance to front vehicle) decreases as the distance increases and as our speed relative to the desired speed increases, which makes sense. The CPTs for "LaneChange" and its children are not significantly different for this network compared to the previous one, so we will not discuss them separately. However, the CPTs for $\ddot{y}$ are much cleaner—the changes in the distribution of $\ddot{y}$ are more systematic as a function of the parents—because of the addition of the parent DETdesiredspeed. Because of the large number of parameters in the distribution, we will not give the CPT here, but the distribution of $\ddot{y}$ tends to increase (skew toward higher values) as our speed relative to the desired speed decreases and the relative speed with respect to the car in front decreases. In general, the acceleration also tends to increase as the distance to the car in front increases, but for longer following distances (e.g., greater than 50 meters) the acceleration tends more toward zero. This makes sense, because as mentioned earlier, the vehicles do not worry about the car in front if it is beyond 40 meters away and the preferred following distance is normally distributed about 25 meters.

## 8.5 Summary

The analysis we have done in this section is an example of a general method we can use to learn driver preferences and other aspects of the belief state. We added nodes to our DPN to represent some hidden aspect of the belief state, learned the network, and found an interpretation for the hidden nodes that is consistent with its CPTs and those of its children. In our analysis, we added nodes to our network so that their parents and children correspond to what we believe are the causes and effects of a lane changing action. We set the initial CPTs based on what we believed was a sensible relationship between the nodes given what we wanted them to represent. We found that the nodes appear to represent the desire to change lanes and the stages of changing lanes as we expected. Further examination of the CPT for the "WantsChange" node gave us information on two driver preferences: preferred following distance and the distance beyond which any car in front is ignored. In this way, we can learn other aspects of driver preference such as the preferred minimum size of the gap in a neighboring lane when changing lanes. We can also use this method to learn other high-level actions such as overtaking and following a car. Since we are using traffic data to learn these models and, ultimately, induce an interpretation upon our hidden nodes, our method is effectively finding those behaviors and attributes that are relevant to understanding real driver behavior rather than our own conception thereof. As a result, our models may even find novel methods of evaluating traffic and driver behavior.

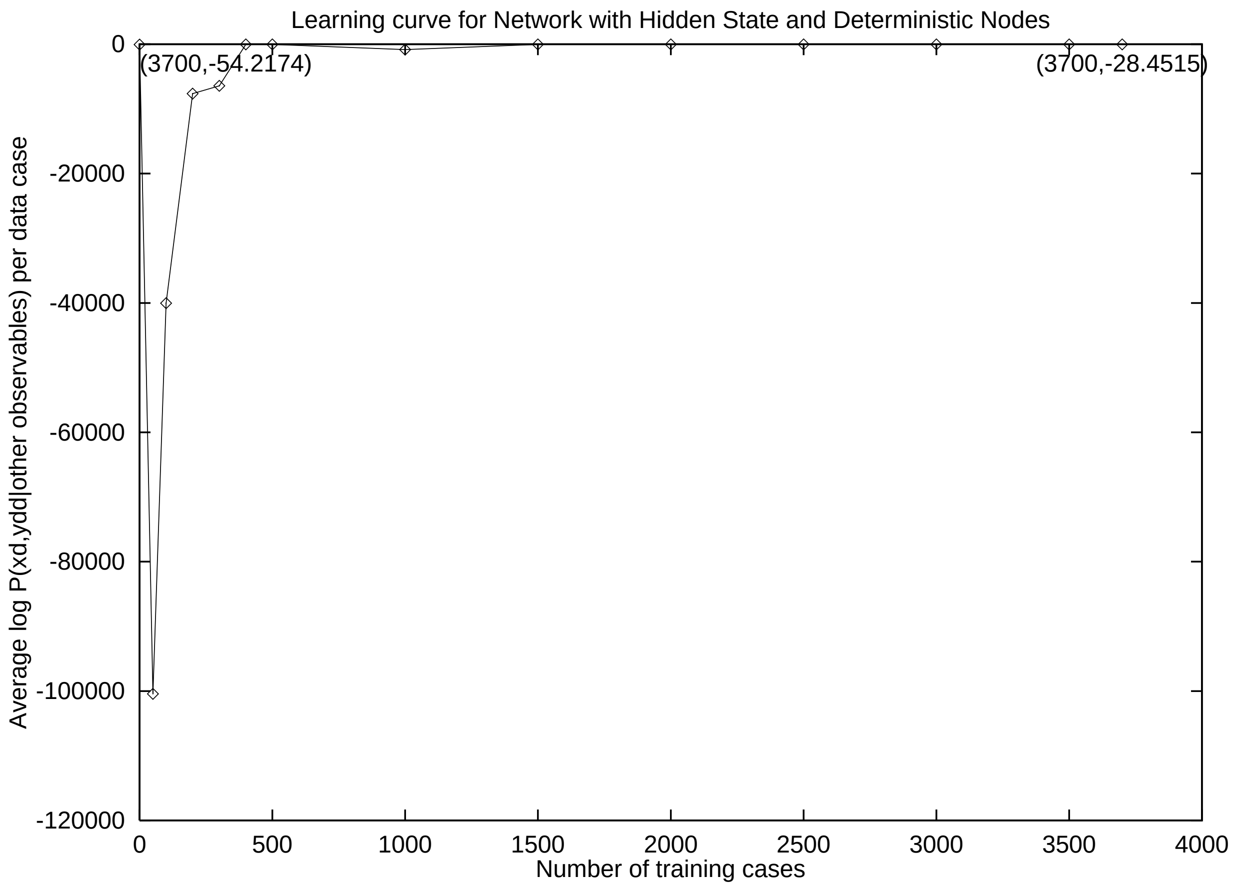


Figure 23: Learning Curve for Network With Hidden State and Deterministic Nodes

## 9 Conclusions and Future Work

We have seen that, for simple scenarios with a large number of cars and a range of congestions from uncongested to peak flow, our models correctly predict driver behavior even though the training data often contain missing values. We have seen that the addition of hidden state can dramatically improve the ability to learn. We need to learn driver behavior for more complicated scenarios—in particular, scenarios with flows near the maximum, where drivers have to take many other cars' actions into account, thereby testing the drivers' cognitive abilities. Attention spans, reaction times, and the ability to observe only a limited part of the environment at any instant may play a more crucial role in modeling driver behavior in higher-flow scenarios. Incorporating other characteristics such as the type of vehicle being driven, the minimum and preferred gaps between vehicles in a neighboring lane when changing lanes, and the driver's intended route may also be helpful in modeling driver behavior more accurately. As described above, we may find new state variables that are useful in modeling driver behavior and analyzing traffic.

Modeling in these more complicated scenarios will require that we use our BAT simulator to yield these higher-flow scenarios through the addition of accidents, stalled vehicles, large trucks, and other events that cause increased density and lower inter-vehicle spacing. We will also need to use continuous variables in our DPN models in many cases where it is natural (such as speeds and accelerations).

We would ultimately like to learn the behavior of real vehicles rather than simulated vehicles. For this, we need computer vision-based vehicle trackers that give us a sequence of observations for every vehicle. Currently, we rely on having accurate bounding boxes on vehicles, thereby giving us accurate locations of and distances between vehicles. We do have access to a videotape library of vehicle traffic, however, all the views of the cars are from cameras on the side of the road aimed downstream similar to what we have in our simulator (Figure 5). There is work in progress here at Berkeley on setting up cameras at a greater

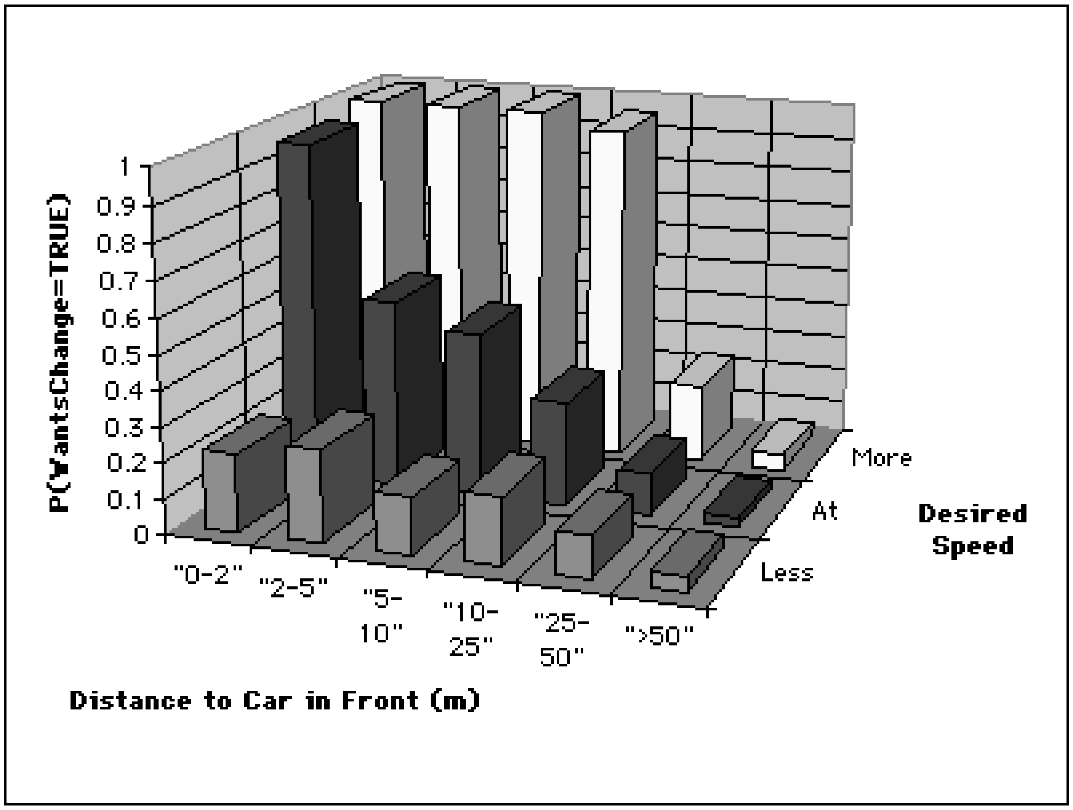


Figure 24: P(WantsChange=TRUE | desired speed, distance to front vehicle)

altitude and at several orientations ranging from perpendicular to the traffic flow to downstream as we have had so far. The greater altitude will reduce occlusion and the altitude combined with the different orientations will allow us to get more accurate measurements of vehicle sizes and locations. For example, the camera aimed perpendicular to the road will get the best measurement of the lengths of the vehicles and their position along the road, thereby allowing better measurements of the distances between vehicles. The cameras aimed downstream will tend to give better measurements of vehicle width and location within their lanes. Additional work is also needed to avoid the now frequent occurrence of only finding small parts of vehicles [20]. Improvements in the cameras, vision algorithms, and possible applications of probabilistic reasoning will likely give more reliable data. Once these improvements are in place, we should be able to get more accurate data on real vehicles.

## 10 Acknowledgments

Thanks to Geoff Zweig for his extensive help with the BAY-SIM belief network package, Jeff Forbes for his extensive assistance with the BAT simulator, and Stuart Russell for many helpful discussions of this material. Thanks to Stuart Russell and Jitendra Malik for reviewing this report. Special thanks go to my parents for their general support and for giving that additional push to finish this report.

## 11 References

[1] Jeff Forbes, Tim Huang, Keiji Kanazawa, Stuart Russell, "The BATMobile: Towards a Bayesian Automated Taxi," *Proceedings IJCAI-95* (1995). Montreal, Canada

[2] Win Van Winsum, Adriaan Heino, "Choice of time-headway in car-following and the role of time-to-collision information in braking," *Ergonomics, Vol. 39, No.4* (1996)

[3] A. Dempster, N. Laird, D. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society, 39* (Series B):1-38, 1977.

[4] D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao, S. Russell, "Toward Robust Automatic Traffic Scene Analysis in Real-Time," *Proc. ICPR-94* Tel-Aviv, Israel

[5] Thomas Benz "The Microscopic Traffic Simulator AS (Autobahn Simulator)," *Proc. 1993 European Simulation Multiconference* Lyon, France.

[6] Axel Niehaus, Robert F. Stengel "Probability-Based Decision Making for Automated Highway Driving," *IEEE Transactions on Vehicular Technology, Vol. 43, No. 3* (1994)

[7] Geoffrey Zweig, *BAY-SIM Belief Network Software Package* (1997)

[8] Richard Haberman, *Mathematical Models in Mechanical Vibrations, Population Dynamics, and Traffic Flow* (1977) Prentice Hall, Inc., Englewood Cliffs, New Jersey.

[9] Judea Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* (1990) Morgan Kaufmann, San Mateo, California.

[10] Thomas Dean, Keiji Kanazawa, "Probabilistic Temporal Reasoning," *Proceedings AAAI-88* (1988) Minneapolis, Minnesota.

[11] Alex. P. Pentland, Andrew Liu, "Observing Drivers Intention," *Nissan Cambridge Basic Research Center Technical Report No. TR96-2* (1996) Cambridge, Massachusetts.

[12] David Heckerman, "A Tutorial on Learning With Bayesian Networks," *Microsoft Research Technical Report No. MSR-TR-95-06* (1995) Redmond, Washington.

[13] Jeffrey Forbes, Nikunj Oza, Ronald Parr, Stuart Russell, "Feasibility Study of Fully Automated Vehicles Using Decision-Theoretic Control," *California PATH Research Report UCB-ITS-PRR-97-18* (1997) Institute of Transportation Studies, University of California, Berkeley, California.

[14] M.J. Lighthill and G.B. Whitham, "On Kinematic Waves II. A Theory of Traffic Flow on Long Crowded Roads," *Proceedings of the Royal Society A, 229*, pp. 317-345 (1955).

[15] P.I. Richardson, "Shock Waves on the Highway," *Operations Research 4*, pp. 42-51 (1956).

[16] Carlos Daganzo, *An Informal Introduction to Transportation and Traffic Operations* (1995) Draft of Book, Institute of Transportation Studies, University of California, Berkeley, California.

[17] Qi Yang and Haris N. Koutsopoulos, "A Microscopic Traffic Simulator for Evaluation of Dynamic Traffic Management Systems," *Transportation Research Part C (Emerging Technologies)*, vol. 4C, (no. 3), pp. 113-129.

[18] Axel Niehaus and Robert F. Stengel, "An Expert System for Automated Highway Driving," *IEEE Control Systems Magazine*, vol. 11, (no.3), April 1991, pp. 53-61.

[19] Michael J. Cassidy, "Unique Bivariate Relationships in Highway Traffic," *Institute of Transportation Studies Research Report UCB-ITS-RR-96-5* June 1996, University of California, Berkeley, California.

[20] David Beymer, Philip McLauchlan, Benn Coifman, and Jitendra Malik, "A Real-time Computer Vision System for Measuring Traffic Parameters," *Proc. IEEE CVPR*, Puerto Rico, June 1997, pp. 495-501.