On Learning Discontinuous Human Control Strategies*

Michael C. Nechyba,[1,]† Yangsheng Xu[2,]‡
[1]*Department of Electrical and Computer Engineering, Machine Intelligence Laboratory, Benton 311, P.O. Box 116200, University of Florida, Gainesville, Florida 32611-6200*
[2]*Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong*

Models of human control strategy (HCS), which accurately emulate dynamic human behavior, have far reaching potential in areas ranging from robotics to virtual reality to the intelligent vehicle highway project. A number of learning algorithms, including fuzzy logic, neural networks, and locally weighted regression exist for modeling continuous human control strategies. These algorithms, however, may not be well suited for modeling discontinuous human control strategies. Therefore, we propose a new stochastic, discontinuous modeling framework, for abstracting human control strategies, based on hidden Markov models (HMM). In this paper, we first describe the real-time driving simulator which we developed for investigating human control strategies. Next, we demonstrate the shortcomings of a typical continuous modeling approach in modeling discontinuous human control strategies. We then propose an HMM-based method for modeling discontinuous human control strategies. The proposed controller overcomes these shortcomings and demonstrates greater fidelity to the human training data. We conclude the paper with further comparisons between the two competing modeling approaches and we propose avenues for future research. © 2001 John Wiley & Sons, Inc.

1. INTRODUCTION

In recent years, a number of different researchers have endeavored to abstract models of human skill directly from observed human input−output data.[1,2] Much of the work to date attempts to model human skill by *learning* the mapping from sensory inputs to control action outputs. Although the choice of learning algorithm varies, the most frequently used—including fuzzy logic, neural networks, and locally weighted regression—are all examples of *continu-*

*This work was supported in part by Carnegie Mellon University and graduate fellowships from the National Science Foundation and the Department of Energy.
†Author to whom correspondence should be addressed; e-mail: nechyba@mil.ufl.edu.
‡e-mail: ysxu@mae.cuhk.edu.hk.

ous function approximators (FAs). For each of these algorithms, control outputs are continuous and deterministic functions of model inputs.

Powerful as these may be, however, continuous learning algorithms may not be able to faithfully reproduce control strategies where discrete events or decisions introduce discontinuities in the input–output mapping. An example of this type of discontinuous control occurs in human driving, which requires control of (1) steering and (2) acceleration. While steering tends to vary continuously with model inputs, acceleration control of the vehicle is decidedly discontinuous, since it involves explicit switching between the gas and the brake pedals.

To adequately model such control behavior, we therefore propose a new stochastic, discontinuous learning algorithm, based on hidden Markov models, where control actions are modeled as individual HMMs. During run-time execution of the algorithm, a control action is then selected stochastically, as a function of both prior probabilities and posterior HMM-evaluation probabilities. Thus, we model the discontinuous acceleration control not as a deterministic functional mapping (as we do with the continuous steering control), but rather as a probabilistic relationship between sensory inputs and discontinuous outputs.

In this paper, we first review previous work in modeling human driving. We then describe the real-time graphic driving simulator for which we have recorded human control data, and for which we wish to abstract the corresponding driving control strategies. Next, we illustrate the difficulty of modeling a discontinuous control strategy using a continuous learning framework, and we propose a new HMM-based, discontinuous learning architecture for abstracting discontinuous human control strategies. We show that the resulting discontinuous modeling framework demonstrates better fidelity to the human training data than the continuous modeling approach. Finally, we offer some comparisons between the two learning architectures and we suggest directions for future research.

2. REAL-TIME DRIVING SIMULATOR

Several approaches to skill learning in human driving have been implemented. Neural networks have been trained to mimic human behavior for a simulated, circular racetrack.[3,4] The task essentially involves avoiding other computer-generated cars; no dynamics are modeled or considered in the approach. Pomerleau implements real-time road-following with data collected from a human driver.[5,6] A static feedforward neural network with a single hidden layer, ALVINN, learns to map coarsely digitized camera images of the road ahead to a desired steering direction, whose reliability is given through an input-reconstruction reliability estimator. The system has been demonstrated successfully at speeds up to 70 mi/h. Subsequently, a statistical algorithm called RALPH[7] was developed for calculating the road curvature and the lateral offset from the road median. Neuser et al. control the steering of an autonomous vehicle through preprocessed inputs to a single-layer feedforward neural network.[8] These preprocessed inputs include the car's yaw angle with respect to the road, the instantaneous and time-averaged road curvature, and the instantaneous and time-averaged lateral offset. Driving data is again collected from a human operator. Other authors provide a control theoretic model of human driver steering control.[9] Finally, Pentland and Liu apply HMMs toward inferring a particular driver's high-level intentions, such as turning and stopping.[10]

In our work, we are interested in abstracting models of *dynamic* human control strategies, including steering *and* acceleration. Figure 1 shows the real-time, dynamic, graphic driving simulator which we developed for collecting and analyzing human control strategy data. In the interface, the human operator has full control over the steering of the car, the brake and the accelerator, although the simulator does not allow both the gas and brake pedals to be pushed at the same time. The state of the car is given by $\{v_\xi, v_\eta, \omega\}$, where v_ξ is the lateral velocity of the car, v_η is the longitudinal velocity of the car, and ω is the angular velocity of the car; the controls are given by

$$-8000\,\text{N} \le \phi \le 4000\,\text{N} \tag{1}$$

$$-\,0.2\,\text{rad} \le \delta \le 0.2\,\text{rad} \tag{2}$$

where ϕ is the user-applied longitudinal force on the front tires and δ is the user-applied steering angle. The nonlinear dynamic model of the car—omitted here for space reasons—is fully described elsewhere.[1,11,12]

Because of input device constraints, the force (or acceleration) control ϕ is limited during each $1/50$ s time step, based on its present value. If the gas pedal is currently being applied ($\phi > 0$), then the operator can either increase or decrease the amount of applied force by a user-specified constant $\Delta\phi_g$ or switch

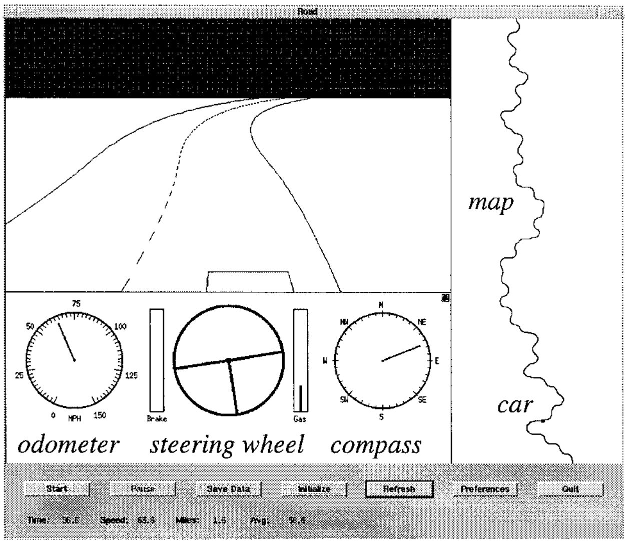


Figure 1. The driving simulator gives the user a perspective preview of the road ahead. The user has independent controls of the steering, brake, and accelerator (gas).

to braking. Similarly, if the brake pedal is currently being applied ($\phi < 0$) the operator can either increase or decrease the applied force by a second user-specified constant $\Delta\phi_b$ or switch to applying positive force. Thus, the $\Delta\phi_g$ and $\Delta\phi_b$ constants define the responsiveness of each pedal. If we denote $\phi(k)$ as the current applied force and if we denote $\phi(k+1)$ as the applied force for the next time step, we can write in concise notation that,

$$\phi(k+1) \in \left\{\phi(k), \min\left(\phi(k) + \Delta\phi_g, 4000\right), \max\left(\phi(k) - \Delta\phi_g, 0\right), -\Delta\phi_b\right\} \qquad \phi(k) \geq 0 \quad (3)$$

$$\phi(k+1) \in \left\{\phi(k), \max(\phi(k) - \Delta\phi_b, -8000), \min(\phi(k) + \Delta\phi_b, 0), \Delta\phi_g\right\} \qquad \phi(k) < 0 \quad (4)$$

For the experiments in this paper, we collect human driving data across randomly generated roads like the 20 km one shown in the map of Figure 1. The roads are described by a sequence of (1) straight-line segments and (2) circular arcs, connected in a manner that ensures continuous first derivatives across segments. The length of each straight-line segment, as well as the radius of curvature of each arc, lie between 100 and 200 m. Roads are defined to be 10 m wide, and the visible horizon is set at 100 m. For notational convenience, let d_ξ denote the car's lateral offset from the road median.

3. CONTINUOUS CONTROL

Below, we motivate the development of the discontinuous HMM-based learning architecture by first illustrating the learning problems that occur when attempting to model a discontinuous control strategy with a continuous learning architecture. For given human driving data, steering will tend to vary *continuously* with sensory inputs, while acceleration will tend to vary *discontinuously* with sensory inputs. This is not merely an artifact of the constraints in Eqs. 3 and 4, but is caused primarily by the necessary switching between the brake and the gas pedals, as is also the case for real driving. As we will show, continuous HCS models, while abstracting convergent strategies, qualitatively bear little resemblance to the original human control strategy. This is not only a shortcoming of our neural-network function approximators; rather, we will show that *any* continuous function approximator is doomed to fail in a similar manner when attempting to model control strategies that involve discontinuous switching.

A. Cascade Learning

Here, we briefly summarize the cascade neural network learning architecture. Further details, which are omitted for space reasons, may be found elsewhere.[2,11,13] Initially, there are no hidden units in the network, only direct input–output connections which are trained first. When no appreciable error reduction occurs, a first hidden unit is added to the network from a pool of

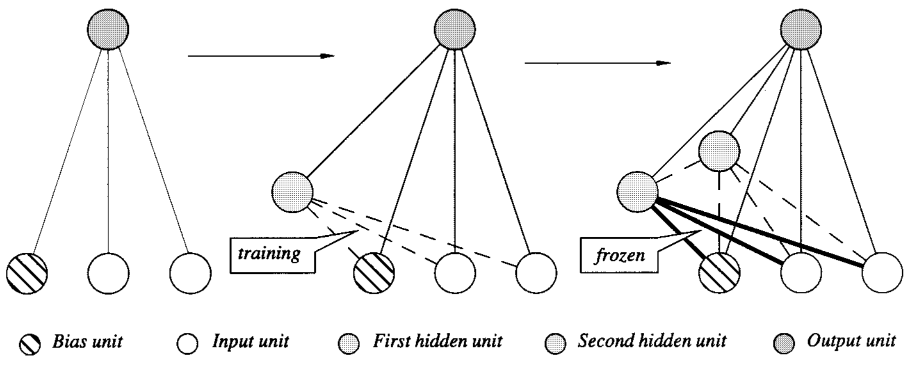


Figure 2. The cascade learning architecture adds hidden units one at a time to an initially minimal network. All connections in the diagram are feedforward. The striped circles represent the bias unit, the open circles represent the input unit, the lightly shaded circles represent the first hidden unit, the shaded circles represent the second hidden unit, and the dark shaded circles represent the output unit.

candidate units, which are trained independently and in parallel with different random initial weights. Once installed, the hidden unit input weights are frozen, while the weights to the output units are retrained. This process which is illustrated in Figure 2, is repeated with each additional hidden unit, which receives input connections from both the network inputs and all the previous hidden units, resulting in a cascading structure.

In the experiments reported in this paper, we enhance the basic cascade learning framework in two ways: (1) we allow new hidden units to have variable activation functions,[11] increasing the functional flexibility of the learning architecture; and (2) we train the neural-network weights through node-decoupled extended Kalman filtering (NDEKF),[4] as opposed to gradient techniques, such as quickprop or backpropagation. Both of these modifications have been shown to significantly improve learning speed and error convergence of the cascade learning architecture.[1,14]

A necessary condition for successful learning is, of course, that the model be presented with those state and environmental variables upon which the human operator relies. We hypothesize that the human's control strategy, which we are trying to learn, relies not only on the present state of the car, but also a recent time history of the car; experiments, which vary the number and type of inputs to the learning model, support this hypothesis.[1] Thus, we need to map a *dynamic* system (i.e., the human control strategy) onto the *static* mapping capacity of the cascade learning architecture.

In general, we can approximate any dynamic system through a difference equation,[15]

$$\mathbf{u}(k+1) = \Gamma\big[\mathbf{u}(k), \mathbf{u}(k-1), \ldots, \mathbf{u}(k-n_u+1), \mathbf{x}(k), \mathbf{x}(k-1), \ldots, \mathbf{x}(k-n_x+1), \mathbf{z}(k)\big] \quad (5)$$

where $\Gamma(\cdot)$ is some (possibly nonlinear) map, $\mathbf{u}(k)$ is the control vector, $\mathbf{x}(k)$ is the system state vector, and $\mathbf{z}(k)$ is a vector describing the external environment at time step k. The order of the dynamic system is given by the constants n_u and n_x. Thus, a static model can abstract a dynamic system, provided that time-delayed histories of the state and command vectors are presented to the model as input. Consequently, the inputs to the cascade neural network should include, (1) current and previous state information $\{\nu_\xi, \nu_\eta, \omega\}$, (2) previous output (control) information $\{\delta, \phi\}$, and (3) a description of the road visible from the current car position. More precisely, the network input vector $\boldsymbol{\zeta}(k)$ at time-step k is given by

$$\{\nu_\xi(k-n_x),\ldots,\nu_\xi(k-1),\nu_\xi(k),\nu_\eta(k-n_x),\ldots,\nu_\eta(k-1),\nu_\eta(k),\omega(k-n_x),\ldots,\omega(k-1),\omega(k)\} \tag{6}$$

$$\{\delta(k-n_u),\ldots,\delta(k-1),\delta(k),\phi(k-n_u),\ldots,\phi(k-1),\phi(k)\} \tag{7}$$

and

$$\{x_1(k),x_2(k),\ldots,x_{n_r}(k),y_1(k),y_2(k),\ldots,y_{n_r}(k)\} \tag{8}$$

For the road description, we partition the visible view of the road ahead into n_r equivalently spaced, body-relative (x, y) coordinates of the road median, and we provide that sequence of coordinates as input to the network. Thus, the total number of inputs to the network are $3n_x + 2n_u + 2n_r$. The outputs of the cascade network are $\{\delta(k+1), \phi(k+1)\}$, the steering and acceleration commands at the next time step, respectively.

B. Experiments

We now conduct the following experiment. First, we ask a human operator to drive over two different randomly generated 20 km roads ρ_1 and ρ_2, similar to the one in Figure 1. We then use the first run (ρ_1) to train a cascade neural network, and we reserve the second road ρ_2 for testing the network model. In this paper, we present results for two different individuals, Curly and Moe. By searching the space of possible inputs parameterized by $\{n_x, n_u, n_r\}$, we arrive at the following suitable input space representation for each individual's model,

$$\text{Curly: } n_x = n_u = 6 \qquad n_r = 10 \tag{9}$$

$$\text{Moe: } n_x = n_u = 3 \qquad n_r = 10 \tag{10}$$

These input representations ensure that the cascade neural networks form convergent control models.[1] Figures 3 and 4 compare each individual's control data with their corresponding model-generated control over test road ρ_2, while Table I compares some aggregate statistics between the human and the model data. For each person, the model consists of a simple linear cascade network (i.e., no hidden units).

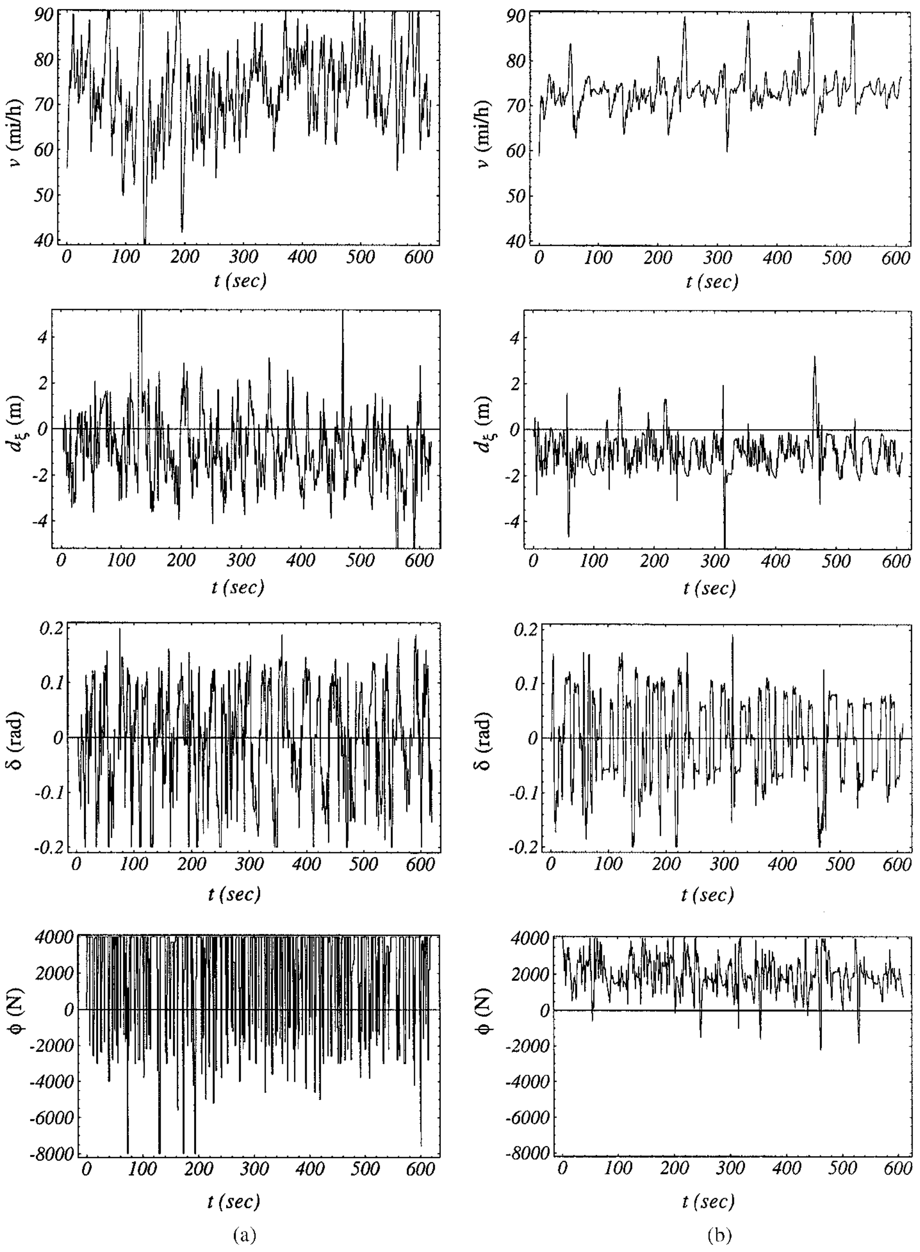


Figure 3. Curly's (a) control data and (b) corresponding linear model data over the test road.

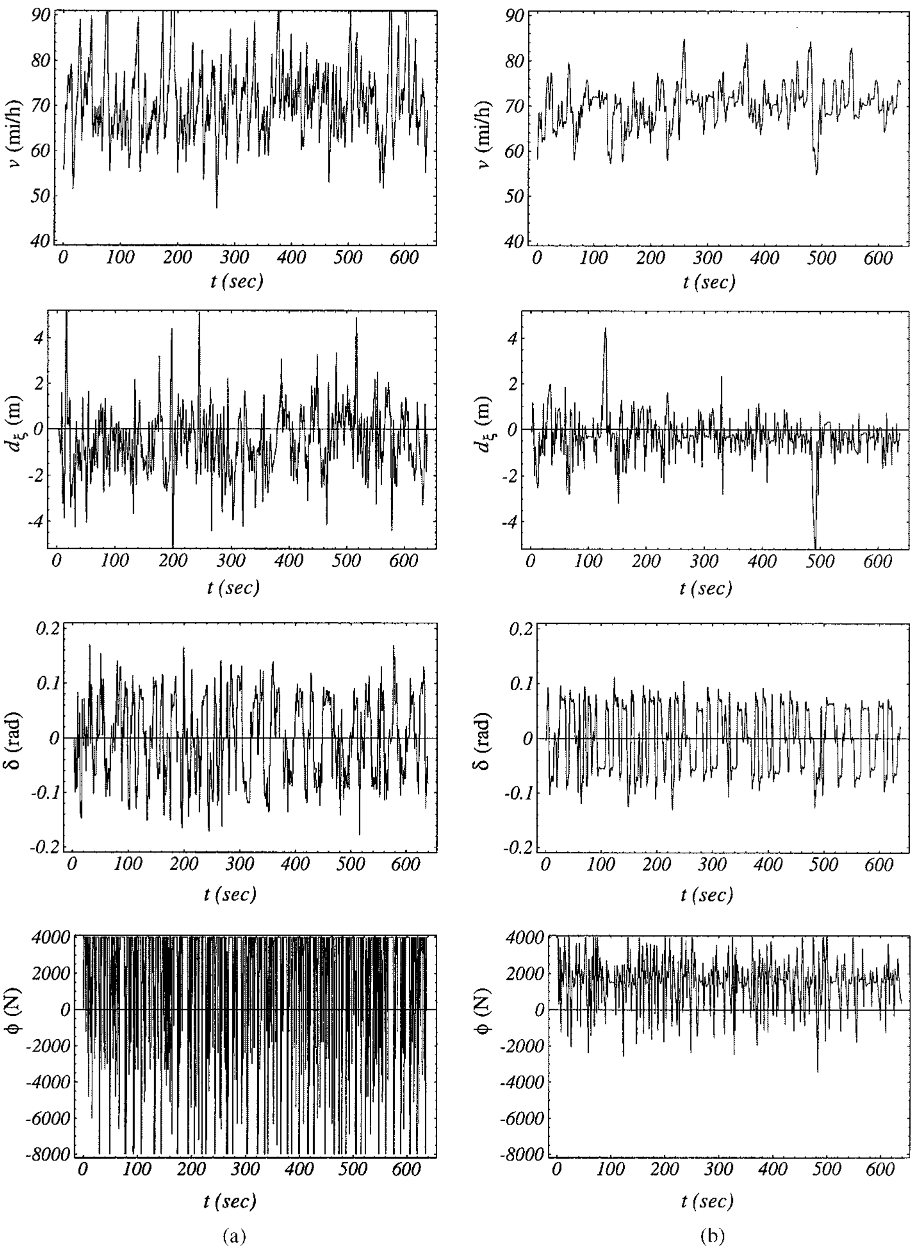


Figure 4. Moe's (a) control data and (b) corresponding linear model data over the test road.

Table I. Statistical comparison.

	Curly		Moe	
Road ρ_2	Human Data	Linear Model	Human Data	Linear Model
ν (mph)	73.1 ± 9.5	74.1 ± 3.6	70.8 ± 8.3	71.0 ± 3.7
d (m)	-0.77 ± 1.97	-1.05 ± 0.58	-0.53 ± 1.42	-0.34 ± 0.56
δ (rad)	± 0.092	± 0.066	± 0.073	± 0.056
ϕ (N)	2190 ± 2770	1790 ± 810	1850 ± 3340	1490 ± 1090

C. Discussion

From Figures 3 and 4 and Table I, we make several observations. First, Curly's and Moe's driving behavior is representative of other runs recorded by them, as well as other individuals, in that (1) the steering control is reasonably continuous; (2) the acceleration control has significant discontinuities due to rapid switching between the brake and gas pedals; and (3) Curly and Moe manage to stay on the road (± 5 m deviation from the road median) for much of the time, with only a few brief off-road episodes in especially tight turns.

Perhaps most importantly, the linear models, despite the discontinuous acceleration command, are able to learn *something*; that is, the models keep the vehicle on the road. Not only that, but they do so at approximately the same average speed and lateral distance from the road median using a similar steering control strategy as Curly and Moe, respectively. In some respects, the models' control can even be considered superior to their respective human counterparts; each model engages the brake only sparingly, and maintains tighter lateral road position.

If we judge the models on how faithfully they reproduce each individual's acceleration control strategy, however, the models rate significantly worse; that is, neither Curly's nor Moe's model acceleration control looks anything like the original human data. Adding hidden units to impart nonlinearity to the model introduces additional high-frequency components to the control, but does not improve model fidelity to the human data much. Figure 5, for example, illustrates Curly's model acceleration control when two hidden units are introduced to the model.

To better appreciate why this is the case, we would like to visualize how different input vectors in the training data map to different acceleration outputs $\phi(k+1)$. As an example, consider Curly's control strategy data and let $n_x = n_u = 6$, $n_r = 10$ as before. For these input space parameters, the input training vectors $\boldsymbol{\zeta}(k)$ are of length 50. Since it is impossible to visualize a 50-dimensional input space, we decompose each of the input vectors $\boldsymbol{\zeta}(k)$ in the training set into the principal components (PCs)[16] over Curly's entire data set, such that,

$$\boldsymbol{\zeta}(k) = c_1^k \gamma_1 + c_2^k \gamma_2 + \cdots c_{50}^k \gamma_{50} \tag{11}$$

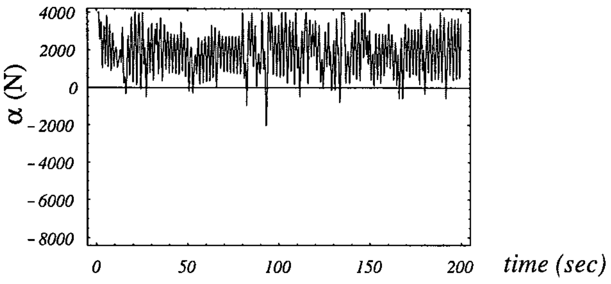


Figure 5. Adding hidden units to the linear model does not bring Curly's model significantly closer to his acceleration control strategy.

where γ_i is the principal component corresponding to the ith largest eigenvalue σ_i. Now, for Curly's control data we have that

$$|\sigma_2/\sigma_1| = 0.44 \qquad |\sigma_i/\sigma_1| \leq 0.05 \qquad i \in \{3, 4, \ldots, 50\} \tag{12}$$

so that we coarsely approximate the input vectors $\boldsymbol{\zeta}(k)$ as

$$\boldsymbol{\zeta}(k) = c_1^k \gamma_1 + c_2^k \gamma_2 \tag{13}$$

By plotting the PC coefficients (c_1^k, c_2^k) in two-dimensional space, we can now visualize the approximate relative location of the input vectors $\boldsymbol{\zeta}(k)$. Figure 6(a) and (b) shows the results for $\phi(k) < 0$ (brake), and $\phi(k) \geq 0$ (gas), respectively. In each plot, we distinguish points by whether or not $\phi(k+1)$ indicates a discontinuity (i.e., a switch between braking and accelerating) such that

$$\phi(k) < 0 \quad \text{and} \quad \phi(k+1) > 0 \; [\text{Fig. 6(a)}] \tag{14}$$

or

$$\phi(k) > 0 \quad \text{and} \quad \phi(k+1) < 0 \; [\text{Fig. 6(b)}] \tag{15}$$

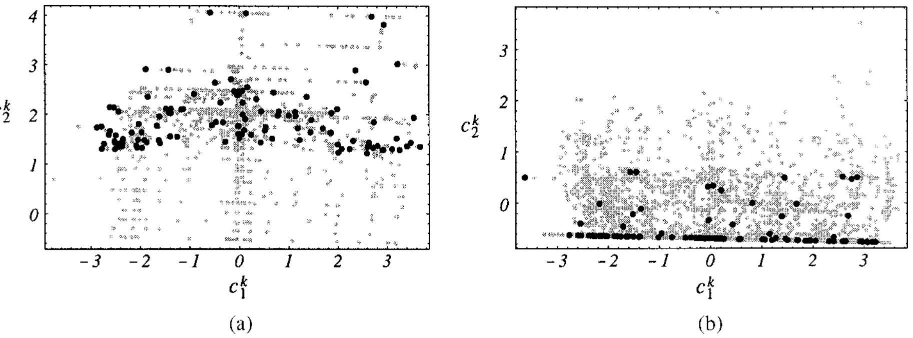


Figure 6. Switching actions (black) significantly overlap other actions (gray) when the current applied force is (a) negative (brake), and (b) positive (gas).

Those points that involve a switch are plotted in black, while a representative sample (20%) of the remaining points are plotted in gray.

We immediately observe from Figure 6 that—at least in the low-dimensional projection of the input vectors—the few training vectors that involve a switch overlap the many other vectors that do not. In other words, very similar inputs $\zeta(k)$ potentially lead to radically different outputs $\phi(k+1)$. Consequently, Curly's acceleration control strategy may not be easily expressible in a functional form, let alone a smooth functional form. This poses an impossible learning challenge not just for cascade neural networks, but for *any* continuous function approximator. In theory, no continuous function approximator will be capable of modeling the switching of the acceleration control ϕ (Fig. 7).

4. DISCONTINUOUS CONTROL

In this section, we propose a stochastic, discontinuous learning algorithm to overcome the problems discussed above. As before, we use a cascade neural network for the steering control δ; now, however, we model the acceleration control ϕ through individual statistical models. During run-time execution of the algorithm, a control action is then selected stochastically, as a function of both prior probabilities and posterior evaluation probabilities. We will show that the resulting controller overcomes the shortcomings of continuous modeling approaches in modeling discontinuous control strategies, and that the resulting model strategies appear to exhibit a higher degree of fidelity to the human training data. Figure 8 illustrates the overall continuous–discontinuous hybrid modeling architecture.

A. General Statistical Framework

Below, we view the discontinuous acceleration control not as a deterministic functional mapping (as we do with the continuous steering control), but rather as a probabilistic relationship between sensory inputs and discontinuous outputs. For now, we make the following assumptions. First, assume a control task where at each time-step k, there is a choice of one of N different control actions A_i, $i \in \{1, \ldots, N\}$. Second, assume that we have sets of input vector

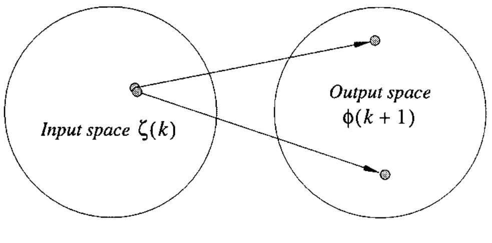


Figure 7. Switching causes very similar inputs to be mapped to radically different outputs.

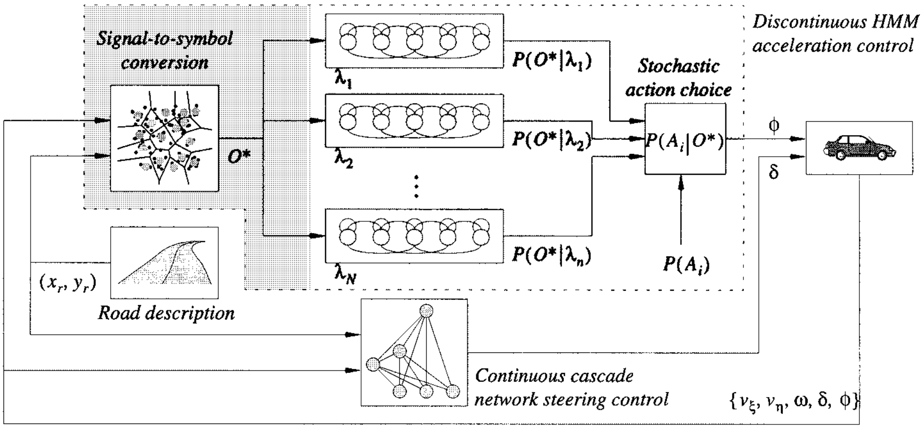


Figure 8. Overall control structure. Steering is controlled by a cascade neural network, while the discontinuous acceleration command is controlled by the statistical HMM-based controller (shaded box).

training examples $\{\zeta_i^j\}$, $j \in \{1, 2, \ldots, n_i\}$, where each ζ_i^j leads to control action A_i at the next time step. Finally, assume that we can train statistical models λ_i, so that

$$\prod_{j=1}^{n_i} P(\lambda_i \mid \zeta_i^j), \qquad i \in \{1, \ldots, N\} \tag{16}$$

is maximized, where $P(\lambda_i \mid \zeta_i^j)$ denotes the probability of model λ_i given ζ_i^j. Given an unknown input vector ζ^*, we would like to choose an appropriate, corresponding control action A^*. Since model λ_i corresponds to action A_i, we define

$$p(\zeta^* \mid A_i) \equiv p(\zeta^* \mid \lambda_i) \tag{17}$$

where $p(\zeta^* \mid A_i)$ denotes the likelihood of ζ^* given A_i. By Bayes rule,

$$P(A_i \mid \zeta^*) = \frac{p(\zeta^* \mid A_i) P(A_i)}{p(\zeta^*)} \tag{18}$$

where

$$p(\zeta^*) \equiv \sum_{i=1}^{N} p(\zeta^* \mid A_i) P(A_i) \tag{19}$$

serves as a normalization factor, $P(A_i)$ represents the *prior* probability of selecting action A_i, and $P(A_i \mid \zeta^*)$ represents the *posterior* probability of selecting action A_i given the input vector ζ^*.

We now define the following stochastic policy for A^*. Let

$$A^* = A_i \text{ with probability } P(A_i \mid \zeta^*) \tag{20}$$

so that, at each time-step k, the control action A^* is generated stochastically as a function of the current model inputs (ζ^*) and the prior likelihood of each action.

B. Statistical Model

Hidden Markov models[17] are powerful, trainable statistical models which have previously been applied in a number of areas, including speech recognition,[17,18] modeling open-loop human actions,[19] and analyzing similarity between human control strategies.[20] Because of their capacity to model arbitrary statistical distributions, we choose HMMs to be the trainable statistical models λ_i of the previous section.

A discrete−hidden Markov model§ consists of a set of n states, interconnected through probabilistic transitions, and is completely defined by $\lambda = \{A, B, \pi\}$, where A is the probabilistic $n \times n$ state transition matrix, B is the $L \times n$ output probability matrix with L discrete-output symbols, and π is the n-length initial state probability distribution vector. For an observation sequence O of discrete symbols, we can locally maximize $P(\lambda \mid O)$ (i.e., probability of model λ given observation sequence O) using the Baum−Welch expectation-maximization (EM) algorithm. We can also evaluate $P(O \mid \lambda)$ through the efficient forward−backward algorithm.

Using discrete HMMs, note from Figure 8 that the discontinuous part of the HCS model consists of three distinct steps:

(1) Input-space signals ζ^* are first converted to an observation sequence of discrete symbols O^*, in preparation for hidden Markov model (HMM) evaluation.
(2) The resulting observation sequence O^* is then evaluated on a bank of discrete-output HMMs, each of which represents a possible control action A_i and each of which has previously been trained on corresponding human control data $\{\zeta_i^j\}$.
(3) Finally, the HMM evaluation probabilities are combined with prior probabilities for each action A_i according to Eqs. 18 and 20 to stochastically select and execute action A^* corresponding to input observation sequence O^*.

C. Signal-to-Symbol Conversion

To use discrete-output HMMs, we must first convert the multidimensional real-valued input space, to a sequence of discrete symbols. At a minimum, this process involves vector quantizing the input-space vectors $\zeta(k)$ to discrete symbols. We choose the well-known LBG VQ algorithm,[21] which iteratively generates vector codebooks of size $L = 2^l$, $l \in \{0, 1, \ldots\}$, and can be stopped at an appropriate level of discretization, as determined by the amount of available

§Although continuous and semicontinuous HMMs are developed, discrete HMMs are often preferred in practice because of their relative computational simplicity and reduced sensitivity to initial parameter settings during training.[17]

data. By optimizing the vector codebook on the human training data, we seek to minimize the amount of distortion introduced by the vector quantization process.

Now, suppose that we want to provide the models λ_i with m time-delayed values of the state and the control variables as input. There are at least two ways to achieve this. First, we could set

$$n_x = n_u = 1 \tag{21}$$

in Eqs. 6 and 7 and then we could train the models λ_i on observable sequences of length $n_O = m$. Alternatively, we could set

$$n_x = n_u = m \quad \text{and} \quad n_O = 1 \tag{22}$$

In the first case, we vector quantize shorter input vectors but we provide a longer sequence of observables $n_O > 1$ for HMM training and evaluation. In the second case, we vector quantize the entire input vector into a single observable, and we base our action choice solely on that single observable. This necessarily forces the HMMs λ_i to single-state models, such that each model is completely described by its corresponding output probability vector $\mathbf{B}_i$.

While in theory both choices start from identical input spaces, the single-observable, single-state case works better in practice. There are two primary reasons for this. Because the amount of data we have available for training comes from finite-length data sets, and is therefore necessarily limited in length, we must be careful that we do not overfit the models λ_i. Assuming fully forward-connected, left-to-right models λ_i, increasing the number of states from n_s to $(n_s + 1)$ increases the number of free (trainable) parameters by $n_s + L$, where L is the number of observables. Thus, having too many states in the HMMs substantially increases the chance of overfitting, since there may be too many degrees of freedom in the model. Conversely, by minimizing the number of states, the likelihood of overfitting is minimized.

A second reason that the single-observable, single-state case performs better relates to the vector quantization process. To understand how, consider that each input vector $\boldsymbol{\zeta}(k)$ minimally includes $2n_r$ road inputs. If we let $n_r = 10$, then for $n_x = n_u = 1$, 80% of the input dimensions will be road-related, while only 20% will be state-related. Thus, the vector quantization will most heavily minimize the distortion of the road inputs, while in comparison neglecting the potentially crucial state and previous command inputs. With larger values of n_x and n_u, the vector quantization process relies more equally on the state, previous control, and road inputs, and therefore forms more pertinent feature (prototype) vectors for control. For Eq. 22 above,

$$P(A_i \mid \boldsymbol{\zeta}^*) \propto P(O^* \mid \lambda_i) P(A_i) = b(j)_i P(A_i) \tag{23}$$

where $b(j)_i$ denotes the jth element in the λ_i model's output probability vector $\mathbf{B}_i$. Equation 23 defines a learned stochastic policy,

$$\pi(o, a) = P\big(a = A_i \mid o = O_j\big), \qquad \forall i, j \tag{24}$$

for each possible input observable O_j and action A_i.

D. Action Definitions

As we point out in Eqs. 3 and 4, the acceleration command ϕ is limited at each time-step k to the following actions. When $\phi(k) \geq 0$ (the gas is currently active),

$$A_1 : \phi(k+1) = \phi(k) \tag{25}$$

$$A_2 : \phi(k+1) = \min\big(\phi(k) + \Delta\phi_g, 4000\big) \tag{26}$$

$$A_3 : \phi(k+1) = \max\big(\phi(k) - \Delta\phi_g, 0\big) \tag{27}$$

$$A_4 : \phi(k+1) = -\Delta\phi_b \tag{28}$$

and when $\phi(k) < 0$ (the brake is currently active),

$$A_5 : \phi(k+1) = \phi(k) \tag{29}$$

$$A_6 : \phi(k+1) = \max\big(\phi(k) - \Delta\phi_b, -8000\big) \tag{30}$$

$$A_7 : \phi(k+1) = \min\big(\phi(k) + \Delta\phi_b, 0\big) \tag{31}$$

$$A_8 : \phi(k+1) = \Delta\phi_g \tag{32}$$

Actions A_1 and A_5 correspond to no action for the next time step; actions A_2 and A_6 correspond to pressing harder on the currently active pedal; actions A_3 and A_7 correspond to easing off the currently active pedal; and actions A_4 and A_8 correspond to switching between the gas and the brake pedals. The constants $\Delta\phi_g$ and $\Delta\phi_b$ are set by each human operator to the pedal responsiveness level he or she desires. We estimate the priors $P(A_i)$ by the frequency of occurrence of each action A_i in the human control training data. For $\phi(k) \geq 0$,

$$P(A_i) = \begin{cases} n_i / \sum_{k=1}^{4} n_k & i \in \{1,2,3,4\} \\ 0 & i \in \{5,6,7,8\} \end{cases} \tag{33}$$

where n_i denotes the number of times action A_i was executed in the training data set; similarly, for $\phi(k) < 0$,

$$P(A_i) = \begin{cases} 0 & i \in \{1,2,3,4\} \\ n_i / \sum_{k=5}^{8} n_k & i \in \{5,6,7,8\} \end{cases} \tag{34}$$

E. Experiments

Here, we follow the same procedure as in Section 3 for modeling the control strategies of Curly and Moe, except that we now model the acceleration control using the discontinuous framework developed above. We use the same values for n_x, n_u, and n_r as in Eqs. 9 and 10, and $n_O = 1$, as discussed previously. Furthermore, we vector quantize the input vectors $\boldsymbol{\zeta}(k)$ to $L = 512$ discrete observables. Figures 9 and 10 compare each individual's control data

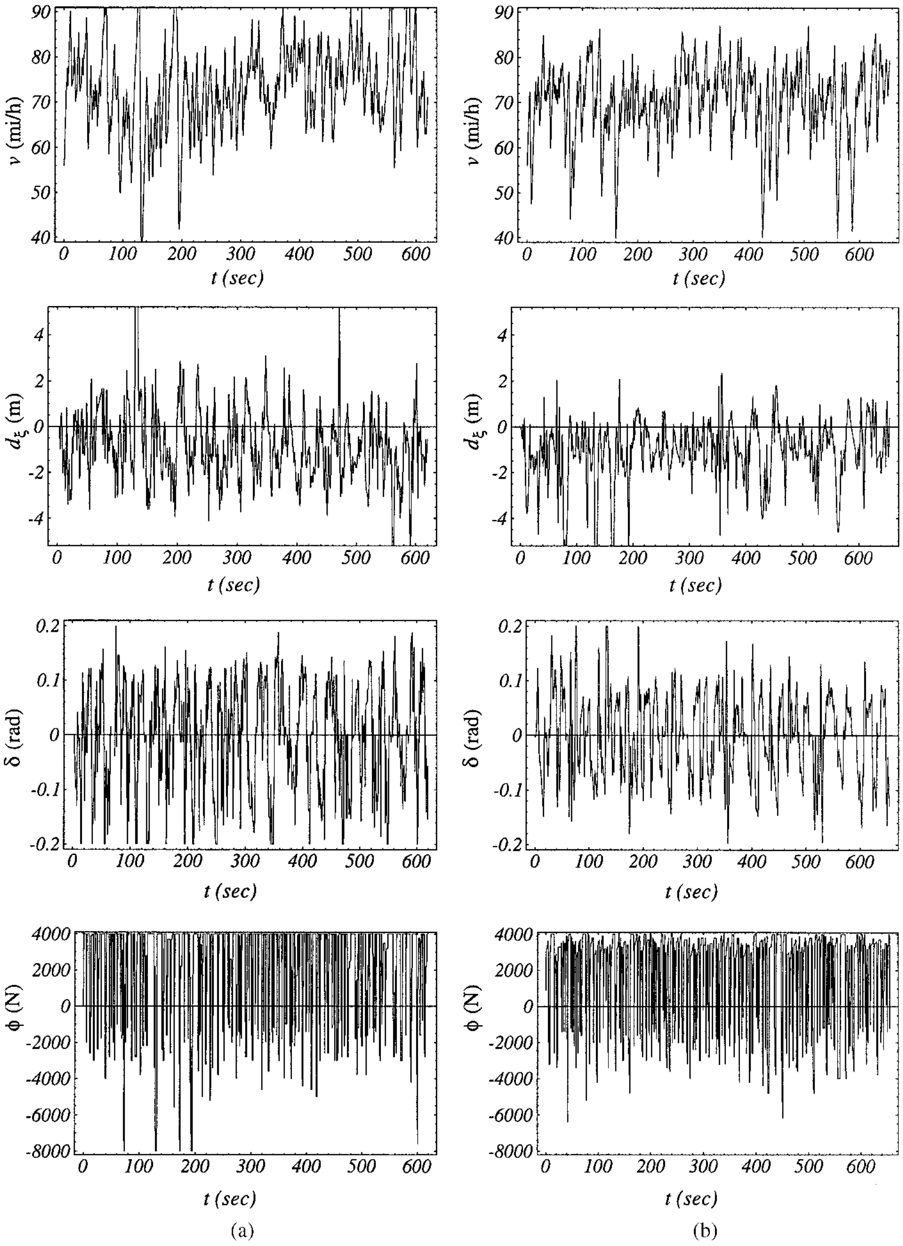
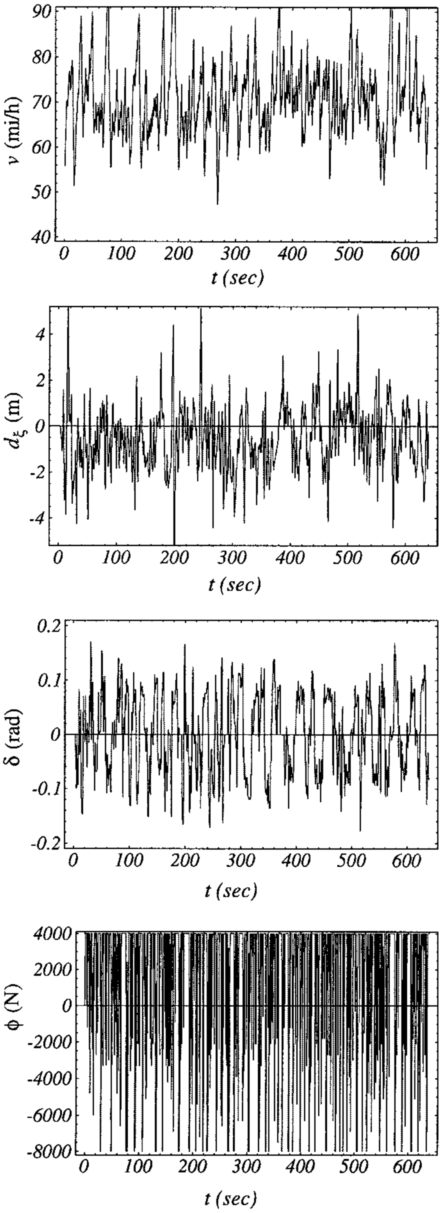
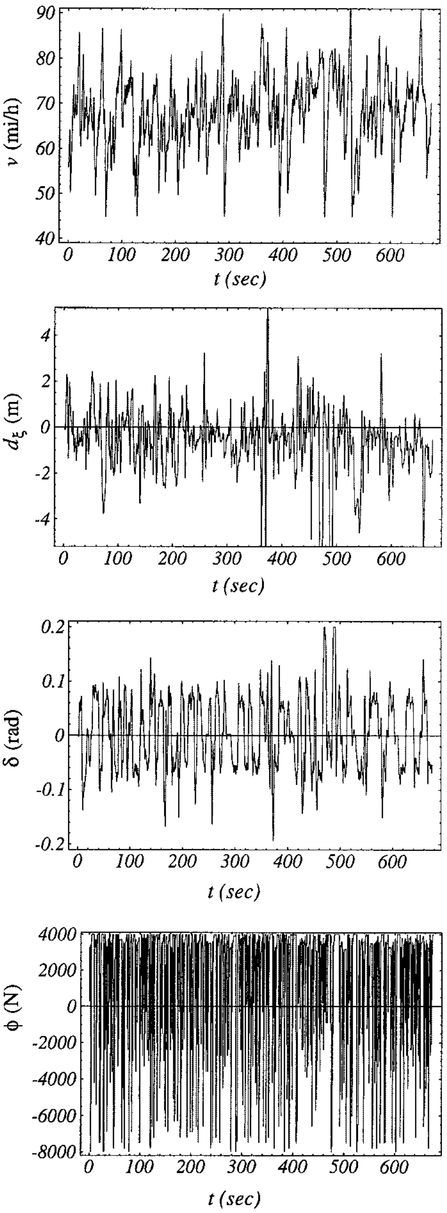


Figure 9. Curly's (a) control data and (b) corresponding hybrid controller data over the test road.



(a)



(b)

Figure 10. Moe's (a) control data and (b) corresponding hybrid controller data over the test road.

Table II. Statistical comparison.

	Curly		Moe	
Road ρ_2	Human Data	Hybrid Model	Human Data	Hybrid Model
ν (mph)	73.1 ± 9.5	71.4 ± 7.6	70.8 ± 8.3	69.5 ± 8.9
d (m)	-0.77 ± 1.97	-0.95 ± 1.27	-0.53 ± 1.42	-0.52 ± 1.77
δ (rad)	± 0.092	± 0.081	± 0.073	± 0.069
ϕ (N)	2190 ± 2770	1940 ± 2450	1850 ± 3340	1790 ± 3130

with their corresponding hybrid-model generated control over test road ρ_2, while Table II compares some aggregate statistics between the human and the hybrid model data.

5. DISCUSSION AND FUTURE WORK

A. Continuous vs. Hybrid Control

We see from Figures 9 and 10 that the stochastic controller also appears to have learned a convergent control strategy. The big question is: Which controller, the continuous neural-network controller, or the discontinuous HMM controller, performs better? The answer to that question depends on what precisely is meant by "better."

If we evaluate the two controllers based on absolute performance criteria, the neural-network controller probably performs better. It minimizes variations in the lateral position of the vehicle, conserves fuel by rarely "switching" to use the brake, and averages a higher overall speed. By comparison, the HMM controller runs off the road more often and resorts to braking much more frequently than the neural-network controller. Simply put, the neural-network controller appears to be more stable than its hybrid counterpart.

If, on the other hand, we evaluate the two controllers on how closely they approximate the operators's control strategy, the verdict changes drastically. As we have already noted, the neural network acceleration control looks nothing like the human control; the discontinuous acceleration control, on the other hand, appears to be a much better approximation of Curly's and Moe's driving, including more frequent off-road incidents. This greater similarity may also be observed in individual turning maneuvers. In Figure 11, for example, we compare Moe's control strategy through a 150 m-radius, 120° curve with the corresponding hybrid-model generated control trajectory. Moe's model initially brakes approximately 1/2 s before Moe himself does, albeit with somewhat less force. Thereafter, the hybrid model closely emulates Moe's strategy of rapid switching between the brake and the accelerator while in the turn. Note that the human and the model maneuvers take almost exactly the same amount of time (less than 1% difference).

We can further quantify the degree of similarity between each individual and his corresponding model using a stochastic similarity measure σ which we

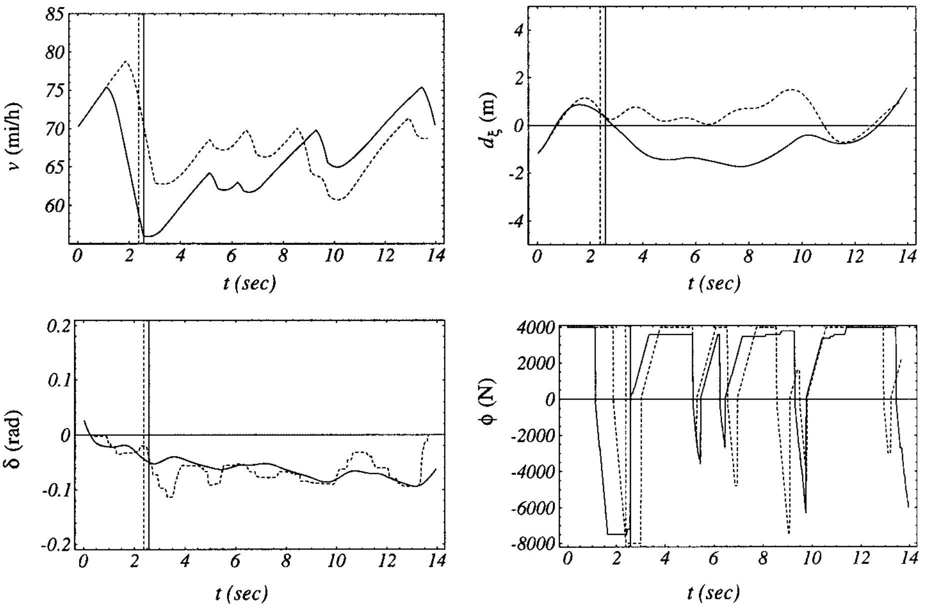


Figure 11. Moe's (dashed) and his hybrid model's (solid) control through a given turn.

developed previously for comparing different human control strategies.[20] The similarity measure is capable of comparing stochastic, multidimensional trajectories and yields a value between 0 and 1, with larger values indicating greater similarity. For the similarity comparison here, we include all relevant state and control variables $\{\nu_\xi, \nu_\eta, \omega, \delta, \phi\}$; Table III reports similarity results between the human data and each type of model. Note from Table III that the hybrid controllers exhibit much higher fidelity to the human control data than the continuous controllers.

B. Probability Profile

The most important reason behind the success of the hybrid controller is that it is able to successfully model the switching behavior between the gas and the brake pedals as a probabilistic event, since the *precise time* that a switch

Table III. Similarity between human and model data.

σ	Curly	Moe
Curly's continuous model	0.096	N/A
Curly's hybrid model	0.458	N/A
Moe's continuous model	N/A	0.088
Moe's hybrid model	N/A	0.555

occurs is not that important (as is the case, for example, in Fig. 11). What is more important is that the switch take place in some *time interval* around the time that the human operator would have executed the switch. Consider, for example, Figure 12, which plots the posterior probabilities $P(A_i \mid O)$ for a small segment of Curly's hybrid model control. We see that switches between the gas

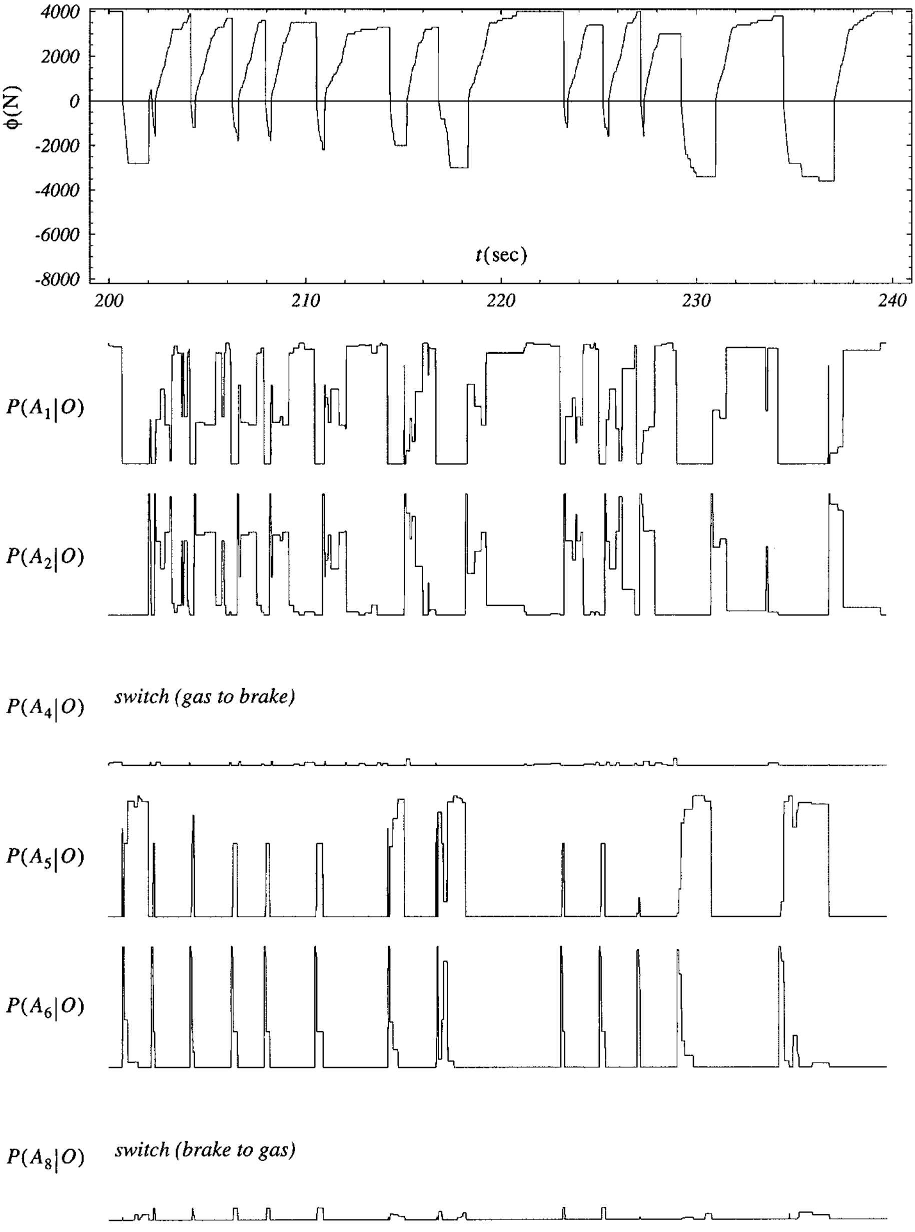


Figure 12. Posterior probabilities for Curly's model control ($P(A_3) = P(A_7) = 0$).

and the brake pedals (actions A_4 and A_8), while never very likely for any individual time step, are modeled as intervals where,

$$P(A_4 \mid O) = p > 0 \quad \text{or} \quad P(A_8 \mid O) = p > 0 \tag{35}$$

The probability that a switch will occur after m time steps given the constant probability p is given by

$$1 - (1 - p)^m \tag{36}$$

Figure 13 plots this probability as a function of time (at 50 Hz) for $p = 0.1$ and $p = 0.05$. Thus, we see that even for small values of p, the likelihood of a switch rises quickly as a function of time.

Because we train *separate* models λ_i for each action A_i, the hybrid modeling approach does not encounter the same one-to-many mapping problem, illustrated in Figures 6 and 7, that the continuous neural networks encounter. The relatively few occurrences of switching in each control data set are sufficient training data, since the switching models λ_4 and λ_8 see *only* that data during training. Including the priors $P(A_i)$ in the action selection criterion Eq. 20 then ensures that the model is not overly biased toward switching.

C. Modeling Extensions and Improvements

Suppose the acceleration control ϕ were not constrained by Eqs. 3 and 4, and thus were not as readily expressible through discrete actions. For example, suppose that the separate gas and brake commands could change by an arbitrary amount for each time step, not just by $\Delta\phi_g$ and $\Delta\phi_b$. How would this change the proposed control framework?

Figure 14 suggests one possible solution. Initially, we train two separate continuous controllers, the first corresponding to $\phi(k) \geq 0$, and the second corresponding to $\phi(k) < 0$. Since these controllers would not be required to model switches between braking and accelerating, the control outputs will vary continuously and smoothly with model inputs; hence a continuous function approximator should be well suited for these two modeling tasks.

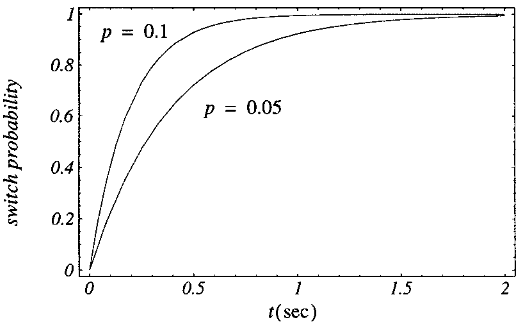


Figure 13. Probability of switch after t s (at 50 Hz) when the probability of a switch at each time step is p.

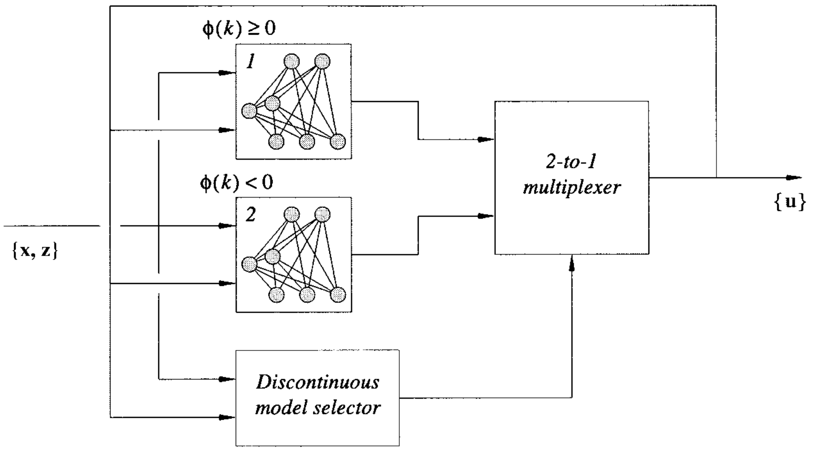


Figure 14. Alternative architecture for discontinuous strategies.

Then, we train four statistical models $\tilde{\lambda}_i$, corresponding to actions $\tilde{A}_i$, $i \in \{1, 2, 3, 4\}$, where actions $\tilde{A}_1$ and $\tilde{A}_2$ correspond to *no switch* at the next time step for $\phi(k) \geq 0$ and $\phi(k) < 0$, respectively, and actions $\tilde{A}_3$ and $\tilde{A}_4$ correspond to *a switch* at the next time step for $\phi(k) \geq 0$ and $\phi(k) < 0$, respectively. This discontinuous action model would then regulate which of the continuous models is active at each time-step k. Although the discontinuous controller's function in this scheme is reduced, it does preserve the critical role of the discontinuous controller in properly modeling the switching behavior, without the introduction of high-frequency noise. In fact, Figure 14 offers a modeling architecture which is applicable whenever discrete events or actions disrupt the continuous mapping from inputs to outputs.

Of course, our proposed statistical framework does have some limitations in comparison to functional modeling approaches. Because we vector quantize the input space, the stable region of operation for the hybrid controller is strictly limited to the input space spanned by the VQ codes. In fact, we observe from the modeling results that the continuous linear models are more stable than the hybrid discontinuous–continuous models. Previously, we attempted to address the stability problem[2] of the hybrid HCS models by reasoning that the stability of the system (i.e., the simulated car) is directly related to the kinetic energy T,

$$T \propto \sum_k \phi(k) \tag{37}$$

that is pumped into the system, where the expected value of T, $E[T]$, is given by

$$E[T] \propto \sum_k E[\phi(k)] \tag{38}$$

Thus, in an attempt to improve the stability margin of the system, we adjusted the model to generate $\phi'(k)$ so that

$$E[\phi'(k)] < E[\phi(k)] \tag{39}$$

Condition 39 can be realized by increasing the priors for those actions that decrease $E[\phi(k)]$—namely, A_3 or A_4, by some small amount ε_s, and, to stay within probabilistic constraints, by decreasing the priors A_2 or A_1, respectively, so that

$$P'(A_3) = P(A_3) + \varepsilon_s \quad \text{and} \quad P'(A_2) = P(A_2) - \varepsilon_s \tag{40}$$

or

$$P'(A_4) = P(A_4) + \varepsilon_s \quad \text{and} \quad P'(A_1) = P(A_1) - \varepsilon_s \tag{41}$$

where $\varepsilon_s > 0$ determines the degree to which $E[\phi'(k)]$ is reduced. While modifications of Eqs. 40 and 41 do improve the stability of the hybrid HCS models, we have recently begun to look at a more principled, less *ad hoc* approach for improving model stability.

In on-going work, we apply reinforcement learning and the theory of partially observable Markov decision processes (POMDPs) to the stability problem with some very promising early results.[22] Through reinforcement learning, we have been able to dramatically improve model stability to levels that compare favorably with the continuous models of Section 3, while, at the same time, maintaining a high degree of fidelity to the human training data. Although more detailed experiments are needed, we believe that these results open up a promising directions for future research by combining learning through observation (from humans) with subsequent reinforcement-learning-based optimization.

6. CONCLUSION

In this paper, we developed a discontinuous modeling framework for abstracting discontinuous human control strategies, and we compared the proposed approach to a competing continuous learning architecture. Which control approach is preferred ultimately depends on the specific application for the HCS model. If the model is being developed toward the eventual control of a real robot or vehicle, then the continuous modeling approach might be preferred as a good starting point. Continuous models can operate for a larger range of inputs, can show greater inherent stability, and can lend themselves more readily to theoretical performance analysis. If, on the other hand, the model is being developed to simulate different human behaviors in a virtual reality simulation or game, then the discontinuous control approach might be preferred, since fidelity to the human training data and random variations in behavior would be the desired qualities of the HCS model. Thus, depending on the application, we believe a need exists for both types of modeling approaches.

We thank the many people who patiently "drove" through our driving simulator. Special thanks go to "Curly" and "Moe" for their patience as they navigated through some 50 miles of simulated road.

References

1. Nechyba MC. Learning and validation of human control strategies; Ph.D. thesis. The Robotics Institute, Carnegie Mellon University; 1998.
2. Nechyba MC, Xu Y. Human control strategy: Abstraction, verification and replication. IEEE Contr Syst Mag 1997;17:48−61.
3. Fix E, Armstrong HG. Modeling human performance with neural networks. In: Proceedings of the International Joint Conference on Neural Networks, 1990, Vol. 1, p 247−252.
4. Fix E, Armstrong HG. Neural network based human performance modeling. In: Proceedings of the IEEE National Aerospace and Electronic Conference, 1990, Vol. 3, p 1162−1165.
5. Pomerleau DA. Neural network perception for mobile robot guidance; Ph.D. thesis. School of Computer Science, Carnegie Mellon University; 1992.
6. Pomerleau DA. Reliability estimation for neural network based autonomous driving. Robot Autonomous Syst 1994;12:113−119.
7. Pomerleau DA, Jochem T. Rapidly adapting machine vision for automated vehicle steering. IEEE Expert 1996;11:19−27.
8. Neusser S, Nijhuis J, Spaanenburg L, Hoefflinger B, Franke U, Fritz H. Neurocontrol for lateral vehicle guidance. IEEE Micro 1993;13:57−66.
9. Modjtahedzadeh A, Hess RA. A model of driver steering control behavior for use in assessing vehicle handling qualities. ASME J Dynamic Syst Measurement Contr 1993;115:456−464.
10. Pentland A, Liu A. Toward augmented control systems. Proc Intell Vehicles 1995; 1:350−355.
11. Nechyba MC, Xu Y. Learning and transfer of real-time human control strategies. J Advanced Comput Intell 1997;1:137−154.
12. Hatwal H, Mikulcik EC. Some inverse solutions to an automobile path-tracking problem with input control of steering and brakes. Vehicle Syst Dynamics 1986; 15:61−71.
13. Fahlman SE, Baker LD, Boyan JA. The Cascade 2 learning architecture; Technical Report, CMU-CS-TR-96-184, Carnegie Mellon University, 1996.
14. Nechyba MC, Xu, Y. Cascade neural networks with node-decoupled extended Kalman filtering. In: Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation, 1997, Vol. 1, p 214−219.
15. Narendra KS, Parthasarathy K. Identification and control of dynamical systems using neural networks. IEEE Trans Neural Networks 1990;1:4−27.
16. Press WH, Teukolsky SA, Vetterling WT, Flannery BP. Numerical recipes in C: The art of scientific computing, 2nd ed. Cambridge, UK: Cambridge Univ. Press; 1992.
17. Rabiner LR. A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE 1989;77:257−286.
18. Huang XD, Ariki Y, Jack MA. Hidden Markov models for speech recognition. Edinburgh: Edinburgh Univ. Press; 1990.
19. Yang J, Xu Y, Chen CS. Human action learning via hidden Markov model. IEEE Trans Syst Man Cybern A 1997;27:34−44.
20. Nechyba MC, Xu Y. Stochastic similarity for validating human control strategy models. IEEE Trans Robot Automat 1998;14:437−451.
21. Linde Y, Buzo A, Gray RM. An algorithm for vector quantizer design. IEEE Trans Commun 1980;28:84−95.
22. Nechyba MC, Bagnell JA. Stabilizing human control strategies through reinforcement learning. In: Proceedings of the IEEE International Symposium on Robotics and Control, 1999, Vol. 1, p 39−44.