On Discontinuous Human Control Strategies

Michael C. Nechyba[1] and Yangsheng Xu[1,2]

[1]The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA
[2]Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong

Abstract

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. In this paper, we first describe the real-time driving simulator which we have developed for investigating human control strategies. Next, we demonstrate the shortcomings of a typical continuous modeling approach in modeling a discontinuous human control strategy. We then propose an HMM-based method of modeling discontinuous human control strategies, and show that 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.*

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 (see [1] for an overview of the literature). 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 *continuous* 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 will tend to vary continuously with model inputs, acceleration control of the vehicle is decidedly discontinuous, since it involves explicit switching between the gas and brake pedals.

To adequately model such control behavior, we therefore propose a new stochastic, discontinuous learning algorithm, based on Hidden Markov Models. The proposed algorithm models possible control actions 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.

In this paper, we first 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. We then illustrate the difficulty of modeling a discontinuous control strategy using a continuous learning framework. Next, 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 suggest potential applications for each modeling approach.

2. Real-time driving simulator

Figure 1 shows the real-time graphic driving simulator which we have developed as an experimental platform for modeling human control strategies (HCS). In the simulator, the human operator has independent 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 described by [2,3], $\{v_\xi, v_\eta, \omega\}$, 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,

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

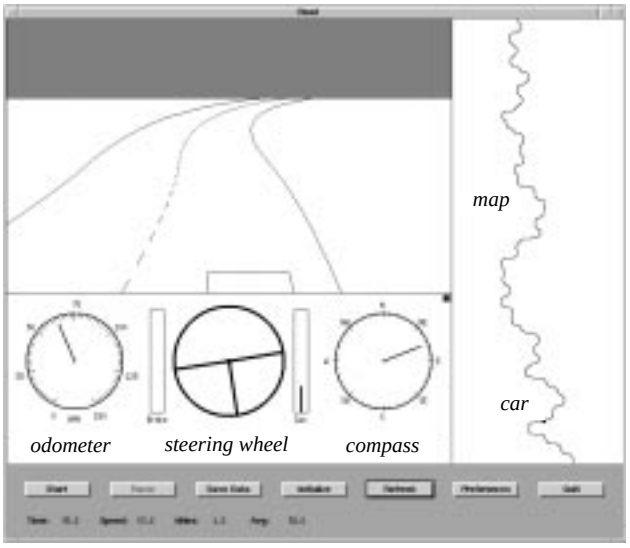

Fig. 1: The driving simulator gives the user a perspective preview of the road ahead. The user has independent controls of the steering, brake, and accelerator (gas).

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

where α is the user-applied longitudinal force on the front tires and δ is the user-applied steering angle.

Because of input device constraints, the force (or acceleration) control α is limited during each 1/50 second time step, based on its present value. If the gas pedal is currently being applied ($\alpha > 0$), then the operator can either increase or decrease the amount of applied force by a constant $\Delta\alpha_g = 200\text{N}$ or switch to braking. Similarly, if the brake pedal is currently being applied ($\alpha < 0$) the operator can either increase or decrease the applied force by a second constant $\Delta\alpha_b = 200\text{N}$ or switch to applying positive force. Thus, the $\Delta\alpha_g$ and $\Delta\alpha_b$ constants define the responsiveness of each pedal. In concise notation, denote $\alpha(k)$ as the current applied force and $\alpha(k+1)$ as the applied force for the next time step. Then, for $\alpha(k) \ge 0$,

$$\begin{aligned}\alpha(k+1) \in \{&\alpha(k), \min(\alpha(k) + \Delta\alpha_g, 4000),\\ &\max(\alpha(k) - \Delta\alpha_g, 0), -\Delta\alpha_b\}\end{aligned} \tag{3}$$

and for $\alpha(k) < 0$,

$$\begin{aligned}\alpha(k+1) \in \{&\alpha(k), \max(\alpha(k) - \Delta\alpha_b, -8000),\\ &\min(\alpha(k) + \Delta\alpha_b, 0), \Delta\alpha_g\}\end{aligned} \tag{4}$$

For the experiments in this paper, we collect human driving data across randomly generated roads like the 20km one shown in the map of Figure 1. The roads are described by a sequence of (1) straight-line segments and (2) circular arcs. The length of each straight-line segment, as well as the radius of curvature of each arc, lies between 100 and 200 meters. Finally, the visible horizon is set at 100m.

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. While we choose *cascade neural network learning* for this purpose, we will show that the same problems would, in fact, be encountered by *any* continuous function approximator.

3.1 Cascade learning

Here, we briefly summarize the cascade neural network learning architecture. Further details, which are omitted for space reasons, may be found in [1,2,4]. 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 *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 is repeated with each additional hidden unit, which receives input connections from both the network inputs and all 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 [2], increasing the functional flexibility of the learning architecture; and (2) we train the neural network weights through node-decoupled extended Kalman filtering (NDEKF) [5], as opposed to gradient-descent 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.

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. Thus, the inputs to the cascade neural network should include, (1) current and previous state information $\{v_\xi, v_\eta, \omega\}$, (2) previous output (control) information $\{\delta, \alpha\}$, and (3) a description of the road visible from the current car position. More precisely, the network input vector $\zeta(k)$ at time step k is given by,

$$\begin{aligned}&\{v_\xi(k-n_s), \ldots, v_\xi(k-1), v_\xi(k)\\ &v_\eta(k-n_s), \ldots, v_\eta(k-1), v_\eta(k)\\ &\omega(k-n_s), \ldots, \omega(k-1), \omega(k)\}\end{aligned}\,, \tag{5}$$

$$\begin{aligned}&\{\delta(k-n_c), \ldots, \delta(k-1), \delta(k)\\ &\alpha(k-n_c), \ldots, \alpha(k-1), \alpha(k)\}\end{aligned}\,, \tag{6}$$

$$\{x_1(k), x_2(k), \ldots, x_{n_r}(k), y_1(k), y_2(k), \ldots, y_{n_r}(k)\}\,. \tag{7}$$

where n_s is the length of the state histories and n_c is the length of the previous command histories presented to the network as input. 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 provide that sequence of coordinates as input to the network. Thus, the total number of inputs to the network are $3n_s + 2n_c + 2n_r$. The outputs of the cascade network are $\{\delta(k+1), \alpha(k+1)\}$, the steering and acceleration commands at the next time step, respectively.

3.2 Experiment

We ask Larry to drive over two different randomly generated 20km roads ρ_1 and ρ_2. A part of Larry's second run, each of which lasts about 10 minutes, is shown in Figure 2 below. Larry's driving behavior is representative of other runs recorded by him, 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) Larry manages to stay on the road (± 5m deviation from the road median) for most of the run, with only a few brief off-road episodes in especially tight turns.

Now, we use Larry's first run (ρ_1) to train a cascade neural network, and reserve the second road ρ_2 for testing the network model. By searching the space of possible inputs parameterized by $\{n_s, n_c, n_r\}$, we arrive at the following suitable input space representation for Larry's cascade network model:

$$n_s = n_c = 6\,,\; n_r = 10 \tag{8}$$

Thus, there are 50 inputs to the model. This input representation ensures that the cascade neural network forms a convergent control model. Figure 3 plots part of the neural network's driving control strategy over road ρ_2 for a linear cascade network (i.e. no hidden units), while Table 1 compares some aggregate statistics for Larry's second run and the linear model's run.

3.3 Discussion

From Figure 3 and Table 1, we make several observations. Most importantly, the linear model, despite the discontinuous acceleration command, is able to learn *something*; that is, the model keeps the vehicle on the road (except for one high-curvature turn that Larry himself was not able to handle properly). Not only that, but it does so at approximately the same average speed and lateral distance from the road median using a similar steering control strategy as Larry. In some respects, the model's control can even be considered superior to Larry's control. The model only rarely engages the brake, and maintains tighter lateral road position.

If we judge the model on how faithfully it reproduces Larry's acceleration control strategy, however, it rates significantly worse; that is, the model's acceleration control looks nothing like Larry's. Adding hidden units to impart nonlinearity to the model introduces additional high-frequency components to the control, but does not bring the model much closer to Larry's control strategy. Figure 4, for example, illustrates the model's acceleration control when two hidden units are introduced to the model.

To better appreciate what is happening, we would like to visualize how different input vectors in the training data map to different acceleration outputs. Since it is impossible to visualize a 50-dimensional input space, we decompose each of the input vectors $\zeta(k)$ in the training set into the principal components (PCs) for Larry's entire first run such that,

$$\zeta(k) = c_1^k\gamma_1 + c_2^k\gamma_2 + \ldots c_{50}^k\gamma_{50}\,, \tag{9}$$

where γ_i is the PC corresponding to the ith largest eigenvalue σ_i. For Larry's control data, we have that,

$$\left|\sigma_2/\sigma_1\right| = 0.44\,,\ \left|\sigma_i/\sigma_1\right| \le 0.05\,,\ i \in \{3, 4, \ldots, 50\} \tag{10}$$

so that we can approximate the input vectors $\zeta(k)$ by,

Table 1: Statistical comparison

Road ρ_2	*Larry*	*Linear model*
v (mph)	71.9 ± 9.0	73.6 ± 2.5
d (m)	-0.72 ± 1.46	-1.00 ± 0.60
δ (rad)	± 0.094	± 0.068
α (N)	2240 ± 2620	1780 ± 760

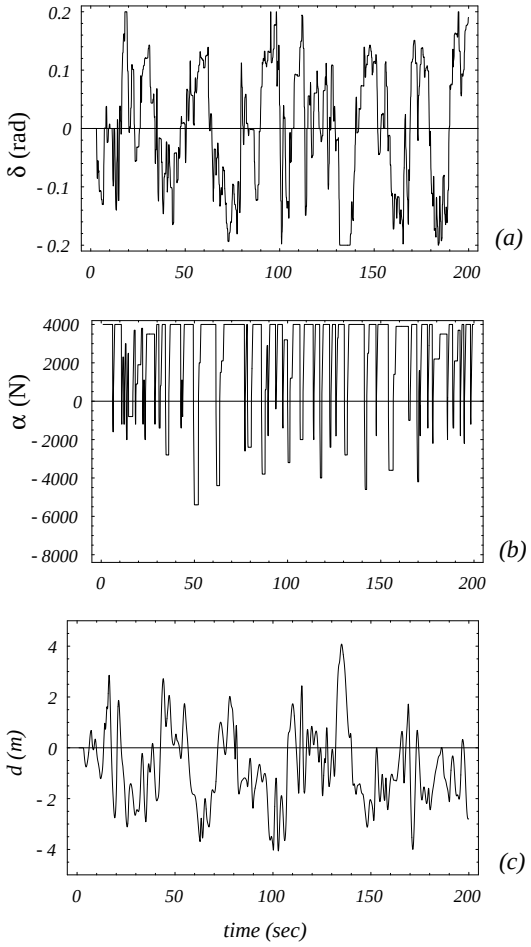

Fig. 2: Part of Larry's (a) steering command, (b) acceleration command and (c) resulting lateral distance from the road median. for the second road.

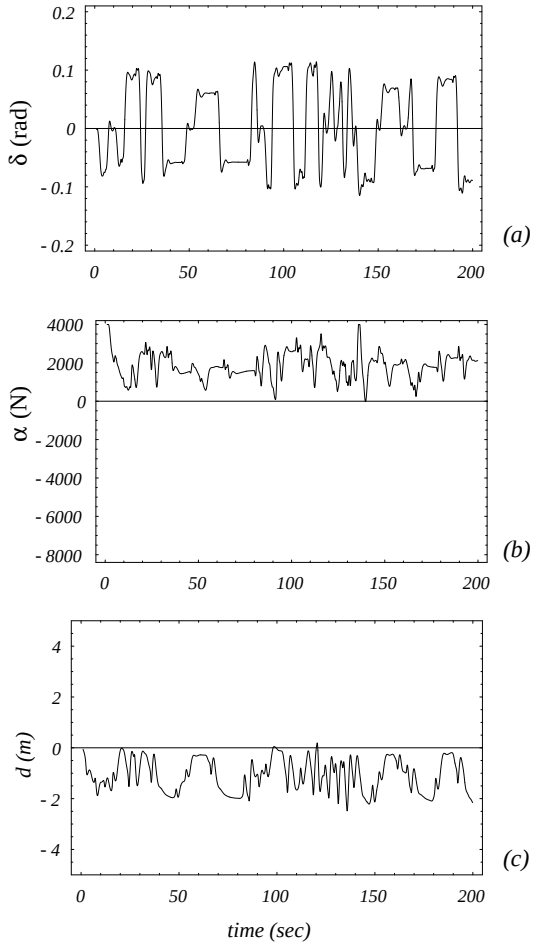

Fig. 3: Part of the linear model's (a) steering command, (b) acceleration command and (c) resulting lateral distance from the road median for the second road.

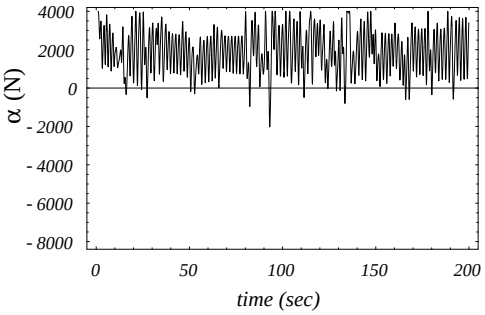

Fig. 4: Adding hidden units to the linear model does not bring the model significantly closer to Larry's acceleration control strategy.

$$\zeta(k) \approx c_1^k \gamma_1 + c_2^k \gamma_2 \,. \tag{11}$$

Now, we can visualize the relative location of each $\zeta(k)$ by plotting its PC coefficients (c_1^k, c_2^k) in 2D space. Figure 5(a) and (b) show the results for $\alpha(k) < 0$ (brake), and $\alpha(k) > 0$ (gas), respectively. In each plot, we distinguish points by whether or not $\alpha(k+1)$ indicates a discontinuity (i.e. a switch between braking and accelerating) such that,

$$\alpha(k) < 0 \text{ and } \alpha(k+1) > 0 \text{ [Figure 5(a)]} \tag{12}$$

$$\alpha(k) > 0 \text{ and } \alpha(k+1) < 0 \text{ [Figure 5(b)]} \tag{13}$$

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

We immediately observe from Figure 5 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 input spaces can lead to radically different outputs $\alpha(k+1)$. Consequently, Larry'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 *any* continuous function approximator.

4. Discontinuous control

To cope with the problems discussed above, we propose a new HMM-based framework for modeling discontinuous control strategies. Hidden Markov Models [6] are trainable statistical models which can be applied to model human control strategy, not as a deterministic functional mapping, but rather as a probabilistic relationship between sensory inputs and control actions outputs. They have previously been applied in a number areas, including speech recognition [6, 7], modeling open-loop human actions [8], and analyzing similarity between human control strategies [9].

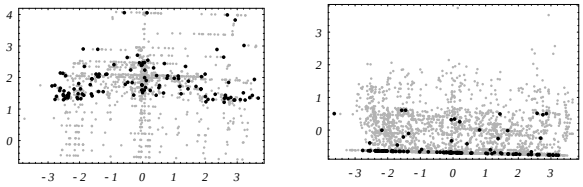

Fig. 5: Switching actions (black) significantly overlap other actions (grey) when the current applied force is (a) negative (brake), and (b) positive (gas).

Although continuous and semi-continuous HMMs have been developed, discrete-output HMMs are often preferred in practice because of their relative computational simplicity and reduced sensitivity to initial parameter settings during training [6]. A discrete Hidden Markov Model consists of a set of n states, interconnected through probabilistic transitions, and is completely defined by the triplet, $\lambda = \{A, B, \pi\}$, 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 $P(\lambda|O)$ (i.e.probability of model λ given observation sequence O) using the Baum-Welch Expectation-Maximization (EM) algorithm. We can also evaluate $P(O|\lambda)$ through the efficient Forward-Backward algorithm.

Figure 6 provides an overview of the resulting hybrid controller, where continuous outputs are modeled as before, and discontinuous outputs are modeled using the new framework. As shown in Figure 6, the discontinuous controller consists of three separate phases:

1. Input-space signals 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.
3. Finally, the HMM evaluation probabilities are combined with prior probabilities for each action A_i to stochastically select and execute action A^* corresponding to input observation sequence O^*.

Below, we describe each of these steps in turn.

4.1 Signal-to-symbol conversion

In order to use discrete-output Hidden Markov Models, we must first convert the multi-dimensional 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 [10], which iteratively generates vector codebooks of size 2^m, $m \in \{0, 1, \ldots\}$, and can be stopped at an appropriate level of discretization, as determined by the amount of available 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.

Once we have trained a vector codebook on all the input vectors $\zeta(k)$ in the human training data, and assuming that we want to train the Hidden Markov Models on sequences of length n_O, the sequence of input vectors,

$$\{\zeta(k - n_O + 1), \zeta(k - n_O + 2), \ldots, \zeta(k)\} \tag{14}$$

can be converted to,

$$O^k = \{o_1, o_2, \ldots, o_{n_O}\} \tag{15}$$

where o_i is the index of the codebook vector which minimizes the SSE distortion for $\zeta(k - n_o + i)$.

4.2 Stochastic, discontinuous controller

For the moment, assume that we wish to model a control task where at each time step k, we can choose one of N different discrete control actions A_i, $i \in \{1, \ldots, N\}$. Furthermore, assume that we have N groups of training observation sequences $\{O_i^1, O_i^2, \ldots, O_i^{n_i}\}$, $i \in \{1, \ldots, N\}$, where observation sequence O_i^j leads to control action A_i at the next time step. Then, using the Baum-Welch algorithm, we can train N different left-to-right HMMs λ_i in order to maximize,

$$\prod_{j=1}^{n_i} P(\lambda_i | O_i^j) \tag{16}$$

for model λ_i. Given this bank of HMM models, and a new observation sequence O^*, we would now like to choose an appropriate action A^*. For each model, we can evaluate $P(O^* | \lambda_i)$ using the Forward-Backward algorithm. Since model λ_i corresponds to action A_i,

$$P(O^* | A_i) \equiv P(O^* | \lambda_i) \tag{17}$$

By Bayes Rule,

$$P(A_i | O^*) = \frac{P(O^* | A_i) P(A_i)}{P(O^*)} \tag{18}$$

where,

$$P(O^*) \equiv \sum_{i=1}^{N} P(O^* | A_i) P(A_i) \tag{19}$$

and $P(A_i)$ represents the prior probability of selecting action A_i. We now propose the following stochastic control strategy:

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

Thus, at each time step k, a control action is generated stochastically as a function of the current model inputs (O^*) and the prior likelihood of each action.

4.3 Acceleration control

For the acceleration control in our application, we have a total of $N = 8$ possible actions, as given in equations (3) and (4). When $\alpha(k) \geq 0$,

$$A_1 : \alpha(k+1) = \alpha(k) \tag{21}$$

$$A_2 : \alpha(k+1) = \min(\alpha(k) + \Delta\alpha_g, 4000), \tag{22}$$

$$A_3 : \alpha(k+1) = \max(\alpha(k) - \Delta\alpha_g, 0), \tag{23}$$

$$A_4 : \alpha(k+1) = -\Delta\alpha_b, \tag{24}$$

and when $\alpha(k) < 0$,

$$A_5 : \alpha(k+1) = \alpha(k) \tag{25}$$

$$A_6 : \alpha(k+1) = \max(\alpha(k) - \Delta\alpha_b, -8000), \tag{26}$$

$$A_7 : \alpha(k+1) = \min(\alpha(k) + \Delta\alpha_b, 0), \tag{27}$$

$$A_8 : \alpha(k+1) = \Delta\alpha_g, \tag{28}$$

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 brake pedals. We estimate the priors $P(A_i)$ by the frequency of occurrence of action A_i in the human control training data:

$$P(A_i) = n_i / \sum_{k=1}^{N} n_k \tag{29}$$

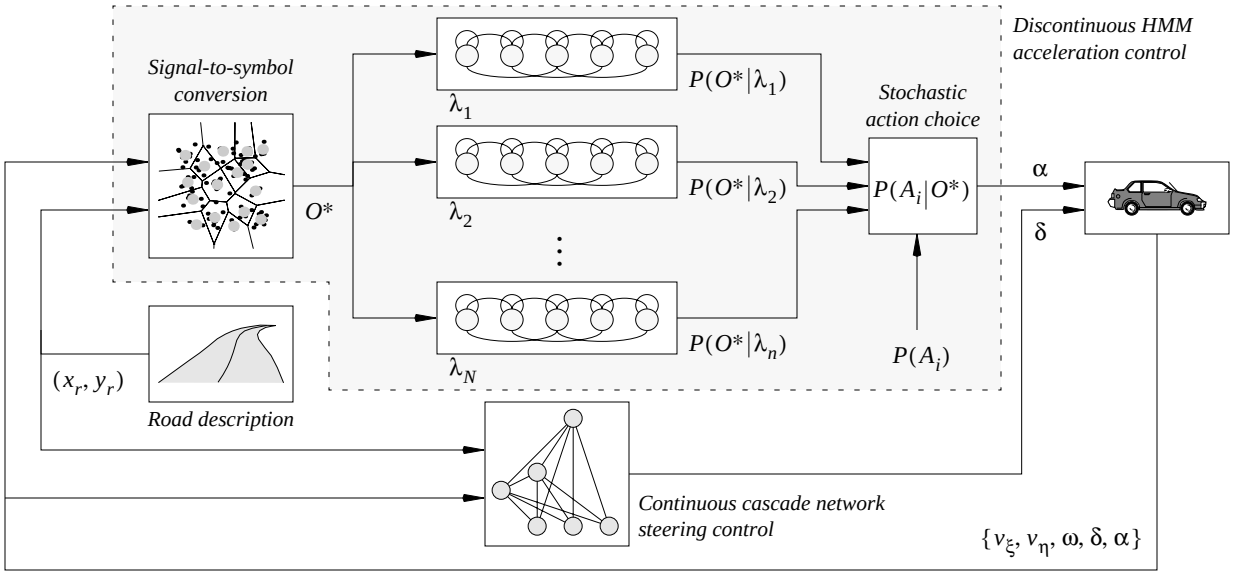

Fig. 6: Overall control structure. Steering is controlled by the cascade network, while the discontinuous acceleration command is controlled by the HMM-based controller (shaded box)

5. Experiment

5.1 Simulation

We once again try to model Larry's driving control strategy, only this time, we use the HMM-based approach to model the acceleration control α. The steering control δ is modeled by a cascade neural network as before (see Figure 6). Once again, we will use Larry's first run on road ρ_1 for training, and will reserve road ρ_2 for testing.

In processing Larry's control data for HMM training, we choose the following input representation and observation sequence length:

$$n_s = n_c = 1,\ n_r = 10,\ n_O = 6, \tag{30}$$

These choices ensure that approximately the same level of information is provided to the HMM controller as was provided to the cascade neural network in Section 3.2. The resulting input vectors $\zeta(k)$ are vector quantized to 512 codes, and three-state HMMs λ_i are trained for each possible action A_i.

Using equation (29), we calculate the priors $P(A_i)$ for Larry's first run as given in Table 2 below. We note from Table 2 that Larry never eases off either the gas or brake pedal, but rather opts to switch between braking and accelerating to achieve his desired control. Even so, the prior likelihood of a pedal switch is relatively low, which is part of the reason the neural network has difficulty modeling Larry's acceleration control strategy.

Figure 7, plots one of the stochastic HMM-based driving control strategies for road ρ_2, while Table 3 compares aggregate statistics for Larry's second run and the HMM controller's run.

5.2 Discussion

We see from Figure 7 that the stochastic HMM 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 HMM counterpart.

Table 2: Prior probabilities

Road ρ_1	$\alpha(k) \geq 0$	$\alpha(k) < 0$
$P(A_1)$	0.819	0.000
$P(A_2)$	0.176	0.000
$P(A_3)$	0.000	0.000
$P(A_4)$	0.005	0.000
$P(A_5)$	0.000	0.730
$P(A_6)$	0.000	0.249
$P(A_7)$	0.000	0.000
$P(A_8)$	0.000	0.021

Table 3: Statistical comparison

Road ρ_2	*Larry*	*HMM controller*
v (mph)	71.9 ± 9.0	70.7 ± 8.1
d (m)	-0.72 ± 1.46	-1.38 ± 1.70
δ (rad)	± 0.094	± 0.081
α (N)	2240 ± 2620	1970 ± 2340

If, on the other hand, we evaluate the two controllers on how closely they approximate the operator's (Larry's) control strategy, the verdict changes drastically. As we have already noted previously, the neural network control does not look anything like

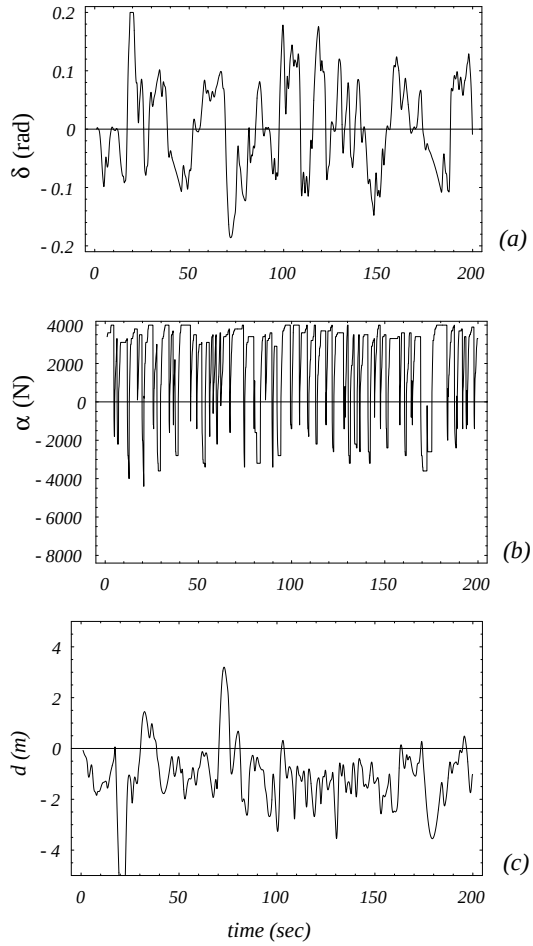

Fig. 7: Part of the HMM controller's (a) steering command, (b) acceleration command and (c) resulting lateral distance from the road median for the second road.

Larry's control; the HMM control trajectory appears to be a much better approximation of Larry's actual acceleration control strategy, including the consequent increases in the number of off-road incidents. We can quantify the degree of similarity to Larry's control strategy for each controller using a stochastic similarity measure σ which we have developed previously [9] for comparing different human control strategies. The similarity measure is capable of comparing stochastic, multi-dimensional 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 $\{v_\xi, v_\eta, \omega, \delta, \alpha\}$, and arrive at the following similarity values:

$$\sigma(Larry, HMM) = 0.628 \quad (31)$$

$$\sigma(Larry, NN) = 0.101 \quad (32)$$

Hence, the HMM controller shows greater fidelity to the source training data than does the neural network controller.

The most important reason behind the HMM controller's success is that it is able to successfully model the switching behavior between the gas and brake pedals as a probabilistic event. Because we train a *separate* HMM for each action A_i, this modeling approach does not encounter the same one-to-many mapping problem, illustrated in Figure 5, that the neural network encounters. The relatively few occurrences of switching are sufficient training data, since the switching HMMs λ_4 and λ_8 see *only* that data during HMM training. Including the priors $P(A_i)$ in the action selection criterion then ensures that the model is not overly biased towards switching.

Now, suppose the acceleration control α were not constrained by equations (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 $\Delta\alpha_g$ and $\Delta\alpha_b$. How would this change the proposed control framework? In such a case, two separate continuous controllers can be trained, one for controlling the gas pedal, and the other for controlling the brake pedal. Furthermore, the HMM controller's role would be reduced to simply regulating the switching behavior between the two pedals. Despite its reduced function, however, the general HMM framework would still retain a critical part in modeling the overall acceleration control accurately.

Of course, the HMM control 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 HMM controller is strictly limited by the range of the vector codebook. When, in rare circumstances, an unfortunate sequence of stochastic action selections causes the controller to stray too far from that limited codebook range, the VQ distortion dramatically increases, and the behavior of the controller becomes less predictable. In such cases, by monitoring the VQ distortion, it might be possible to temporarily suspend the HMM controller in favor of a neural network controller until the VQ distortion once again returns to an acceptable level. Alternatively, we are investigating whether semicontinuous HMMs [7], which model the VQ codebook as a family of Gaussian pdfs, would (1) improve performance, and (2) be computationally tractable in real- time execution.

A second limitation of the HMM approach is the inclusion of the prior probabilities $P(A_i)$ in the stochastic selection criterion. By including the priors, we are assuming an environment similar to the training environment. Radically different environmental conditions would presumably change the values of the priors, and therefore make the action selection criterion less valid.

6. Conclusion

In this paper we have developed a discontinuous modeling framework for abstracting discontinuous human control strategies, and have 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 towards 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, show greater inherent stability, and lend themselves more readily to theoretical performance analysis. If, on the other hand, the model is being developed in order 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.

References

[1] M. C. Nechyba and Y. Xu, "Human Control Strategy: Abstraction, Verification and Replication," *IEEE Control Systems Magazine*, vol. 17., no. 5, pp. 48-61, 1997.

[2] M. C. Nechyba and Y. Xu, "Learning and Transfer of Real-Time Human Control Strategies," to appear in *Journal of Advanced Computational Intelligence*, vol. 1, no. 2, 1997.

[3] H. Hatwal and E. C. Mikulcik, "Some Inverse Solutions to an Automobile Path-Tracking Problem with Input Control of Steering and Brakes," *Vehicle System Dynamics*, vol. 15, pp. 61-71, 1986.

[4] S. E. Fahlman, L. D. Baker and J. A. Boyan, "The Cascade 2 Learning Architecture," Technical Report, CMU-CS-TR-96-184, Carnegie Mellon University, 1996.

[5] M. C. Nechyba and Y. Xu, "Cascade Neural Networks with Node-Decoupled Extended Kalman Filtering," *Proc. IEEE Int. Symp. on Computational Intelligence in Robotics and Automation*, vol. 1, pp. 214-9, 1997.

[6] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257-86, 1989.

[7] X. D. Huang, Y. Ariki and M. A. Jack, *Hidden Markov Models for Speech Recognition*, Edinburgh Univ. Press, 1990.

[8] J. Yang, Y. Xu and C. S. Chen, "Human Action Learning Via Hidden Markov Model," *IEEE Trans. Systems, Man and Cybernetics, Part A*, vol. 27, no. 1, pp. 34-44, 1997.

[9] M. C. Nechyba and Y. Xu, "Stochastic Similarity for Validating Human Control Strategy Models," *Proc. IEEE Conf. on Robotics and Automation*, vol. 1, pp. 278-83, 1997.

[10] Y. Linde, A. Buzo and R. M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Trans. Communication*, vol. COM-28, no. 1, pp. 84-95, 1980.