Human Control Strategy: Abstraction, Verification, and Replication

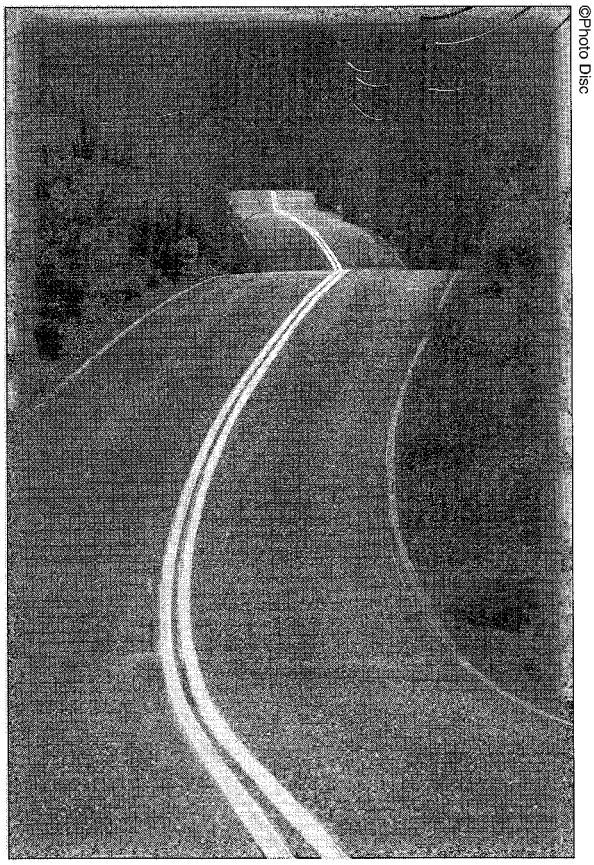
Michael C. Nechyba and Yangsheng Xu

In this article, we describe and develop methodologies for modeling and transferring human control strategy (HCS). This research has potential application in a variety of areas such as the Intelligent Vehicle Highway System (IVHS), human-machine interfacing, real-time training, space telerobotics, and agile manufacturing. We specifically address the following issues: (1) how to efficiently model human control strategy through learning cascade neural networks, (2) how to select state inputs in order to generate reliable models, (3) how to validate the computed models through an independent, Hidden Markov Model-based procedure, and (4) how to effectively transfer human control strategy. We have implemented this approach experimentally in the real-time control of a human driving simulator, and are working to transfer these methodologies for the control of an autonomous vehicle and a mobile robot. In providing a framework for abstracting computational models of human skill, we expect to facilitate analysis of human control, the development of human-like intelligent machines, improved human-robot coordination, and the transfer of skill from one human to another.

Introduction

Although humans are quite adept at mastering complex and dynamic skills, we are far less impressive in formalizing our behavior into algorithmic, machine-codable strategies. Therefore, it has been difficult to duplicate the types of intelligent skills and actions we witness every day as humans, in robots and other machines. This limits not only the capabilities of individual robots, but also the extent to which humans and robots can safely interact and cooperate with one another. Nevertheless, human actions are currently our only examples of truly "intelligent" behavior. As such, there exists a profound need to abstract human skill into computational models, capable of realistic emulation of dynamic human behavior. With such models of human skill, we can transfer intelligent control behaviors to robots. This is especially critical for robots that have to operate in remote or inhospitable environments, where humans cannot function. In other robotic applications, we would like robots to carry out tasks that humans have traditionally performed. For example, the Intelligent Vehicle Highway System (IVHS), currently being developed through massive initiatives in the United States, Europe, and Japan [1,2], envisions automating much of the driving on our highways. The required automated vehicles will need significant intelligence to interact safely with variable road conditions and other traffic. Modeling human intelligence offers one way of building up the necessary skills for this type of intelligent machine.

With increased intelligence and sophistication in robotic systems, analysis of human-robot coordination in tightly coupled human-machine systems will become increasingly relevant. In IVHS, for example, there will be ubiquitous interaction between autonomous vehicles and their human drivers/passengers. Moreover, the currently limited application domain for robots may broaden into other aspects of consumer life, where household and service robots will interact primarily with non-experts. To ensure safe coordination with humans in a shared workspace, we must incorporate appropriate models of human behavior into the world model of the robots. We can assess the quality of joint human-machine systems by including computational models of human behavior in the overall system analysis.



©Photo Disc

Realistic simulation of human behavior is required not only in human-machine systems, but also in the burgeoning field of virtual reality. As graphic displays become increasingly lifelike, the dynamic behavior of the virtual world will need to match the increased visual realism. Computational models of human skill can impart the necessary sense of realism to the actions and behaviors of virtual humans in the virtual world.

M. Nechyba and Y. Xu are with the Robotics Institute at Carnegie Mellon University in Pittsburgh, Pa.

0272-1708/97/$10.00©1997IEEE

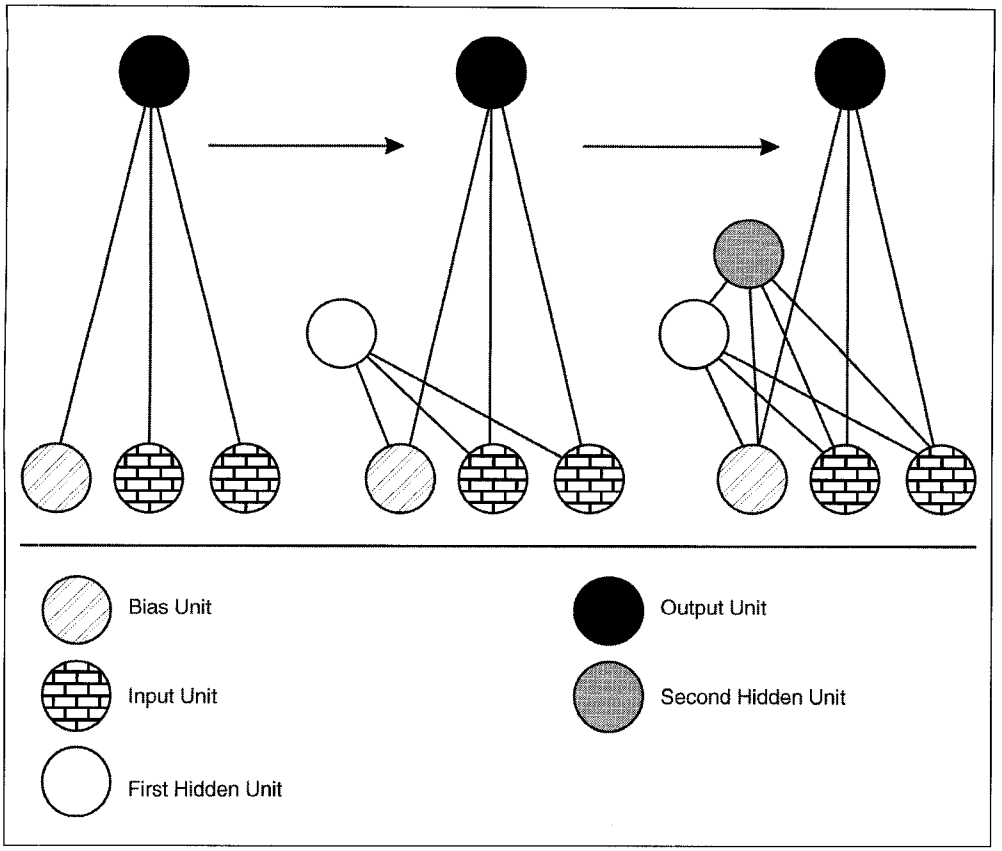


Fig. 1. The cascade learning architecture adds hidden units one at a time to an initially minimal network. All connections in the diagram are feedforward.

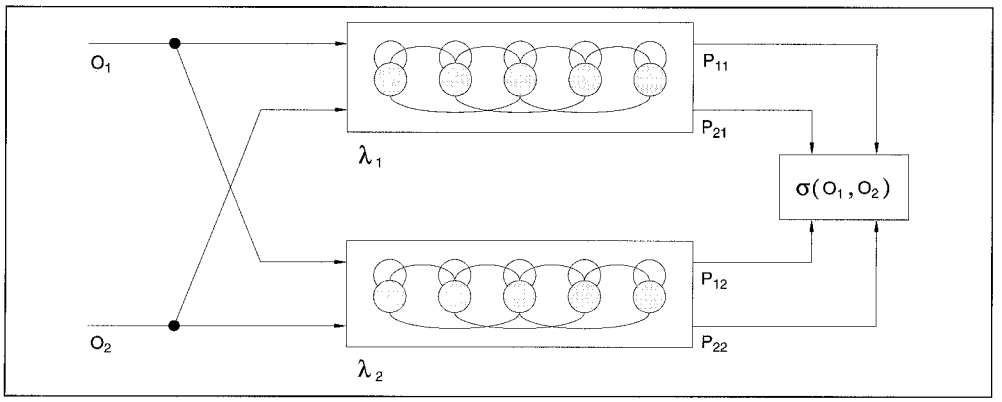


Fig. 2. Four normalized probability values make up the similarity measure.

Finally, accurate models of human skill can contribute to improved expert training and human-computer interfacing (HCI). Consider, for example, the tasks of teleoperating robots in remote environments or learning to fly a high-performance jet. Training for both of these tasks is difficult, expensive, and time-consuming for a novice [3]. We can accelerate learning for the novice operator by providing on-line feedback from virtual teachers in the form of skill models, which capture the control strategies of expert operators. Through the use of human skill models, operator performance can be monitored during training or actual task execution as information is displayed through different sensor modalities and layouts.

Some work has been done in recent years toward learning skills directly from humans. In fuzzy control schemes [4, 5], human experts are asked to specify "if-then" control rules with fuzzy linguistic variables, which they believe guide their control actions. For complex systems, this approach is prone to error since important details may be inadvertently omitted by the expert. A number of researchers have solved the inverted pendulum problem through learning from a human expert [6-8]. Robot learning from human experts has recently been applied to a deburring robot by Asada and Liu [9], where human input patterns are associated with corresponding output actions for the robot. Lee and Chen [10] use feasible state transition graphs through self-organizing data clusters to abstract skill from human data. Skills are modeled as optimal sequences of one-step state transitions that transform the current state into the goal state. The approach is verified on demonstrated human Cartesian teleoperation skill. Xu and Yang [11] implement a different state-based approach to open-loop skill learning and telerobotics using Hidden Markov Models (HMMs). Several approaches to skill learning in human driving have been implemented. In [12], neural networks are trained to mimic human behavior for a simulated circular racetrack. Pomerleau [13] implements real-time road-following with data collected from a human driver. A static feedforward neural network, with a single hidden layer, 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 research thus far has not addressed a number of important issues. Most recent work in learning from human data deals with either action skills, static or quasi-static skills, or higher-level abstractions of human skill (e.g., assembly), and does not focus on abstracting dynamic human control strategy. As such, we need to develop an *efficient and flexible learning architecture* for model-

Table 1. Cascade Learning for Networks with Different Activation Functions: Quickprop and NDEKF

$\Gamma(\cdot)^a$		Quickprop Algorithm				Node-Decoupled Extended Kalman Filtering			
		$\overline{N}^b$	$\sigma(N)^c$	$\overline{N}^b$	$\sigma(e_{RMS})^e$	$\overline{N}$	$\sigma(N)$	$\overline{e}_{RMS}$	$\sigma(e_{RMS})$
$f_1(x)$	*variable*	734	33	1.4×10^{-3}	3.5×10^{-4}	182	13	1.2×10^{-4}	1.2×10^{-4}
	sinusoidal	768	55	1.6×10^{-3}	3.2×10^{-4}	187	9	1.9×10^{-4}	1.2×10^{-4}
	sigmoidal	2109	103	6.0×10^{-3}	2.5×10^{-3}	183	7	1.7×10^{-3}	8.2×10^{-4}
$f_2(x)$	*variable*	723	45	1.4×10^{-3}	5.3×10^{-4}	155	15	9.1×10^{-4}	3.5×10^{-4}
	sinusoidal	663	56	1.9×10^{-3}	9.8×10^{-4}	162	12	1.5×10^{-3}	6.4×10^{-4}
	sigmoidal	2077	86	1.3×10^{-2}	2.6×10^{-3}	172	8	5.7×10^{-3}	2.3×10^{-3}
$f_3(x)$	*variable*	770	57	7.4×10^{-3}	2.9×10^{-3}	154	15	3.1×10^{-3}	1.2×10^{-3}
	sinusoidal	745	31	6.4×10^{-3}	3.3×10^{-3}	153	8	4.7×10^{-3}	9.0×10^{-4}
	sigmoidal	2140	140	2.3×10^{-2}	9.2×10^{-3}	110	25	5.4×10^{-2}	1.7×10^{-2}

a. Each cascade network $\Gamma(\cdot)$ can have either (1) variable, (2) sinusoidal, or (3) sigmoidal hidden units.
b. Average number of epochs (N) required to build a 10-hidden-unit network (over 25 trials).
c. Standard deviation of the number of epochs required to build a 10-hidden-unit network (over 25 trials).
d. Average root-mean-squared (RMS) error (e_{RMS}) for a 10-hidden-unit network (over 25 trials).
e. Standard deviation of the average RMS error for a 10-hidden-unit network (over 25 trials).

ing human control strategy. We require this architecture to learn dynamic, nonlinear, stochastic, and possibly discontinuous control strategies, where the input space for these control strategies can be relatively large (i.e., on the order of 10-100 inputs for simple control strategies, significantly more than 100 for complicated strategies). In the next section, we propose the *cascade learning architecture* with *extended Kalman filtering* as the flexible and efficient basis for modeling human control strategy.

Since we are learning HCS models from experimental data, we must independently verify or *validate* the computed models' fidelity to the source data. Standard cross-validation in neural network training may not suffice, since we are not interested in similarity between individual training patterns, but similarity between overall system trajectories. In the third section, we propose an independent stochastic *model validation* procedure, based on *Hidden Markov Models (HMMs)* to ensure that our learned HCS models are a true representation of the human control training data. Model fidelity is characterized by a stochastic *similarity measure* that compares the dynamic trajectories of the human source data and the learned HCS model through HMM observation probabilities.

To date, virtually all learning regimes (including neural networks) require that a specific *input representation* be chosen prior to learning. In general, learning performance degrades significantly if uncorrelated inputs are presented to the learning algorithm. For human control strategy, however, we do not know the best input representation *a priori*, as it will vary from one individual to the next. We propose an algorithm that automatically selects the best *input representation* for the HCS models to maximize model fidelity to the source training data. We combine *simultaneously perturbed stochastic approximation (SPSA)* with the stochastic validation procedure to automatically refine the HCS model's input representation. A human's *controller order, controller granularity,* and *control delay* are thereby automatically extracted in the process.

We address important issues in transferring human skill to robots and other humans. We propose to use HCS models as *virtual teachers* in *human-to-human control strategy transfer.* HCS models, rather than human instructors, are used to teach a novice operator an expert's human control strategy. We address issues in *learning monitoring*, and the *selection of "good" virtual teachers*. Finally, we briefly discuss future work and additional avenues of research.

Learning Human Control Strategy

In modeling human control strategy (HCS), as with other poorly understood phenomena, we must rely on modeling by observation, or *learning*, rather than theoretical or physical derivation. An individual's HCS is characterized by unique, complex, and unknown properties; as such, we require a learning paradigm that can cope with many difficult challenges. First of all, little if anything is known *a priori* about the (1) structure, (2) order, (3) granularity, or (4) control delay inherent in a particular individual's internal controller. Second, human control strategy is dynamic, stochastic, and nonlinear in nature. Humans are not machines, and their actions are prone to errors and gradual changes over time. In addition, human control actions can vary smoothly as well as discontinuously with sensory inputs. Thus, human control strategy is a stochastic, nonlinear, possibly discontinuous mapping from present and prior sensory inputs and control actions to future control action outputs. To address these challenges, we use *cascade neural networks* with *variable activation functions* as the foundation for learning human control strategy.

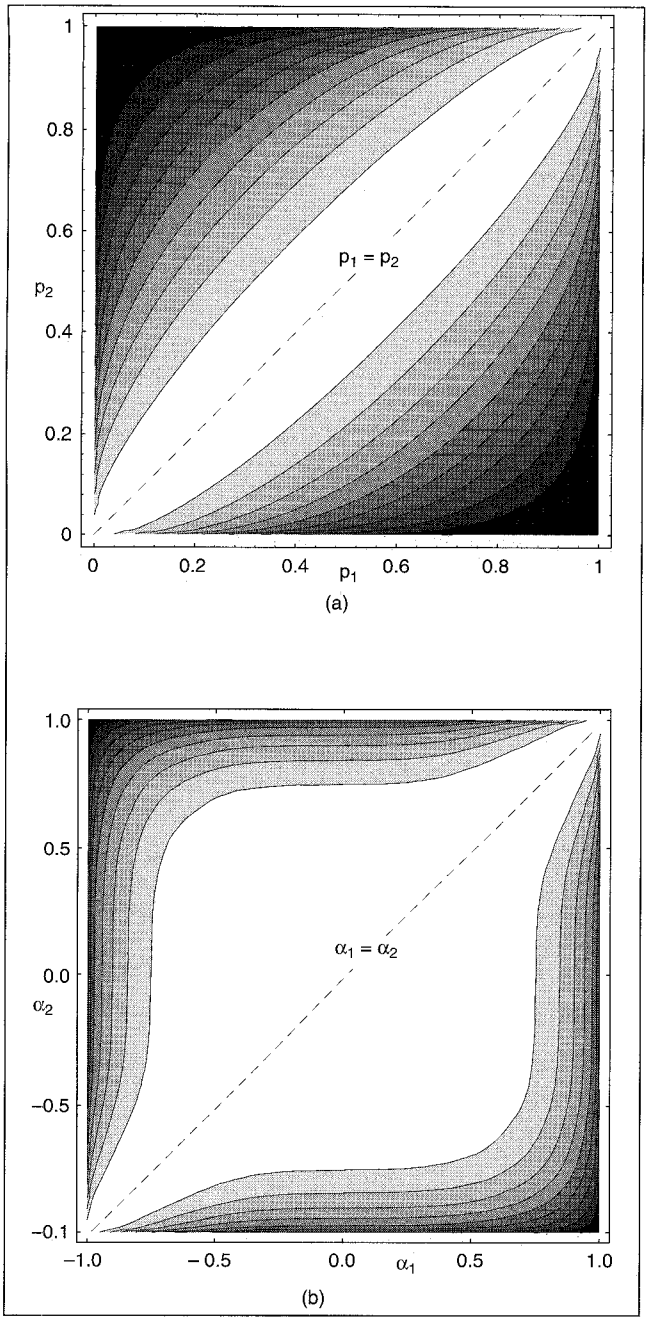


Fig. 3. (a) Similarity measure for two binomial distributions (lighter colors indicate greater similarity); (b) the similarity measure changes predictably as a function of HMM structure.

Cascade Neural Network Architecture

Cascade neural networks [14] are ideally suited for learning complex, nonlinear HCS mappings. Unlike more conventional neural network architectures, the structure of cascade neural networks is not fixed before learning begins, but evolves as part of learning. As such, the cascade learning algorithm combines both aspects of function approximation—namely, (1) the selection of an appropriate functional form and (2) the adjustment of free parameters in the functional model to optimize some error criterion. This is especially important in learning HCS models, since so little is known about the underlying human controller structure.

The flexible cascade learning architecture combines the following two notions to adjust the structure of the neural network as part of learning: (1) a feedforward cascade architecture, in which hidden units are automatically added one at a time to an initially minimal network, and (2) the learning algorithm, which creates and installs new hidden units as the learning requires in order to reduce the RMS error (e_{RMS}) between the network's outputs and the training data.

As originally formulated in [14], network training proceeds in several steps. Initially, there are no hidden units in the network, only direct input-output connections. These weights are trained first, thereby capturing any linear relationship between the inputs and outputs. With no further depreciable decrease in the error measure, a first hidden unit is added to the network from a pool of *candidate* units. Using the quickprop algorithm [15], these candidate units are trained independently and in parallel with different random initial weights.

Again, after no more appreciable error reduction occurs, the best candidate unit is selected and installed in the network. Once installed, the hidden unit input weights are frozen, while the weights to the output units are retrained. By freezing the input weights for all previous hidden units, each training cycle is equivalent to training a three-layer feedforward neural network with a single hidden unit. This allows for much faster convergence of the weights during training than in a standard backpropagation network where many hidden-unit weights are trained simultaneously. The process is repeated until the algorithm succeeds in reducing e_{RMS} sufficiently for the training set or the number of hidden units reaches a specified maximum number. Fig. 1 illustrates, for example, how a two-input, single-output network grows as two hidden units are added. We note that a cascade network with n_i input units (including the bias unit), n_h hidden units, and n_o output units, will have n_w connections where,

$$n_w = n_i n_o + n_h(n_i + n_o) + (n_h - 1)n_h / 2 . \quad (1)$$

Recent theorems which hold that standard layered neural networks are universal function approximators [16-19] also hold for the cascade network topology, since any multi-layer feedforward neural network with k hidden units arranged in m layers, fully connected between consecutive layers, is a special case of a cascade network with k hidden units with some weight connections equal to zero. Below, we discuss two ways to augment standard cascade learning to improve functional flexibility, learning speed, and error convergence: (1) variable activation functions and (2) node-decoupled extended Kalman filtering.

Variable Activation Functions

The cascade architecture relaxes *a priori* assumptions about the functional form of the model to be learned by dynamically adjusting the network size. We further relax these assumptions by allowing new hidden units to have *variable activation functions*. This idea is similar to projection pursuit regression, a statistical procedure where mappings are approximated through an iterative sum of separable nonlinear functions of the input variables [20, 21]. In fact, Cybenko [19] shows that sigmoidal functions are not the only possible activation functions that allow for

Table 2. Variable Cascade Networks vs. Multi-Layer Feedforward Networks

	Multi-Layer Feedforward Neural Network				Variable Cascade Network with NDEKF[a]		
	Topology	Free Parameters	$\overline{N}$[b]	$\overline{e}_{RMS}$[c]	Free Parameters	$\overline{N}$	$\overline{e}_{RMS}$
$f_1(x)$	1-25-1	76	2150	1.8×10^{-2}	77	182	1.2×10^{-4}
	1-105-1	81	4200	8.6×10^{-3}	77	182	1.2×10^{-4}
$f_2(x)$	1-25-1	76	5350	4.0×10^{-3}	77	155	9.1×10^{-4}
$f_3(x)$	1-25-1	76	4750	5.9×10^{-2}	77	154	3.1×10^{-3}

a. Each cascade network is allowed to grow to 10 hidden units.
b. Average number of epochs (N) over 25 trials.
c. Average root-mean-squared (RMS) error (e_{RMS}) over 25 trials.

universal function approximation. There are other nonlinear functions, such as *sine* and *cosine*, for example, that are complete in the space of n-dimensional continuous functions. In the pool of candidate units, we can assign a different nonlinear activation function to each unit, rather than just the standard sigmoidal function. During candidate training, the algorithm will select for installment whichever candidate unit reduces e_{RMS} for the training data the most. Hence, the unit with the most appropriate activation function at that point during training is selected. Typical alternatives to the sigmoidal activation function are the Gaussian function, Bessel functions, and sinusoidal functions of various frequency.

Node-Decoupled Extended Kalman Filtering

While quickprop is an improvement over the standard backpropagation algorithm for adjusting the weights in the cascade network, it is still essentially a gradient-descent based algorithm, which, although simple, can require many iterations until satisfactory convergence is reached [15, 22]. Thus, we modify standard cascade learning by replacing the quickprop algorithm with *node-decoupled extended Kalman filtering (NDEKF)* [23], which has been shown to have better convergence properties and faster training times than gradient-descent techniques for multi-layer feedforward networks.

In *general extended Kalman filtering (GEKF)* [22], an m x m conditional error covariance matrix P, which stores the interdependence of each pair of m weights in a given neural network is explicitly generated. NDEKF reduces this computational and storage complexity by—as the name suggests—decoupling weights by node, so that we consider only the interdependence of weights feeding into the same unit (or node). This, of course, is a natural formulation for cascade learning, since we only train the input-side weights of one hidden unit and the output units at any one time; we can partition the m weights by unit into $n_o + 1$ groups—one group for the current hidden unit, n_o groups for the output units. In fact, by iteratively training one hidden unit at a time and then freezing that unit's weights, we minimize the potentially detrimental effect of the node-decoupling.

Denote ω_k^i as the input-side weight vector of length m_i at iteration k, for unit $i \in \{0, 1, ..., n_o\}$, where $i = 0$ corresponds to the current hidden unit being trained, and $i \in \{1, ..., n_o\}$ corresponds to the ith output unit. The NDEKF weight-update recursion is then given by

$$\omega_{k+1}^i = \omega_k^i + \left\{\left(\psi_k^i\right)^T \left(A_k \xi_k\right)\right\} \phi_k^i , \tag{2}$$

where ξ_k is the n_o-dimensional error vector for the current training pattern, ψ_k^i is the n_o-dimensional vector of partial derivatives of the network's output unit signals with respect to the ith unit's net input, and

$$\phi_k^i = P_k^i \zeta_k^i , \tag{3}$$

$$A_k = \left[I + \sum_{i=0}^{n_o} \left\{ \left(\zeta_k^i\right)^T \phi_k^i \right\} \left[\psi_k^i \left(\psi_k^i\right)^T \right] \right]^{-1} , \tag{4}$$

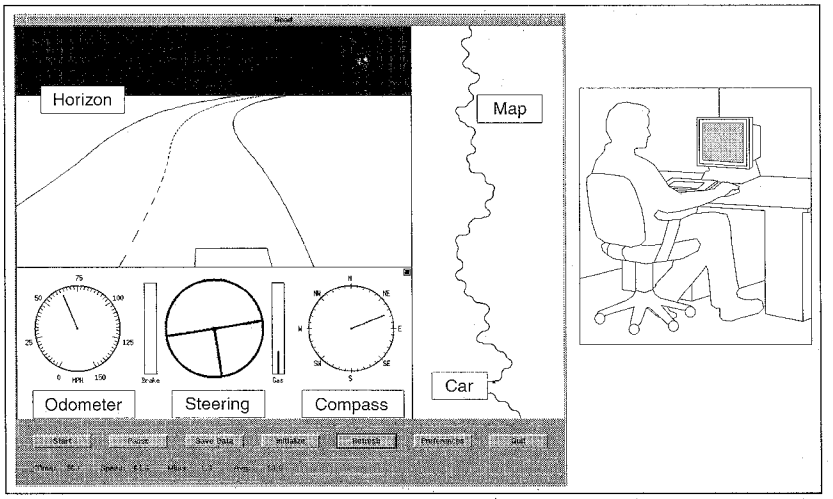


Fig. 4. The driving simulator gives the user a perspective preview of the road ahead. The user has independent controls of the steering, brake, and accelerator (gas).

$$P_{k+1}^{i} = P_{k}^{i} - \left\{ \left(\psi_{k}^{i}\right)^{T} \left(A_{k}\psi_{k}^{i}\right) \right\} \left[\phi_{k}^{i} \left(\phi_{k}^{i}\right)^{T} \right] + \eta I \,, \tag{5}$$

where ζ_k^i is the m_i-dimensional input vector for the ith unit, P_k^i is the m_i x m_i approximate conditional error covariance matrix for the ith unit, and η is a small number (0.0001) which alleviates singularity problems for P_k^i [23]. In (2) through (5), { }'s, ()'s, and []'s evaluate to scalars, vectors and matrices, respectively. The computational complexity for cascade learning with NDEKF is given by

$$O\left(n_o^3 + \sum_{i=1}^{n_o} m_i^2 \right), \tag{6}$$

while the storage complexity is given by

$$O\left(\sum_{i=0}^{n_o} m_i^2 \right). \tag{7}$$

Table 1 summarizes cascade learning results for three sample continuous functions:

$$f_1(x) = x^3 + 0.3x^2 - 0.4x, x \in [-1, 1] \tag{8}$$

$$f_2(x) = 1 / (1 + x^2), x \in [-4, 4] \tag{9}$$

$$f_3(x) = 0.6 \sin(\pi x) + 0.3 \sin(3\pi x) + 0.1 \sin(5\pi x), x \in [-1, 1] \tag{10}$$

Since as many as 80 percent of all variable activation functions selected during training are of some sinusoidal type [24,25], we include results for cascade networks with only (1) variable, (2) sinusoidal and (3) sigmoidal hidden units. The results in Table 1 are averaged over 25 trials each, with 10 hidden units allowed in each cascade network. For each example, 1500 randomly selected data points are chosen for training, and 1500 points for cross validation. Table 2 compares the performance of variable cascade networks (trained using NDEKF) to that of comparably sized multi-layer feedforward neural networks (trained using quickprop) for the same three examples.

From Tables 1 and 2, we note the following. First, without exception, variable cascade networks (trained using NDEKF) converge to the smallest approximation error, followed closely by sinusoidal cascade networks (trained using NDEKF). In many instances, the improvement over cascade networks with sigmoidal units or those trained using quickprop is an order of magnitude difference. Second, cascade networks trained with NDEKF converge (by far) in the fewest number of epochs. Therefore, we feel well motivated in preferring this learning architecture over others due to (1) its efficiency in learning speed, (2) its flexibility in functional form and (3) its good function approximation properties. All the cascade networks described in the remainder of this article have either variable or exclusively sinusoidal hidden units.

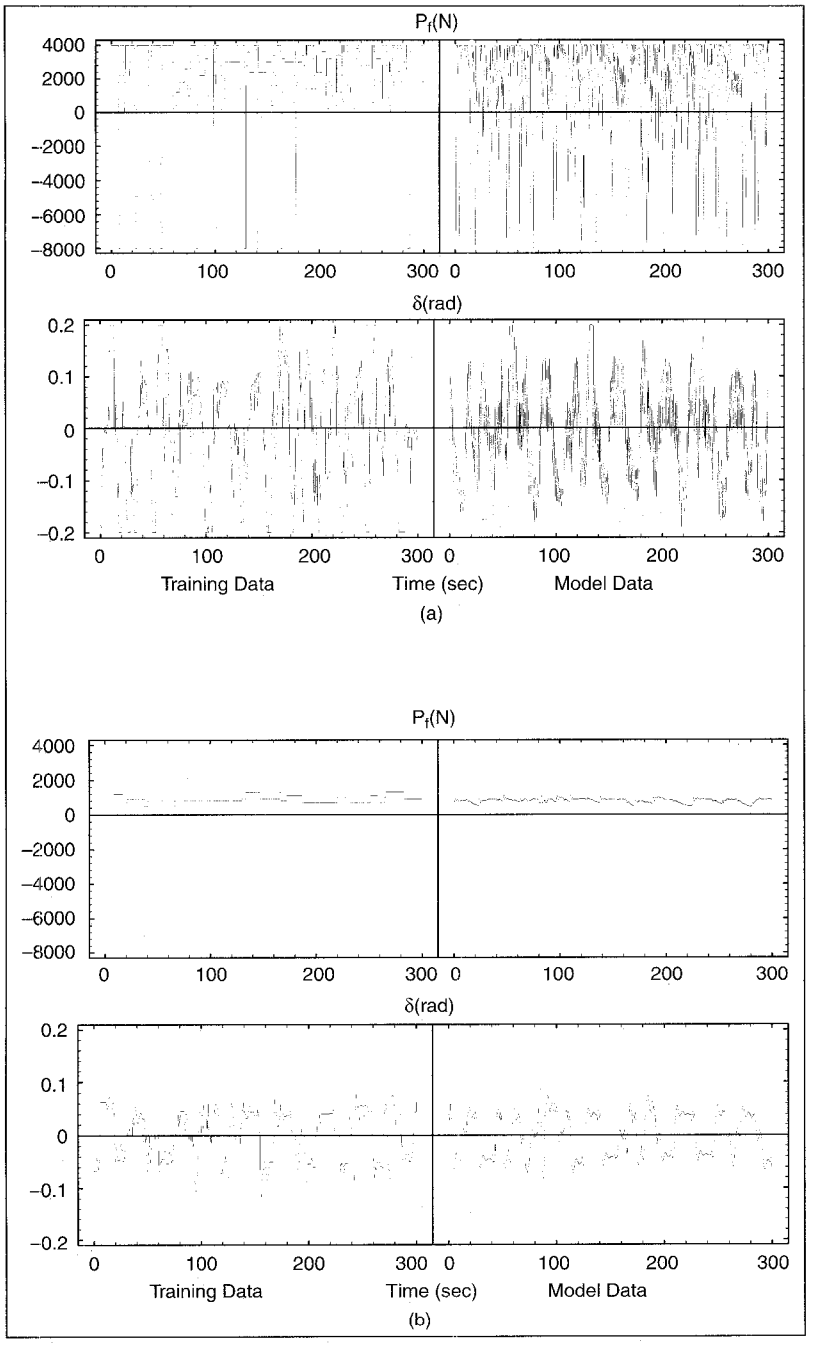


Fig. 5. (a) Oliver's driving data, and (b) Stan's driving data. On the left of each figure, we show part of the source training data; on the right of each figure we show the corresponding model-generated data.

Mapping Dynamic Systems into Static Cascade Networks

The cascade architecture approximates only static mappings; human control strategy is dy-

namic, however. That is, HCS depends not only on current sensory perceptions, but also on a recent time history of the sensory data. Therefore, a dynamic system (i.e., the HCS) must be mapped onto a static map (i.e., the cascade network). We can approximate a dynamic system by a difference equation of the general form,

$$\bar{u}\left[(k+1)\tau+\delta\right]=\Gamma\left(\bar{x}[k\tau]\right),\bar{x}\left[(k-1)\tau\right],\ldots.,$$
$$\bar{x}\left[(k-m)\tau\right],\bar{u}[k\tau],\bar{u}\left[(k-1)\tau\right],\ldots,\bar{u}\left[(k-n)\tau\right], \quad (11)$$

where $\Gamma(\cdot)$ is some arbitrary unknown function, $\bar{u}(k)$ is a vector of control outputs, $\bar{x}(k)$ is a vector of sensory inputs (including both state and environment variables) at time step k, τ indicates the controller resolution or granularity, and δ indicates the control delay. The order of the dynamic system is given by the constants n and m, which may be infinite. From (11), we observe that a static neural network can learn the unknown dynamic mapping $\Gamma(\cdot)$ by providing a sufficient time history of data as input to the neural network [26, 27].

Model Validation

The main strength of modeling by learning, is that no explicit physical model is required; however, this also represents its biggest weakness. On the one hand, we are not restricted by the limitations of current scientific knowledge, and are able to model HCS for which we have not yet developed adequate biological or psychological understanding. On the other hand, the lack of scientific justification detracts from the confidence that we can show in these learned models. This is especially true when the unmodeled process is (1) dynamic and (2) stochastic in nature, as is the case for human control strategy. For a dynamic process, model errors can feed back on themselves to produce trajectories that are not characteristic of the source process or are even potentially unstable. For a stochastic process, a static error criterion, based on the difference between the training data and predicted model outputs may be inadequate and inappropriate to gauge the fidelity of a learned model to the source process. Yet most learning approaches today, including the cascade learning algorithm, utilize a static error measure (e_{RMS}) as a test of convergence. While this measure is very useful during training, it offers no guarantees, theoretical or otherwise, about the dynamic behavior of the resulting learned model (see example in [28]). We therefore require a procedure that evaluates the complete trajectories generated by the learned model and compares those to the trajectories of the original human control data. Such a procedure allows us to perform multiple tasks in the context of human control strategy modeling:

- Validate the dynamic performance of existing models.
- Compare different methodologies of modeling.
- Characterize differences in control strategies among different individuals.

To this end, we have developed a *stochastic similarity measure* based on Hidden Markov Models, capable of comparing dynamic, multi-dimensional trajectories.

Stochastic Similarity with Hidden Markov Models

Similarity measures or metrics have been given considerable attention in computer vision [29-31], image database retrieval [32], and 2-D or 3-D shape analysis [33, 34]. These methods, however, generally rely on the special properties of images, and are therefore not appropriate for analyzing sequential trajectories. Other work has focused on classifying temporal patterns using standard statistical techniques [35], wavelet analysis [36],

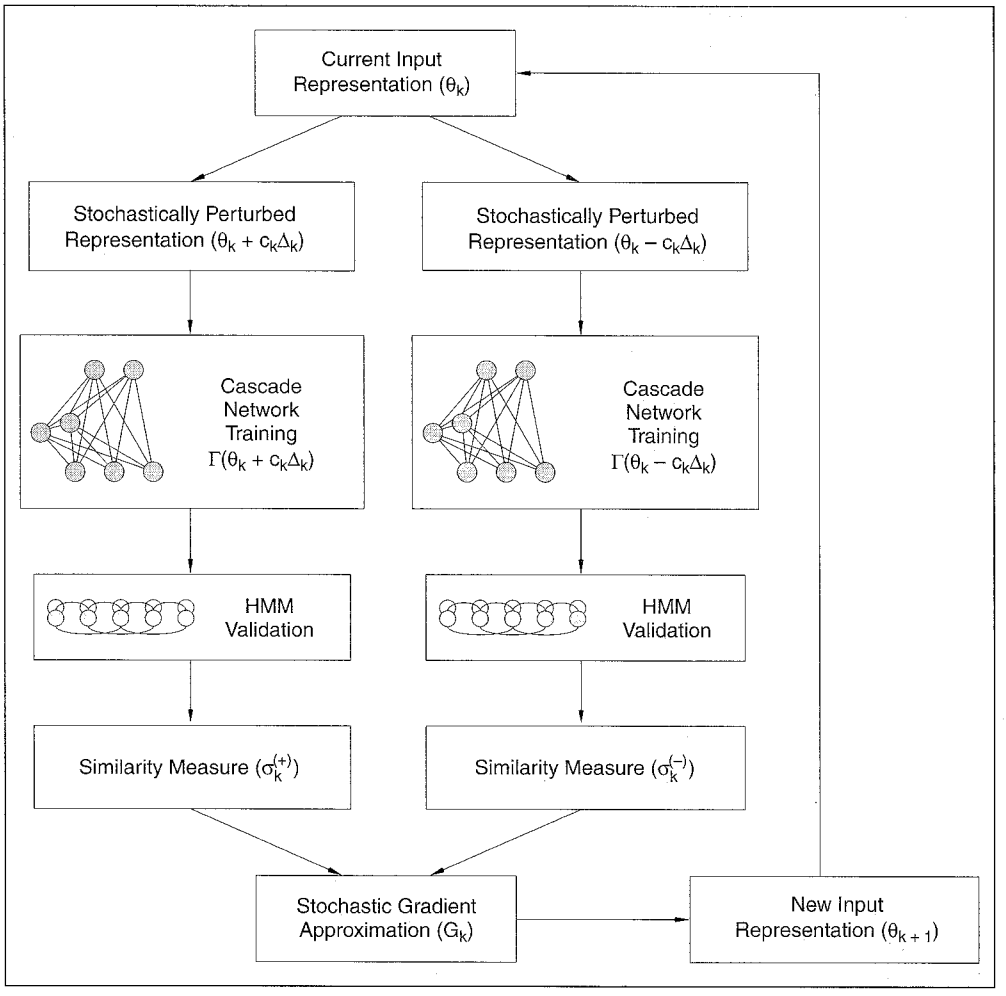


Fig. 6. Overall approach to stochastic optimization of controller structure.

Table 3. Similarity Between Model Data and Human Data				
σ	Stan	Stan's HCS Model	Oliver	Oliver's HCS Model
Stan	1.000	0.748	0.009	0.007
Stan's HCS Model	0.748	1.000	0.006	0.004
Oliver	0.009	0.006	1.000	0.349
Oliver's HCS Model	0.007	0.004	0.349	1.000

neural networks [37, 38], and Hidden Markov Models (see discussion below). Much of this work, however, analyzes only short-time trajectories or patterns, and, in many cases, generates only a binary classification, rather than a continuously valued similarity measure. Prior work has not addressed the problem of comparing long, multi-dimensional, stochastic trajectories, especially of human control data. Thus, we propose to evaluate stochastic similarity between two dynamic, multi-dimensional trajectories using *Hidden Markov Model (HMM)* analysis.

Rich in mathematical structure, HMMs are trainable statistical models, with two appealing features: (1) no *a priori* assumptions are made about the statistical distribution of the data to be analyzed, and (2) a high degree of sequential structure can be encoded by the Hidden Markov Models. As such, they have been applied for a variety of stochastic signal processing, such as speech recognition [39, 40], transient sonar signal classification [41], task structure classification in teleoperation [42], open-loop action skill learning [43], and human gesture recognition [44].

A Hidden Markov Model consists of a set of n states, interconnected through probabilistic transitions; each of these states has some output probability distribution associated with it. Although algorithms exist for training HMMs with both discrete and continuous output probability distributions, and although most applications of HMMs deal with real-valued signals, discrete HMMs are preferred to continuous HMMs in practice, due to their relative computational simplicity and lesser sensitivity to initial random parameter settings [45]. Using discrete HMMs for analysis of real-valued signals requires that data be converted to discrete symbols through pre-processing and vector quantization [28,46]. Thus, a discrete HMM is completely defined by the triplet $\lambda = \{A, B, \pi\}$ [40], where A is the probabilistic n x n state transition matrix, B is the L x n output probability matrix with L discrete output symbols $\chi \in \{1, 2, \ldots, L\}$, and π is the n-length initial state probability distribution vector for the HMM.

For an observation sequence O of discrete symbols, Hidden Markov Model parameters can be locally optimized to maximize $P(\lambda \mid O)$ (i.e., the probability of the model λ given the observation sequence O) off-line using the Baum-Welch Expectation-Maximization (EM) algorithm [40, 47], or on-line, as presented in [48, 49]. We can also evaluate $P(O \mid \lambda)$ (i.e., the probability that a given observation sequence O is generated from the model λ).

Now, we derive a stochastic similarity measure, based on discrete-output HMMs. Let O_i, $i \in \{1, 2, \ldots\}$, denote a distinct observation sequence of discrete symbols with length T_i. Also let $\lambda_j = \{A_j, B_j, \pi_j\}, j \in \{1, 2, \ldots\}$, denote a discrete HMM locally optimized (using the Baum-Welch algorithm) to maximize $P(\lambda_j \mid O_j)$. Similarly, let $P(Oi \mid \lambda_j)$ denote the probability of the observation sequence Oi given the model λ_j, and let

$$P_{ij} = \hat{P}\left(O_i \middle| \lambda_j\right) = P\left(O_i \middle| \lambda_j\right)^{\frac{1}{T_i}} \tag{12}$$

denote the probability of the observation sequence O_i given the model λ_j, normalized with respect to T_i. In practice, we calculate P_{ij} as

$$P_{ij} = 10^{\log P(O_i|\lambda_j)/T_i} \tag{13}$$

to avoid problems of numerical underflow for long observation sequences.

Fig. 2 illustrates our overall approach to evaluating similarity between two observation sequences. Each observation sequence is first used to train a corresponding HMM; this allows us to evaluate P_{11} and P_{22}. Furthermore, we subsequently cross-evaluate each observation sequence on the other HMM (i.e. $P(O_1 \mid \lambda_2)$, $P(O_2 \mid \lambda_1)$) to arrive at P_{12} and P_{21}. Given these four normalized probability values, we now propose the following similarity measure between O_1 and O_2 (In [28], we proposed a different similarity measure which gives potentially inconsistent and misleading results for certain data. The similarity measure in (14) corrects these problems):

$$\sigma(O_1, O_2) = \sqrt{\frac{P_{21}P_{12}}{P_{11}P_{22}}} \tag{14}$$

This measure takes the ratio of the cross probabilities over the training probabilities, and normalizes for the multiplication of the two probability values in the numerator and denominator by taking the square root. We note that in practice, we calculate the P_{ij} not on λ_j itself, but rather $\hat{\lambda}_j$, which is a smoothed version of λ_j, where zero elements in the matrices $\{A_j, B_j, \pi_j\}$ are replaced by $\varepsilon > 0$ and renormalized to fit probabilistic constraints. The results reported below use $\varepsilon = 0.0001$ as the smoothing value.

We now describe some of the properties of the similarity measure defined above. First we note that (by definition)

$$\sigma(O_1, O_2) = \sigma(O_2, O_1)\,. \tag{15}$$

Second, consider the class of single-state, discrete HMMs given by

$$\lambda_j = \{A_j, B_j, \pi_j\} = \{[1], [b_{j1} \ldots b_{jL}]^T, [1]\}\,. \tag{16}$$

These HMMs essentially encode only the distribution of symbols, without capturing any of the sequential properties of obser-

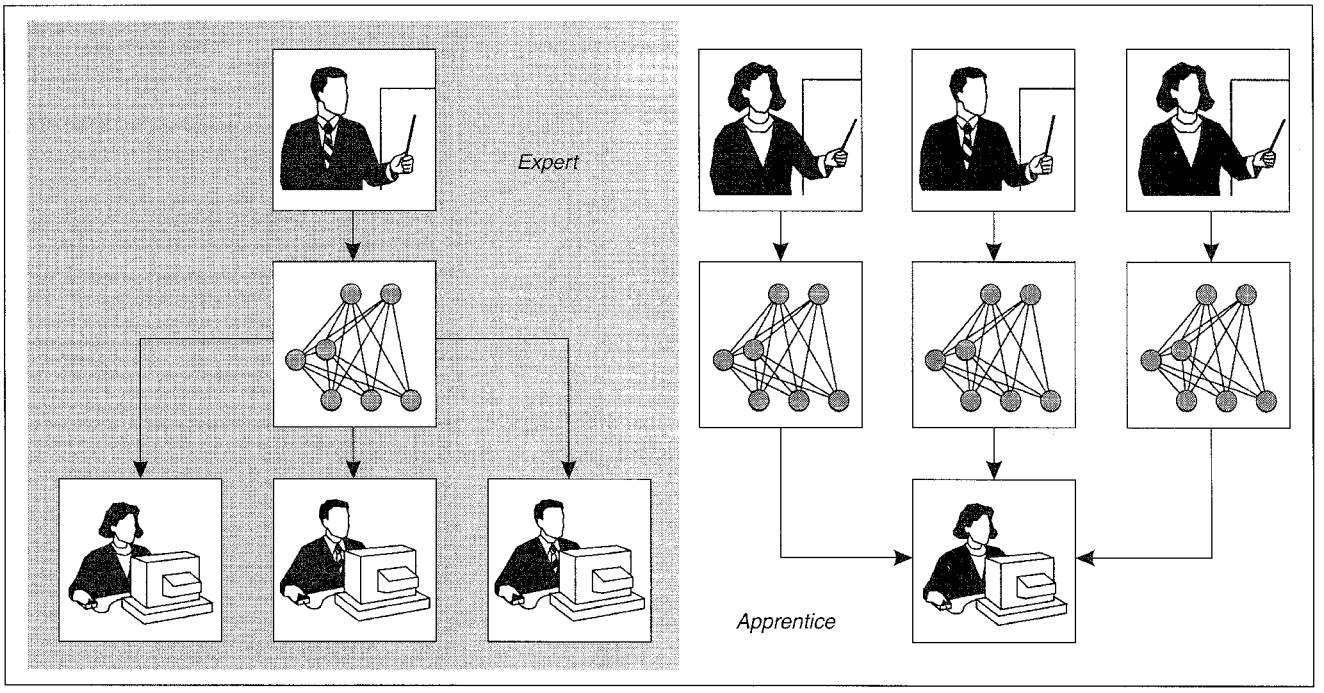


Fig. 7. One expert can teach many apprentices (left), and many experts can contribute to the learning of a single apprentice (right).

vation sequence Oj. It is easy to show that for two such models λ_1 and λ_2, the similarity measure reduces to [46],

$$\sigma(O_1, O_2) = \prod_{k=1}^{L} \left(\frac{b_{1k}}{b_{2k}} \right)^{\frac{(b_{2k} - b_{1k})}{2}}, \tag{17}$$

which reaches a maximum when $b_{1k} = b_{2k}$, $\forall k \in \{1, 2, \ldots, L\}$, or simply, $B_1 = B_2$, and that that maximum is equal to one. Furthermore, since $P_{ij} \geq 0$, $\sigma(O_1, O_2)$ itself is lower bounded by zero. Hence,

$$0 \leq \sigma(O_1, O_2) \leq 1 \tag{18}$$

and

$$\sigma(O_1, O_2) = 1 \text{ if and only if } \lambda_1 = \lambda_2 . \tag{19}$$

As an example, consider the case where

$$B_1 = [p_1 \; 1 - p_1]^T \quad B_2 = [p_2 \; 1 - p_2]^T , \tag{20}$$

which is graphed in Fig. 3(a) as a contour plot for $0 < p_1, p_2 < 1$.

For a number of reasons, a similar proof is not possible for multi-state HMMs. No known analytic solution exists for $P(O \mid \lambda)$ for general O and λ. Furthermore, $P(O \mid \lambda)$ can only be locally—not globally—maximized using the Baum-Welch algorithm. Finally, although two HMMs λ_1 and λ_2 may appear quite different such that,

$$A_1 \neq A_2 \qquad B_1 \neq B_2 \qquad \pi_1 \neq \pi_2 , \tag{21}$$

they may in fact evaluate to identical $P(O \mid \lambda)$. The following two HMM models, for example, are equivalent:

$$\lambda_1 = \left\{ [1], [0.5 \; 0.5]^T, [1] \right\} \quad \lambda_2 = \left\{ \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, [0.5 \; 0.5]^T \right\} \tag{22}$$

Below, we show one empirical example of how the similarity measure changes, not as a function of different symbol distributions, but rather as a function of varying HMM structure. Consider the following Hidden Markov Model:

$$\lambda(\alpha) = \left\{ \begin{bmatrix} \frac{1+\alpha}{2} & \frac{1-\alpha}{2} \\ \frac{1-\alpha}{2} & \frac{1+\alpha}{2} \end{bmatrix}, \begin{bmatrix} \frac{1+\alpha}{2} & \frac{1-\alpha}{2} \\ \frac{1-\alpha}{2} & \frac{1+\alpha}{2} \end{bmatrix}, [0.5 \; 0.5]^T \right\}, -1 < \alpha < 1 \tag{23}$$

and corresponding observation sequences, $O(\alpha)$, stochastically generated from model $\lambda(\alpha)$. For all $\lambda \in (-1, 1)$, $O(\alpha)$ will have an equivalent aggregate distribution of symbols 0 and 1—namely 1/2 and 1/2. As $|\alpha|$ increases, however, $O(\alpha)$ will become increasingly structured. For example,

$$\lim_{\alpha \to -1} O(\alpha) = \{..., 1, 0, 1, 0, 1, 0, 1, 0, ...\} \tag{24}$$

$$\lim_{\alpha \to 0} \lambda(\alpha) = \left\{ [1], [0.5 \; 0.5]^T, [1] \right\} \text{(equivalent to unbiased coin toss)} \tag{25}$$

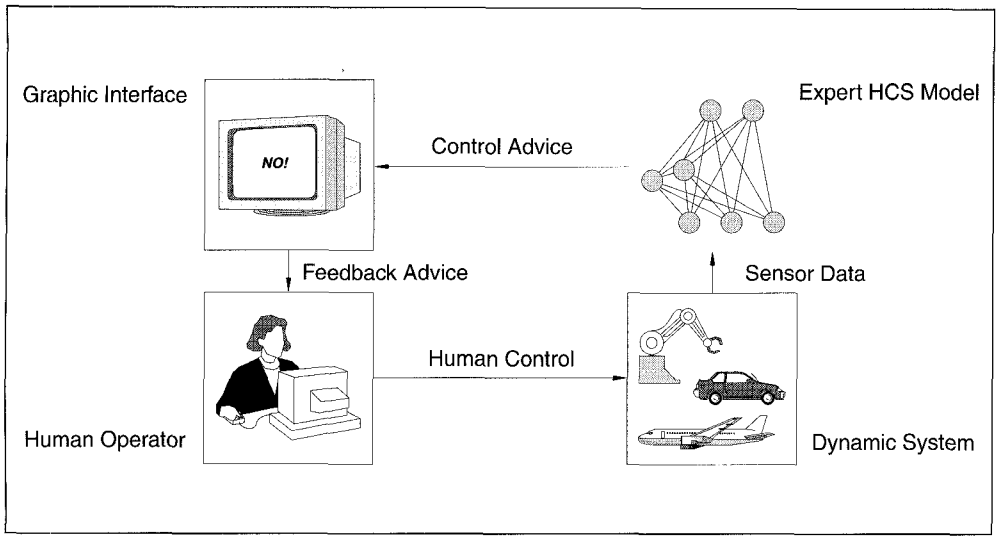


Fig. 8. Human-to-human control strategy transfer utilizes expert HCS models to train human operators.

$$\lim_{\alpha \to 1} O(\alpha) = \{..., 1, 1, 1, 0, 0, 0, ..., 0, 0, 0, 1, 1, 1, ...\} \quad (26)$$

Fig. 3(b) graphs $\sigma[O(\alpha_1), O(\alpha_2)]$ as a contour plot for $-1 < \alpha_1, \alpha_2 < 1$, where each observation sequence $O(\alpha)$ of length $T = 10,000$ is generated stochastically from the corresponding HMM $\lambda(\alpha)$. Greatest similarity is indicated for $\alpha_1 = \alpha_2$, and around the origin, ($\alpha_1 = 0, \alpha_2 = 0$, while greatest dissimilarity occurs for $\alpha_1 \to 1, \alpha_2 = -1$), and ($\alpha_1 \to -1, \alpha_2 = 1$).

Signal-to-Symbol Conversion

Since we choose to work with discrete-output HMMs, multidimensional, real-valued human control data must be converted to a sequence of discrete symbols in order to apply the similarity measure defined in Equation (14). Such conversion involves two steps: (1) spectral preprocessing and (2) vector quantization. The primary purpose of the spectral preprocessing is to extract meaningful feature vectors for the vector quantizer. In this work, we rely on the fast Fourier transform (FFT) and the fast Walsh transform (FWT), the $O(n \log n)$ algorithmic counterparts of the discrete Fourier transform (DFT) and the discrete Walsh transform (DWT), respectively. Instead of sinusoidal basis functions, the Walsh transform decomposes a signal based on the orthonormal Walsh functions [50]. Certain types of sharply discontinuous human control data are characterized more concisely through the Walsh PSD than the Fourier PSD [46].

For each dimension of the human control data, we partition the data into overlapping window frames and perform either a short-time FFT or FWT on each frame. Generally, we select the FFT for *state trajectories*, and the FWT for *command trajectories*, since these trajectories tend to have sharp discontinuities for the experimental data in this article. In the case of the FFT, the data in each frame is filtered through a Hamming window before applying the FFT, so as to compensate for the windowing effect. The spectral coefficients are then converted to power spectral density (PSD) vectors. In preparation for the vector quantization, the PSD vectors along each dimension of the system trajectory are normalized and concatenated into one long feature vector per frame. We quantize the resulting sequence of long feature vectors using the iterative LBG VQ algorithm [51]. This vector quantizer generates codebooks of size 2^m, $m \in \{0, 1, 2, ...\}$, and can be stopped at an appropriate level of discretization given the amount of available data and complexity of the system trajectories. Assuming that we segment the data into window frames of length k with 50% overlap, the original multidimensional, real-valued signal of length t is thus converted to a sequence of discrete symbols of length $T = \text{int}(2t/k)$.

Experiment

Here, we present validation results for the task of human driving [46]. With independent steering, acceleration, and braking control, a human operator is asked to negotiate different roads in a driving simulator, whose interface is shown in Fig. 4. The state of the car is given by [52]

$$\dot{\theta} = \text{angular velocity of the car,} \quad (27)$$

$$v_\xi = \text{lateral velocity of the car,} \quad (28)$$

$$v_\eta = \text{longitudinal velocity of the car,} \quad (29)$$

and the controls are given by

$$-8000\text{N} \le (P_f = \text{longitudinal force on front tires}) \le 4000\text{N}, \quad (30)$$

$$-0.2 \text{ rad} \le (\delta = \text{steering angle}) \le 0.2 \text{ rad}. \quad (31)$$

Note that the separate brake and gas commands for the human are, in fact, the single P_f variable, where the sign indicates whether the brake or the gas is active.

We collect data from two human operators, Stan and Oliver. For each individual we train a simple HCS model using cascade networks. The inputs to the cascade networks for each individual include (1) current and previous state information, (2) previous control information, and (3) a description of the road:

$$\left\{ \begin{matrix} v_\xi(k-n_s), ..., v_\xi(k-1), v_\xi(k), v_\eta(k-n_s), \\ ..., v_\eta(k-1), v_\eta(k), \dot{\theta}(k-n_s), ..., \dot{\theta}(k-1), \dot{\theta}(k) \end{matrix} \right\}, \quad (32)$$

$$\{\delta(k - n_c), ..., \delta(k - 1), \delta(k), P_f(k - n_c), ..., P_f(k - 1), P_f(k)\} \quad (33)$$

$$\{x(1), x(2), ..., x(n_r), y(1), y(2), ..., y(n_r)\}, \quad (34)$$

where n_s = length of state history to include as input, and n_c = length of command history to include as input. State and command histories are spaced at $\tau = 0.02$ sec. For the road description, we discretize the visible view (given a 100m horizon) 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, for fixed n_s, n_c, and n_r, the total number of inputs to the network is

$$n_{inputs} = 3n_s + 2n_c + 2n_r(35) \tag{35}$$

The outputs for the cascade network are, of course, $\{\delta(k+1), P_f(k+1)\}$ (i.e., the steering and acceleration command for the next time step).

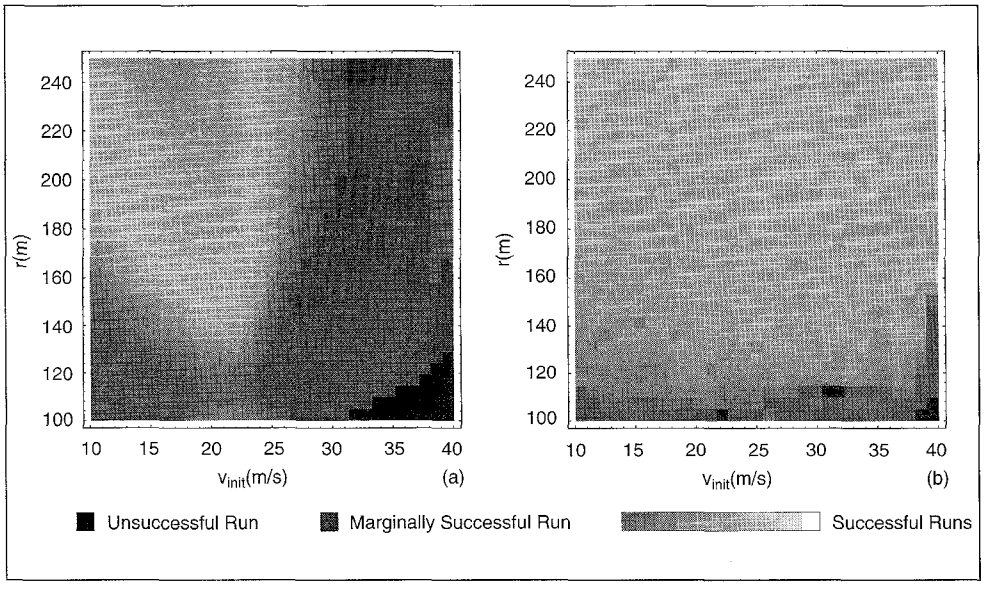


Fig. 9. Stability profiles for (a) Stan and (b) Oliver through an s-curve with varying radii and initial speeds.

The left sides of Figure 5(a) and (b) show part of the driving data collected from two individuals, Oliver and Stan. Note that the driving styles for the two individuals are quite different for the same road. The right sides of Figure 5(a) and (b) show part of the cascade model-generated command trajectories for Oliver and Stan. Since Stan's control strategy is relatively simple, his control strategy model requires 30 hidden units and only the previous two states as input (i.e., $n_s = n_c = 2$). Oliver's more complicated control strategy model, on the other hand, requires 32 hidden units and relies on the previous ten states ($n_s = n_c = 10$) to stay on the road. We determine these input histories experimentally to achieve stable road following for each HCS model. For both models, we let $n_r = 15$.

To conduct the similarity analysis, we preprocess the state trajectories using the 16-point FFT with 50% overlap, and we preprocess the command trajectories using the 16-point FWT with 50% overlap. The resulting feature vectors are quantized to 128 levels. Table 3 summarizes the similarity results for the data in Fig. 5 for eight-state HMMs.

The similarity results in Table 3 confirm two qualitative assessments of the data in Fig. 5. First, we observe that the two driving styles are objectively quite different. This fact is reflected in the low similarity measures between one individual's model and the other individual's source and model-generated data. Second, Stan's model is a better reflection of his driving style than Oliver's model is of his, as reflected in the two respective similarity measures, 0.748 and 0.349. This indicates that Oliver's sharply discontinuous driving strategy is more difficult to learn by a single cascade network than Stan's calmer approach. Indeed, Oliver's model generates significant oscillatory behavior, of which Oliver himself is not guilty. Thus, a low similarity measure can point to problems in the model itself, including improper assumptions about the controller order, granularity, and control delay of the HCS. We address these problems in the following section.

Finally, we note that we have applied the similarity measure in comparing control data across individuals. We collect driving data from five individuals over three different roads, and then attempt to correctly classify each individual's data. While the similarity measure correctly classifies every run, the more traditional Bayes optimal classifier incorrectly classifies about 1/3 of all the runs [46]. Hence, it appears that the similarity measures encodes more than simply aggregate statistics.

Input Selection

For the example, Stan's driving model requires state and command histories of length 2, while Oliver's driving model requires state and command histories of at least length 10 to ensure stable behavior. In human control of an inverted pendulum system, we have found similar variations among individuals in controller granularity and control delay [8]. Thus, we require a procedure that automatically refines the input representation for HCS models, so as to arrive at better approximations of the actual human control strategy. To this end, we propose combining the model validation procedure with *simultaneously perturbed stochastic approximation* (SPSA) [53] to select the best model input representation.

Based on (11), in order to select the best input representation for the HCS model, we need to optimize the parameter vector,

$$\theta = [m \;\; n \;\; \tau \;\; \delta]^T, \tag{36}$$

for some given experimental HCS data. Let $\Gamma(\theta)$ denote a trained HCS model with input representation θ, and let $\sigma[\Gamma(\theta)] = \sigma(\theta)$ denote the similarity measure for model $\Gamma(\theta)$. Now, the best input representation θ^* is defined in terms of the similarity measure σ, such that

$$\sigma(\theta^*) > \sigma(\theta), \; \forall \theta \neq \theta^* \tag{37}$$

This optimization is difficult in principle since (1) we have no explicit gradient information,

$$G(\theta) = \frac{\partial}{\partial\theta}\sigma(\theta) \tag{38}$$

and (2) each measurement of σ is computationally expensive. Thus, we resort to *simultaneously perturbed stochastic approximation* to perform the optimization.

We propose to evaluate the similarity measure σ for two values of θ in order to arrive at a new estimate for θ at iteration k. Fig. 6 illustrates the overall input-selection feedback loop, where Δ_k is a vector of mutually independent, mean-zero random variables (e.g., symmetric Bernoulli distributed), the sequence $\{\Delta_k\}$ is independent and identically distributed, and the $\{a_k\}$, $\{c_k\}$ are positive scalar sequences. We can verify the proposed method on dynamic systems whose input parameter vector θ is known *a priori.*

Skill Transfer

Here, we discuss the important application of HCS models as *virtual expert instructors*. Rather than getting advice from the human expert directly, an apprentice can get guidance from an expert's HCS model, which acts as the virtual teacher. The model-generated advice can be presented continuously to the apprentice, while exploiting multiple sensor modalities. This has the potential to improve both learning speed and the quality of learning by the apprentice, as is qualitatively shown in human subject experiments for an inverted-pendulum system [8]. In this approach, apprentice training need no longer be one-to-one. A single expert can efficiently train many apprentices through his/her HCS model, without increased demands on the expert's time; conversely, a single apprentice can efficiently benefit from diverse advice of many experts at once (Fig. 7). Moreover, since HCS models are trained on physically plausible human data, feedback advice from the HCS model does not require unreasonably high precision or control fidelity from the apprentice.

Fig. 8 illustrates our overall approach to human-to-human control strategy transfer. Suppose we wish to train an apprentice operator to control some dynamic system, such as a robot, car, or airplane. We first collect data from an expert to train the expert HCS model. We then utilize this expert model to display recommended actions to a human operator. The human operator then acts and learns based upon the recommendations of the expert model (i.e., the *virtual expert instructor*). During learning by the apprentice, it is advisable to replace the actual dynamic system with a simulator, as we do not wish the apprentice to do harm either to the system or to himself/herself. It has been shown that simulated training of this nature still dramatically improves performance once the apprentice transitions to control of the real dynamic system [54, 55].

Good vs. Bad Virtual Teacher

Just as an apprentice needs to feel comfortable with the teaching characteristics of a human instructor, so will the apprentice require a comfort level with his/her virtual teacher. In other words, alternate expert HCS models can result in varying learning quality and learning speed by the apprentice; in general, different apprentices will prefer HCS models of different character and complexity during training. By allowing apprentices to experiment with differing HCS models, and select which they, in fact, prefer, we can evaluate both the accepted and rejected HCS models in terms of the performance criteria discussed previously, and identify those performance criteria (if any) which differentiate the "good" from the "bad" virtual teachers.

Monitoring Apprentice Learning

We can monitor the progress of an apprentice's learning online. Let, λ_Γ denote a Hidden Markov Model trained on trajectories from the expert HCS model Γ. During learning, we can continuously monitor,

$$p = P(O_{[t,\ t+\tau]} \mid \lambda_\Gamma) \tag{39}$$

where $O_{[t,\ t+\tau]}$ denotes the symbol-converted data from the apprentice in time window $[t,\ t+\tau]$ during training. Increasing values of p indicate positive learning by the apprentice. Once p reaches some threshold value, we can test the quality of the apprentice's learning by momentarily switching off the feedback advice and monitoring the value of p. If p remains steady, the apprentice has truly learned the expert's human control strategy; if p drops sharply, however, it indicates that the apprentice is still relying heavily on the feedback advice, and more learning by the apprentice is required.

Future Work

There are of course many additional areas of research in human control strategy modeling. One such area which has important implications for the effective use of HCS models is *skill evaluation*, without which it is difficult to rank or prefer one control strategy over another. As the phrase, "garbage in, garbage out" asserts, a model's fidelity to human training data offers no estimator of the quality of the demonstrated skill itself. Model validation is a necessary, but not sufficient condition for the appropriate application of HCS models. Skill is not a unique figure of merit; rather, skill consists of an entire set of task-independent as well as task-dependent performance criteria, which may or may not be in conflict. Consider, for example, the driving control strategies in Fig. 5 for Oliver and Stan. Which control strategy demonstrates better skill? On the one hand, Stan's control strategy offers a smoother ride compared to Oliver's strategy, and is significantly more fuel-efficient to boot. On the other hand, Oliver's strategy achieves higher average speeds over equivalent roads (72 m.p.h. vs. 56 m.p.h.) and exhibits stability for a broader range of initial and environmental conditions (Fig. 9), where each control strategy model is asked to steer through s-curves of various radii and different initial velocities.

By definition, task-dependent criteria for skill evaluation cannot be enumerated. Consider, however, the task of driving as an example of human control strategy. There are any number of task-specific skill criteria in human driving such as (1) average fuel consumption per mile, (2) RMS distance from the center of the lane, (3) maximum exhibited lateral and longitudinal forces, (4) performance in tight turns, (5) average velocity, and (6) control output (i.e. steering and power) magnitudes. Task-independent criteria, on the other hand, include such performance indices as (1) long-term consistency, (2) generalizability, and (3) robustness.

Given specific skill criteria, we may be interested in optimizing a HCS model's performance. Since the unoptimized HCS model already gives an initial, stable control law, adaptive optimal control, in terms of a specified performance criterion, is now possible. Specifically, we can optimize the performance in stable HCS models through simultaneously perturbed stochastic approximation (SPSA) by measuring a specified performance index J for $2p$ perturbed models of the original HCS model, approximating the gradient of J with respect to the weights in the model, and generating a new estimate for the optimal weight vector.

Conclusion

Modeling and transferring human control strategies is an important area of research, with potential impact in a number of diverse applications. The most significant impact of this research is in the design and control of dynamic systems where humans demonstrate good skill, and where humans and machines must interact with one another. This article identifies key problems in modeling and transferring human control strategy, offers methodologies to deal with each problem in turn, and summarizes some results of this ongoing work. We explicitly address the learning of HCS models, independent validation, and transfer of human skill from human to robot, as well as human to human. We have implemented much of the work proposed herein in the real-time control of two simulated testbeds, and are working to transfer the methodologies for the navigation and control of autonomous and teleoperated robots and land vehicles.

References

[1] K. Chen and R. D. Ervin, "Worldwide IVHS Activities: A Comparative Overview," *Proc. CONVERGENCE'92 - Int. Congress on Transportation Electronics*, pp. 339-349, 1992.

[2] W.C. Collier and R.J. Weiland, "Smart Cars, Smart Highways," *IEEE Spectrum*, vol. 31, no. 4, pp. 27-33, 1994.

[3] D. O'Hare and S. Roscoe, Flight Deck Performance: The Human Factor, Iowa State University Press, Ames, 1990.

[4] C.C. Lee, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller—Part I," *IEEE Trans. Systems, Man and Cybernetics*, vol. 20, no. 2, pp. 404-418, 1990.

[5] C.C. Lee, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller—Part II," *IEEE Trans. Systems, Man and Cybernetics*, vol. 20, no. 2, pp. 419-435, 1990.

[6] B. Widrow, "The Original Adaptive Neural Net Broom Balancer," *Proc. IEEE Symp. on Circuits and Systems*, vol. 2, pp. 351-357, 1987.

[7] A. Guez and J. Selinsky, "A Trainable Neuromorphic Controller," *Journal of Robotic Systems*, vol 5., no. 4, pp. 363-388, 1988.

[8] M. Nechyba and Y. Xu, "Human Skill Transfer: Neural Networks as Learners and Teachers," *Proc. IEEE Int. Conf. on Intelligent Robots and Systems*, vol. 3, pp. 314-319, August 1995.

[9] H. Asada and S. Liu, "Transfer of Human Skills to Neural Net Robot Controllers," *Proc. IEEE Int. Conf. on Robotics and Automation*, vol. 3, pp. 2442-2447, 1991.

[10] S. Lee and J. Chen, "Skill Learning from Observations," *Proc. IEEE Int. Conf. on Robotics and Automation*, vol. 4, pp. 3245-3250, 1994.

[11] Y. Xu and J. Yang, "Towards Human-Robot Coordination: Skill Modeling and Transferring via Hidden Markov Model," *Proc. IEEE Int. Conf. on Robotics and Automation*, vol. 1, pp. 906-911, 1995.

[12] E. Fix and H.G. Armstrong, "Modeling Human Performance with Neural Networks," *Proc. Int. Joint Conf. on Neural Networks*, vol. 1, pp. 247-252, 1990.

[13] D.A. Pomerleau, "Neural Network Perception for Mobile Robot Guidance," Ph.D. thesis, School of Computer Science, Carnegie Mellon University, 1992.

[14] S.E. Fahlman and C. Lebiere, "The Cascade-Correlation Learning Algorithm," *Advances in Neural Information Processing Systems* 2, ed. D. S. Touretzky, Morgan Kaufmann Publishers, pp. 524-532, 1990.

[15] S.E. Fahlman, "An Empirical Study of Learning Speed in Back-Propagation Networks," *Technical Report, CMU-CS-TR-88-162*, Carnegie Mellon University, 1988.

[16] K. Funahashi, "On the Approximate Realization of Continuous Mappings by Neural Networks," *Neural Networks*, vol. 2, no. 3, pp. 183-192, 1989.

[17] K. Hornik, M. Stinchcomb, and H. White, "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks,* vol. 2, no. 3, pp. 359-366, 1989.

[18] V. Kurkova, "Kolmogorov's Theorem and Multilayer Neural Networks," *Neural Networks*, vol. 5, no. 3, pp. 501-506, 1992.

[19] G. Cybenko, "Approximation by Superposition of a Sigmoidal Function," *Mathematics of Control, Signals, and Systems*, vol. 2, no. 4, pp. 303-314, 1989.

[20] J.H. Friedman and W. Stuetzle, "Projection Pursuit Regression," *Journal of the American Statistical Association*, vol. 76, no. 376, pp. 817-823, 1981.

[21] J. Hwang and I. Hang, "A Comparison of Projection Pursuit and Neural Network Regression Modeling," *Advances in Neural Information Processing Systems* 4, ed. J.E. Moody, S.J. Hanson and R.P. Lippmann, Morgan Kaufmann Publishers, pp. 1159-1166, 1992.

[22] S. Singhal and L. Wu, "Training Multilayer Perceptrons with the Extended Kalman Algorithm," *Advances in Neural Information Processing Systems* 1, ed. D.S. Touretzky, Morgan Kaufmann Publishers, pp. 133-140, 1989.

[23] G.V. Puskorius and L.A. Feldkamp, "Decoupled Extended Kalman Filter Training of Feedforward Layered Networks," *Proc. Int. Joint Conf. on Neural Networks*, vol. 1, pp. 771-777, 1991.

[24] M. Nechyba and Y. Xu, "Neural Network Approach to Control System Identification with Variable Activation Functions," *Proc. IEEE Int. Symp. Intelligent Control*, pp. 358-363, 1994.

[25] M. Nechyba and Y. Xu, "Towards Human Control Strategy Learning: Neural Network Approach with Variable Activation Functions," *Technical Report, CMU-RI-TR-95-09*, Carnegie Mellon University, 1995.

[26] K.S. Narendra, "Adaptive Control of Dynamical Systems Using Neural Networks," *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, ed. D.A. White and D.A. Sofge, pp. 141-184, 1992.

[27] K.S. Narendra and K. Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Trans. Neural Networks*, vol. 1, no. 1, pp. 4-27, 1990.

[28] M. Nechyba and Y. Xu, "On the Fidelity of Human Skill Models," *Proc. IEEE Int. Conf. in Robotics and Automation*, vol. 3, pp. 2688-2693, 1996.

[29] R. Basri and D. Weinshall, "Distance Metric Between 3D models and 2D Images for Recognition and Classification," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. 18, no. 4, pp. 465-479, 1996.

[30] M. Boninsegna and M. Rossi, "Similarity Measures in Computer Vision," *Pattern Recognition Letters*, vol 15., no. 12, pp. 1255-1260, 1994.

[31] M. Werman and D. Weinshall, "Similarity and Affine Invariant Distances Between 2D point Sets," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 810-814, 1995.

[32] R. Jain, S. Murty, et al., "Similarity Measures for Image Databases," *Proc. IEEE Int. Conf. on Fuzzy Systems*, vol. 3, pp. 1247-1254, 1995.

[33] K.Y. Kupeev and H.J. Wolfson, "On Shape Similarity," *Proc. of 12th IAPR Int. Conf. on Pattern Recognition*, vol. 1, pp. 227-231, 1994.

[34] H.Y. Shum, M. Hebert, and K. Ikeuchi, "On 3D Shape Similarity," *Technical Report, CMU-CS-95-212*, Carnegie Mellon University, 1995.

[35] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons, New York, 1973.

[36] M. Sun, G. Burk and R.J. Sclabassi, "Measurement of Signal Similarity Using the Maxima of the Wavelet Transform," *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, vol. 3, pp. 583-586, 1993.

[37] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley Publishing, Redwood City, 1991.

[38] L.G. Sotelino, M. Saerens, and H. Bersini, "Classification of Temporal Trajectories by Continuous-Time Recurrent Nets," *Neural Networks*, vol. 7, no. 5, pp. 767-776, 1994.

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

[40] L.R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proc. of the IEEE*, vol. 77, no. 2, pp. 257-286, 1989.

[41] A. Kundu, G.C. Chen, and C.E. Persons, "Transient Sonar Signal Classification Using Hidden Markov Models and Neural Nets," *IEEE Journal of Oceanic Engineering*, vol. 19, no. 1, pp. 87-99, 1994.

[42] B. Hannaford and P. Lee, "Hidden Markov Model Analysis of Force/Torque Information in Telemanipulation," *Int. Journal of Robotics Research*, vol. 10, no. 5, pp. 528-539, 1991.

[43] J. Yang, Y. Xu, and C.S. Chen, "Hidden Markov Model Approach to Skill Learning and Its Application to Telerobotics," *IEEE Trans. Robotics and Automation*, vol. 10, no. 5, pp. 621-631, 1994.

[44] J. Yang, Y. Xu and C.S. Chen, "Gesture Interface: Modeling and Learning," *Proc. IEEE Int. Conf. on Robotics and Automation*, vol. 2, pp. 1747-52, 1994.

[45] L.R. Rabiner, et. al., "Some Properties of Continuous Hidden Markov Model Representations," *AT&T Technical Journal*, vol. 64, no. 6, pp. 1211-1222, 1986.

[46] M. Nechyba and Y. Xu, "Stochastic Similarity for Validating Human Control Strategy Models," *Technical Report, CMU-RI-TR-96-29*, Carnegie Mellon University, 1996.

[47] L.E. Baum, T. Petrie, G. Soules, and N. Weiss, "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains," *Annals of Mathematical Statistics*, vol. 41, no. 1, pp. 164-171, 1970.

[48] P. Baldi and Y. Chauvin, "Smooth On-Line Learning Algorithm for Hidden Markov Models," *Neural Computation*, vol. 6, no. 2, pp. 307-318, 1994.

[49] V. Krishnamurthy and J.B. Moore, "On-Line Estimation of Hidden Markov Model Parameters Based on the Kullback-Leibler Information Measure," *IEEE Trans. Signal Processing*, vol. 41, no. 8, pp. 2557-2573, 1993.

[50] D.F. Elliott and K.R. Rao, *Fast Transforms: Algorithms, Analyses, Applications*, Academic Press, New York, 1982.

[51] 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.

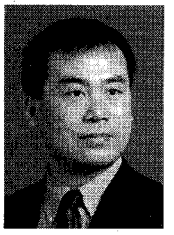
[52] H. Hatwal and E.C. Mikulcik, "Some Inverse Solutions to an Automobile Path-Tracking Problem with Input Control of Steering and Brakes," *Vehicle System Dynamics*, vol. 15, pp. 61-71, 1986.

[53] J.C. Spall, "Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation," *IEEE Trans. Automatic Control*, vol. 37, no. 3, pp. 332-341, 1992.

[54] D. Gopher, M. Weil, and T. Bareket, "The Transfer of Skill from a Computer Game Trainer to Actual Flight," *Proc. Human Factors Society 36th Annual Meeting*, vol. 2, pp. 1285-1290, 1992.

[55] W.L. Shebilske and J.W. Regian, "Video Games, Training, and Investigating Complex Skills," *Proc. Human Factors Society 36th Annual Meeting*, vol. 2, pp. 1296-1300, 1992.

Michael C. Nechyba was born in Vienna, Austria, in 1970. He received a B.S. degree in electrical engineering from the University of Florida in 1992. Since 1992, he has been working toward a Ph.D. degree in robotics at Carnegie Mellon University. His research interests include the learning and transfer of human control strategy, dynamically stable systems, neural networks, Hidden Markov Models, and applications in space robotics and teleoperation. Mr. Nechyba is an NSF Graduate Fellow, as well as a DOE Doctoral Fellow.



Yangsheng Xu worked in the GRASP lab at the University of Pennsylvania where he received his Ph.D. degree in 1989. Since then, he has been working at the Robotics Institute of Carnegie Mellon University as Research Scientist. At Carnegie Mellon, he has been leading research in the areas of space robotics, intelligent control, and manufacturing automation. With his colleagues, he developed several novel robot systems including the well-known space station walker, the Self-Mobile Space Manipulator (SM^2), and a ground-based space telerobotics testbed including zero-gravity compensation systems. His current interests include real-time modeling and transfer of human control strategies, dynamically stable robots, and design and control of high-performance mechatronics systems.