Proceedings of the 1999 IEEE
International Conference on Robotics & Automation
Detroit, Michigan • May 1999

Transfer of Human Control Strategy Based on Similarity Measure

Jingyan Song[1], Yangsheng Xu[2,3], Michael C. Nechyba[4], Yeung Yam[2]

[1]Department of Systems Engineering & Engineering Management
[2]Department of Mechanical and Automation Engineering
The Chinese University of Hong Kong, Shatin, NT, Hong Kong
[3]The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213 USA
[4]Department of Electrical and Computer Engineering
University of Florida, Gainesville, FL 32611 USA

Abstract

In this paper, we address the problem of transferring human control strategies (HCS) from an expert model to an apprentice model. The proposed algorithm allows us to develop useful apprentice models that nevertheless incorporate some of the robust aspects of the expert HCS models. We first describe our experimental platform – a real-time graphic driving simulator – for collecting and modeling human control strategies. Then, we discuss an adaptive neural network learning architecture for abstracting HCS models. Next, we define a hidden Markov model (HMM) based similarity measure which allows us to compare different human control strategies. This similarity measure is combined subsequently with simultaneously perturbed stochastic approximation to develop our proposed transfer learning algorithm. In this algorithm, an expert HCS model influences both the structure and the parametric representation of the eventual apprentice HCS model. Finally we describe some experimental results of the proposed algorithm.

1 Introduction

Transferring human control strategy (HCS) has potential application in a number of research areas ranging from virtual reality and robotics to intelligent highway systems. In previous work, transfer of human control strategies from human experts to human apprentices has been studied [1]. In this paper, we investigate the transfer of human control strategies, not between human expert and human apprentice, but rather between an expert HCS model and an apprentice HCS model.

In developing an algorithm to accomplish this transfer of control strategies, we rely on a number of related results in human control strategy research. First, we need to be able to successfully abstract a human control strategy to a reliable computational model. Since human control strategies are dynamic, nonlinear stochastic processes, however, developing good analytic HCS models tends to be difficult. Therefore, recent work in modeling HCS has focused on learning empirical models from real-time input-output human control data, through, for example, fuzzy logic [2, 3, 4, 5], and neural network techniques [6].

Second, we need to be able to evaluate our resulting models both with respect to how well they approximate the human control data, and how well they meet certain performance criteria. Since the HCS models are empirical, this is especially important, as few theoretical guarantees exist about their stability or performance. In [7], a stochastic similarity measure, based on hidden Markov model (HMM) analysis, was developed for validating the fidelity of HCS models to the source training data, while in [8, 9], several performance criteria were developed for evaluating the skill inherent in learned HCS models.

This previous work forms the basis for our transfer learning algorithm proposed herein. The algorithm requires that we first collect control data from experts and apprentices. From this data, we can then abstract two types of HCS models, one for the expert and one for the apprentice, using previously developed techniques. Our main goal in this paper is to improve the apprentice model's performance while still retaining important characteristics of the apprentice model. To do this, we look towards the stochastic similarity measure proposed in [7], which is capable of comparing long, multi-dimensional, stochastic trajectories, and has been applied previously towards validating models of human control strategy.

In our transfer learning algorithm, we propose to raise the similarity between an expert HCS model and an apprentice HCS model. Alternatively, we can think of the expert model guiding the actions of the apprentice model. The overall algorithm consists of two steps. In the first step, we let the expert model influence the eventual structure of the HCS model. Once an appro-

0-7803-5180-0-5/99 $10.00 © 1999 IEEE

priate model structure has been chosen, we then tune the parameters of the apprentice model through simultaneously perturbed stochastic approximation, an optimization algorithm that requires no analytic formulation, only two empirical measurements of a user-defined objective function.

In this paper, we first describe our experimental platform for collecting and modeling human control data – a real-time graphic driving simulator. We then describe an adaptive neural network learning architecture for abstracting HCS models given human control data. Next, we define the notion of similarity between control trajectories, and develop our transfer learning algorithm for expert and apprentice HCS models. Finally, we describe experimental results of the learning algorithm in the driving domain.

2 HCS modeling

For this work, we collect expert and apprentice driving data through a real-time graphic simulator, whose interface is shown in Figure 1. In the simulator, the

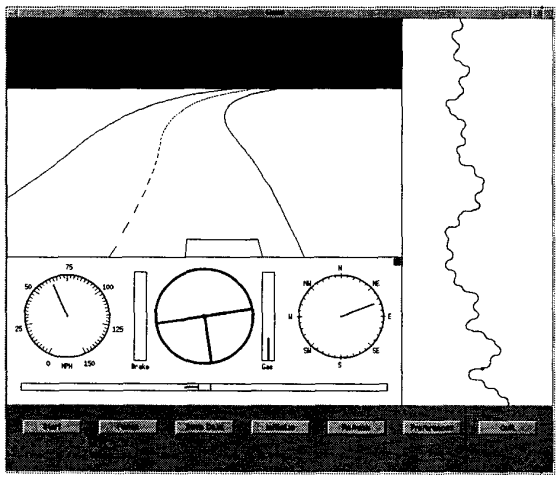

Figure 1: The driving simulator gives the user a perspective preview of the road ahead.

driving operator has independent control of the vehicle's steering as well as the brake and gas pedals. We ask different individuals to navigate across a number of different randomly generated roads, which consist of a sequence of (1) straight-line segments, (2) left turns, and (3) right turns. The map in Figure 1, for example, illustrates one randomly generated 20km road for which human (expert or apprentice) driving data was recorded. Each straight-line segment as well as the radius of curvature for each turn range in length between 100m and 200m. Nominally, the road is divided into two lanes, each of which has width $w = 5$m. The human operator's view of the road ahead is limited to 100m, and the entire simulator is run at 50Hz.

Using the collected data, we choose the flexible cascade neural network architecture with node-decoupled extended Kalman filtering (NDEKF) [10] for modeling the human driving data. We prefer this learning architecture over others for a number of reasons. First, no *a priori* model structure is assumed; the neural network automatically adds hidden units one at a time to an initially minimal network as the training requires. Second, hidden unit activation functions are not constrained to be a particular type. Rather, for each new hidden unit, the incremental learning algorithm can select that functional form which maximally reduces the residual error over the training data. Typical alternatives to the standard sigmoidal function are sine, cosine, and the Gaussian function. Finally, it has been shown that node-decoupled extended Kalman filtering, a quadratically convergent alternative to slower gradient descent training algorithms (such as backpropagation) fits well within the cascade learning framework and converges to good local minima with less computation [10].

The flexible functional form which cascade learning allows is ideal for abstracting human (expert or apprentice) control strategies, since we know very little about the underlying structure of each individual's internal controller. By making as few *a priori* assumptions as possible in modeling the human driving data, we improve the likelihood that the learning algorithm will converge to a good model of the human control data.

In order for the learning algorithm to properly model each individual's human control strategy, the model must 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 $\{\nu_\xi,\ \nu_\eta,\ \dot{\theta}\}$, (2) previous output (command) information $\{\delta,\ P_f\}$, and (3) a description of the road visible from the current car position. More precisely, the network inputs are,

$$\begin{aligned} &\{\nu_\xi(k-n_s),\cdots,\nu_\xi(k-1),\nu_\xi(k),\\ &\nu_\eta(k-n_s),\cdots,\nu_\eta(k-1),\nu_\eta(k),\\ &\dot{\theta}(k-n_s),\cdots,\dot{\theta}(k-1),\dot{\theta}(k)\} \end{aligned} \quad (1)$$

$$\begin{aligned} &\{\delta(k-n_c),\cdots,\delta(k-1),\delta(k),\\ &P_f(k-n_c),\cdots,P_f(k-1),P_f(k)\} \end{aligned} \quad (2)$$

$$\{x(1),x(2),\cdots,x(n_r),y(1),y(2),\cdots,y(n_r)\} \quad (3)$$

where $\nu_\xi(k)$ is the lateral velocity of the car at time k, $\nu_\eta(k)$ is the longitudinal velocity of the car at time k, $\dot{\theta}(k)$ is the angular velocity of the car at time k, $\delta(k)$ is the steering command at time k, $P_f(k)$ is the acceleration command at time k,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 n_i is given by,

$$n_i = 3n_s + 2n_c + 2n_r \tag{4}$$

The two outputs of the cascade network are $\{\delta(k+1), P_f(k+1)\}$. For the system as a whole, the cascade neural network can be viewed as a feedback controller, whose two outputs control the driving of the vehicle.

3 HCS similarity

In this section, we describe a similarity measure [7], which has previously been applied towards validating HCS models, and which we will use in a subsequent section for our HCS model comparisons. This similarity measure is based on hidden Markov models (HMMs), which 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 degree of sequential structure can be encoded by the hidden Markov models. As such, they have previously been applied in a number of different stochastic signal processing applications.

A discrete-output HMM is completely defined by the following triplet,

$$\lambda = \{A, B, \pi\} \tag{5}$$

where A is the probabilistic $n_s \times n_s$ state transition matrix, B is the $L \times n_s$ output probability matrix with L discrete output symbols $l \in \{1, 2, \cdots, L\}$, and π is the n-length initial state probability distribution vector for the HMM.

Below, we define a stochastic similarity measure, based on discrete-output HMMs. Assume that we wish to compare observation sequences from two stochastic processes Γ_1 and Γ_2. Let $\bar{O}_i = \{O_i^{(k)}\}, k \in \{1, 2, \cdots, n_i\}, i \in \{1, 2\}$, denote the set of n_i observation sequences of discrete symbols generated by process Γ_i. Each observation sequence is of length $T_i^{(k)}$, so that the total number of symbols in set $\bar{O}_i$ is given by,

$$\bar{T}_i = \sum_{k=1}^{n_i} T_i^{(k)}, i \in \{1, 2\}. \tag{6}$$

Also let $\lambda_j = \{A_j, B_j, \pi_j\}, j \in \{1, 2\}$, denote a discrete HMM locally optimized with the Baum-Welch algorithm to maximize,

$$P(\lambda_j | \bar{O}_j) = \prod_{k=1}^{n_i} P(\lambda_j | O_j^{(k)}), j \in \{1, 2\}, \tag{7}$$

and let,

$$P(\bar{O}_i | \lambda_j) = \prod_{k=1}^{n_i} P(O_i^{(k)|\lambda_j}) \tag{8}$$

$$P_{ij} = P(\bar{O}_i | \lambda_j)^{1/\bar{T}_i}, i, j \in \{1, 2\} \tag{9}$$

denote the probability of the observation sequences $\bar{O}_i$ given the model λ_j, normalized with respect to the sequence lengths $\bar{T}_i$. .

Using definition (9), we now define the following similarity measure between $\bar{O}_1$ and $\bar{O}_2$:

$$\sigma(\bar{O}_1, \bar{O}_2) = \sqrt{\frac{P_{21} P_{12}}{P_{11} P_{22}}} \tag{10}$$

In other words, we first use each set of observation sequences to train a corresponding HMM; this allows us to evaluate P_{11} and P_{22}. We then cross-evaluate each observation sequence on the other HMM to arrive at P_{12} and P_{21}. Finally we take the ratio of these probabilities and take the square root.

For any two sets of observation sequences, the similarity measure σ will range between 0 and 1. For very similar observation sequences, σ will be close to one, while for very dissimilar observation sequences, σ will be close to zero. Additional detailed discussion of the similarity measure's properties can be found in [7].

Finally, we note that in order to apply this similarity measure towards comparing human control strategies, we need to convert real-valued trajectories to a sequence of discrete symbols. To achieve this, we follow a two-step signal-to-symbol conversion process First, we window the data into frames and filter the data through spectral transforms such as the well-known short-time Fourier transform. Then, the resulting spectral coefficient vectors are vector-quantized into discrete symbols. For further details on this processing, please consult [7].

4 HCS transfer

In this section, we develop a learning algorithm for transferring skill from an expert HCS model to an apprentice HCS model. Rather than simply discard the apprentice HCS model, we attempt to preserve important aspects of the apprentice model, while at the same time improving the apprentice model's performance. In the proposed algorithm, the expert model serves as the guide or teacher to the apprentice model, and influences both the eventual structure and parametric representation of the apprentice model. As we will demonstrate shortly, the similarity measure defined in the previous section will play a crucial role in this transfer learning algorithm.

4.1 Structure learning

Recall that in cascade neural network learning, the structure of the neural network is adjusted during training, as hidden units are added one at a time until satisfactory error convergence is reached. Suppose that we train a HCS model from human control data provided by an expert using cascade learning. The final trained expert model will then consist of a given structure, which, in cascade learning, is completely defined by the number of hidden units in the final model.

Now, suppose that we have collected training data from an apprentice – that is, from an individual less skilled than the expert. What final structure should his learned model assume? One answer is that we let the model converge to the "best" structure as was the case for the expert model. Since, we already have an expert model at hand, however, we can let the expert model inform the choice of structure for the apprentice model.

Figure 2 illustrates our approach for structure learning in apprentice HCS models, as guided by an expert HCS model. We first train the expert HCS model in the usual fashion. Then, we train the apprentice HCS model, but impose an additional constraint during learning – namely, that the final structure (i.e. the number of hidden units) of the apprentice model and the expert model be the same.

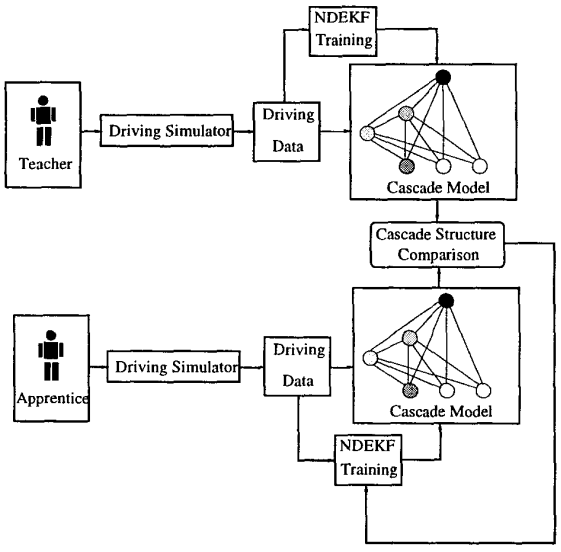

Figure 2: Structure learning in apprentice HCS model.

4.2 Parameter learning

Even though we impose the same structure on the expert and apprentice HCS models, they will clearly converge to different parametric representations. The expert model will be similar to the expert's control strategy, while the apprentice model will be similar to the apprentice's control strategy. We would now like to tune the apprentice model so as to retain part of the control strategy encoded within, while at the same time improving the performance in the apprentice model. Once again we will use the expert model as a guide in this learning, by examining the similarity (as defined previously) between the expert and apprentice models.

Let,

$$\omega = [w_1 \;\; w_2 \;\; \cdots \;\; w_n] \tag{11}$$

denote a vector consisting of all the weights in the apprentice HCS model $\Gamma(\omega)$. Also let $\sigma(O_e, O_p(\omega))$ denote the HMM similarity between the apprentice HCS model and expert HCS model. We would now like to determine a weight vector ω^* which raises the expert/apprentice similarity $\sigma(O_e, O_p(\omega^*))$, while at the same time retaining part of the apprentice control strategy.

Determining a suitable ω^* is difficult in principle because (1) we have no explicit gradient information

$$G(\omega) = \frac{\partial}{\partial \omega}\sigma(O_e, O_p(\omega)) \tag{12}$$

(2) each experimental measurement of $\sigma(O_e, O_p(\omega))$ requires a significant amount of computation. We lack explicit gradient information, since we can only evaluate the similarity measure empirically. Hence, gradient-based optimization techniques, such as steepest-descent and Newton-Raphson are not suitable. Furthermore, because each similarity evaluation is potentially computationally expensive, genetic optimization, which can require many iterations to converge, also does not offer a good alternative. Therefore we turn to *simultaneously perturbed stochastic approximation (SPSA)* to adjust ω.

Stochastic approximation (SA) is a well known iterative algorithm for finding roots of equations in the presence of noisy measurements. Simultaneously perturbed stochastic approximation (SPSA) [11] is a particular multivariate SA technique which requires as few as two measurements per iteration and shows fast convergence in practice. Hence, it is well suited for our application. Denote ω_k as our estimate of ω^* at the kth iteration of the SA algorithm, and let ω_k be defined by the following recursive relationship:

$$\omega_{k+1} = \omega_k - \alpha_k \bar{G}_k \tag{13}$$

where $\bar{G}_k$ is the simultaneously perturbed gradient approximation at the kth iteration,

$$\bar{G}_k \;=\; \frac{1}{q}\sum_{i=1}^{q} G_k^i \approx \frac{\partial}{\partial \omega}\bar{\sigma}(O_e, O_p(\omega)) \tag{14}$$

$$G_k^i = \frac{\bar{\sigma}_k^{(+)} - \bar{\sigma}_k^{(-)}}{2c_k} \begin{bmatrix} 1/\Delta_{kw_1} \\ 1/\Delta_{kw_2} \\ \cdots \\ 1/\Delta_{kw_n} \end{bmatrix} \quad (15)$$

Equation (14) averages q stochastic two-point measurements G_k^i for a better overall gradient approximation, where,

$$\bar{\sigma}_k^{(+)} = \sigma(O_e, O_p(\omega_k + c_k\Delta_k)) \quad (16)$$

$$\bar{\sigma}_k^{(-)} = \sigma(O_e, O_p(\omega_k - c_k\Delta_k)) \quad (17)$$

$$\Delta_k = [\Delta_{kw_1}\Delta_{kw_2}\cdots\Delta_{kw_n}]^T \quad (18)$$

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 $\{\alpha_k\}$, $\{c_k\}$ are positive scalar sequences satisfying the following properties:

$$\alpha_k \to 0, \quad c_k \to 0 \; as \; k \to \infty, \quad (19)$$

$$\sum_{k=0}^{\infty} \alpha_k = \infty, \quad \sum_{k=0}^{\infty} (\frac{\alpha_k}{c_k})^2 < \infty \quad (20)$$

For our problem, we define the objective function to be $\bar{\sigma} = 1 - \sigma(O_e, O_p(\omega))$. The weight vector ω_0 is of course the weight representation in the initially stable apprentice model. Finally, we note that while larger values of q in equation (14) will give more accurate approximations of the gradient, in practice, two measurements ($q = 1$) per iteration is often sufficient. Figure 3 illustrates the overall parameter tuning algorithm.

5 Experiment

Here, we test the transfer learning algorithm on control data collected from three individuals, Tom, Dick and Harry. In previous work [8], we have observed that Harry's driving model performs better with respect to certain important performance measures. Therefore, we view Harry as the expert, and Dick and Tom as the apprentices. Furthermore, in order to simplify the problem somewhat, we keep the applied force constant at $P_f = 300N$. In other words, we ask each driver to control the steering δ only.

For this experiment, we first train the expert model on Harry's control data. The final trained model consists of two hidden units with $n_s = n_c = 3$, and $n_r = 15$; because we are keeping P_f constant, the total number of inputs for the cascade network model therefore is $n_i = 42$. Keeping in mind that we want the final structure of the apprentice models to be the same as the expert model's structure, we also train Dick's and Tom's models to two hidden units each, with the same number of inputs ($n_i = 42$).

We now would like to improve the performance of the apprentice models, while still retaining some

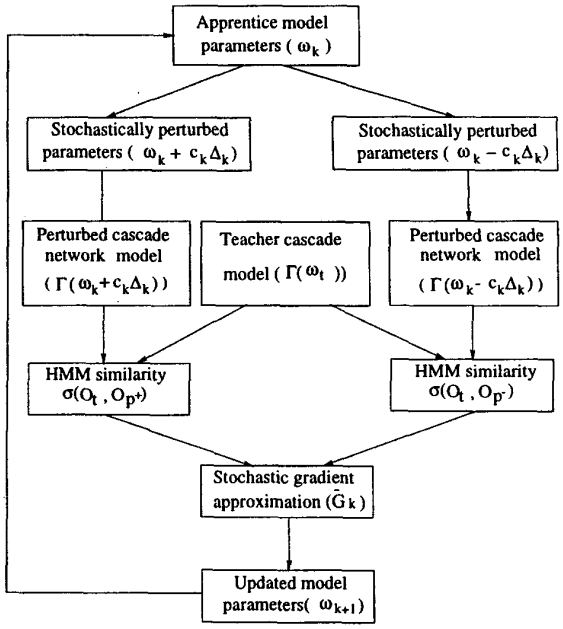

Figure 3: Stochastic transfer learning algorithm for HCS models.

aspects of the apprentice control strategies. In other words, we would like to improve the similarity $\sigma(O_e, O_p)$ defined in equation (10) between Harry's expert model and each of the apprentice models using the SPSA algorithm as discussed in the previous section.

In the SPSA algorithm, we empirically determine the following values for the scaling sequences $\{\alpha_k\}$, $\{c_k\}$:

$$\alpha_k = 0.00001/k^2, \quad k > 0 \quad (21)$$

$$c_k = 0.001/k^{1.25}, \quad k > 0 \quad (22)$$

Furthermore, we set the number of measurements per gradient estimation in equation (14) to $q = 1$. Finally, we denote $\sigma(O_e, O_p^k)$ as the similarity $\sigma(O_e, O_p)$ after iteration k of the learning algorithm; hence, $\sigma(O_e, O_p^0)$ denotes the similarity prior to any weight adjustment in the apprentice models.

Figure 4 plots the similarity measure between Tom's and Dick's apprentice models and Harry's expert HCS model, respectively, as a function of iteration k in the transfer learning algorithm. We observe that for Dick's model, the similarity to Harry's model improves from $\sigma(O_e, O_p^0) = 41.5\%$ to $\sigma(O_e, O_p^{15}) = 78.5\%$. Although for Tom's model the change is less dramatic, his model's similarity nevertheless rises from $\sigma(O_e, O_p^0) = 55.0\%$ to $\sigma(O_e, O_p^{15}) = 72.2\%$. Thus, the transfer learning algorithm improves the similarity of Dick's model by approximately 37% and Tom's model by about 17.2% over their respective initial models.

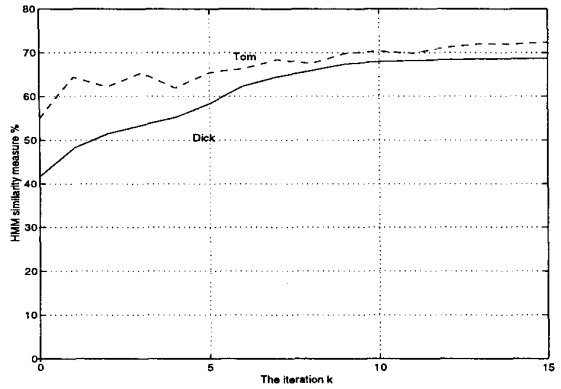

Figure 4: Similarity between Harry's and Dick's (solid) and Harry's and Tom's (dashed) models during transfer learning.

Since the similarity in control strategies improves, we would expect that apprentice performance improves as well. In order to test this, we examine model performance as measured by the obstacle avoidance performance criterion J defined previously in [8]. Let J_e denote the performance criterion value for Harry's expert model. Also, let J_p^0 denote apprentice performance before learning, and let J_p^{15} denote apprentice performance after transfer learning. We note that the performance criterion is defined so that smaller values indicate better obstacle avoidance performance.

Table 1 lists these performance values for Dick and Tom.

	J_p^0	J_p^{15}
Harry	0.51	0.51
Tom	1.23	0.87
Dick	1.37	1.13

Table 1: Obstacle avoidance performance measure

From Table 1, we note that Harry's performance does not change, of course, since we keep his model fixed. The models for Dick and Tom do, however, improve with respect to the obstacle avoidance performance measure. The adjusted apprentice models can be viewed as hybrid models, which combine the apprentice control strategy with some of the improved techniques of the expert model.

6 Conclusion

In this paper, we have proposed an iterative learning algorithm, based on simultaneously perturbed stochastic approximation (SPSA), for improving the similarity between apprentice and expert models of human control strategy. The transfer algorithm consists of two steps. In the first step, we ensure that the apprentice model reaches the same overall structure as the expert model. In the second step, we tune the parameters (i.e. weights) in the apprentice model to improve its similarity with the expert model. The resulting model will preserve some aspects of the original apprentice model while at the same time improving performance. The proposed algorithm requires no analytic formulation, only two experimental similarity measurements per iteration. In initial experiments, we have demonstrated that apprentice performance improves with the help of the expert model.

Acknowledgments

This work is supported in part by Hong Kong Research Council Grant No. CUHK4138/97E.

References

[1] M. C. Nechyba and Y. Xu, "Human Skill Transfer: Neural Networks as Learners and Teachers," *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, vol. 3, pp. 314-9, 1995.

[2] M. Sugeno and T. Yasukawa, "A Fuzzy-Logic-Based Approach to Qualitative Modeling," *IEEE Trans. on Fuzzy Systems*, vol. 1, no. 1, pp. 7-31, 1993.

[3] C.C. Lee, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller–Part I," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 20, no. 2, pp. 404-18, 1990.

[4] C.C. Lee, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller–Part II," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 20, no. 2, pp. 419-35, 1990.

[5] U. Kramer, "On the Application of Fuzzy Sets to the Analysis of the System Driver-Vehicle-Environment," *Automatica*, vol. 21, no. 1, pp. 101-7, 1985.

[6] 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.

[7] M. C. Nechyba and Y. Xu, "Stochastic Similarity for Validating Human Control Strategy Models," *IEEE Trans. on Robotics and Automation*, vol. 14, no. 3, pp. 437-51, 1998.

[8] J. Song, Y. Xu, M. C. Nechyba and Y. Yam, "Two Measures for Evaluating Human Control Strategy," *Proc. IEEE Conf. on Robotics and Automation*, vol. 3. pp. 2250-5, 1998.

[9] J. Song, Y. Xu, Y. Yam and M. C. Nechyba, "Optimization of Human Control Strategies with Simultaneously Perturbed Stochastic Approximation", *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and System, October, 1998.*

[10] 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.

[11] J. C. Spall, "Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation," *IEEE Trans. on Automatic Control*, vol. 37, no. 3, pp. 332-41, 1992.