Stabilizing Human Control Strategies through Reinforcement Learning

Michael C. Nechyba
nechyba@mil.ufl.edu

J. Andrew Bagnell
dbagnell@mil.ufl.edu

Machine Intelligence Laboratory
Electrical and Computer Engineering
University of Florida

Abstract

Humans are, and for the foreseeable future remain our best and only example of true intelligence. In comparison, even advanced robots are still embarrassingly stupid. Consequently, one popular approach for imparting intelligent behaviors to robots and other machines abstracts models of human control strategy (HCS), learned directly from human control data. This type of approach can be broadly classified as "learning through observation." A competing approach, which builds up complex behaviors through exploration and optimization over time, is reinforcement learning. We seek to unite these two approaches and show that each approach, in fact, complements the other. Specifically, we propose a new algorithm, rooted in reinforcement learning, for stabilizing learned models of human control strategy. In this paper, we first describe the real-time driving simulator which we have developed for investigating human control strategies. Next, we motivate and describe our framework for modeling human control strategies. We then illustrate how the resulting HCS models can be stabilized through reinforcement learning and finally report some positive experimental results.

1. Introduction

Models of human skill, which accurately emulate dynamic human behavior, have far reaching potential in areas ranging from robotics to virtual reality to the intelligent vehicle highway system. Thus, a number of different researchers have endeavored in recent years to abstract models of human skill directly from observed human input-output data (see [1] for an overview of the literature). Unfortunately, capturing intelligent behaviors through human modeling suffers from some potential weaknesses. Because human control strategies are dynamic, nonlinear, stochastic processes, *analytic* models of human actions tends to be quite difficult, if not impossible, to abstract. Therefore, HCS models are usually derived *empirically*, rather than analytically from real-time human input-output data. As such, traditional performance or stability guarantees, like those in linear control for example, are typically not available.

In previous work, we have sought to address this issue through task-specific performance measures and post-training performance optimization [2]. Here, however, we propose a new algorithm for improving the performance of learned HCS models through *reinforcement learning*. Reinforcement learning denotes a class of adaptive techniques that seek to learn to predict and control the behavior of an autonomous agent through that agent's interaction with his/her environment. Modern reinforcement learning borrows heavily from the fields of operations research and optimal control; in particular, the agent's environment is approximated as a Markov Decision Process (MDP) [3], so that the agent is asked to maximize rewards emitted by the MDP.

Reinforcement learning too, however, suffers from some significant weaknesses. First, reinforcement learning techniques often do not scale well to problems with high-dimensional input spaces. Furthermore, much of the literature in reinforcement learning deals only with problems where the agent has perfect knowledge about the state of his/her environment. This condition is rarely met in real life (e.g. noisy sensors, finite precision, etc.) and techniques tailored for a perfect-knowledge environment can degenerate to give arbitrarily poor results when uncertainty about the agent's state exists [4].

In this paper, we propose to combine reinforcement learning with the modeling of human control strategies in order to address the weaknesses inherent in each approach by itself. The HCS model aids reinforcement learning by intelligently partitioning a high-dimensional input space into regions that are meaningfully different to the reinforcement learner. Even more significantly, the HCS model can serve to "jump-start" the policy of the reinforcement learner. At the same time, reinforcement learning can take an initially imperfect HCS model and — as we will show — improve its performance substantially.

2. Real-time driving simulator

Driving is a prototypical example of human control strategy that offers a rich environment for studying HCS modeling. The task is inherently multi-input, multi-output (MIMO), and includes control outputs which vary both continuously and discontinuously with sensor inputs. Below we describe a driving simulator that we have developed for investigating human control strategies. We choose virtual driving over real driving for a number of reasons: (1) it is safer for the human operator, (2) it allows us better control of our experimental environment, and (3) it allows us to vary the control difficulty of the task without fear of accident or injury.

Figure 1 shows the real-time graphic driving simulator which we have developed as an experimental platform. In the simulator, the human operator has independent control over the steering of the car, the brake and the accelerator, although the simulator does not allow both the gas and brake pedals to be pushed at the same time. The state of the car is described by $\{v_\xi, v_\eta, \omega\}$ [1], where v_ξ is the lateral velocity of the car, v_η is the longitudinal velocity of the car and ω is the angular velocity of the car; the controls are given by $\{\alpha, \delta\}$, where α is the user-applied longitudinal force on the front tires and δ is the user-applied steering angle.

Because of input device constraints, the force (or acceleration) control α is limited during each 1/50 second time step, based on its present value. If the gas pedal is currently being applied ($\alpha > 0$), then the operator can either increase or decrease the

amount of applied force by a constant $\Delta\alpha_g$ or switch to braking. Similarly, if the brake pedal is currently being applied ($\alpha < 0$) the operator can either increase or decrease the applied force by a second constant $\Delta\alpha_b$ or switch to applying positive force. Thus, the $\Delta\alpha_g$ and $\Delta\alpha_b$ constants define the responsiveness of each pedal.

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

3. Human control strategy modeling

In modeling human control strategies, we want to map sensory inputs to control action outputs. For our case, the sensory inputs to the model are a vector ζ of (1) current and time-delayed states $\{v_\xi, v_\eta, \omega\}$, (2) previous control outputs $\{\delta, \alpha\}$, and (3) a description of the road visible from the current car position. The control action outputs of the model are the steering and acceleration commands at the next time step (see [1,5] for details). Viewed as a mapping from inputs to outputs, note that the two controls — steering and acceleration — are fundamentally quite different. For given human driving data, steering will tend to vary *continuously* with sensory inputs, while acceleration will tend to vary *discontinuously* with sensory inputs. This is *not* merely an artifact of the input device constraints, but is caused primarily by the necessary switching between the brake and gas pedals, as is also the case for real driving.

As was demonstrated in [5], continuous learning architectures — whether they be fuzzy logic-based, neural network-based, or memory-based — cannot faithfully reproduce control strategies where discrete events or decisions introduce discontinuities in the input-output mapping. Therefore, we have developed a hybrid continuous/discontinuous modeling framework for handling the two different control types [1,5]. The resulting architecture is illustrated in Figure 2. Note that the continuous steering control is modeled by a *cascade neural network*, a powerful continuous nonlinear function approximator, which makes few *a priori* assumptions about the underlying structure of the human controller [1, 6]. The discontinuous part of the overall model, which is the primary focus of this paper, is described in somewhat greater detail below.

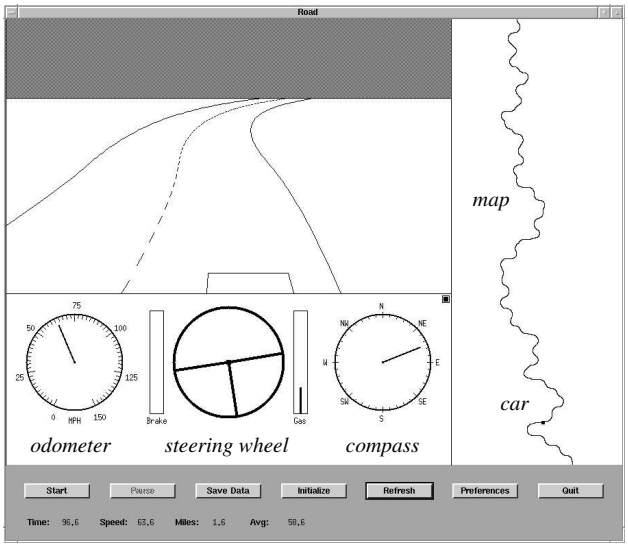

Fig. 1: The experimental driving simulator.

3.1 Discontinuous control

We view the discontinuous acceleration control not as a deterministic functional mapping (as we did the continuous steering control), but rather as a probabilistic relationship between sensory inputs and discontinuous outputs. For each possible control action A_i, we train a corresponding statistical model λ_i to maximize,

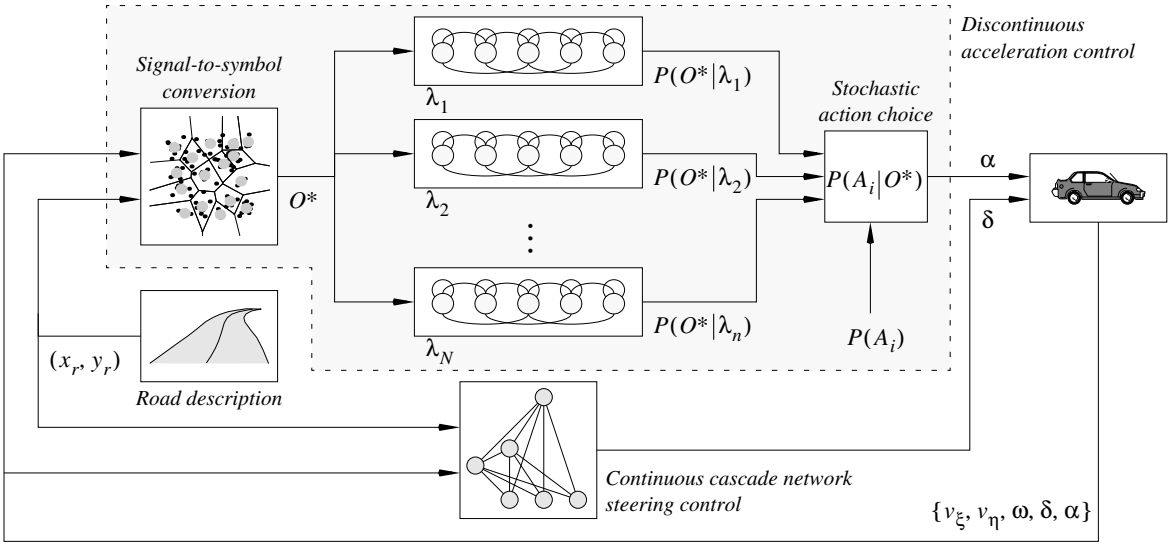

Fig. 2: Overall control structure. Steering is controlled by a cascade neural network, while the discontinuous acceleration command is controlled by the HMM-based controller (shaded box).

$$\prod_{j=1}^{n_i} p(\zeta_i^j | \lambda_i) \,,\; i \in \{1, \ldots, N\} \,. \tag{1}$$

where $\{\zeta_i^j\}$ denotes the set of sensory input vectors in the human control data that led the human to execute action A_i, and $p(\zeta_i^j | \lambda_i)$ denotes the likelihood of ζ_i^j given the model λ_i.

After training, given an unknown input vector ζ^*, we now choose action A^* stochastically so that,

$$A^* = A_i \text{ with probability } P(A_i | \zeta^*) \,, \tag{2}$$

where,

$$P(A_i | \zeta^*) = p(\zeta^* | A_i) P(A_i) / p(\zeta^*) \text{ (Bayes Rule)}, \tag{3}$$

$p(\zeta^* | A_i) \equiv p(\zeta^* | \lambda_i)$, $p(\zeta^*)$ is a normalization factor, and $P(A_i)$ represents the *prior* probability of selecting action A_i.

As with the continuous steering control, the statistical models λ_i should not be needlessly complicated and should make as few *a priori* assumptions about the underlying distribution of the human control data as possible. Moreover, computing the maximum-likelihood estimate of the parameters in each model (i.e. maximizing (1) above) should be relatively easy. Hidden Markov Models (HMMs) [7], which are powerful, trainable statistical models that have previously been applied in a number of stochastic signal processing applications, appear to meet the above criteria, since well-known algorithms exist for training them on *arbitrary* statistical distributions.

The simplest type of HMM consists of a single state, with a discrete-output probability distribution. In order to train these HMMs for our problem, we must first convert the input vectors ζ to discrete symbols O. To minimize the resulting distortion and loss of information, we choose the well-known LBG vector-quantization algorithm [8], which iteratively generates vector codebooks of size $L = 2^m$, $m \in \{0, 1, \ldots\}$, and can be stopped at an appropriate level of discretization, as determined by the amount of available human control data.

Each statistical model λ_i therefore reduces to a probability vector $\mathbf{b}_i$, whose j th element $\mathbf{b}_i(j)$ is equivalent to the relative frequency of input symbol j leading to control action A_i. Consequently,

$$P(A_i | \zeta^*) \propto P(O^* = j | \lambda_i) P(A_i) = \mathbf{b}_i(j) P(A_i) \tag{4}$$

so that equation (2) reduces to a learned stochastic policy,

$$\pi(o, a) = P(a = A_i | o = j) \,,\; \forall i, j \,, \tag{5}$$

for each possible input observable j and action A_i.

It should be noted that the discretized input observables implicitly encode time-dependent information since the input vectors ζ, corresponding to the discrete observables, contain histories of both the state and previous outputs. A more detailed discussion of the trade-offs in modeling complexity and accuracy between single-state and multi-state HMMs, omitted here for space reasons, can be found in [1].

Finally, to make the definition of $\pi(o, a)$ complete, we note that the priors $P(A_i)$ are easily estimated as the relative frequency of occurrence of each action in the human control data. In our driving example, there are a total of eight possible actions, as outlined in Section 2.

3.2 Experiment

We ask Larry to drive over two different randomly generated 20km roads ρ_1 and ρ_2, where each run lasts approximately 10 minutes. A part of Larry's second run, for example, is shown in Figure 3(a) below. Larry's driving behavior is representative of typical human driving in the simulator, in that (1) the steering control is reasonably continuous; (2) the acceleration control has significant discontinuities due to rapid switching between the brake and gas pedals; and (3) Larry manages to stay on the road ($\pm$5m deviation from the road median) for most of the run, with only a few brief off-road episodes in especially tight turns.

Now, we use Larry's first run (ρ_1) to train a hybrid continuous/discontinuous HCS model (Figure 2), and reserve the run on road ρ_2 for testing the learned model. We include six time-delayed values of the state and previous control outputs in the input-space representation and quantize the resulting input vectors $\{\zeta\}$ to $L = 512$ observables. Figure 3(b) plots a typical control trajectory over road ρ_2 for the resulting HCS model. In order to benchmark the performance of the hybrid discontinuous/continuous HCS model, we also train a second HCS model which maps both the steering δ *and* the acceleration α with continuous cascade neural networks, using the same input representation. Figure 3(c) plots part of the control trajectory over road ρ_2 for this strictly continuous HCS model. Table 1 compares some aggregate statistics for Larry's original data and the two different HCS models.

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

If we judge the continuous model on how faithfully it reproduces Larry's acceleration control strategy, however, it rates significantly worse; that is, the model's acceleration control looks nothing like Larry's. It was shown in [5] that Larry's acceleration control is not easily expressed in a continuous functional form, such as a neural network, since the switching discontinuities in the acceleration control essentially require very similar input vectors to be mapped to radically different output vectors.

Table 1: Statistical comparison

Road ρ_2	*Larry*	*hybrid model*	*continuous model*
v (mph)	71.9 ± 9.0	70.7 ± 8.1	73.6 ± 2.5
d (m)	-0.72 ± 1.46	-1.38 ± 1.70	-1.00 ± 0.60
δ (rad)	± 0.094	± 0.081	± 0.068
α (N)	2240 ± 2620	1970 ± 2340	1780 ± 760

The hybrid continuous/discontinuous controller [Figure 3(b)] appears to do a much better job in modeling Larry's driving control strategy. In fact, we can quantify the degree of similarity between Larry's control strategy and each of the two models, by computing a stochastic similarity measure σ which we have developed previously [9] for comparing human control strategies. The similarity measure is capable of comparing stochastic, multidimensional trajectories and yields a value between 0 and 1, with larger values indicating greater similarity. For the similarity comparison here, we include all relevant state and control variables $\{v_\xi, v_\eta, \omega, \delta, \alpha\}$, and arrive at the following similarity values:

$$\sigma(\text{Larry, hybrid model}) = 0.57 \quad (6)$$

$$\sigma(\text{Larry, continuous model}) = 0.08 \quad (7)$$

Hence, the hybrid controller is significantly more faithful to Larry's control strategy than the strictly continuous controller.

Unfortunately though, the hybrid controller also tends to be significantly less stable than its continuous counterpart. Note, for example, from Figure 3(c) that the hybrid controller veers off the road at $t = 20$ sec. To understand why this may be happening, consider, for example, Figure 4, where we plot a small part of Larry's first run. We observe that Larry's trajectory takes him close to the edge of the road; what keeps him from driving off the road is the switch from the gas to the brake at time t_s. Now, because the action selection criterion in equation (2) is stochastic, it is possible that the stochastic controller will only brake at time $t_s + \tau$, even if the time t_s is modeled as the most likely time for a control switch. Braking at time $t_s + \tau$, however, may be too late for the car to stay in contact with the road.

4. Reinforcement learning stabilization

In previous work [1], we attempted to address the stability problem of the hybrid HCS models by increasing the prior probability of switching from accelerating to braking by some small ε_s. The problem with this *ad hoc* modification is two-fold. First, specific values of ε_s need to be experimentally determined for every new HCS model, and second, we cannot be sure that this adjustment addresses all sources of instability. Therefore, we look towards reinforcement learning for a more principled and less *ad hoc* approach to improving model stability.

4.1 Introduction

Our overall approach for stability improvement proposed below seeks to modify the initial hybrid HCS model through reinforcement learning. We look towards the theory of Partially Observable Markov Decision Processes (POMDP) (see [10] for an excellent discussion) as an appropriate modeling framework for our driving domain. In other words, we propose that in the driving domain there exist meaningful underlying states that are sufficient statistics to determine the next state of the driving agent, and therefore to make optimal decisions, but that the agent is limited to observing a small number of messages from the environment that encode and compress this information.

In order to apply the theory of POMDPs to our problem, we choose the following simple assignment of rewards R. All states that are off the road are assigned a reward of -1, a form of pun-

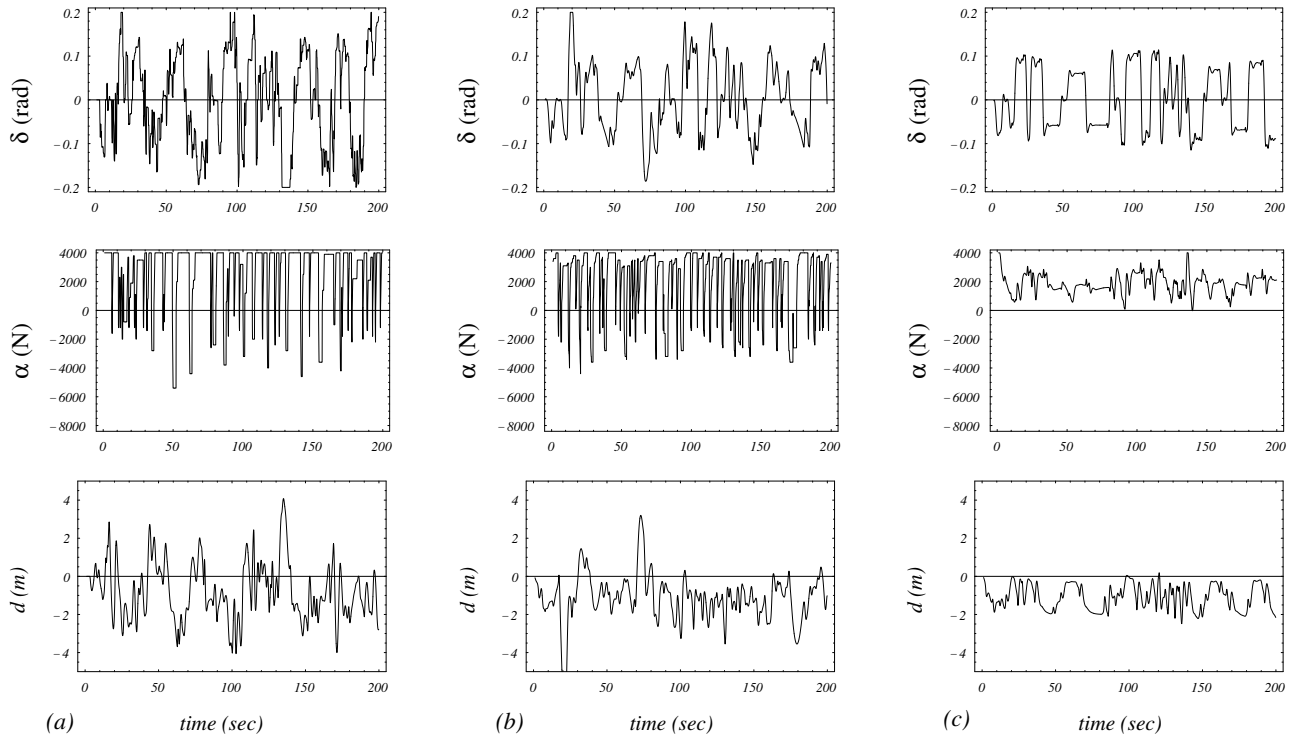

Fig. 3: (a) Part of Larry's steering, acceleration and lateral offset trajectories over time; (b) part of the hybrid continuous/ discontinuous HCS model's steering, acceleration and lateral offset trajectories; and (c) part of the purely continuous HCS model's steering, acceleration and lateral offset trajectories.

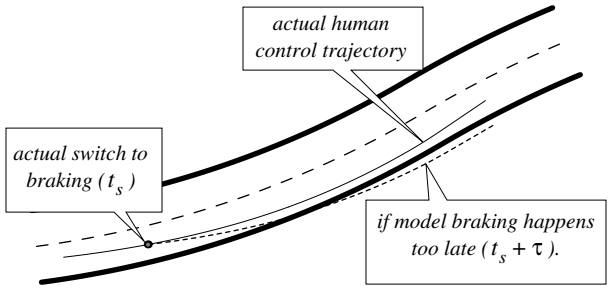

Fig. 4: Instability can result if the hybrid controller switches to braking too late.

ishment. Further, if the driving agent loses sight of the road (deemed a "catastrophic failure"), we reset the agent to its starting position after applying the -1 penalty for the previous 50 steps. This extended penalty prevents the agent from learning that if it veers off the road, it should aim to reach a catastrophic state so as to quickly return to the good initial position.

4.2 RL algorithm for stabilization

Well-studied algorithms in reinforcement learning (RL), like *Q*-learning [3], suffer from significant weaknesses in our current setting. First, little work has been done with stochastic policies in reinforcement learning. Second, traditional RL techniques are not capable of maintaining fidelity to prior human control data, since they typically modify a policy at every step of policy evaluation. Third, many of the current techniques in reinforcement learning have problems dealing with partial observability. Defining an optimal policy can be difficult, as an agent cannot, in general, maximize the value of all observations simultaneously. Moreover, the one-state updates that most algorithms employ do not accurately estimate the value of each observation [4]. Finally, in this domain, there are often many steps between punishments.

To deal with these difficulties, we propose an algorithm very similar to the one described in [11]. Let (o, a) denote an observation/action pair; let $\pi(o, a)$ denote the current stochastic policy for all (o, a); let $h(o, a)$ denote the current relative value for all (o, a); let $e(o, a)$ denote the current eligibility for all (o, a); and let $v(o, a)$ denote the current number of times that the pair (o, a) has been visited. Also, let $\gamma \in [0, 1)$ denote the temporal-difference learning discount factor; let tot denote the total number of actions performed; and let ρ denote the current average reward. Then the algorithm proceeds as follows:

---

1. **Initialization**:

   At the start of the algorithm, we initialize the following values:

   $$h(o, a) = 0,\ e(o, a) = 0,\ v(o, a) = 0,\ \forall o, a \tag{8}$$

   $$\rho = 0,\ tot = 0 \tag{9}$$

   $$\pi(o, a) = \text{learned HCS model [equation (5)]} \tag{10}$$

   $$o = \text{initial observation} \tag{11}$$

2. **Policy evaluation**:

   In evaluating the current policy, we iterate the steps below for max steps. First, we choose an action a stochastically according to the current policy $\pi(o, a)$ and the current observation o, and then execute a. Consequently $v(o, a)$ and tot are incremented,

   $$v(o, a) = v(o, a) + 1, \tag{12}$$

   $$tot = tot + 1 \tag{13}$$

   Defining,

   $$r = \text{reward after execution of } a, \tag{14}$$

   $$o' = o \text{ (previous observation), and} \tag{15}$$

   $$o = \text{current observation after execution of } a, \tag{16}$$

   we compute the differential reward Δ for action a versus the reward we would otherwise have predicted,

   $$\Delta = r - \rho + h(o) - h(o', a), \tag{17}$$

   where $h(o)$ is the value of the observation (state) o defined as,

   $$h(o) \equiv \sum_b h(o, b)\pi(o, b) \tag{18}$$

   We then update the value function $h(o, a)$,

   $$h(o', a) = h(o', a) + \Delta / v(o', a), \tag{19}$$

   $$h(o', a') = h(o', a') + \frac{\Delta \cdot e(o', a')}{v(o', a)},$$

   $$\forall (o', a') \neq (o', a), \tag{20}$$

   and the eligibility function $e(o, a)$,

   $$e(o', a) = 1, \tag{21}$$

   $$e(o', a') = \gamma \cdot e(o', a'),\ \forall (o', a') \neq (o', a). \tag{22}$$

   Finally, we update the average reward ρ,

   $$\rho = \rho + \Delta / tot \tag{23}$$

3. **Policy improvement**:

   We are now interested in updating the policy $\pi(o, a)$ for those observations (states) o' with low values $h(o')$ (i.e. poor stability). Hence, if,

   $$h(o') < 0, \tag{24}$$

   and,

   $$\max_{a'} [h(o', a')] > h(o') + \Delta_{min}, \tag{25}$$

   then we modify our current policy by,

   $$\pi(o', a') = (1 - \varepsilon)\pi(o', a') + \varepsilon\pi' \tag{26}$$

   for some constant Δ_{min} and,

   $$\pi' = \begin{cases} 1 & \operatorname{argmax}_{a'} [h(o', a')] \\ 0 & \text{otherwise} \end{cases} \tag{27}$$

   In other words, the policy for states with poor stability is modified if the value of the best action for those states is significantly larger than the value of those states themselves [equation (25)].

Our algorithm applies the Policy Improvement Theorem in [11], which states that perturbing our policy toward actions that maximize $h(o, a)$ will improve the average reward as long as the perturbations ε are small. Thus, we carefully choose to perturb

the policy only for those observations o' that stand to have a significant improvement, as indicated by equation (25).

4.3 Experiment

We now apply our algorithm to the hybrid HCS model learned from Larry's control data. The stabilization algorithm is executed on roads that are statistically similar to road ρ_1 with,

$$\varepsilon = 0.05\,,\ \gamma = 0.96\,,\ max = 20,000\,,\ \Delta_{min} = 0.12 \quad (28)$$

The resulting modified model exhibits a dramatic improvement in stability. Where before Larry's model rarely could travel for more than 5km without a catastrophic failure, the modified HCS model completes100km courses without any catastrophic failures. Furthermore, it does so with little drop in average speed and fidelity, as illustrated in Table 2 below.

Table 2: Comparisons

Data set	v_{avg} (mph)	% off-road	similarity σ
Larry	73.1	1.63	0.75[a]
Original model	72.4	**9.02**	0.57
Perturbed model	70.1	**1.23**	0.45

a. Self-similarity between Larry's two runs.

4.4 Discussion

Table 2 clearly demonstrates that the RL optimization of the initial HCS model improves the model's stability. It was not possible to achieve similar such improvements using *ad hoc* stabilization procedures. Let us illustrate schematically why this might be the case. In Figure 5, the policy $\pi(o, a)$ is drawn as a grid, where each box in the grid represents one observation/action pair. (For readability, we drew a grid with only 20 possible observations o.) Boxes in the grid that are shaded are modified after training the HCS model, while boxes that are white represent unmodified states. As part (a) of Figure 5 indicates, an *ad hoc* stabilization procedure modifies a set number of (o, a) pairs by a fixed amount. In the RL stabilization procedure [Figure 5(b)], however, any (o, a) pair with initial probability greater than zero can potentially be modified by some not predetermined amount. It therefore has significantly greater flexibility in stabilizing the initial HCS model, while still retaining its fidelity to the human control data. Moreover, the RL stabilization is based on active exploration and consequent optimization of the actual system.

5. Conclusion

We believe that this work opens up a promising direction for future research by combining learning through observation (from humans) with subsequent reinforcement-learning-based optimization. Little work has been done previously to incorporate prior knowledge into reinforcement learning, and this paper offers one potentially successful approach for tackling this problem. Currently, we are exploring the modeling framework of this paper with more varied data (from different humans), varied learning parameters, algorithmic variations and applications in other learning domains.

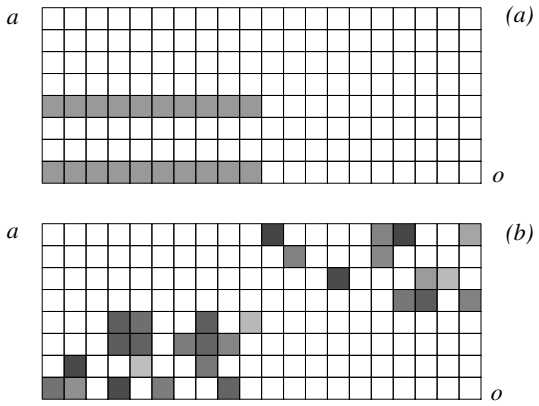

Fig. 5: Visualizing (a) *ad hoc vs.* (b) RL stabilization.

References

[1] M. C. Nechyba, *Learning and validation of human control strategies*, Ph.D. thesis, Carnegie Mellon University, 1998 (http://www.mil.ufl.edu/people/nechyba/phd.html).

[2] J. Song, Y. Xu, Y. Yam and M. C. Nechyba, "Optimization of Human Control Strategies with Simultaneously Perturbed Stochastic Approximation," *Proc. IEEE Int. Conference on Intelligent Robots and Systems*, vol. 2, pp. 983-8, 1998.

[3] L. P. Kaelbling, M. L. Littman and A. W. Moore, "Reinforcement Learning: A Survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237-85, 1996.

[4] S. P. Singh, T. Jaakkola and M. Jordan, "Learning Without State Estimation in Partially Observable Markovian Decision Processes," *Proc. Eleventh Int. Conf. on Machine Learning*, 1994.

[5] M. C. Nechyba and Y. Xu, "On Discontinuous Human Control Strategies," *Proc. IEEE Int. Conference on Robotics and Automation*, vol. 3, pp. 2237-43, 1998.

[6] 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.

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

[8] 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.

[9] 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.

[10] L. P. Kaelbling, M. L. Littman and A. R. Cassandra, "Planning and Acting in Partially Observable Stochastic Domains," *Artificial Intelligence*, vol. 101, no. 1-2, pp. 99-134, 1998.

[11] T. Jaakkola, S. P. Singh and M. I. Jordan, "Monte-Carlo Reinforcement Learning in Non-Markovian Decision Problems," *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. S. Touretzky and T. K. Lee, eds., MIT Press, Cambridge, 1995.