JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 8, March 2011

Supervised vs Unsupervised Learning for Operator State Modeling in Unmanned Vehicle Settings

Yves Boussemart[*] , Mary L. Cummings[†] , Jonathan Las Fargeas[‡] , and Nicholas Roy[§]
Massachusetts Institute of Technology, Cambridge, Massachusetts 02139

DOI: 10.2514/1.46767

In this paper, we model operator states using hidden Markov models applied to human supervisory control behaviors. More specifically, we model the behavior of an operator of multiple heterogeneous unmanned vehicle systems. The hidden Markov model framework allows the inference of higher operator states from observable operator interaction with a computer interface. For example, a sequence of operator actions can be used to compute a probability distribution of possible operator states. Such models are capable of detecting deviations from expected operator behavior as learned by the model. The difficulty with parametric inference models such as hidden Markov models is that a large number of parameters must either be specified by hand or learned from example data. We compare the behavioral models obtained with two different supervised learning techniques and an unsupervised hidden Markov model training technique. The results suggest that the best models of human supervisory control behavior are obtained through unsupervised learning. We conclude by presenting further extensions to this work.

Nomenclature

a_{ij}	Probability of going from state i to state j
$\hat{a}_{ij}$	Maximum likelihood estimate of a_{ij}
A	State transition matrix
$b_i(c)$	Probability of observing symbol c in state i
$\hat{b}_i(c)$	Maximum likelihood estimate of $b_i(c)$
B	Observable emission matrix
H	Hidden Markov Model
l_s	Length of observation sequence s
M	Number of observable symbols
N	Number of hidden states

Received 17 August 2009; accepted for publication 6 December 2010. Copyright © 2011 by the Yves Boussemart, Mary L. Cummings, Jonathan Las Fargeas and Nicholas Roy. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission. Copies of this paper may be made for personal or internal use, on condition that the copier pay the $10.00 per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923; include the code 1542-9423/11 $10.00 in correspondence with the CCC.
[*] Ph.D. Candidate, Engineering Systems Division, Humans and Automation Lab, 77 Mass. Ave. 33-407, Cambridge, MA 02139, yves@mit.edu.
[†] Associate Professor, Department of Aeronautics and Astronautics, Humans and Automation Lab, 77 Mass. Ave. 33-311, Cambridge, MA 02139. Associate Fellow.
[‡] B.Eng. Candidate, Department of Aeronautics and Astronautics, Humans and Automation Lab, 77 Mass. Ave. 33-407, Cambridge, MA 02139.
[§] Associate Professor, Computer Science and Artificial Intelligence Lab, Humans and Automation Lab, 77 Mass. Ave. 33-315, Cambridge, MA 02139.

O_t^s	t-th observation in sequence s
S	Set of possible hidden states
S_i^t	Property of being at state i at time t
V	Set of possible observables
$\alpha_t(j)$	Forward probability
$\beta_t(j)$	Backward probability
$\delta_t(i)$	Viterbi backtrack pointer
$\gamma_t(i)$	Probability of the observation sequence and that the state at time t is i
$\mathcal{L}(H)$	Likelihood of model H
π_i	Initial probability of being in state i
Π	Initial probability vector
$\xi_t(i, j)$	Probability of the observation sequence and that the state at time t and $t + 1$ are i and j
$\upsilon_j(c)$	Smoothed estimator for $b_j(c)$
η_ω	Learning rate for all ω_{ij}'s
η_ν	Learning rate for all $\nu_j(c)$'s
ω_{ij}	Smoothed estimator for a_{ij}

I. Introduction

BEHAVIORAL models of human operators engaged in complex, automation-mediated high-risk domains, such as those typical in human supervisory control (HSC) settings, are of great value because of high cost of operator failure [1]. Although generic models of full human behaviors are intractable, smaller scale models of behaviors in specific contexts are desirable both to predict possible system failures in various modes of operation and to predict the impact of design interventions. While many techniques have been used to model different aspects of human behavior [2,3], partially observable models such as hidden Markov models (HMMs) have been shown to be a good fit for modeling unobservable operator states [4,5]. HMMs are stochastic models of time series processes mainly used for the modeling and prediction of sequences of symbols. In the HSC context, HMMs can be used to recognize and predict human operator behavior, given some level of intermittent interaction with an automated system [6].

One difficulty in using HMMs to model any real world process is that the model is parametric; any HMM inference algorithm requires an a priori description of the likelihood of specific observations and the likelihood of state changes. These parameters can be specified according to domain knowledge, or acquired from training data in a "model learning" process. For example, Bayesian model learning computes model parameters so as to maximize the likelihood of the model given a training data set. The learning algorithms can either be unsupervised or supervised. While an unsupervised learning algorithm only makes use of the information contained in the training data set to extract the optimal set of model parameters, supervised learning methods require that the data are augmented with *a priori* information, or labels, to guide the learning process. The labels usually consist of input data associated with the expected model output, defined by a subject matter expert. The supervised methodology has been favored by the machine-learning community in the past for two reasons: (i) the simplest supervised learning methods offer better computational efficiency compared to unsupervised learning methods and (ii) the labels in the training data are assumed to be derived from reliable ground-truth, thereby increasing the amount of information captured in the model. In the context of supervisory control models of human behavior, we argue that it is fundamentally impossible to correctly label the training data because operator cognitive states are not observable. Without reliable ground-truth, we hypothesize that human bias [7,8] is unavoidably introduced into training data labeling, which greatly influences the learning process and may generate uninformative or incorrect models.

To support our hypothesis, we define the HMM formalism along with the mathematical description of three learning techniques that we compare: (i) purely supervised learning [9], (ii) unsupervised learning [9], and (iii) smooth supervised learning [10]. We then introduce the data set used to train the models, present the results, and finally offer our conclusions on the respective values of the different techniques.

II. Hidden Markov Models

A. Formal Definition

HMMs were first formally defined by Baum and Petrie [11] and their application was popularized by Rabiner and Juang [9]. HMMs have since then been used in a large number of different contexts and have proven valuable in diverse fields such as speech recognition [12], financial data prediction [13], signal processing [14], and generic temporal data clustering [15]. HMMs consist of stochastic Markov chains based on a set of hidden states whose value cannot be directly observed. Each hidden state generates an observable symbol according to a specific emission function. Although the sequence of hidden states cannot be observed directly, the probability of being in a specific state can be inferred from the sequence of observed symbols. Transition functions describe the dynamics of the hidden state space. There are thus two types of probability parameters: the state transition probabilities and observable symbol output probabilities. Given a finite sequence of hidden states, all the possible transition probabilities and symbol output probabilities can be multiplied at each transition to calculate the overall likelihood of all the output symbols produced in the transition path up to that point. Summing all such transition paths, one can then compute the likelihood that the sequence was generated by the HMM.

Adopting the classic notation from Rabiner and Juang [9], let N be the number of hidden states $S = \{S_1, S_2, \ldots, S_N\}$ in the HMM and M be the number of observation symbols $V = \{V_1, V_2, \ldots, V_M\}$ (i.e., the dictionary size). Let S_i^t denote the property of being in state i at time t. The state transition probability from state i to state j is $A = \{a_{ij}\}$ where $a_{ij} = P(S_j^{t+1}, S_i^t)$; $i, j = 1, \ldots, N$. The symbol output probability function in state i is $B = \{b_i(c)\}$, where $b_i(c) = P(V_c|S_i)$. The model parameters must be valid probabilities and thus satisfy the constraints:

$$\sum_{j}^{N} a_{ij} = 1, \quad \sum_{c}^{M} b_j(c) = 1. \tag{1}$$

where $a_{ij} \geqslant 0, b_j(c) \geqslant 0$.

The initial probability of being in state i at time $t = 0$ is $\pi = \{\pi_i\}$, where $\pi_i = P(S_i^0)$. Thus, an HMM is formally defined as the tuple: $H = \{S, V, A, B, \pi\}$. Figure 1 illustrates the HMM concept by showing a graphical representation of a three-state model, where the set of hidden states' $\{S_1, S_2, S_3\}$ transition probabilities are defined as a set of α_{ij}'s. Each state has a probability density function of emitting a specific observable.

An HMM is said to respect the first-order Markov assumption if the transition from the current state to the next state only depends on the current state such that $P(S_j^{t+1}|S_i^t) = P(S_j^{t+1}|S_i^t, S_m^{t-1} \ldots S_n^0), \forall t, \forall j, i, m, n \in S$.

Three main computational issues need to be addressed with HMMs: model evaluation, most likely state path, and model learning. The first issue is the evaluation problem, i.e., the probability that a given sequence is produced by the model. This value is useful because, according to Bayes' rule, it is a proxy for the probability of the model given the data presented. We can thus compare different models and choose the most likely one by solving the evaluation problem. The evaluation problem is solved with the forward/backward dynamic programming algorithm. Let O^s the be the *sth* training sequence of length l_s, and the *tth* symbol of O^s be O_t^s, so that $O^s = \{O_1^s \ldots O_{l_s}^s\}$. We can define

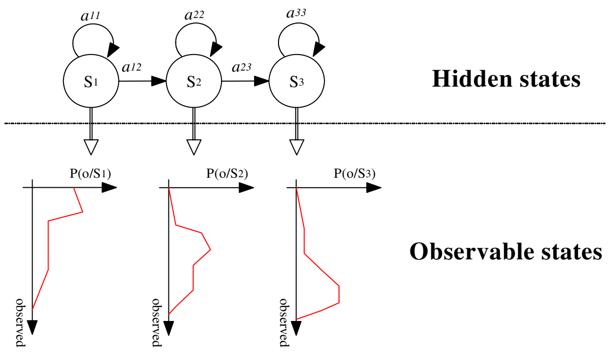

Fig. 1 A Three-state HMM.

the forward probability $\alpha_t(j)$ as the probability that the partial observable sequence $O_1^s \ldots O_t^s$ is generated and that the state at time t is j. The forward probability can be recursively computed by the following method:

$$\begin{aligned} \alpha_t(j) &= \sum_i^N a_{ij} b_j(O_{t+1}^s)\alpha_{t-1}(i), \quad (t = 1, \ldots, l_s) \\ \alpha_{l_s+1}(j) &= \sum_i^N a_{ij}\alpha_{l_s}(i) \end{aligned} \tag{2}$$

where $\alpha_0(j) = 1$ if j can be the first state and $\alpha_0(j) = 0$ otherwise.

Similarly, we can define the backward probability $\beta_t(i)$ as the probability of the partial observable sequence $O_{t+1}^s \ldots O_{l_s}^s$ and that the state at time t is i. The backward probability can also be recursively computed as follows:

$$\begin{aligned} \beta_t(j) &= \sum_j^N a_{ij} b_j(O_{t+1}^s)\beta_{t+1}(j), \quad (t = l_s - 1, \ldots, 0) \\ \beta_{l_s+1}(i) &= \sum_i^N \alpha_{ij}\beta_{l_s+1}(j) \end{aligned} \tag{3}$$

where $\beta_{l_s+1}(i) = 1$ if i can be the last state and $\beta_{l_s+1}(i) = 0$ otherwise.

We can now compute the likelihood that the given training sequence O^s is generated by HMM H and solve the state evaluation problem:

$$P(O^s|H) = \sum_i \alpha_{l_s+1}(i)\beta_{l_s+1}(i) \tag{4}$$

The second issue consists of determining the most probable ("correct") path of hidden states given a sequence of observables: a common way to solve this problem is by the Viterbi algorithm [16]. The Viterbi algorithm is a dynamic programming algorithm that finds the most probable sequence of states $S = \{S_i^1 S_j^2 \ldots S_k^T\}$ given $O = \{O_i^1 O_j^2 \ldots O_K^T\}$ by using a forward–backward algorithm across the trellis of hidden states. More specifically, let $\delta_t(i)$ be the highest probability path across all states which ends at state i at time t:

$$\delta_t(i) = \max_{S_i^1 S_j^2 \ldots S_k^{t-1}} [P(S_l^t = i, O_i^1 O_j^2 \ldots O_k^T | H] \tag{5}$$

The Viterbi algorithm finds the maximum value of $\delta_t(i)$ iteratively, and then uses a backtracking process to decode the sequence of hidden states taken along the path of maximum likelihood.

Finally, the last problem is the learning of the model, which is, given a sequence of observables, what is the maximum likelihood HMM that could produce this string? The different solutions to this problem are described in the next section.

B. Learning Strategies for HMMs

HMMs are useful for modeling sequences of data because their structure provides inferences over unobservable states. The parameters of an HMM H, i.e., the characteristics of the sequences of data being modeled, are trained to maximize $\sum_S \log(P(O^s|H))$, the sum of the posterior log-likelihoods of each training sequence O^s. Conceptually, the easiest way to model data is to start from known examples and associated model states. Provided that a large enough corpus of annotated data (pairs of input and desired output) is available, different supervised learning techniques can be used to estimate the parameters of the HMM. However, if no annotated data are available, it is still possible to automatically derive the probabilistically optimal parameters of the HMM model. This case corresponds to unsupervised learning.

For ease of notation, we introduce $\gamma_t(i)$ as the probability that the sequence O^s is generated by the HMM and that the state at time t is i. We also define $\xi_t(i, j)$ as the probability that the sequence O^s is generated by the HMM and that the state at time t and $t+1$ are i and j, respectively:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O^s|H)} \tag{6}$$

$$\xi_t(i, j) = \frac{\alpha_t(i)a_{ij}b_j(O^s_{t+1})\beta_{t+1}(j)}{P(O^s|H)}, \quad (t = 0, \ldots, l_s - 1) \tag{7}$$

1. Classic Supervised Learning

Classic supervised learning is the simplest way to extract model parameters from labeled data. In essence, the idea is to maximize the probability of the observations. Assuming that our training data consist of sequence of observations O^s, it can be shown that the most likely estimate (MLE) of the emission probability distribution given the training data is distributed according to the frequency of emissions in the data. We can calculate the frequency by counting how often an observation was generated by a state. Similarly, the most likely transition distribution is distributed according to the frequency of state transition.

Ordinarily, we cannot count the frequency of state transitions or observation emissions from a particular state because the state is hidden. However, supervised learning makes the assumption that during training, we have access to the underlying state sequence and can therefore "label" each observation in O^s with the corresponding true, hidden state. The transition matrix of $A = \{a_{ij}\}$ can therefore be computed directly by counting the relative frequency of the transition between all states i and j. Similarly, the emission functions $B = \{b_j(c)\}$ can be computed by counting the number of times a specific observation c has been observed at a given state j. More formally, if we define count$(s \rightarrow s')$ as the number of time state s' follows state s and count$(s \rightsquigarrow c)$ as the number of time state j is paired with emission c:

$$\text{count}(s \rightarrow s') = \sum_{j=1\ldots l_s-1} \left[\!\left[S_j = s \bigwedge S_{j+1} = s' \right]\!\right] \tag{8}$$

$$\text{count}(s \rightsquigarrow c) = \sum_{j=1\ldots l_s} \left[\!\left[s_j = s \bigwedge O_j = c \right]\!\right] \tag{9}$$

The MLE estimates $\hat{a}_{ij}$ of a_{ij} then are:

$$\hat{a}_{ij} = \frac{\sum_{i=1\ldots N} \text{count}(s \rightarrow s')}{\sum_{i=1\ldots N} \sum_{s'} \text{count}(s \rightarrow s')} \tag{10}$$

Similarly, the MLE estimates $\widehat{b_j}(c)$ of $b_j(c)$ are:

$$\widehat{b_j}(c) = \frac{\sum_{i=1\ldots N} \text{count}(i, s \rightsquigarrow x)}{\sum_{i=1\ldots N} \sum_{x} \text{count}(i, s \rightsquigarrow x)} \tag{11}$$

This supervised learning technique to compute the HMM model parameters is relatively simple and runs in $O(l_s)$, where l_s is the length of a sequence.

2. Unsupervised Learning

It may be, however, that even during training it is not feasible to access to the underlying hidden state. For example, it is impossible to know exactly what information an operator is internally processing (e.g., is a thinking person solving a problem or wondering about lunch?) Without such labels, we cannot use the above-said algorithm because when counting state transitions, we no longer know which state transitioned to where at each time step and we no longer know which state to assign credit for each observation. We can, however, use unsupervised learning algorithms to not only infer the labels but also infer the model during training. The most commonly used algorithm for HMMs is a form of expectation maximization (EM) called the Baum–Welch algorithm [11]. Just as in supervised learning, the goal of the Baum–Welch algorithm is to maximize the posterior likelihood of the observed sequence O^s for a given HMM. More formally, Baum–Welch computes the optimal model such that:

$$H^* = \text{argmax}_H \left(\prod_s P(O^s | H) \right) \tag{12}$$

We cannot, however, use equations (10) and (11) directly because we do not know the state at each time step t. EM operates by hypothesizing an initial, arbitrary set of model parameters. These model parameters are then used to estimate a possible state sequence $S_{O^s} = \{s^1 \dots s^{l_s}\}$ via the Viterbi algorithm. This is the expectation or E-Step of the EM algorithm. The model parameters are then *re-estimated* using equations (10) and (11) at the given state labels S_{O^s}. This is the maximization or M-Step of the EM algorithm. We could make the assumption that the state sequence S_{O^s} is correct. However, the state sequence can be very uncertain in cases, and an incorrect assumption can lead to failures in determining model parameters. The EM algorithm takes the uncertainty of the state sequence estimate into account by using the probability of being in state S_i at time t to estimate transition and emission probabilities. The probability $\hat{a}_{ij}$ is re-estimated using $\gamma_t(i)$ and $\xi_t(i, j)$ based not on the frequency of state transitions from i to j in the data, but on the likelihood of being in state i at time t and the likelihood of being in state j at time $t + 1$. Note that our frequencies or counts in equations (10) and (11) are not integer counts but likelihoods and are therefore fractional. Similarly, $\hat{b}_i(c)$ is re-estimated with $\gamma_t(i)$ as the likelihood of being in state i when the observation was c. The equations of re-estimation are:

$$\hat{a}_{ij} = \frac{\sum_S \sum_{t=0}^{l_S} \xi_t(i, j)}{\sum_S \sum_{t=0}^{l_S} \gamma_t(i)} \tag{13}$$

$$\hat{b}_i(c) = \frac{\sum_S \sum_{t=1, O_t^s = c}^{l_S} \gamma_t(i)}{\sum_S \sum_{t=1}^{l_s} \gamma_t(i)} \tag{14}$$

Through this iterative procedure, it can be proven that the Baum–Welch algorithm converges to a local optimum [9]. The process described earlier assumes that the model structure (i.e., the number of hidden states) is known in advance. In most practical settings, this assumption is unrealistic and the structure of the model must be determined through a process called model selection. While there are many different criteria used to determine the validity of a model, we adopt an information-theoretic metric called the Bayesian information criterion (BIC) [17]. This metric allows the comparison of different models, in particular with different number of hidden states, trained on the same underlying data. The BIC penalizes the likelihood of the model by a complexity factor proportional to the number of parameters P in the model and the number of training observations K.

$$BIC = -2 \log(\mathcal{L}(H)) + P \log(K) \tag{15}$$

Such techniques provide a metric by which the number of hidden states in a model can be determined without prior knowledge; the model structure chosen balances modeling fit and generalizability.

3. *Smooth Supervised Learning*

Smooth supervised learning was first introduced by Baldi and Chauvin [18] to avoid issues with sudden jumps or absorbing probabilities of 0 during the parameter update process. The absorption property of null probabilities is an issue because once a transition or emission function is set to 0, it cannot be used again. The idea for the supervised case is to minimize the distance between the a priori labels and the labels estimated as most likely by the HMM. This algorithm can be tailored for sequence discrimination [10], and we can replace the usual a_{ij} and $b_j(c)$ with real-value parameters ω_{ij} and $\upsilon_j(c)$ defined as (with λ being a constant):

$$\begin{aligned} a_{ij} &= \frac{e^{\lambda \omega_{ij}}}{\sum_k e^{\lambda \omega_{ik}}} \\ b_j(c) &= \frac{e^{\lambda \upsilon_j(c)}}{\sum_k e^{\lambda \upsilon_j(k)}} \end{aligned} \tag{16}$$

Let $p_s = P(O^s|H)$ be the target value of the likelihood of the pre-labeled observations and associated symbols given the HMM H. The probability p_s will depend on the length of the sequence, so we introduce δ which scales the probability p_s with respect to the length of the sequence. C_a and C_b are constants that normalize δ for different observation sequence sizes:

$$\begin{aligned} p_s &= \log(P(O, S|H)) \\ \delta_s &= 1 - C_a \frac{p_s}{(C_b - p_s^2)} \end{aligned} \tag{17}$$

The algorithm thus tries to maximize δ to maximize the fit of the model to the data. Given η_ω and η_ν as learning rates, the update rules for ω_{ij} and $\upsilon_j(c)$ are as follows:

$$\begin{aligned} \Delta\omega_{ij} - \eta_\omega \sum_s \delta_S \sum_{t=0}^{l_s} (\varepsilon_t(i, j) - a_{ij}\gamma_t(i)) \\ \Delta\upsilon_j(c) = \eta_\nu \sum_s \delta_s \sum_{t=0}^{l_s} (\gamma_t(j)_{O^s_{t=c}} - b_j(c)\gamma_t(j)) \end{aligned} \tag{18}$$

To reach convergence, the constants and learning rates need to be changed for each new training set. Because the solution space is highly nonlinear, there is no analytical method to choose these parameters appropriately. As a result, the constants and rates have to be found by a very time-consuming process of trial and error to maximize the model likelihood.

C. HMMs of HSC Behavior

We posit that the dual-chained structure of HMMs is well suited to modeling human behavior in a supervisory control system. HMMs are appropriate, because while the operator cognitive states are not directly observable, they can be inferred from his or her actions with the system. This is similar to the HMM notion of hidden states that must be inferred from observable symbols. However, as discussed previously, how such models should be trained, given the inherent hidden nature of operator cognitive states and the lack of a reliable ground truth remains an open question. Using an HSC application in the form of single operator control of multiple unmanned vehicles, we will demonstrate that for HMM models that attempt to infer operator cognitive states, unsupervised methods are superior to supervised ones. While the results we present in this paper are specific to the interface and task described subsequently, the testing platform we use embodies two characteristics embodied in all HSC systems: first, the operator intermittently interacts with highly levels of automation and secondly, the task is performed under time pressure.

III. Experiment and Results

Data used to develop the HMM of an HSC system, specifically that of a single operator controlling multiple unmanned vehicles, was obtained from the experiment described in Nehme et al. [19]. While the goal of the original experiment was to validate a discrete event simulation of an operator controlling multiple heterogeneous unmanned vehicles, the recorded user interface interactions represent a rich corpus of supervisory control behavior. In the experiment, the Research Environment for Supervisory Control of Heterogeneous Unmanned Vehicles (RESCHU) simulator was used to allow a single human operator to control a team of UVs composed of unmanned air and underwater vehicles (UAVs and UUVs) (Fig. 2).

In RESCHU, the UVs perform surveillance tasks with the ultimate goal of locating specific objects of interest in urban coastal and inland settings. UAVs can be of two types: one that provides high-level sensor coverage (high altitude long endurance or HALEs, akin to global hawk UAVs), while the other provides more low-level target surveillance and video gathering (medium altitude long endurance or MALEs, similar to predator UAVs). In contrast, UUVs are all of the same type. Thus, the single operator controls a heterogeneous team of UVs which may consist of up to three different types of platforms.

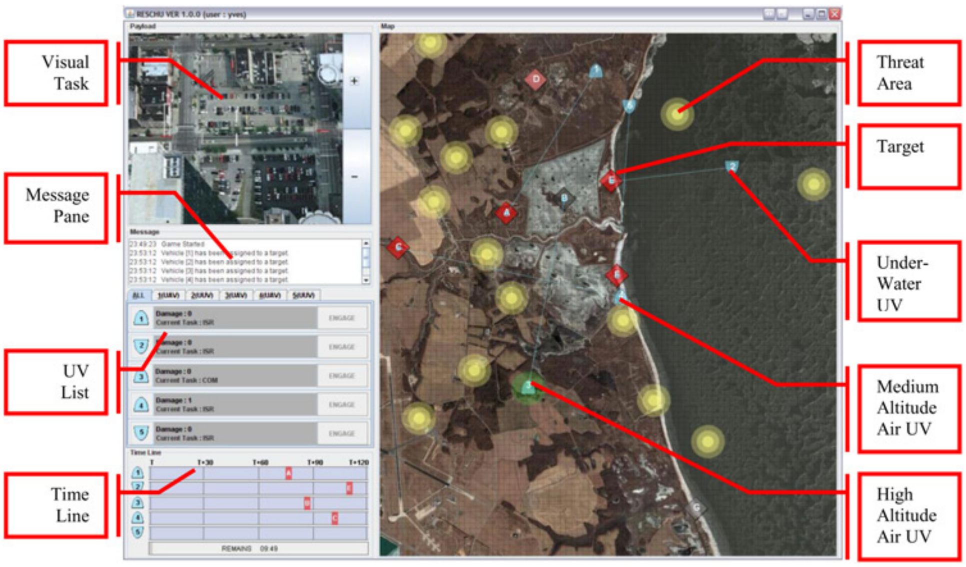

Fig. 2 The RESCHU interface.

In the rules of this simulation, the HALE performs a target designation task (simulating some off-board identification process). Once designated, either MALEs or UUVs perform a visual target acquisition task, which consists of looking for a particular item in an image by panning and zooming the camera view. Once a UV has visually identified a target, an automated planner chooses the next target assignment, creating possibly sub-optimal target assignments that the human operator can correct. Furthermore, threat areas appear dynamically on the map, and entering such an area could damage the UV, so the operator can optimize the path of the UVs by assigning a different goal to a UV or by adding waypoints to a UV path to avoid threat areas.

Participants maximized their score by (i) avoiding threat areas that dynamically changed, (ii) completing as many of the visual tasks correctly, (iii) taking advantage of re-planning when possible to minimize vehicle travel times between targets, and (iv) ensuring a vehicle was always assigned to a target whenever possible. Training was done through an interactive tutorial and an open-ended practice session. After completing a 10-min experiment, the participants could see their score which corresponded to the total number of targets correctly identified. All data were recorded to an online database. The data of interest for this project consisted of user interactions with the interface in the manner of clicks, such as UV selections on the map or on the left sidebar, waypoint operations (add, move, delete) goal changes, and the start and end of visual tasks.

Data were collected on 49 subjects via the experimental procedure described earlier. The data collected amounts to ~8 h of raw experimental data and about 3500 behavioral sample points. The average length for each sequence consisted of about 71 user interface events.

A. Applying HMMs to HSC Behavior

Determining the parameters of an HMM of user behavior requires training the model on the observed behavioral data. The raw behavioral data, which consists of the logged user interface events described previously, cannot be used directly by the HMM learning algorithms, and must be pre-processed. Figure 3 shows this process, which consists of a grammatical and a statistical phase.

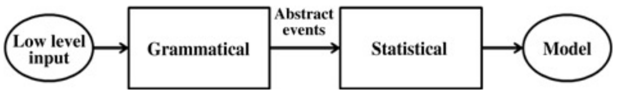

Fig. 3 A combined grammatical and statistical approach infers future behavior from a stream of current behavior.

Table 1 The RESCHU grammar

All						
Underwater UV						
Aerial UV						
High altitude UV						
UV type/mode	Select sidebar	Select map	Waypoint edit	Waypoint add/del	Goal	Visual task/engage

First, in the grammatical phase, low-level input data (the logged raw events) are translated into abstract events by the use of a syntactic parser. The role of the grammar is to abstract low-level user interface interactions into a set of meaningful tasks that can both be learned by the algorithm, as well as interpreted by a human modeler. Thus, the grammar performs a state space reduction and defines the scope of the observable space usable by the machine-learning process. In the context of RESCHU, this solution space corresponds to the set of tasks that operators must perform to complete the mission. The grammatical rules were established through a cognitive task analysis (CTA) [20] of the RESCHU interface and demonstrated which interface interactions belonged to which task set. The CTA yielded the grammatical space shown in Table 1. User interactions were first discriminated based on the type of UV under controlled, the y-axis of the grammar. Then, the interactions with each of the UV types were separated into different modes on the x-axis, representing different task types: selection on either the sidebar or on the map, waypoint manipulation (addition, deletion, and modification), goal changes and, finally, the visual task engagement. The sequences of low-level user interactions were thus translated into a sequence of integers suitable for the statistical phase.

The second step of the process is the statistical learning phase that models the time sequenced data. The goal of this phase is to find statistically linked clusters of abstract events, which we call operator states. During this phase, the HMM parameters are trained using sequences of either labeled or unlabeled data. The initial HMM parameters were randomly chosen. In this experiment, we used 49 data traces of individual user trials, and used a three-fold cross-validation by training the algorithm on 46 subjects and keeping 3 subjects as a test set. This procedure was repeated multiple times to ensure the learning process was not over-fitting the data.

B. State Labeling

For the HMM leveraging supervised learning, labels are needed. However, due to the futuristic nature of the single operator–multiple unmanned vehicle system, no subject matter expert is available to label the data by hand. Through the cognitive task analysis, we derived an initial set of labels known to be recurrent based on sets of previously identified common cognitive functions for UAV tasks [21], such that user–interface interactions could be grouped as clusters or states. For the supervised learning portion of this work, the a priori labels consisted of:

1. Normal navigation: map selections and interactions with goals,
2. Monitoring: interaction with the UVs based on selection on the sidebar or the map,
3. Visual task: set of action that results in the visual task engagement action, and
4. Preemptive threat navigation: adding a series of waypoints to change the course of the vehicle should the need arise.

This labeling scheme covered about 80% of the training sequences and any observable that did not get labeled was dropped from the training set. These labels align with those basic underlying operator functions that form the core of supervisory control of unmanned vehicles which include navigation, vehicle health and status monitoring, and payload management [22]. In RESCHU, the payload management task is the visual task.

C. Classic Supervised Model

In general, the supervised learning algorithms find the optimal set of parameters for state transitions and the emission functions. The model in Fig. 4 represents the HMM obtained with the classic supervised learning method. All transitions with a weight under 5% are not shown for legibility purposes. Models under this learning paradigm contain four states which may correspond to interaction types as defined by the a priori patterns of most likely observable states. The annotated arrows between the states represent the probability of going from one state to

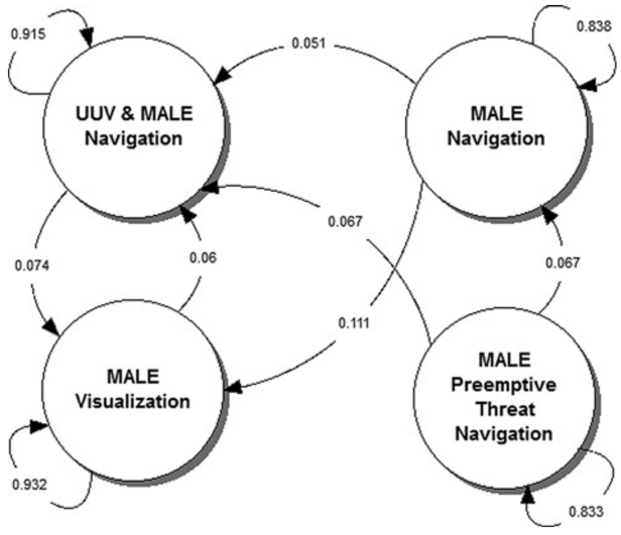

Fig. 4 Supervised learning model of a single operator of multiple unmanned systems.

another. The observable emission probability of each hidden state is not shown graphically. Instead, for each hidden state, the set of most likely observables is determined by analyzing the emission probability function. Then, based on that set of most likely observables events, a label can be attributed to the hidden state. The supervised learning process leverages the pre-defined state labels and learns the most likely set of parameters for the HMM. In the case of Fig. 4, the four states of the model are as follows: the first state aggregates sequences of UUV and MALE navigation interactions (i.e., repeatable map and goal selections between the two vehicle types). The second state focuses uniquely on similar navigation interactions, but for MALEs only. The first state that contains both UUV and MALE navigation is likely due to comparatively lower interaction frequencies between the operator and the UUVs, which could subsume similar low frequency interactions with MALEs. In contrast, the third state embodies MALE threat navigation, which is adding a series of waypoints to change the course of the vehicle if needed. This differs from normal navigation tasks in that the operator acts only on waypoints and does not interact with goals. Finally, the last state is the MALE visualization, which corresponds to the visual target identification task in RESCHU. The obtained model shows that operator interactions with the HALEs do not appear as a distinct state, even though we know that they exist. While operators interacted with the HALEs less than they did with the MALE UAVs and the UUVs, because HALE use was required before use of a MALE for unknown targets, we anticipated that this would be a state with a clearly assigned meaning. However, this was not seen in this supervised learning model.

D. Smooth Supervised Models

The model in Fig. 5 represents the model obtained with the smooth supervised learning method. Again, all transitions with a weight under 5% are not drawn for legibility purposes. The model obtained is somewhat different from the one obtained through classic supervised learning (Fig. 4), although three of the four states are the same. The first state aggregates MALE visualization and UUV navigation tasks. This aggregation denotes that UUV navigation and MALE visualization are statistically clustered together and indicates that a number of MALE visual tasks tended to either precede or follow a UUV navigation interaction. This is possibly due to the spatial distribution of the targets along the water body on the map. The second state expresses MALE visual tasks and the third state represents MALE normal navigation, and finally the last state corresponds to MALE threat navigation. While the transition probabilities between hidden states is less deterministic (as indicated by the higher number of likely transitions between hidden states) than in the classic supervised model, operator interactions with the HALEs again disappear and do not appear as a distinct state as defined by the learning algorithm.

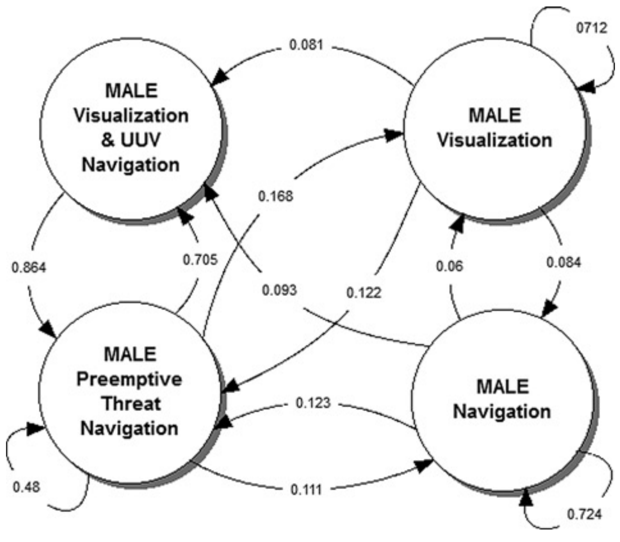

Fig. 5 Smooth supervised learning model of a human operator of multiple unmanned systems.

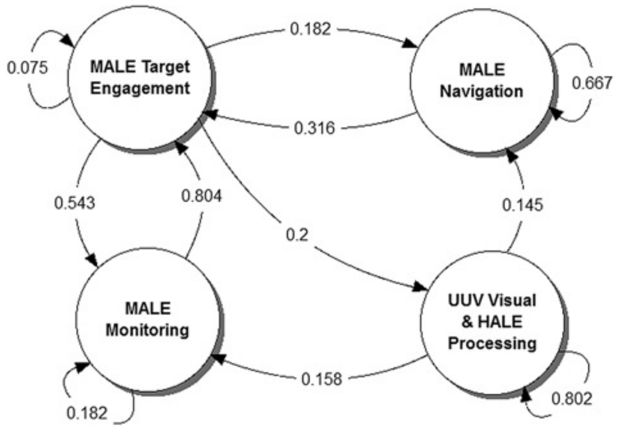

Fig. 6 Unsupervised learning model of a human operator of multiple unmanned systems.

E. Unsupervised Models

Because there is no prior knowledge assumed about the model structure, the unsupervised method first requires performing model selection to determine how many hidden states the model should have. Models with a number of hidden states ranging from 2 to 15 were trained multiple times with different randomly assigned initial parameters to avoid convergence to a local optimum. The training was performed through the unsupervised Baum–Welch learning technique. The BIC for each model was computed and the four-state model was chosen to be the most adequate model structure (Fig. 6). Again, all transitions with a weight under 5% are not included for legibility purposes. The first state represents MALEs engaging targets (i.e., vehicle selection and goal setting), the second state is MALE navigation and the third state groups UUV visual task and HALE processing (i.e., navigation and goal setting). Finally, the last state is MALE health and status monitoring, which are the interactions that allow the operator to verify the task assigned to a specific UV along with the damages it has incurred.

This model is markedly different from the models obtained with the supervised learning techniques, most notably in that the HALE and UUV interactions (i.e., normal and threat navigation along with visual task) are grouped together in a state distinct from the others which focus on MALE interactions. This denotes that the unsupervised

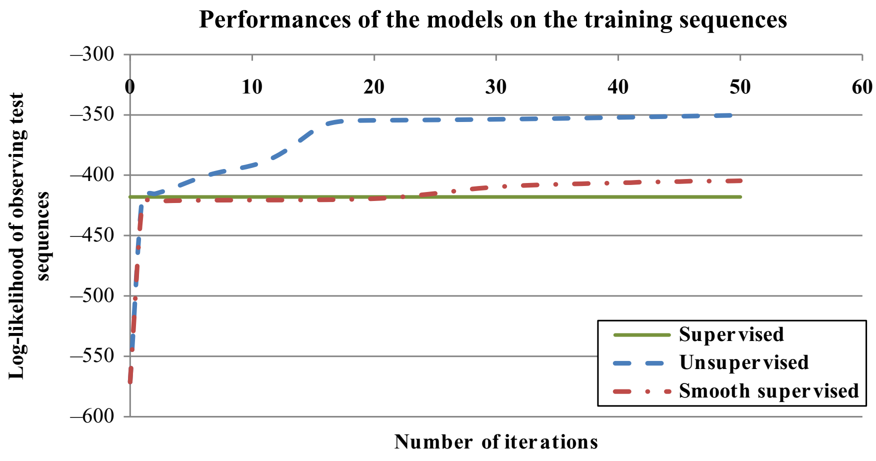

Fig. 7 Model fit in terms of test set likelihood for the three different training techniques.

model was able to recognize the much less frequent interactions with UUVs and HALEs as a distinct state and qualitatively different from that with the MALEs.

IV. Discussion and Model Comparisons

Given the three different models, it is interesting to consider how fast the training converges to a stable set of parameters, and how good the data fit the models after the training process. Figure 7 shows the evolution of the three learning techniques over the course of 50 learning iterations.

As expected, the supervised algorithm converges in the first iteration and provides a constant performance baseline. The first few iterations of the smooth supervised algorithm, conversely, are quite poor. However, at the 25th iteration, the smooth supervised model surpasses the classic supervised model and plateaus at around the 30th iteration. The first few learning iterations of the unsupervised model behave very closely to the smooth supervised. After the 3rd iteration, however, while the smooth supervised model plateaus for the first time, the unsupervised algorithm log-likelihood continues to increase and converges at the 20th learning iteration. In terms of log-likelihood, the performance differences are clear in that the unsupervised learning method gives rise to a model that is more likely than both supervised methods. The smooth supervised model provides slightly superior posterior log-likelihoods than the classic supervised one.

Adopting a human-centric and cost-benefit point of view, it is interesting to compare how much human effort was required to generate the above-mentioned models. For both supervised methods, the cost of labeling the data was quite high, as our initial undertaking was to execute a cognitive task analysis of the single operator—multiple unmanned systems to define a likely set of behaviors. Cognitive task analyses are labor intensive and are somewhat subjective, so there is no guarantee that the outcome behaviors are correctly identified.

Moreover, these a priori-defined patterns then had to be tagged in all the sequences to construct the corpus of training and testing data. To avoid the known risks of human judgment bias [23] in the state definition process, an iterative approach was adopted in which multiple acceptable sets of state definitions were compared with the data. The set of definitions that provided the better explanation for the states was then chosen. It is important to note that expert knowledge of the task was required in all phases of this lengthy process. Thus, in addition to being extremely time intensive, it is recognized that expert labeling is a costly and sometimes subjective process that can unnecessarily constrain the resulting models to the types of behaviors seen as important by human experts[5], which could ultimately be flawed.

The classic supervised algorithm was straightforward in execution and, by design, converged in a single iteration. In the case of the smooth supervised algorithm, however, we had to spend a considerable amount of effort tweaking the learning parameters and the different constants in the error distance function to achieve a reasonable convergence point. Unfortunately, due to the highly nonlinear features of the solution space, it was impossible to use standard

Table 2 Qualitative performance summary of the learning algorithms

Algorithm	Convergence speed	Performance	Human effort required
Straight supervised	Best	Worst	Better
Smooth supervised	Worst	Better	Worst
Unsupervised	Better	Best	Best

optimization algorithms such as grid-based exploration or sequential quadratic programming. Our approach then was to use a process similar to simulated annealing in which we initially chose a large number of random algorithm parameter values and focused the exploration around the most "promising" initial points, where the value of a given set of initial values was measured as the closeness between the expected and modeled distance in label-space. Finally, the Baum–Welch (unsupervised) method required some effort coding, but subsequently did not require human intervention to reach convergence.

Table 2 shows a qualitative summary of the arguments presented earlier. We suggest that even when experts are available to decide which operator states should be labeled, uncertainty and subjective judgment cannot be eliminated from the process. Furthermore, even with a defined state space of labels, it is not possible for an expert to look at the data and unambiguously assign labels to observable states. This is especially true for states with no clear ground truth. The possible bias introduced in the labeling process is detrimental to the fit, and therefore the predictive ability of the model obtained through supervised learning. Therefore, we submit that unsupervised methods should be preferred to the supervised technique in modeling human cognitive interactions with automated systems.

In addition to the quantitative metrics such as convergence speed and performance, it is interesting to analyze the models for the explanatory mechanism they can provide. For the supervised models, the results obtained are similar in that they emphasize the role of the MALEs and UUVs. Both supervised models, based on human-biased grammar, disregard a major part of the problem space: the existence of a third vehicle category (the HALEs). The unsupervised learning technique, on the contrary, segregated the HALE and UUV interactions in a separate state. The unsupervised technique also detected that there were very few marine targets, and thus the majority of the interactions with UUVs were spent in the visual task and not replanning. Such examples show the richness of the interpretation that can be obtained from analyzing a nonbiased model that is based on statistical properties of operator interactions. Such unsupervised approaches could actually be used to augment cognitive task analyses to provide more objective results in what is known to be a very subjective process.

These results, both quantitative (i.e., model likelihood) and qualitative (i.e., model interpretation), demonstrate that for the purpose of modeling HSC operator states, the use of supervised learning is likely inherently flawed. Not only did the supervised models yield poorer prediction rates, but also failed to capture important characteristics of operator behavior. The poor results could be blamed, quite rightly, to poor a priori labeling of the states, and that the results could have been very different with better labeling. While we agree, this again highlights the subjective nature of expert state labeling in the presence of uncertainty. In the specific context of HSC modeling, we argue that it is very difficult, if not impossible, to obtain a correct set of labels. Even using multiple experts to label in hopes of reducing the negative impact of imperfect labeling is not guaranteed to lead to better results [24].

While RESCHU is representative of a large class of HSC settings where an operator has to oversee and interact intermittently with highly automated external entities, our results do not necessarily generalize to significantly different situations. For example, our modeling efforts rely mostly on UI interaction data and thus our results may not be applicable in the subclass of HSC tasks that require monitoring for long periods of time with no system interaction (i.e., nuclear power plant supervision or air traffic control activities). However, the specificity of the interface may not be such a critical factor since the output of the grammatical parsing stays the same, regardless of interface layout. For example, moving a button around the interface would have no impact on the resulting interpretation in the grammar.

The nature of correct, or even close enough, labels is very domain specific, and when modeling high-risk systems such as those in command and control systems, we propose that the more conservative and objective unsupervised approach is superior. Application of supervised learning methods to domains with known ground truths and objective label identifications, even in non-HSC domains such as determining facial expressions [25] or melodic content [26] do not suffer from these limitations. However, our results demonstrate that the lack of known ground-truth to annotate operator cognitive states leads to poor models, and that unsupervised learning should be strongly considered for future work in similar HSC contexts where an operator has to intermittently interact with the system.

One other shortcoming of this work is that classical HMMs do not provide a way to explicitly deal with state durations. In classical HMMs, the probability of staying in a given state has to be geometrically distributed according to the state self-transition probability. Assuming such a state sojourn distribution may not be valid in all contexts, hidden semi-Markov models (HSMMs, also known as explicit duration HMMs) are built to address this specific issue. Because HSMMs incorporate timing information, they can predict not only the most likely future states but also when these future states are likely to occur. Such predictions are of great value in time-sensitive situations commonly found in HSC contexts where operational tempo is critical. Work is underway to address these temporal considerations in HSC models, and extend these results via HSMMs.

V. Conclusion

In this paper, we compared models of HSC obtained through supervised and unsupervised learning. Our results, rooted in the domain of a single operator controlling multiple unmanned vehicles, suggest that not only do the supervised learning methods require significantly more human involvement both in terms of state labeling and parameter adjustments, they also tend to perform worse than the models obtained with unsupervised learning. The lack of accessible ground truth to label operator cognitive states and inherent human decision making biases in the labeling process hinders the application of supervised learning to operator HSC models. Although domain specific, we propose that operator modeling efforts could greatly benefit from using unsupervised learning techniques.

Acknowledgments

This research was supported by Boeing Phantom Works.

References

[1] Leveson, N. G., "Software Safety: Why, What, and How," *Computing Surveys*, Vol. 18, 1986, pp. 125–163.

[2] Wayne, D. G., Bonnie, E. J., and Michael, E. A., "The Precis of Project Ernestine or an Overview of a Validation of GOMS," *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, Monterey, CA, 1992.

[3] Anderson, J. R., *Rules of the Mind*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1993.

[4] Hayashi, M. "Hidden Markov Models to Identify Pilot Instrument Scanning and Attention Patterns," *Proceedings of the IEEE International Conference on Man and Cybernetics*, 2003.

[5] Hoey, J., "Value-Directed Human Behavior Analysis from Video Using Partially Observable Markov Decision Processes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 29, 2007, pp. 1118–1132.

[6] Boussemart, Y., and Cummings, M. L., "Behavioral Recognition and Prediction of an Operator Supervising Multiple Heterogeneous Unmanned Vehicles," *Proceedings of the Humans Operating Unmanned Systems*, *HUMOUS'08*, Brest, France, 2008.

[7] Kahneman, D., and Tversky, A. "Prospect Theory: An Analysis of Decision under Risk," *Econometrica*, Vol. 47, 1979, pp. 263–292.

[8] Tversky, A., and Kahneman, D., "Judgement Under Uncertainty: Heuristics and Biases," *Science*, Vol. 185, 1974, pp. 1124–1131.

[9] Rabiner, L., and Juang, B., *An Introduction to Hidden Markov Models*, ASSP Magazine, IEEE [see also IEEE Signal Processing Magazine], Vol. 3, No. 1, 1986, pp. 4–16.

[10] Hiroshi, M., "Supervised Learning of Hidden Markov Models for Sequence Discrimination," *Proceedings of the First Annual International Conference on Computational Molecular Biology*, ACM, Santa Fe, New Mexico, 1997.

[11] Baum, L. W., and Petrie, T., "Statistical Inference for Probabilistic Functions of Finite State Markov Chains," *The Annals of Mathematical Statistics*, Vol. 37, No. 6, 1966, pp. 1554–1563.

[12] Chien, J.-T., and Furui, S., "Predictive Hidden Markov Model Selection for Decision Tree State Tying," *Eurospeech 2003*, Geneva, 2003.

[13] Zhang, Y., *Prediction of Financial Time Series with Hidden Markov Models*, School of Computer Science, Simon Fraser University, 2004.

[14] Kil, D. H., and Shin, F. B., "Pattern Recognition and Prediction with Applications to Signal Characterization," AIP Series in Modern Acoustics and Signal Processing, AIP Press, Woodbury, NY, 1996, xvi, 418 p.

[15] Li, C., and Biswas, G., "Finding Behavior Patterns from Temporal Data Using Hidden Markov Model based on Unsupervised Classification," *Proceedings of the International ICSC Symposium on Advances in Intelligent Data Analysis (AIDA'99)*, Rochester, NY, 1999.

[16] Forney, G. D., "The Viterbi Algorithm," *Proceedings of the IEEE*, Vol. 61, No. 3, 1973, pp. 268–278.

[17] Burnham, K. P., and Anderson, D. R., *Model Selection and Multimodel Inference, A Practical Information Theoretic Approach*, Springer, 2002.

[18] Baldi, P., and Chauvin, Y., "Smooth On-Line Learning Algorithms for Hidden Markov Models," *Neural Computation*, Vol. 6, 1994, pp. 307–318.

[19] Nehme, C. E., Crandall, J., and Cummings, M. L., "Using Discrete-Event Simulation to Model Situational Awareness of Unmanned-Vehicle Operators," *Proceedings of the 2008 Capstone Conference*, Norfolk, VA, 2008.

[20] Schraagen, J. M., Chipman, S., and Shalin, V. E., *Cognitive Task Analysis*, Erlbaum, Mahwah, NJ, 2000.

[21] Nehme, C. E., Crandall, J. W., and Cummings, M. L., "An Operator Function Taxonomy for Unmanned Aerial Vehicle Missions," *Proceedings of the International Command and Control Research and Technology Symposium*, Newport, Rhode Island, 2007.

[22] Cummings, M. L., Bruni, S., Mercier, S., and Mitchell, P. J., "Automation Architecture for Single Operator, Multiple UAV Command and Control," *The International Command and Control Journal*, Vol. 1, No. 2, 2007, pp. 1–24.

[23] Tversky, A., and Kahneman, D., "The Framing of Decisions and the Psychology of Choice," *Science, New Series*, Vol. 211, No. 4481, 1981, pp. 453–458.

[24] Victor, S. S., Foster, P., and Panagiotis, G. I., "Get Another Label? Improving Data Quality and Data Mining Using Multiple, Noisy Labelers," *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* ACM, Las Vegas, NV, 2008.

[25] Dong, L., Jie, Y., Zhonglong, Z., and Yuchou, C., "A Facial Expression Recognition System Based on Supervised Locally Linear Embedding," *Pattern Recognition Letters*, Vol. 26, No. 15, 2005, pp. 2374–2389.

[26] Kyogu, L., and Malcolm, S., "Automatic Chord Recognition from Audio Using a Supervised HMM Trained with Audio-from-symbolic Data," *Proceedings of the 1st ACM Workshop on Audio and Music Computing Multimedia*, ACM, Santa Barbara, CA, 2006.

Mike Hinchey
Editor-in-Chief