Controlling Eye Movements with Hidden Markov Models

Raymond D. Rimey Christopher M. Brown

The University of Rochester
Computer Science Department
Rochester, New York 14627

Technical Report DRAFT

Abstract

Advances in technology and in active vision research allow and encourage sequential visual information acquisition. Hidden Markov models (HMMs) can represent probabilistic sequences and probabilistic graph structures: here we explore their use in controlling the acquisition of visual information. We include a brief tutorial with two examples: (1) use input sequences to derive an aspect graph and (2) similarly derive a finite state machine for control of visual processing.

The first main topic is the use of HMMs in both their learning and generative modes, and their augmentation to allow inputs sensed during generation to modify the generated outputs temporarily or permanently. We propose these augmented HMMs as a theory of adaptive skill acquisition and generation. The second main topic builds on the first: the augmented HMMs can be used for knowledge fusion. We give an example, the what-where-AHMM, which creates a hybrid skill from separate skills based on object location and object identity. Insofar as low-level skills can be learned from the output of high-level cognitive processes, AHMMs can provide a link between high-level and low-level vision.

This material is based on work supported by the NSF under grants CDA-8822724 and IRI-8920771, and by U.S. Air Force research grant no. AFOSR-89-0222. The government has certain rights in this material.

1 Overview

The advent of the "active vision" paradigm (*e.g.* [1; 3; 9; 11]), partly driven by the availability of sophisticated hardware for sensor/platform control and real-time image processing, raises many issues of how active sensors can help with the vision problem. Here we explore the control of time sequences of camera motions for evidence gathering in task-oriented vision. The results apply whenever spatially-variant image processing resources can be flexibly deployed to implement attentional shifts. We propose hidden Markov models (HMMs) as a representation for action sequences, and also for relational structures in the world that can be discovered by action sequences.

Section 2 introduces the two novel uses we wish to make of HMMs: (a) implementing visual information-gathering skills that are self-modifying on short and long time scales, and (b) information fusion and links to high-level vision by combining separately-acquired skills into more complex skills. Both goals are met by using the generative capabilities of augmented versions of HMMs. Section 3 presents a brief introduction to the HMM, including the formal model and a summary of the basic algorithms for parameter estimation, sequence classification, and sequence generation. The qualities of the HMM that make it interesting for computer vision applications are summarized, and two examples are presented to illustrate HMMs in recovery of spatial and control structure for computer vision.

Section 4 introduces the Augmented HMM (AHMM), which allows an explicit action sequence to adapt to visual stimuli during its execution. In Section 5, the AHMM is used to learn "skill structure graphs", generate skilled sequences of camera fixations, and modify them as a result of real-time image variations. Directed graphs representing sequences of object-locations can be learned, as can those that sequence, or index, through objects based on descriptive features. Experimental results from our implementation of visual skills are presented, including an illustration of how robust the AHMM is to noise in either the training examples or the visual feedback signal.

Our ultimate purpose for studying adaptable action sequences is to incorporate them with a high-level vision system. Complex control structures can be built up using AHMMs, implementing sophisticated interactions between learned skills and ongoing cognitive activity. As an initial step towards this goal, Section 6 shows how two skill structure graphs can be combined, using the (AHMM) feedback modification mechanism for information fusion

to create an efficient representation of the combined knowledge. The principle allows low-level and high-level knowledge to be combined. The result is a system that gathers evidence for solving a task (*e.g.* object recognition) by sequentially positioning a camera so that it views a sequence of desired features in their expected locations in the image. The system is demonstrated with a simple face recognition application.

2 The Problems

2.1 Visual Skills

In active and animate vision [3] the sensors can make selective movements to direct resources or attention to different areas of the scene. The traditional artificial intelligence view of this process is highly cognitive: Vision is like problem-solving — the observer is iteratively reasoning from an internal model, acting, and obtaining information. This explanation of actions ignores important and large repertoires of behavior that are necessary to survival — they are the motor and visuo-motor skills, which are not often studied but are of interest in their own right. To understand the basic, qualitative difference between skilled behavior and behavior mediated by cognition and feedback, consider the differences between playing the piano and learning to play the piano, or between signing your name and copying calligraphy in an unknown script.

Both cognition (*e.g.* planning, problem solving) with sensory feedback and reflexive (reactive) processes lead to *emergent sequences* of actions from the observer. The resulting output sequences themselves are not represented, remembered, or available in advance. In a context that calls for skill, the resulting performance is painfully inadequate, and compared to skilled behavior there may be a qualitative inadequacy in competence as well. As computer vision and robotics become more capable of real-time performance and as more is demanded of them, the "cognition, action, feedback" loop must be augmented by something like skilled behaviors. To deal with skills, we need structures that represent explicit sequences that can be modified, retrieved, and generally can appear in computations. Creating a representation for the emergent action sequence efficiently captures the effects of the slow and expensive cognitive process applied to the task domain. Clearly cognitive and skilled behavior are related: we should be able to learn a skill by proceeding through a stage involving cognition and feedback.

Our goal is thus a well-founded theory of skilled and adaptive visuo-motor behavior: for this we need a learnable, tunable, explicit representations for sequences. The particular type of skill addressed here involves a foveation sequence: a sequence of (*where*) locations or (*what*) features to which the system should apply high-resolution sensing. Skilled visual sequences may be found in industrial inspection, medical image interpretation, some photointerpretation, or habitual (eye movement) patterns during driving. Some skills proceed too fast for any sensory feedback, while other action sequences are modified in real time to suit the current situation. Formal models for visuo-motor skills are needed so they can be integrated with systems, *e.g.* [23], for continuously controlling the gaze of an active vision system. Explicit representations of foveation sequences have been proposed [22; 30], as has the idea of motor programs [34]. A foveal-peripheral dichotomy in the visual system almost implies the need for intelligent or skilled sensor management. A spatially variant sensory device can be created in several ways, including custom VLSI sensors, resolution pyramids, or simply by using two cameras with two different focal lengths. Computer vision research predicated on a fovea and periphery is beginning to appear [11; 10; 7; 35].

The solution we present here is a variation on the hidden Markov model (HMM). It models a *set* of sequential orders in which to gather evidence. The specific sequential order is determined by information in the image. We first show how to use HMMs to acquire visuo-motor skills. Using HMMs to generate output is unusual. We introduce Augmented HMMs (AHMMs) to allow the acquired skill to be modified by feedback from the current situation.

2.2 Fusing Low- and High-Level Skills

Although visual skills are often acquired by a slow cognitively-mediated learning phase, they are by definition a "low-level" phenomenon. We should like a mechanism to link more directly the "emergent" commands from a cognitive system and the explicit action sequences of a skill. A system that applies decision theoretic techniques to maximize expected utility is an example of a system generating emergent commands. We propose that when the necessary calculations can not be performed within real-time limits, an explicit behavioral sequence may be substituted. The challenge is to develop an explicit model with a limited form of the capabilities and features of the more sophisticated emergent system, and we

propose AHMMs for the task.

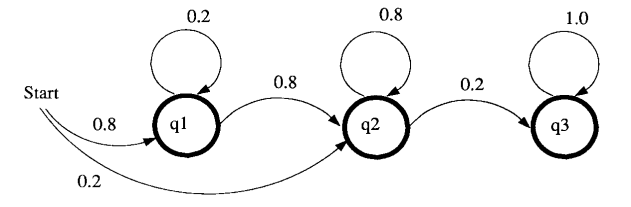
This general approach has already been followed in mobile robot navigation [14], using a small set of "strategies" (explicit action sequences), where a specific strategy is selected via a maximum expected utility criterion computed using a temporal Bayes net (TBN). The distinction between emergent and explicit models for behavior is also mentioned in [14], and interestingly the TBN structure is similar to the lattice structure used by the classic HMM algorithms in [24; 25].

The issue of merging skills and plans is of increasing importance. The basic idea of explicit sequences or "strategies" has been increasingly popular in AI, especially the planning literature, over the past several years. Cached plans or macros are stored action sequences that are applied in predetermined situations. Some reactive planning systems learn policy functions that associate an action with each world state. The space of legal transitions between world states can be viewed as a finite state transition diagram that is similar to AHMM models [33].

Below, we present the *what-where-AHMM* as an example link between emergent, high-level systems and skilled low-level systems. It is just one application of a general idea and technique using AHMMs: combine graph structures representing the execution sequences of two visual skills into a single coordinated skill (and an equivalent, but virtual, fused graph representation). In this case the result is a system that sequentially positions a camera to view a sequence of *desired features* (or objects) *in their expected locations* in the image. The system also incorporates visual feedback cues and a control scheme for verifying expectations using foveal image data.

3 Hidden Markov Models

The hidden Markov model (HMM) is an elegant formalism for classifying, generating, and learning sequences. For details on the following algorithms see the excellent tutorial by Rabiner [24; 25]. The hidden Markov model (HMM) is like a probabilistic finite state machine. State transitions have associated probabilities, and each state has a probability distribution that determines what symbols it outputs. The HMM models a sequence of symbols, called an observation sequence. The HMM is "hidden" because only the symbol sequence is observable: the internal state sequence producing the symbols is *not* observable.



v1 (left)	0.5	0.7	0.0
v2 (up)	0.5	0.3	1.0

Figure 1: Example of a simple HMM.

An HMM is formally defined as $\lambda = (A, \pi, B)$ with states $Q = \{q_1, ..., q_N\}$ and symbols $V = \{v_1, ..., v_M\}$. The probability of transitioning at time step t from state q_i to state q_j is given by $A = \{a_{ij}\}$ where $a_{ij} = P(q_j \text{ at } t+1 \mid q_i \text{ at } t)$. The initial state is determined by $\pi = \{\pi_i\}$ where $\pi_i = P(q_i \text{ at } t = 1)$. If the HMM is in state q_j it produces symbol v_k according to the probability $B = \{b_j(k)\}$ where $b_j(k) = P(v_k \text{ at } t \mid q_j \text{ at } t)$.

Figure 1 shows an example HMM. The state sequence it models tend to contain a short subsequence of q1 states, a larger subsequence of q2 states, and q3 states until the end of the sequence. Assuming that symbols v1 and v2 are associated with "left" and "up"-ward (incremental) camera movements, and that the HMM remains in state q1 for several steps, then it tends to move the cameras to the upper left during that time.

The graph structure of an HMM may be arbitrary. Pre-structuring the graph is useful when something is known about the information that the graph will model. For example, a graph with a left-to-right flow, as in Figure 1, may be appropriate for certain non-cyclic sequences. The parameters of an HMM, including the a_{ij} parameters that determine the graph structure, are estimated from a set of training sequences. By setting the initial a_{ij} parameters of an HMM to zero, the learned graph can be restricted to certain types of structures. In practice the particular graph structure is related to performance – one approach is to start with a completely connected graph and let the weights on unimportant arcs go to zero with learning.

3.1 HMM Algorithms

HMMs have three associated capabilities, at least one of which is used in any application. They can classify sequences, they can generate sequences, and they can be learned from

examples. In each case, the basic idea is summarized below. Complete algorithms can be found in [24; 25].

The HMM is based on a finite state machine. Trainable probabilistic finite state automata have a long history. A recent survey, with an emphasis on policy functions, is [5]. Hidden grammars have apparently been proposed [2]. Characterization of the class of languages defined by an HMM, and some variants, has recently been reported [12]. Links between the HMM model and other models in the areas of neural networks and control theory have been studied recently [8; 19; 17].

Classification

Classification is the most common application. It is very common in the speech recognition community [24; 25]. In classification, a single HMM λ_i is associated with each possible class ω_i. A complete description of the HMM is assumed to be known, including the states Q_i, the set of possible symbols V_i, and the probabilities λ_i (and thus the graph structure). (See below for how λ_i is normally estimated from a set of training examples.) An observed sequence O is classified as the most likely class, according to $P(O \mid \lambda_i)$, as

$$\omega = \text{argmax}_i P(O \mid \lambda_i).$$

$P(O \mid \lambda_i)$ could be naively computed by enumerating every possible state sequence and computing the probability of the observation sequence in each case, but this requires exponential time. Fortunately an efficient dynamic programming algorithm exists for computing $P(O \mid \lambda_i)$, which essentially considers all possible state sequences.

Parameter estimation

The parameters λ_i for each HMM are estimated separately using a training set of examples and an iterative estimation algorithm. The user must first specify the states Q_i and the set of possible symbols V_i. The user also specifies an initial value for the probabilities $\lambda_i = (A_i, \pi_i, B_i)$, since the estimation algorithm is iterative and needs a starting point. In particular, the structure of the HMM transition graph can be constrained by setting the initial values of specific a_{ij} parameters to zero. (Zero values are unchanged in the iteration.) Finally, the user must supply a set of training examples for each class ω_i. The algorithm yields the final λ_i value.

Generation

Sequence generation from HMMs is not used in classical applications, but is crucial in the active vision context. As in classification, a complete description of the HMM is assumed to be known. Sequences may be generated using random numbers chosen from appropriate probability distributions in λ_i (first for a state transition and then for an output symbol). The length of a sequence is determined by a bound on $P(O \mid \lambda_i)$ or by using a fixed length. This randomness means the sequence is not "optimal" (say maximum likelihood) for a static, well-known world. However, the *visual feedback* of the AHMM introduced in the next section allows the sequences to adapt to the state of the world, and the randomness lends an exploratory nature to the behavior that allows adaptation to long-term changes.

3.2 Examples of HMMs in Computer Vision

The HMM is an elegant probabilistic model for any sequence of symbols. Its parameters are learned by example. It can classify a sequence and it can also generate example sequences. It is particularly good at learning to model homogeneous, but variable length, subsequences and their boundaries within a larger overall sequence.

The idea of hidden states is particularly appealing to computer vision (and AI in general) since in many cases observations can be made of the world, but the underlying world structure that explains those observations is not directly observable. Rosenschein's "situated automata" proposal contains a similar idea [27]. In some cases world structure may *theoretically* be observable, but available resources may constrain or limit perceptual ability. Examples of limited perception include the fovea/periphery organization of the human eye, selective attention in humans, and the limited sensing used to keep state descriptions small in reinforcement learning applications [33].

HMM performance is sensitive to the number of states in the HMM transition graph and their initial connectivity. A priori knowledge or experimentation is necessary to achieve the best results.

Classification

The HMM can fill the role of any standard pattern classification algorithm. The main choices to make are the application domain and the particular features used for the HMM

symbols. One-dimensional texture profiles were classified in [15]. Written characters and 2-D shapes have also been classified: Two-dimensional shapes can be described using a sequence of curvature values, and the "Ring HMM" can model cyclic sequences [20; 16].

Generation

HMMs have rarely been used to generate sequences. Since HMM sequence generation is our main topic, we defer discussion and examples to Sections 4 and 5.

Structure Learning

In some cases, interesting chunks of the world can be reasonably represented by graph structures (*e.g.* finite state worlds for mobile robot applications [21]). We assume that the observations or experiences of an active vision system in the world can thus be modeled as sequences isomorphic to paths through the relevant graph structure. The problem is to recover the structure of the world from the sequences of inputs. In the remainder of this section we give tutorial applications of HMMs to high-level vision. The first example does aspect graph learning from view sequences, the second uses more complex input to learn a (quasi) finite machine for control.

The familiar feature aspect graph [18; 32] is a way to represent the varying appearance of a 3D object as observed from different viewpoints. The nodes of the graph represent the regions corresponding to the distinct "principal views" of an object, and the links give the adjacency relations between the regions. An active computer vision system can observe objects from varying viewpoints. The problem is to learn aspect graph representations of objects from observation sequences. One approach is described in [29]. We present a solution using an HMM.

Figure 2 shows a simple prism shaped object and one possible aspect graph for it. The object has five faces, labeled F, B, L, R and X for front, back, left, right and bottom respectively. A node in the aspect graph (a view) is labeled by the list, in alphabetical order, of the labels of faces visible from that viewpoint. We assume that the faces are visually distinguishable, and that this vision problem is solved. Thus an observation is a sequence of symbols for sets of visible faces, such as (BR,R,FR,BFR...).

A training set of 800 such view symbol sequences, each of 16 symbols, was synthesized for the object of Figure 2 by choosing a random node in the true aspect graph as the start of

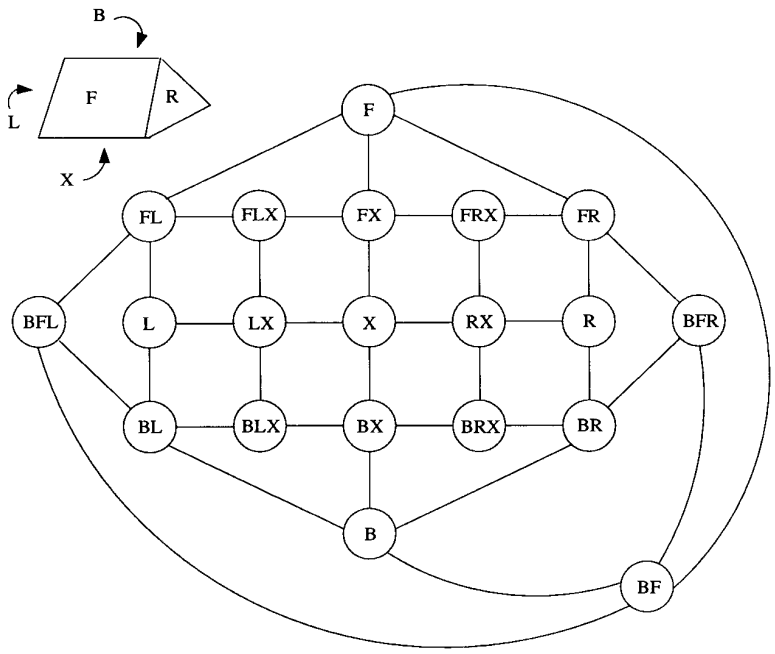


Figure 2: An aspect graph for a prism shaped object. All arcs are bi-directional.

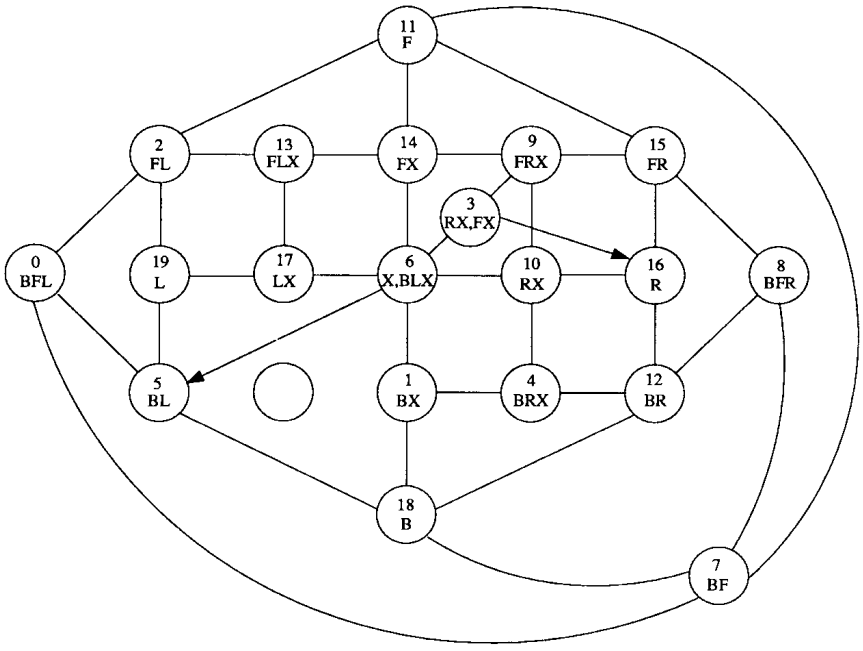


Figure 3: Learned aspect graph. A link without arrows is bi-directional.

a random walk through the graph. An HMM was trained using four iterations through this data set. The initial HMM graph structure contained 20 states and was fully connected. The graph structure learned by the HMM is illustrated in Figure 3. The circle drawn for each node contains a number, which is the state's index number, and a list of view labels. Links in the HMM are directional (although links in the aspect graph are bidirectional). The graph shows a bidirectional link when the HMM learned both directional links, but a uni-directional link is also possible.

In this example the HMM has the same number of nodes as the true graph. When we started the HMM learning with a larger number of states, a node in the real aspect graph was often represented by one, two, or more nodes in the learned graph. Further, some nodes in the real aspect graph were still not represented. Reducing the number of states in the HMM graph produces more interesting results: the HMM merges "neighboring" states (in terms of the original graph adjacency relations). This forced generalization of nodes is a grouping of smaller components in the scene – in fact it is a method of segmentation loosely related to graph partitioning with interesting properties (which we have not yet explored.)

Predictably, longer training sequences yield better quality learned graphs. Of course, random walks through the training graph are far from optimal: Using a small set of sequences with a more directed, non-random motion of the viewing position, as might arise from a purposive observer, gave much better results. Even better but not implemented is the idea of using the helpful information available to the observer about the observer's movement between observations. The observation sequences were assumed to be perfect. Sequences with noise or dropout are certainly not fatal to HMMs, which are robust tools in speech recognition, but this example does not show that. (Section 5.3 addresses the question of robustness of HMMs and AHMMs to noise in the visuo-motor skill domain.) Similarly, the observable sets of features from different viewpoints need not differ — the back face may look exactly like the front, for instance. The learning algorithm will use context (adjacent symbols) in the sequence to create separate nodes in the learned graph for such duplicated feature sets in the aspect graph.

The second example is similar to the first in that a graph is learned from sequences. Here the graph is a "Quasi Finite State Machine" (QFSM) that serves as a control structure in a multi-representational computer vision system [6]. It defines when the representation at one level merits an attempt to construct a more detailed representation at the next level.

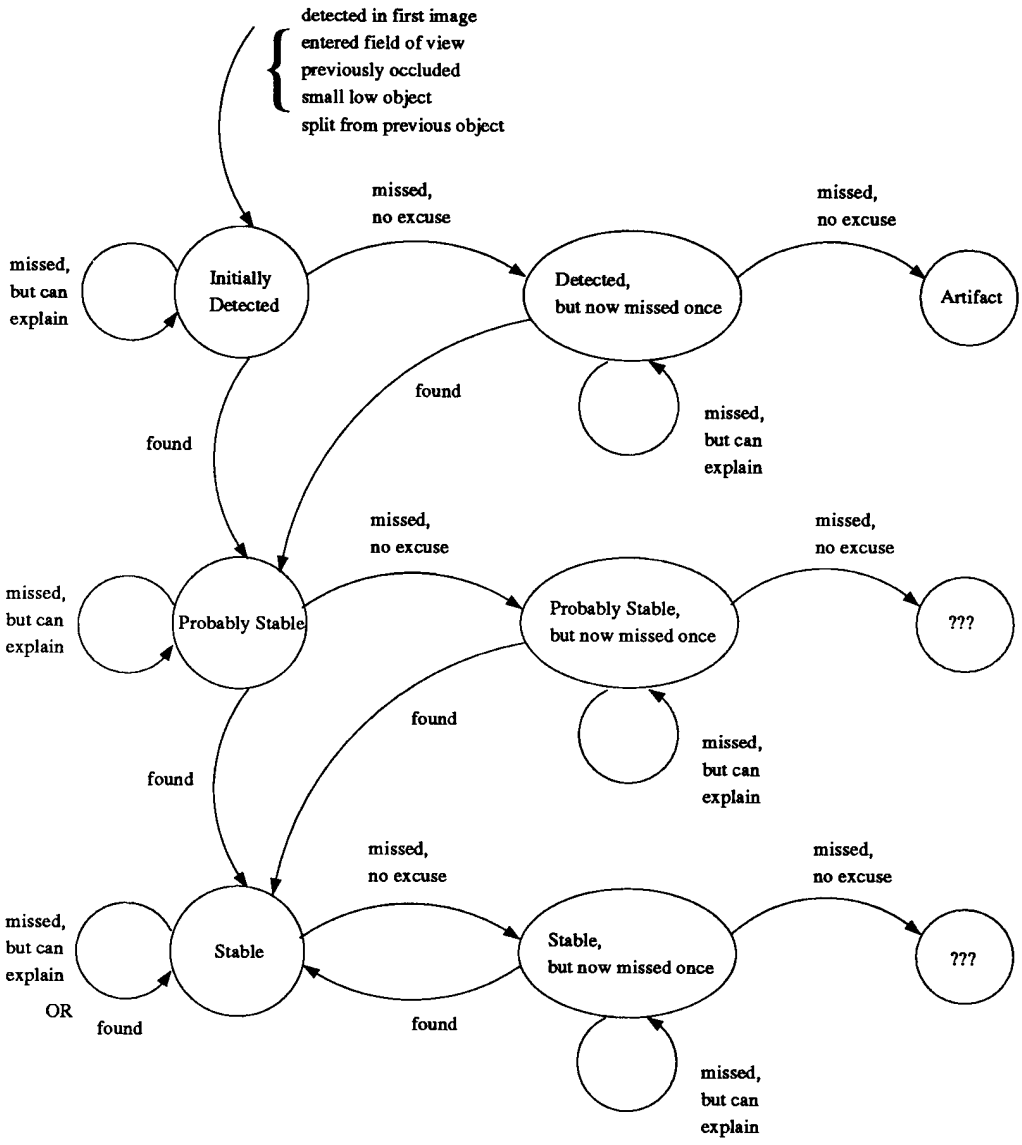


Figure 4: True QFSM.

Figure 4 shows an example of a QFSM, with states corresponding to varying degrees or types of stability (certainty) in the representation of the object being constructed.

The QFSM model contains two types of symbols: A "report" from a vision module both provides its result and also an explanation of the results (*e.g.* the object was missed in the last image frame but the module has additional information that can reasonably explain why it was missed). A "world state" describes the stability of the object's overall representation. The QFSM models a sequence of alternating "reports" and subsequent opinion about the current "world state". Sequences of this sort might be obtained by running the vision module to generate the "reports" and asking a human to provide subjective evaluation of the current state of the object's representation. Bobick and Bolles applied their experience with their system to construct their QFSM by hand.

After a training regime similar to that for the aspect graph, the graph structure learned by the HMM was that of Figure 5. The circle drawn for each node in this figure contains an integer number, the index number of the node in the HMM graph. Two or more numbers indicate that several nodes learned essentially the same function. The learned graph makes efficient use of nodes with "report" labels. For example, a "found" link is required from node 23 to node 7/26, and another "found" link is required from node 24 to the same destination. These two labeled links are produced using only a single node, 4, with the "found" label.

These examples suggest that HMMs are flexible and general tools for learning graph structures from sequences. There are several open questions and opportunities for research to build robust learning systems based on graph-learning algorithms — we have tried to hint at the possibilities.

4 Augmented Hidden Markov Models

We want to develop models for visuo-motor skills, specifically skills for controlling camera movements to point a high resolution sensor at selective areas of a scene. We shall use the HMM as a tool to learn, represent, and generate camera movement sequences, one HMM for each different skill behavior. To couple sequence generation to scene information, we introduce the Augmented HMM (AHMM).

An HMM encodes several different sequences, since there are several possible paths

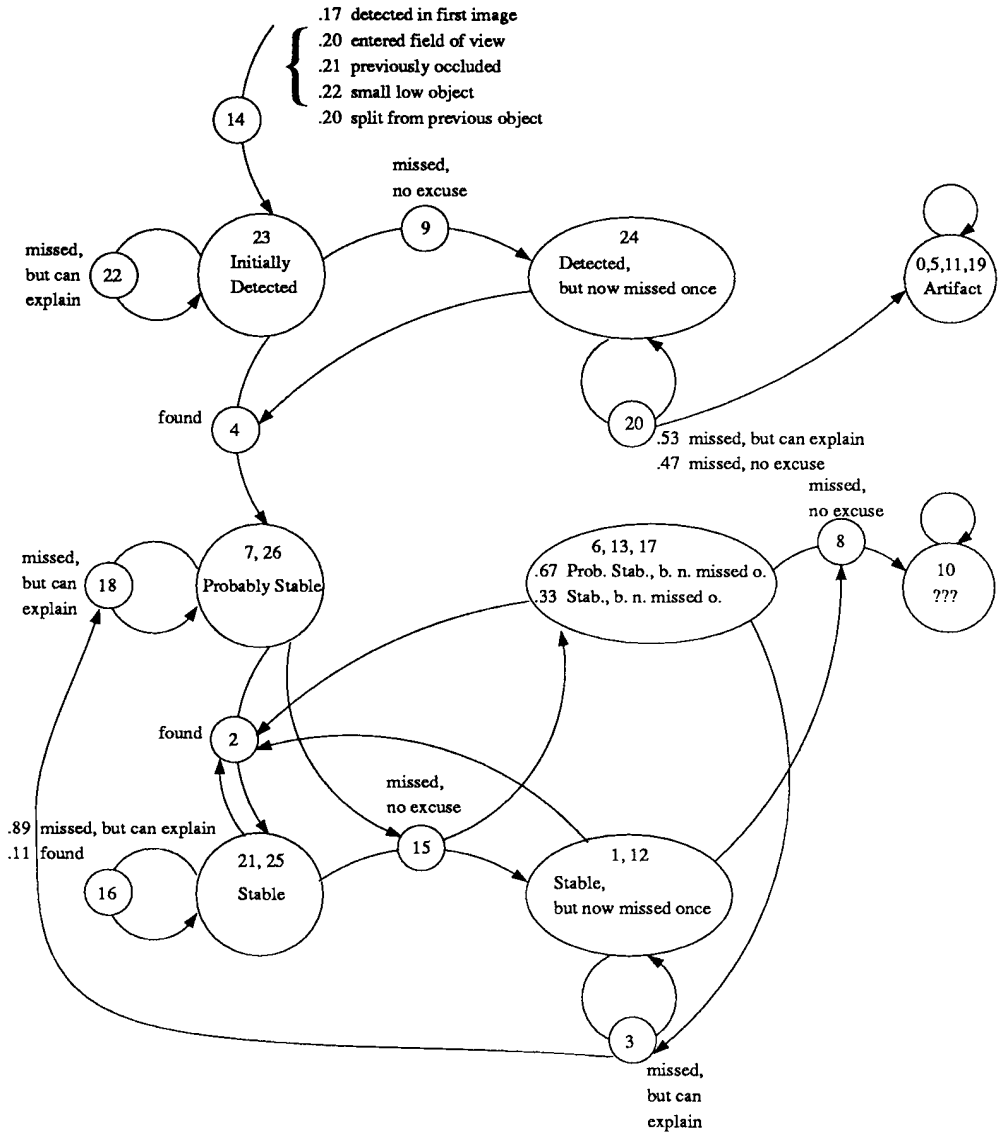


Figure 5: Learned QFSM.

through the HMM's probabilistic transition graph. The choice of a specific path should be affected by scene information. The AHMM treats scene information as visual feedback, specifically the visual feedback is considered as a prediction of the next symbol in the sequence and that prediction modifies the AHMM's future behavior. Here we only consider prediction one time step into the future.

The AHMM has parameters that vary as a function of time, denoted as $\lambda^{t+1} = (A^{t+1}, \pi, B^{t+1})$. Assume that the AHMM is at time step t, that it has already output a sequence of symbols $O_1, ..., O_t$, and that the current state is q_i. The feedback symbol S_t is available, where the value of S_t is v_k. S_t is used to modify λ^t into λ^{t+1}, then the AHMM uses λ^{t+1} to generate the next symbol, O_{t+1}. Obviously the choice of the next generated symbol depends partially on the few most recent feedback symbols. The λ^t parameters also slowly decay over time to their original values. So as long as consistent feedback symbols are available, their effect on the AHMM parameters will endure, but eventually the parameters return to their original values. The initial state probabilities π do not vary in time since a feedback symbol is not available at time $t = 0$. The equations for computing λ^{t+1} are summarized below.

A weighting factor. The equations for modifying the AHMM parameters use three key values: i, k, and w_j^t. The value i is determined from the state q_i of the AHMM at the current time step, t. The value k is determined from the value of the current feedback symbol, $S_t = v_k$. The values for i and k are known and will be assumed in all the equations below.

w_j^t is a probabilistic weight computed from i and k: w_j^t is the probability of being in state q_j at time $t + 1$, given the information that the AHMM is in state q_i at time t and will output symbol v_k at time $t + 1$.

$$w_j^t = \frac{a_{ij}^t b_j^t(k)}{\sum_{l=1}^{N} a_{il}^t b_l^t(k)} \qquad 1 \leq j \leq N. \tag{1}$$

The current feedback symbol (S_t) is assumed to be a prediction of the next output symbol (O_{t+1}), so w_j^t provides an indication of how consistent the immediately possible state transitions are with the current feedback, and it can be used to bias the state transition probabilities.

Modification of a_{ij}^t. The new values for the transition probabilities $\tilde{a}_{ij}^{t+1}$ that are most consistent with the feedback are $\tilde{a}_{ij}^{t+1} = w_j^t$. As in reinforcement learning, it seems desirable

to parameterize the rate of modification of a_{ij}^t from slow change to immediate replacement. Therefore only a fraction, $r_f w_j^t$, is mixed with the current value. The new equation is

$$\tilde{a}_{ij}^{t+1} = r_f w_j^t + (1 - r_f) a_{ij}^t \qquad 1 \leq j \leq N \tag{2}$$

where r_f $(0 \leq r_f \leq 1)$ is a modification gain. Larger gain values emphasize the feedback modified transition probability over the original probability.

Modification of $b_j^t(l)$. The emission probabilities in the AHMM must be updated for each state q_j that can be reached in one time step from the current state q_i. The equations for the updated emission probabilities, denoted $\tilde{b}_j^{t+1}(l)$, and already incorporating a mixing gain, are as follows.

$$\tilde{b}_j^{t+1}(l) = \frac{e_j^t(l)}{\sum_{m=1}^{M} e_j^t(m)} \quad 1 \leq j \leq N, \quad 1 \leq l \leq M \tag{3}$$

$$e_j^t(l) = s_f w_j^t d_j^t(l) + (1 - s_f) b_j^t(l) \tag{4}$$

$$d_j^t(l) = \begin{cases} 1.0 & \text{if } l = k \\ 0.0 & \text{otherwise.} \end{cases} \tag{5}$$

The denominator in equation (3) ensures that $\tilde{b}_j^{t+1}(l)$ is a valid probability. The modification gain is s_f $(0 \leq s_f \leq 1)$, where small gain values emphasize the original emission probability over the feedback modified ones. The key term in these equations, the one contributed by the feedback, is $w_j^t d_j^t(l)$.

Time decay. Equations (2) and (3) are the basis for modification at any instant in time. Since these modifications should only be maintained as long as they are justified by feedback information, the modifications are made to decay in time as follows

$$a_{ij}^{t+1} = r_d \tilde{a}_{ij}^{t+1} + (1 - r_d) \hat{a}_{ij} \qquad 1 \leq j \leq N \tag{6}$$

$$b_j^{t+1}(l) = s_d \tilde{b}_j^{t+1}(l) + (1 - s_d) \hat{b}_j(l) \quad 1 \leq j \leq N, \quad 1 \leq l \leq M \tag{7}$$

where $\hat{a}_{ij}$ and $\hat{b}_j(l)$ are the original values, which do not change over time. The decay gains are r_d and s_d $(0 \leq r_d, s_d \leq 1)$. Small decay gains cause the probabilities to decay quickly to their original values. These equations give the final values for a_{ij}^{t+1} and $b_j^{t+1}(l)$.

Multiple feedback signals. The AHMM can easily be extended if several different feedback symbols are available from different feedback sources. The feedback symbols at time t are denoted by a *set* $\mathbf{S_t}$. For example, this set could contain either simultaneous peripheral and foveal feedback symbols, or multiple peripheral feedback symbols. The updating equations are similar to those above [26].

Figure 6: Smooth path movement sequence. Parameters: $\alpha = 0.3$ and $\beta = 0.4$. Oblivious path (white), peripheral (gray) and peripheral-foveal (black) modified paths.

5 Skilled Camera (Eye) Movements

The HMM and AHMM models can now be applied to skilled camera movements. Here we examine the case when HMM symbols specify locations in the image. Experiments are presented with two types of location symbols to produce either smooth or discontinuous (saccadic) movements. Visual feedback is based on a domain-independent saliency image. The next section will examine generated symbols and feedback that index objects by feature vector.

The camera movement examples are implemented on the Rochester robot head and its associated image processing hardware [9]. Computer control of the two cameras on the head permits individual camera pans, a shared tilt, and either smooth path or saccadic movements. A single camera is used for these experiments. In this work the head and background remain fixed and only the pan and tilt facilities of the head are used.

Figure 6 shows a typical experimental scene: a table top with a variety of objects. All images show a low resolution peripheral image (128x128, zoomed 4x), with a central high resolution foveal component (also 128x128). The graphics superimposed on all figures

illustrate the points fixated when the camera executes a movement sequence. The relation between image coordinates and camera pan and tilt coordinates only depends on the camera lenses, is easily determined by a 2-D calibration procedure, and is stored in a table of useful system constants.

We begin by using the HMM for camera movements, follow with a similar AHMM application, and conclude with some observations about the robustness of AHMMs to noise either in the training examples or in the visual feedback.

5.1 Using HMMs

Smooth Path Movement

In smooth path movement, the HMM symbols are $v_i \in [0, 7]$, the eight chain code ("compass point") directions. The camera rotates a fixed increment so that the image shifts by an increment in the given direction. Here, the camera (or attentional) sequence is a relatively smooth path, of the sort arising in contour following, doing vision through a reduction tube, or in some other situations [31].

Assume we have trained a HMM somehow (monitoring human eye movements, monitoring the emergent sequence output of a cognitive process, or drawing paths with a mouse). If the HMM is then used to generate a sequence (exercise its skill), the sequence will be *oblivious* to the data in the image. Let the oblivious sequence of symbols normally generated by this HMM be $O = O_1, ..., O_T$. Before the camera moves, this sequence defines a path through the peripheral image. A very simple modification of the HMM (simpler than the full AHMM) makes the originally-learned sequence output sensitive to the data in the current image. This is useful if the images vary a little over time. We introduce low-level, non-cognitive, skill-modifying feedback cues based on local maxima of a simple "saliency image" – the equally weighted sum of five features derived from the Sobel edge operator and the grayscale variance. A feedback symbol, S_i, is simply the direction towards the local maximum of the saliency data at time step i. Generally, of course, any other kind of feedback cue can be used.

The foveal and peripheral images are used to compute a peripheral feedback cue $S_i^{(p)}$ and a foveal cue $S_i^{(f)}$. First, a periphery modified path, with elements $M_i^{(p)}$, is generated

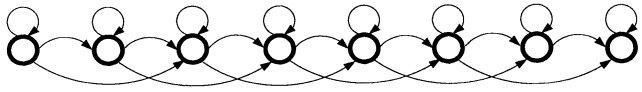


Figure 7: HMM graph for smooth path movement experiment.

according to

$$M_i^{(p)} = \begin{cases} S_i^{(p)} & \text{with probability } \alpha \\ O_i & \text{otherwise.} \end{cases} \tag{8}$$

Where $S_i^{(p)}$ is a symbol representing the peripheral feedback cue at index i in the sequence. Then, foveal modification is performed according to

$$M_i^{(f)} = \begin{cases} S_i^{(f)} & \text{with probability } \beta \\ M_i^{(p)} & \text{otherwise} \end{cases} \tag{9}$$

resulting in the final sequence $M^{(f)} = M_1^{(f)}, ..., M_T^{(f)}$. $S_i^{(f)}$ is a foveal feedback cue. Note that the cameras must actually be moved to make the appropriate foveal data available to compute $S_i^{(f)}$. The parameters α and β vary between 0.0 and 1.0, and regulate the amount of influence from the peripheral and foveal feedback sequences, respectively. In each case, a parameter value of 0.0 completely ignores the feedback data, and a value of 1.0 ignores the HMM-produced value and tracks the immediate image data. This variable tradeoff between prediction and input data is reminiscent of a steady state estimation filter.

Each $S_i^{(\cdot)}$ symbol is the chain code direction towards the local maximum of the respective saliency image. We condition the path (encourage smooth progress through the image) by using a neighborhood function for the local maximum that emphasizes saliency points in the direction of the path and away from the current location.

The initial HMM graph is shown in Figure 7. The HMM was trained on a set of 30 paths drawn on the image with a mouse (for an auto-training version, see the next section). An oblivious path generated by the resulting HMM is shown in white in Figure 6. Such a path might correspond to knowing the desired object is normally kept on the left side of the desk. Peripheral image data modified the oblivious path, resulting in the path shown in gray. Here the peripheral data keeps the path from overshooting the stuffed animals. However, it does not effectively pull the path closer to either the soda cans or the pile of small parts, as would be preferred. When the HMM is run using foveal data modification as well as peripheral, the foveal saliency data attracts the latter half of the path to the small pile of parts (black path).

Figure 8: Saccadic movement sequence generated using HMM which was trained on examples produced by "other" algorithm. Oblivious path (white) and periphery modified path (gray).

Saccadic Movement

Another type of camera movement sequence is saccadic, or point-to-point across large angles at high velocity. The HMM symbols are $v_i = (x, y)$ where x and y are coarsely quantized. The movements are in a fixed coordinate system (alternatively they could be relative to the current camera location). Feedback is determined from the maximum of the peripheral saliency image in the neighborhood of the target. The camera movement is made to a maximum computed after smoothing with a Gaussian filter centered on the target.

This time, the HMM learned the emergent behavior generated by an algorithm similar to that in [13], which produced 20 training examples. The control algorithm iteratively fixates the point in the image with a maximum saliency value, zeroes out the local saliency around that maximum, and then goes to the next largest maximum, until five fixations are made. The initial graph of the HMM is like that of Figure 7 only with four nodes. (Our saccadic movement experiments used shorter sequences than our smooth path experiments, so we chose to use four nodes rather than eight.) Figure 8 shows an oblivious sequence (white) generated by the trained HMM, and the sequence modified using peripheral salience (gray).

Figure 9: Smooth path movement sequence, small foveal feedback gain. Gains are: $r_{f,(p)} = s_{f,(p)} = 0$, $r_{f,(f)} = s_{f,(f)} = 0.2$, $r_d = s_d = 0.9$.

The modified sequence has been drawn to the locally more interesting areas of the image.

5.2 Using AHMMs

Smooth Path Movement

The AHMM can be used in a straightforward manner to generate smooth path movements, using multiple feedback symbols (simultaneous peripheral and foveal feedback). The trained AHMM parameters from Section 5.1 were used, except that any zero valued probabilities were changed to have very small non-zero values. These AHMM parameters served as the initial (fixed in time) version of the model. The saliency image used in these experiments was simply the Sobel edge magnitude image.

Figure 9 shows an example of weak foveal feedback, with foveal gains of $r_{f,(f)} = s_{f,(f)} = 0.2$, and no peripheral feedback. Recall that these gains are used in equations (2) – (5). The path is drawn towards the stuffed animals. Figure 10 shows dual feedback. The peripheral gains are 0.2 and the foveal gains are 0.6. The path is immediately drawn to the detergent boxes, however the effect of the AHMM's trained behavior, in combination with

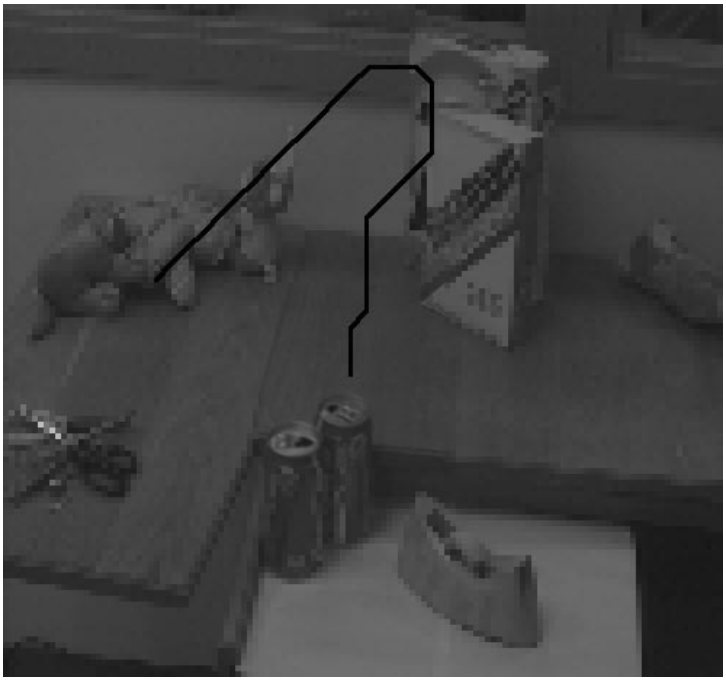


Figure 10: Smooth path movement sequence, dual foveal and peripheral feedback. Gains are: $r_{f,(p)} = s_{f,(p)} = 0.2$, $r_{f,(f)} = s_{f,(f)} = 0.6$, $r_d = s_d = 0.9$.

foveal/peripheral feedback, eventually gains control and the path resumes its course to the lower left towards the stuffed animals.

Saccadic Movement

The AHMM can also be applied to saccadic camera movements, operating in two stages during each time step. First it generates a preliminary output symbol (target location). Then a Gaussian smoothing function is centered on the target location. Local maxima are detected and used as multiple feedback to modify the AHMM parameters. Last, the AHMM generates the final version of the output symbol for the time step. The feedback modifications in the AHMM result in a form of (application independent) averaging of the feedback symbols and the oblivious output symbol. Saccadic movements do not use foveal feedback because by definition saccades are large foveal movements.

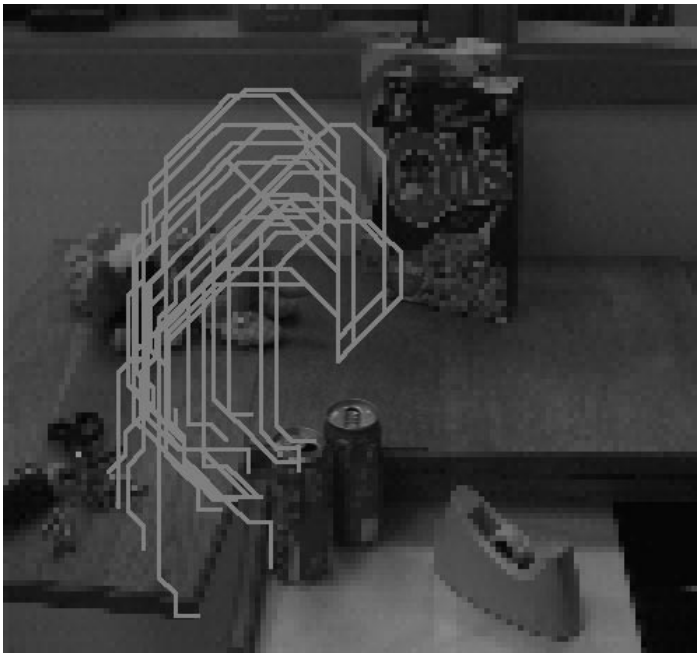


Figure 11: The 30 training sequences for the smooth path movement examples.

5.3 Robustness to Noise

The AHMM is robust to variations and noise either in the set of training examples or in the visual feedback signal. The HMM and AHMM used in the smooth path movement examples above were both trained with the same set of 30 training sequences, each drawn with a mouse. Figure 11 shows all 30 training sequences overlaid on a single image. The HMM parameter estimation algorithm easily deals with the considerable variation in these 30 sequences and learns a reasonable average of the training set. The estimation algorithm can just as easily deal with small "noise" variations. Speech understanding, the most successful application for HMMs, is well known both for the large variety in speech signal sequences and for significant amounts of noise.

The AHMM feedback mechanism can also cope with a considerable amount of noise. As an example consider Figure 12 which illustrates the feedback signals during a smooth path movement sequence. At each step in the sequence (shown in pure white) two small lines protrude in the directions of the peripheral (off-white) and foveal (gray) feedback signals. In some cases both lines protrude in the same direction (shown as a black line) or are in the same direction as the generated sequence (no lines protrude). The feedback signal is clearly

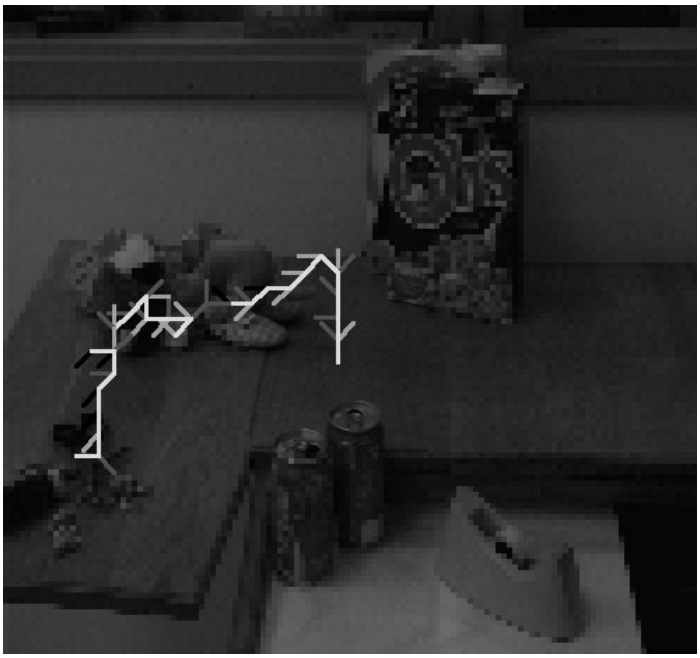


Figure 12: The feedback signals during a smooth path movement sequence. The sequence is pure white. At each step in the sequence two small lines protrude in the directions of the peripheral (off-white) and foveal (gray) feedback signals. In some cases both lines protrude in the same direction (shown as a black line) or are in the same direction as the generated sequence (no lines protrude). Dual foveal and peripheral feedback was used with the gains: $r_{f,(p)} = s_{f,(p)} = 0.3$, $r_{f,(f)} = s_{f,(f)} = 0.3$, $r_d = s_d = 0.9$.

noisy, but the AHMM uses it to generate a final sequence that is correctly attracted toward and passes through the desired objects in the scene. Compare the final feedback modified path to the variety of training examples in Figure 11.

6 Fusing "What" and "Where" Knowledge

We introduce the *what-where-AHMM*, which uses the AHMM to combine two independently learned HMMs. It is just one application of a general idea and technique: combining graph structures representing the execution sequences of two visual skills into a single coordinated skill (and an equivalent, but virtual, fused graph representation). In this case the result is a system that sequentially positions a camera to view a sequence of *desired features* (or objects) *in their expected locations* in the image. The system also incorporates visual feedback cues and a control scheme for verifying expectations using foveal image data. Movement sequences of this sort form part of an attentive vision system that has the goal of sequentially acquiring evidence to classify an object or situation in the scene.

There exists a variety of evidence in the human visual system that "what" and "where" information is processed separately (*e.g.* [28]). The what-where-AHMM fuses the knowledge in independently learned "what" and "where" graph structures. It is easier to learn the knowledge structures for the two subproblems separately. The number of states will be smaller, since a single graph for the overall problem is essentially a cross product of the two (smaller) subproblem graphs. In turn, fewer states require a smaller number of transition and emission probability parameters. Finally, it is likely that the knowledge about the subproblems can be combined in other ways or otherwise be used to solve problems — keeping the what and where representations as primitives allows them to participate in other combinations.

The what-where-AHMM application is only one use for the general hybrid (A)HMM algorithm that underlies it. In fact the algorithm implements a form of *graph matching*. One general method of graph matching [4] constructs an association graph whose nodes are pairs of nodes of the graphs to be matched and whose arcs represent compatibilities between these node pairings. If all matches are compatible the association graph has as nodes the cross product of the node sets. Largest maximal cliques in this association graph represent best (largest compatible) matches between the two graphs. The what-where-AHMM can be

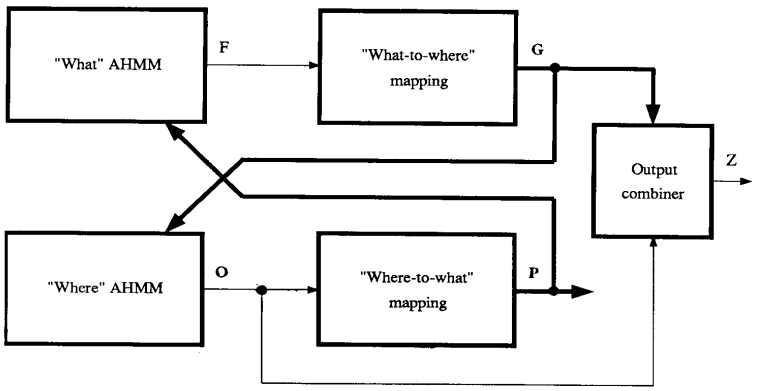


Figure 13: The "what-where" version of the AHMM. A Darker signal path denotes a set rather than a single signal value.

considered as producing a probability-weighted cross product of the node sets of the what and where graphs. It derives a match between the two graphs that associates object labels (what nodes) with their location (where nodes). Thus the feedback construction is a way of producing associative matches between the respective knowledge representations.

6.1 The What-Where-AHMM

The what-where-AHMM has a "what" part, a "where" part, and an output combiner (Figure 13). The what-part contains two stages. The first stage is an AHMM whose output symbols F_t, called *what-symbols*, are feature vectors intended to describe an object or characteristics of objects. Such feature vectors are assumed to have been computed for each pixel in the (peripheral) image. The second stage of the what-part performs a "what-to-where" mapping, meaning that it maps a feature vector into the set $\mathbf{G_t}$ of camera movement commands, called *where-symbols*, that would cause those locations to be foveated (centered in the image).

If each $\mathbf{G_t}$ contains exactly one element, the output sequence will fixate the desired objects in the scene. Each $\mathbf{G_t}$ does not generally contain exactly one element, so some method must be developed to select among the choices. One option is to use a where-AHMM to help pick among the choices. In fact, the what-AHMM can be made to help the where-AHMM with its *own* choices.

The where-part contains two stages, similar to those in the what-part. First it has an AHMM, which outputs a sequence of where-symbols O_t. Secondly it has a "where-to-what"

mapping that determines for each where-symbol O_t the location in the current (peripheral) image it corresponds to, and outputs the set of feature vectors $\mathbf{P_t}$ in that local area of the image. The AHMM in the where-part uses as feedback a sequence of sets of where-symbols $\mathbf{G_t}$, which is the output of the what-part.

Finally, the output combiner determines the overall output of the what-where-AHMM. The overall output at time step t is Z_t, a where-symbol (*i.e.* a camera movement command), selected as the element of the set $\mathbf{G_t}$ that has the smallest distance to the symbol O_t.

The what-where-AHMM operates as follows. At each time step, each of the two parts produces a set of feedback symbols that reflects its own preference for action. Each then updates its own preferences taking the other's into account, and then generates its own final preference for action at that time step. The set of final preferences is reduced to a single output symbol by the output combiner.

6.2 Limited Perception and Foveal Verification

The what-where-AHMM models a sequence of high resolution feature vectors. High resolution feature vectors of points in the scene are only available for the area where the high resolution fovea of the camera is pointing. Features from the low resolution peripheral image data are always available, so feedback in the what-where-AHMM necessarily uses them. The "what-to-where" and the "where-to-what" mappings in the what-where-AHMM also convert from high-to-low and low-to-high resolution feature vectors respectively. For example, in the face recognition experiments presented later, *nose* and *ear* are indistinguishable in low-resolution images, so the features detectable in the periphery are *eye*, *noseEar*, and *mouth*. If the what-AHMM output is the what-symbol *nose*, while the what-where-AHMM receives three *noseEar* feedback symbols, then the what-where-AHMM points the camera at one of the *noseEar* features. (Usually, it will be the real nose since the where-AHMM biases it to look in the expected location for the nose.)

If it points the camera at one of the ears, it immediately gets high resolution foveal data that it is pointed at an *ear*. For these situations we have given the system a "foveal verification" mechanism: If foveation does not produce the expected what-symbol, a "verification failure" signal causes the where-AHMM to generate the next most likely where-symbol, which results in another foveation attempt for the desired what-symbol. Of course, this

verification mechanism also corrects for other erroneous predictions and actions caused by random exploratory behavior.

6.3 Experimental Results

Our experiments used a high-contrast stylized face image (Figure 14), again with the foveal and peripheral distinction. Overlaid across the top of the figure are low-level image analysis results for the peripheral and foveal images. The system uses binary blob shape analysis to classify the parts of the face. We are considering using unconstrained grayscale images, which would require more sophisticated feature extractors but would also motivate multiple representations. The following face parts are distinguishable in the periphery image: *eye*, *noseEar*, and *mouth*. When the fovea is positioned over a face part, it can be determined to be one of: *eye*, *nose*, *ear*, or *mouth*. A nose and an ear have roughly the same shape and can not be differentiated using the low resolution peripheral data, thus the *noseEar* symbol is required. This problem illustrates the use of verification control using high resolution foveal data.

A What-HMM

The HMM can model a sequence of features (shape numbers, graylevel histograms, ...) as well as a sequence of locations. Here, we assume a feature is a symbol representing either a person's face or the parts of a face. The people are denoted by the symbols CB and RR. The parts of a face are: *eye*, *nose*, *ear*, and *mouth*. These symbols are the output of the low-level vision module. Suppose an observer executes the following sequence:

$$(RR, eye, eye, nose, mouth, CB, eye, eye, nose, mouth, RR).$$

The observer is repeatedly looking back and forth between the two subjects, and when looking at either subject the observer is scanning over the parts of that subject's face. An HMM (initially fully connected) was trained on this sequence, yielding the graph shown in Figure 15. The graph knowledge structure efficiently contains a *single* subgraph that models the cycling among the face parts for a subject, and the graph also contains an outermost loop for cycling among the two subjects.

The HMM gratifyingly learns cyclic graph structures when the observation has indications of a cyclic nature. Here, the cyclic nature is suggested by repeating the beginning and

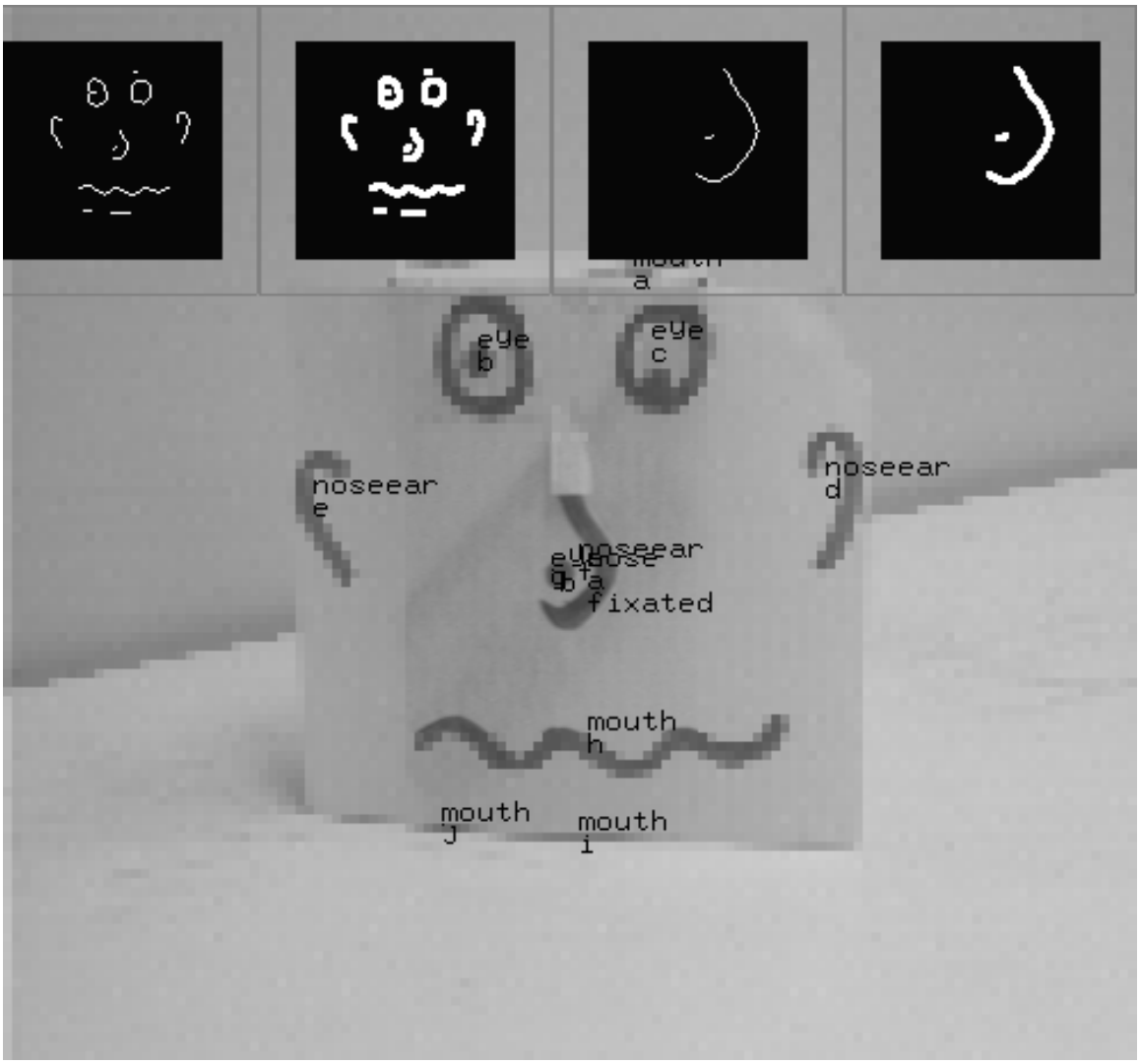


Figure 14: The face scene. Early processing results are shown at the top of the figure. The face features located using both foveal and peripheral image data are shown.

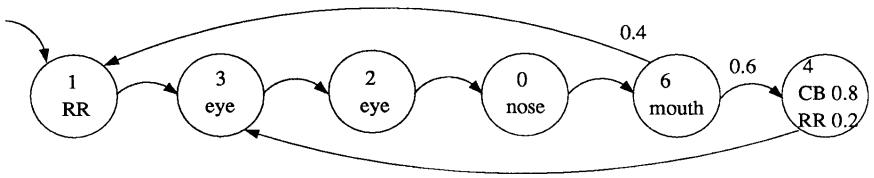


Figure 15: A learned graph for object ("what") indexing.

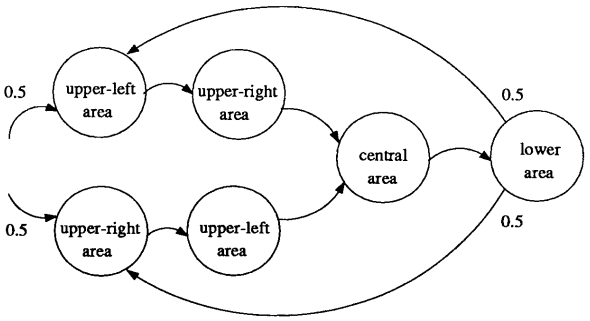


Figure 16: A learned graph for location ("where") indexing.

end of the observation sequence. More complicated application domains may require longer repeated subsequences, doubling the length of the sequence in the extreme case. A special HMM parameter estimation algorithm that enforces cycles has recently been reported [20], but in practice simply "doubling up" each observation sequence produces the desired result.

A Where-HMM

The symbols used by this HMM are absolute image coordinates, (x, y), and each coordinate is quantized to 16 values over the range from 0 to 511. Each part of the example face has dimensions a few times larger than the coordinate quantization. A small training set of sequences was generated using a mouse. For serious domain-dependent research (such as face recognition) the training might involve such expert-provided guidance. More interestingly the face-recognition skill could be learned, as in Section 5.1, by training the HMM with emergent sequences from a cognitively mediated algorithm. Our training set contained sequences that cyclically examined the areas in the image in the following two orders: either (left-eye, right-eye, nose, mouth) or (right-eye, left-eye, nose, mouth). An HMM (initially fully connected) was trained on this data. The graph structure learned by the HMM is shown in Figure 16. The learned structure reflects both the cyclic nature of the training sequences, and the choice of examining the two eye areas in either order.

A What-Where-AHMM

The what-AHMM and where-AHMM were trained independently, essentially as described in the examples above. (The only difference is that the what-AHMM was trained with the face-part symbols, and not the people symbols, CB and RR.) In this experimental

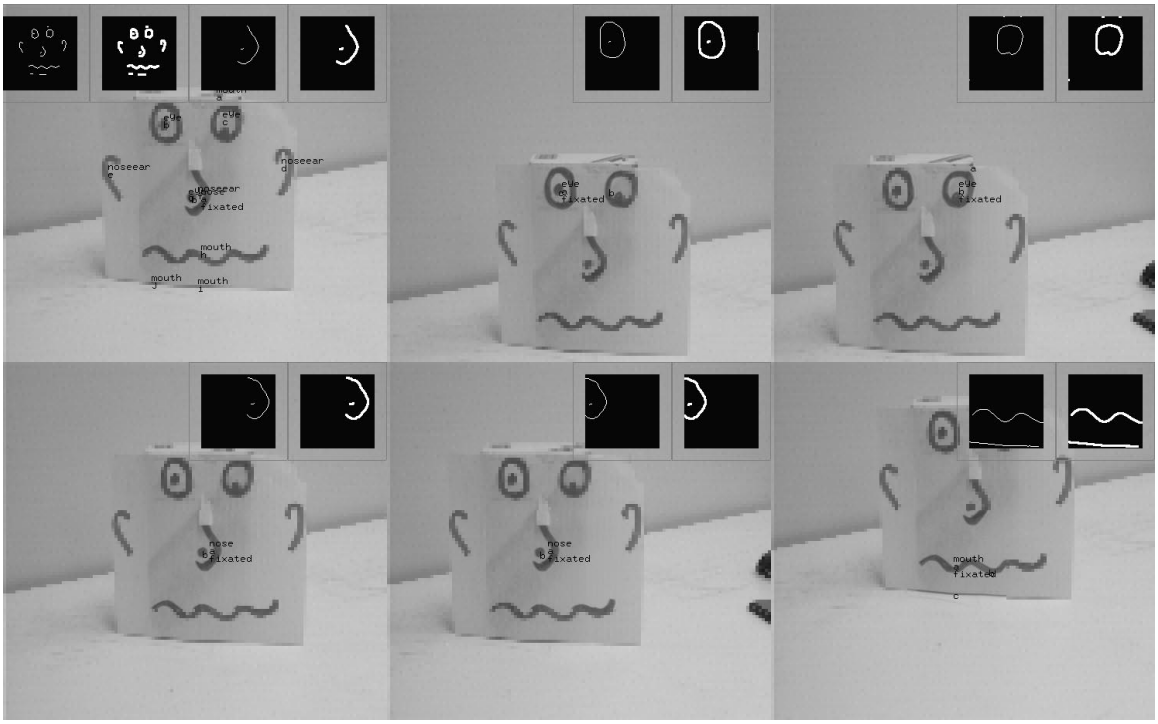


Figure 17: A viewing sequence determined by the what-where-AHMM system on the face scene (see text).

run, the system performed the following actions (Figure 17, scanning left to right, top to bottom).

1. Decide to look for an *eye* in the upper left. Look there. An *eye* is verified using the foveal data.

2. Decide to look for an *eye* in the upper right. Look there. An *eye* is verified using the foveal data.

3. Decide to look for a *nose* in the center. Look there. A *nose* is verified using the foveal data. (Recall that *nose* and *ear* can not be distinguished in peripheral images. If despite the "where" knowledge the system finds an ear while searching for an eye, verification fails and the control causes the system to look in the next most likely location for the desired object.)

4. Decide to look for a *mouth* in the center. Look there. A *mouth* is not found in the foveal data. (Occasionally the system looks in unusual places – this is a feature of its ability to adapt to slowly changing input scenes.)

5. Still looking for a mouth, the control again asks for a likely mouth location (based on peripheral data and previous training). The next most likely location indeed contains a *mouth*.

This foveation sequence is illustrative of the operation of a system whose purpose is to gather evidence about individual features, *e.g.* for a dynamic approach to face recognition.

7 Conclusions

Hidden Markov models can be useful for a variety of vision tasks. We have demonstrated applications in active vision: structure learning (HMM), sensori-motor skill acquisition and production (AHMM), and knowledge fusion (what-where-AHMM). A few other applications of HMMs to vision (mainly low-level) have recently been reported by others. We believe this formalism has further uses.

HMMs can incorporate a certain amount of prior knowledge in the form of the initial graph structure. They learn appropriate behavior sequences and accommodate to varying conditions during execution. The what-where AHMM incorporates some new ideas

in learning and knowledge fusion. At the same time the probabilistic self-structuring and adaptation capabilities of AHMMs leave their performance open to criticism: learning may involve many trials, graph structures are not guaranteed to be duplicated exactly, and outputs can be sub-optimal in any given situation. Also, the performance of the AHMM is heavily influenced by the nature (*e.g.* higher level feedback is better) and quality of the feedback signal.

These characteristics of HMMs and AHMMs have led us to begin investigating other approaches to the problem of visual resource allocation, in particular the "where to look next" problem. We have developed a system that produces emergent action sequences (both foveal and peripheral actions) using a maximum expected utility criterion with an augmented Bayes net or augmented influence diagram model. Its advantages are that it performs "optimally" under some criterion and that it incorporates well-understood information fusion mechanisms. However, it does not exhibit exploratory or self-structuring behavior.

Thus AHMMs raise several varieties of research questions. From an applications standpoint, how can the structures created by HMMs be used efficiently for object recognition, and how can several learned structures be used together? From a systems standpoint, how should reasoning (planning or inference) systems and skill-implementing, learning systems interact? We think progress in these directions will be particularly valuable in the animate vision systems of the future, which will increasingly interact with their environment and which will increasingly have to rely on efficient sensorimotor skills and effective decision-making.

Acknowledgements

We thank Dana Ballard, David Coombs, Steve Whitehead, Mike Swain and the referees for their helpful comments on drafts of this paper.

References

[1] R. Bajcsy. Active perception. *IEEE Proceedings*, 76(8):996–1005, 1988.

[2] J. K. Baker. Trainable grammars for speech recognition. In *97th Meeting of Acoustical Society of America*, 1979. Speech Communications Paper.

[3] D. H. Ballard. Animate vision. *Artificial Intelligence*, 48:57–86, 1991.

[4] D. H. Ballard and C. M. Brown. *Computer Vision*. Prentice-Hall, 1982.

[5] A. Barto, R. Sutton, and C. Watkins. Learning and sequential decision making. Technical Report 89-95, Department of Computer and Information Science, University of Massachusetts at Amherst, September 1989.

[6] A. F. Bobick and R. C. Bolles. Representation space: An approach to the integration of visual information. In *Proceedings: IEEE Conference on Computer Vision and Pattern Recognition*, pages 492–499, 1989.

[7] R. M. Bolle, A. Califano, and R. Kjeldsen. Data and model driven foveation. In *Proceedings: IEEE International Conference on Pattern Recognition*, pages 1–7, 1990.

[8] H. Bourlard and C. J. Wellekens. Links between Markov models and multilayer perceptrons. In *Proceedings: Neural and Information Processing Conference*, pages 502–510, 1988.

[9] C. M. Brown. The Rochester robot. Technical Report 257, Department of Computer Science, University of Rochester, August 1988.

[10] R. A. Browse and M. G. Rodrigues. Propagation of interpretations based on graded resolution input. In *Proceedings: International Conference on Computer Vision*, pages 405–410, 1988.

[11] P. J. Burt. Smart sensing within a pyramid vision machine. *IEEE Proceedings*, 76(8):1006–1015, 1988.

[12] F. Casacuberta. Some relations among stochastic finite state networks used in automatic speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):691–694, 1990.

[13] J. J. Clark and N. J. Ferrier. Modal control of an attentive vision system. In *Proceedings: International Conference on Computer Vision*, pages 514–523, 1988.

[14] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150, 1989.

[15] X. Gong and N. K. Huang. Textured image recognition using hidden Markov model. In *Proceedings: IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 1128–1131, 1988.

[16] Y. He and A. Kundu. Planar shape classification using hidden Markov model. In *Proceedings: IEEE Conference on Computer Vision and Pattern Recognition*, 1991.

[17] A. Kehagias. *Approximation and Estimation of Stochastic Processes by Hidden Markov Models.* PhD thesis, Brown University, Division of Applied Mathematics, 1991.

[18] J. J. Koenderink and A. J. Doorn. The internal representation of solid shape with respect to vision. *Biological Cybernetics*, 32:211–216, 1979.

[19] S. Y. Kung and J. N. Hwang. A unified systolic architecture for artificial neural networks. In *Proceedings: IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1989.

[20] W. D. Mao and S. Y. Kung. An object recognition system using stochastic knowledge source and vlsi parallel architecture. In *Proceedings: IEEE International Conference on Pattern Recognition*, pages 832–836, 1990.

[21] M. C. Mozer and J. Bachrach. Discovering the structure of a reactive environment by exploration. *Neural Computation*, 2:447–457, 1990.

[22] D. Noton and L. Stark. Eye movements and visual perception. *Scientific American*, 224(6):34–43, 1971.

[23] T. J. Olson and D. J. Coombs. Real-time vergence control for binocular robots. *International Journal of Computer Vision*, 7(1):67–89, 1991.

[24] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *IEEE Proceedings*, 77(2):257–286, 1989.

[25] L. R. Rabiner and B. H. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, 1986.

[26] R. D. Rimey and C. M. Brown. Selective attention as sequential behavior: Modeling eye movements with an augmented hidden Markov model. Technical Report 327 (revised), Department of Computer Science, University of Rochester, April 1990.

[27] S. J. Rosenschein. Formal theories of knowledge in ai and robotics. *New Generation Computing*, 3(4):345–358, 1985.

[28] J. G. Rueckl, K. R. Cave, and S. M. Kosslyn. Why are 'what' and 'where' processed by separate cortical visual systems? a computational investigation. *Journal of Cognitive Neuroscience*, 1(2):171–186, 1989.

[29] M. Seibert and A. M. Waxman. Learning aspect graph representations from view sequences. In *Proceedings: Neural and Information Processing Conference*, pages 258–265, 1990.

[30] L. Stark and S. R. Ellis. Scanpaths revisited: Cognitive models direct active looking. In D. F. Fisher, R. A. Monty, and J. W. Senders, editors, *Eye Movements: Cognition and Visual Perception*. Lawrence Erlbaum, 1981.

[31] S. Ullman. Visual routines. *Cognition*, 18:97–157, 1984.

[32] N. A. Watts. Calculating the principal views of a polyhedron. In *Proceedings: IEEE International Conference on Pattern Recognition*, 1988.

[33] S. D. Whitehead and D. H. Ballard. Active perception and reinforcement learning. In *Proceedings: Machine Learning Conference*, pages 179–188, 1990.

[34] C. E. Wright. Controlling sequential motor activity. In D. N. Osherson, S. M. Kosslyn, and J. M. Hollerbach, editors, *An Invitation to Cognitive Science, Volume 2, Visual Cognition and Action*, pages 285–316. MIT Press, 1990.

[35] Y. Yeshurun and E. L. Schwartz. Shape description with a space-variant sensor. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11):1217–1222, 1989.