Learning temporal structure for task based control

Kingsley Sage, A. Jonathan Howell and Hilary Buxton
University of Sussex, Department of Informatics
Centre for Research in Cognitive Science, Brighton, BN1 9QH, UK
{khs20,jonh,hilaryb}@sussex.ac.uk

Antonis Argyros
Institute of Computer Science, Foundation for Research and Technology Hellas
PO Box 1385, GR 711 10 Heraklion, Crete, Greece
argyros@ics.forth.gr

May 24, 2005

Abstract

We present an extension for Variable Length Markov Models (VLMMs) to allow for modelling of continuous input data and show that the generative properties of these VLMMs are a powerful tool for dealing with real world tracking issues. We explore methods for addressing the temporal correspondence problem in the context of a practical hand tracker, which is essential to support expectation in task-based control using these behavioural models. The hand tracker forms a part of a larger multi-component distributed system, providing 3-D hand position data to a gesture recogniser client. We show how the performance of such a hand tracker can be improved by using feedback from the gesture recogniser client. In particular, feedback based on the generative extrapolation of the recogniser's internal models is shown to help the tracker deal with mid-term occlusion. We also show that VLMMs can be used as a means to inform the prior in an Expectation Maximisation (EM) process used for joint spatial and temporal learning of image features.

Keywords. Variable Length Markov Models, Temporal Learning, 3-D Tracking, Data Association, Task-based Control

1 Learning temporal structure and task based control

1.1 Predictive control

A common observation in animate vision is that it is 'purposive', i.e. the visual processing is focussed on a particular task or goal [2]. In cognitive computer vision systems, this notion has to be formalised using models that support online expectation or 'predictive control'. That is, behavioural models must be acquired by the system, e.g. by learning, and exploited for effective prediction at the required level of visual processing, e.g. here object trajectories.

Assuming that spatial localisation of task relevant objects is possible, and the existence of suitable training data, then the categorisation/classification labels and localisation data taken together provide an account of object trajectories. If we have over-arching prior knowledge about the types of trajectories we may observe, the learning of temporal dynamics for a specific target is simply a matter of 'best fitting' the localisation data to the available models (e.g. N-th order polynomial curve fitting).

In the absence of such a 'global' perspective of candidate trajectories, we can still make useful assumptions that constrain the learning of temporal dynamics. For example, we assume there is some underlying process that describes the motion and that the motion is smooth and continuous (except, for example, under occlusion). Additional assumptions will determine the generative properties of the resulting model.

Markov models have been used extensively in the modelling and recognition of human activities involving highly structure and semantically rich behaviours such as sign language (Starner and Pentland [20], Vogler and Metaxas [21]), dance and aerobics (Galata and Hogg [9]) and vehicle movements around parked aircraft [10]. Learning of first-order Markov models is a well-established field. Unfortunately, as Galata and Hogg point out [9], Hidden Markov Models (HMMs) do not easily encode high order temporal dependencies. Local optima are frequently encountered by iterative optimisation techniques when learning HMMs with many free parameters. Galata and Hogg use Variable Length Markov Models (VLMMs) [16] which are able to locally optimise memory length within the model. This approach captures long term temporal dependencies in some parts of behaviour and short term dependencies elsewhere. Their approach uses discrete VLMMs modelling transitions between a predetermined set of prototypes. Clouse et al.[4] show how time delay neural networks can be used to induce and represent finite state machines with long memory lengths (cf. higher order Markov models). Johnson and Hogg [12] model highly non-linear behaviours for a pedestrian tracking activity using a Gaussian mixture over system state change and observed history. Their approach is unusual in that it does not assume a discretisation of state space and models behaviours as both continuous and non-linear. However, one drawback of this approach is that, while the distributions are learnt and effective in online control, the prediction used Kalman filters to constrain the search from frame to frame. As shown later in this paper, more powerful predictive control, robust to long term occlusions is possible using our temporal structure learning. This is because the VLMMs are able to use their longer length optimised state histories to provide a better account of likely future state using stochastic sampling techniques than a first-order model. So repeated applications of stochastic sampling produce a more structured and constrained prediction of likely state sequences.

1.2 Interpretation of human activities - the ActIPret project

The ActIPret project (`actipret.infa.tuwien.ac.at`) aims to develop a cognitive vision methodology that interprets and records the activities of people handling tools. The focus is on active observation and interpretation of activities, on parsing the sequences into constituent behaviour elements, and on extracting the essential activities and their functional dependence. The ActIPret demonstrator is a distributed system with low-level data-driven vision components and high level task driven behavioural reasoning components. These components are organised into a nominal hierarchy and higher level components interact with the lower level components through a common service interface. The resulting system implements a scheme of task-based visual control that can be though of as data-driven (bottom-up) processing limited in scope by task-based (top-down) control.

The work described in this paper has application in the interactions between just two of the ActIPret components, a Gesture Recogniser (GR) that identifies task relevant functional gestures (activities such as 'pickup' and 'putdown' rather than communicative gestures), and a Hand Tracker (HT) that provides 3-D hand positional data to the GR. The work has more general applicability across a range of problems in Cognitive Vision where perception is guided by expectation.

1.3 The Hand Tracker

The FORTH Hand Tracker (see [1]) uses a non-parametric method for skin detection and performs tracking in a non-Bayesian framework. A YUV 4:2:2 based skin colour representation is learned through an off-line procedure. The skin colour detection model is adaptive, based on the recent history of tracked skin-coloured objects. Thus, without relying on complex models, it is able to robustly and efficiently detect skin-coloured objects even in the case of changing illumination conditions. Tracking over time is performed by employing a novel technique that can cope with multiple skin-coloured objects moving in complex patterns in the view of a potentially moving camera. The Hand Tracker provides real time hand candidate centroid data to the GR.

1.4 Combining the HT and GR

In the ActIPret system, the GR is tasked with identifying task relevant functional gestures for specific hand objects. The selection of the relevant candidate hand objects is determined by an abstract reasoning engine at a higher level than the GR (Sage, Howell and Buxton [19]). For multiple handed tasks, or early attentive processing where we are interested in identifying all possible task relevant hand candidates, we may need to track an arbitrary number of candidate hand objects. The higher level reasoning component may have cause to request attentive processing for multiple objects because of task ambiguity (it can pursue concurrent multiple lines of reasoning), or because there were multiple hand candidates (i.e. less than perfect segmentation) reported during earlier processing. Consistent hand candidate labelling is a key factor for the performance of the ActIPret system as a whole. Whereas short term consistency may be acceptable for discrete gesture recognition (e.g. the hand has 'reached out' from the torso), it is problematic for longer temporal scale recognition of behavioural activities (e.g. hand(x) picked up object(y) and put it down on object(z)). Consistency over longer temporal scales requires robustness against many factors such as lighting variation and, in particular, mid-term occlusion and ambiguity caused by intersecting multiple hand trajectories. In order to achieve candidate hand labelling consistency, the HT has to reason locally about such occlusion and ambiguity in order to unify different labels should they arise (e.g. to determine that hand(1) is the same physical hand as hand(2) when the new label is created when the original object comes out of occlusion).

In conventional approaches to solving this temporal correspondence or 'data association' problem, we might use predictive tools such as Kalman filters, or adopt graph based spatial matching. In ActIPret we seek opportunities through the hierarchical service structure. Normally, the GR makes a service request to the HT for positional data for a specific object or set of objects. The HT then provides the data to the GR, which then applies its own processing to determine whether gestures have been completed. When the HT is not able to provide data positional data, the GR contains useful information about what gestures might have been occurring at the time the hand object was lost. At that point, the GR can generatively extrapolate positional data for the missing hand object, maximising the odds that disparate hand object labels can be re-unified when positional data is available for the hand once more.

The aim of the experiments described in Section 3 of this paper is to learn relevant gesture models using hand positional data derived from FORTH's tracker and show that these gesture models can be used within a component such as the GR to provide a robust means of dealing with tracking issues such as mid-order temporal occlusion. This capability is provided by strong generative properties of the underlying VLMM gesture models. The experiments described in section 3 demonstrate such generative extrapolation in action.

2 From first-order to variable order temporal dynamics

2.1 HMM Gesture Recognition

A Hidden Markov Model (HMM) is a doubly stochastic process, i.e. there is an underlying stochastic process that is not observable (hidden) but can only be observed through another set of stochastic processes that produce the sequence of observed symbols [15]. The HMM is characterised by a triple $\lambda = (\pi, A, B)$, where A is a square $N * N$ matrix of probabilities for transitions between N discrete hidden states, π is a vector of probabilities describing the initial state of the model and B is a $N * M$ matrix accounting for the mapping between the N hidden states and the M output (observable) symbols.

There are three general problems we may solve using HMMs. Given a set of observation symbols O and a model λ we can calculate the probability of that sequence $p(O|\lambda)$ (forward evaluation). Given O and λ we can deduce the most likely sequence of hidden states (Viterbi decoding). Finally given O we can estimate model parameters λ that maximise the probability of O.

To capture gesture models, we used training observation sequences represented sequences of 3-valued vectors (3-D hand velocities $V = [v_x, v_y, v_z]$) to train a continuous output HMM with hidden states modelled as 3-component mixtures of Gaussian functions. We then varied the number of hidden states to explore the underlying dimensionality of the training set (which corresponds approximately to the number of distinct gesture phases) and demonstrated the ability of the HMM to distinguish the learned gesture from other gestures, as in some of our previous work[18].

2.2 Overview of VLMM

Ron et al's [16] formulation of the VLMM is based on optimisation of the statistical prediction of a Markov Model measure by the instantaneous Kullback-Liebler (KL) divergence of the following symbols (the statistical surprise of the model when presented with the next symbol). The memory is extended when such a surprise is significant until the overall statistical prediction of the model is 'sufficiently good' for a user-defined error ϵ [17]. The original formulation worked with training sequences that were strings of discrete symbols (such as in language modelling). The training process leads to a prediction suffix tree that predicts the probability that a symbol $\sigma_t \in \Sigma$ follows a variable length string $s_{t-N}, ...s_{t-1}$. Throughout this paper, we have adopted the notation of σ as a single alphabet symbol (length 1 string) and s as string of arbitrary integer length > 1. Where s is shown subscripted, this is intended to show that we are indexing a single alphabet symbol from within the string s. Further, to prevent confusion between strings and joint probability, subscripted strings are sometimes shown in braces. Distributions are shown in upper case, single values are shown in lower case. Thus, for example, $P(\Sigma, T)$, where Σ is the alphabet and T is time, is a 2-D array of values and $p(\sigma, t)$ is a single value.

To learn VLMM gesture models from hand training data, we first fit a set of N Gaussian mixtures over the data using the standard first-order HMM learning procedure. We discard the learned HMM state transition matrix and prior vector and use just the mixture mean μ and co-variance ϕ parameters. Our VLMM alphabet Σ then corresponds to the mixture indices $(1, 2, ..., N)$. The probability of being in a state σ_t is defined as the fit between a piece of observed data x_t and the N Gaussian mixtures. In this paper, states are assumed to be numbered $1, 2, ..., N$. More sophisticated modelling could expand Σ to model the mixture index plus other parameters such as probabilistic context [18].

The central element in generating the prediction suffix tree is the KL divergence measure for the statistical surprise associated with a prediction for a symbol, s, compared with the longer string σs:

$$Err(\sigma s, s) = \sum\nolimits_{\sigma' \in \Sigma} p(\sigma s \sigma') log \frac{p(\sigma s \sigma')}{p(\sigma'|s)p(\sigma s)} \tag{1}$$

The source probabilities $p(s)$ and $p(\sigma|s)$ are estimated from sums and products of the empirical counts (#) of the appearances of their constituent alphabet symbols in the distribution $P(\Sigma, T)$ defined by:

$$p(\sigma_t) = \frac{\mathcal{N}(x_t; \mu_\sigma, \phi_\sigma)}{\sum_{\sigma'=1}^{\sigma'=N} \mathcal{N}(x_t; \mu_{\sigma'}, \phi_{\sigma'})} \tag{2}$$

$p(\sigma_t)$ is drawn from the distribution $P(\Sigma, T)$ and is the probability of alphabet symbol σ at time t (i.e. the fit between the observation data x at time t and the set of N Gaussian mixtures specified by $\{\mu_1, \phi_1\}...\{\mu_N, \phi_N\}$. Following Ron's original paper and applying Laplace's rule of succession:

$$p(s) \approxeq \frac{\#s + 1}{\sum_{s' \in \sum^{|s|}} \#s' + |\Sigma|} \tag{3}$$

$$p(\sigma|s) \approxeq \frac{\#(\sigma|s) + 1}{\sum_{\sigma' \in \Sigma} \#(\sigma'|s) + |\Sigma|} \tag{4}$$

The notation $\Sigma^{|s|}$ taken from Ron's original paper refers to the set of strings of length $|s|$ that can be derived from the alphabet. For example, for a two symbol alphabet $\Sigma = \{1, 2\}$, if we wanted to find $p(\{1, 2\})$ we would need to evaluate $p(\{1, 1\})$, $p(\{1, 2\})$, $p(\{2, 2\})$ and $p(\{2, 1\})$ for the denominator term. Fortunately, it is easy to show that the denominator terms are constant for any given Σ and $|s|$. We dispensed with the smoothing terms by simply removing the $+1$ from the numerator terms and the $|\Sigma|$ from the denominator terms. Here, $|\Sigma| = N$ i.e. the alphabet size is the number of Gaussian mixtures. As $P(\Sigma, t)$ is a continuous valued vector of length N, and s is a variable length string (potentially 1 symbol or >1 consecutive symbols), the $\#s$ term becomes a sum of products of probability values:

$$\#s \equiv \sum_{t \in T} p(\{s_{t-|s|+1}, \ldots, s_t\}) = \sum_{t \in T} \left(\prod_{t'=t-|s|+1}^{t'=t} p(s_{t'}) \right) \tag{5}$$

rather than a simple count and likewise the $\#\sigma|s$ becomes:

$$\#\sigma|s \equiv \sum_{t \in T} p(\{s_{t-|s|}, \ldots, s_{t-1}, \sigma_t\}) = \sum_{t \in T} \left(p(\sigma_t) \prod_{t'=t-|s|}^{t'=t-1} p(s_{t'}) \right) \tag{6}$$

This extension to the VLMM formulation to the continuous case enables us to estimate $p(s)$ and $p(\sigma|s)$ under noisy conditions and proves to degrade gracefully in practice towards a first-order (length=1) Markov model with $1/|\Sigma|$ state transitions with high levels of noise. We use these terms to build a Prediction Suffix Tree.

A Prediction Suffix Tree (PST) is a tree graph representation of a set of variable length nodes that describe a training data set. The nodes are those essential to the variable length structure of the domain and are determined by applying the KL divergence statistical test previously described. Each node consists of a label (a set of alphabet symbols of length $1 < \mathit{length} \leq L$ and a set of $|\Sigma|$ transition probabilities. L is a user defined parameter that determines the maximum allowed length of any variable length node to describe the domain (the maximum PST tree depth). The transition probabilities specify the probability of a subsequent single alphabet symbol having observed the variable length node. Following Ron et al's original formulation, the PST also has a root node denoted 'e' (of nominal length 0) that specifies the probabilities of observing single alphabet symbol label nodes. There is one additional user-defined parameter ϵ that determines the level of statistical surprise required before a node is inserted into the PST.

An example Markov source (used to generate training data) and the corresponding PST generated for it are shown in Fig. 1.

Once learned, a PST can be converted into a Probabilistic Finite Automaton (PFA) 5-tuple (Q,Σ,τ,γ,π) where Q is the finite set of M variable length states derived (approximately) from the leaves of the prediction suffix tree, Σ is an alphabet of size N (the mixture indices), $\tau : Q \times \Sigma \rightarrow Q$ is the transition function, $\gamma : Q \times \Sigma \rightarrow [0, 1]$ is the output probability function and $\pi : Q \rightarrow [0, 1]$ is the probability distribution over the starting states.

In the simplest case, we can then use the PFA to create a forward evaluation matrix similar to calculate the distribution $P_e(Q, T)$ as we would with a first-order HMM. We first create a forward evaluation trellis of size $M * T$. The probability values $p(q_t)$ are calculated as follows:

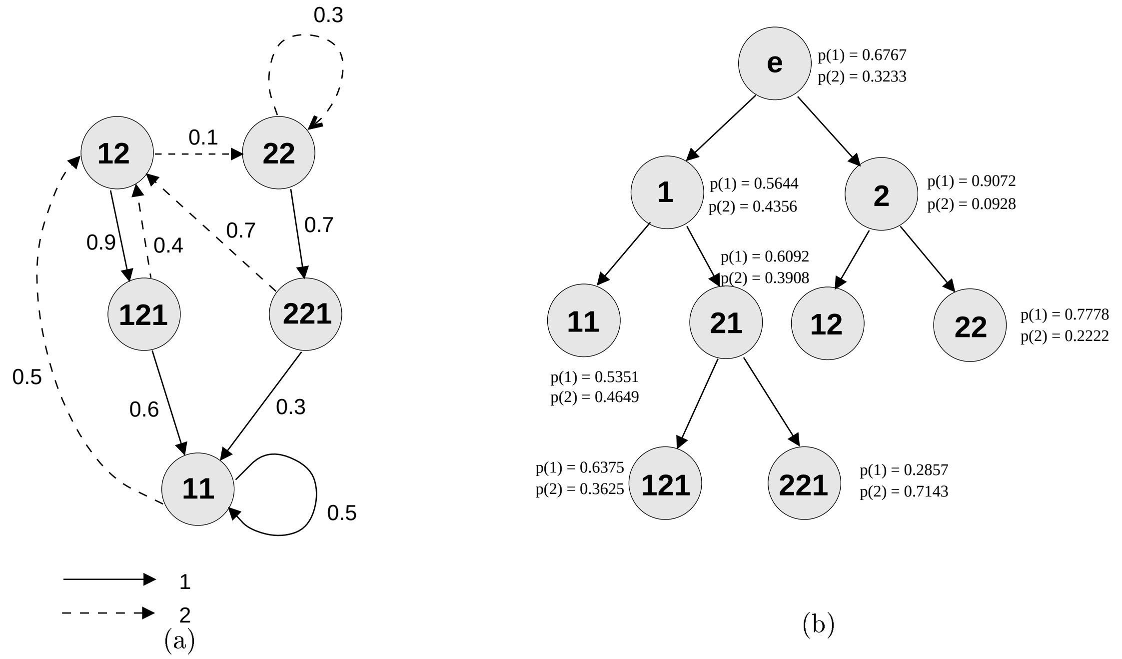


Figure 1: (a) Markov model used to generate discrete symbol string (b) Prediction Suffix Tree learned from string for $\epsilon = 0.001$ and $L = 3$

$$p_e(q_t) = \mathcal{N}(o_t; \mu_{ts(q)}, \phi_{ts(q)}) \sum_{q' \in Q: \sigma = ts(q): \exists \tau(q', \sigma \rightarrow q)} p_e(q', t-1)\gamma(q', \sigma) \tag{7}$$

where the mapping between the M states in Q and the N Gaussian mixtures is defined by the notion of the terminal symbol $ts(q)$. $ts(q)$ is the last symbol in the state q e.g. $ts(123) = 3$. So μ_q is defined as $\mu_{ts(q)}$.

We assume a uniform $1/M$ distribution for π such that $\Sigma P_Q = 1$ for each timestep. Finally, we can form a $|\Sigma| * T$ trellis by adding rows of the $M * T$ trellis so that the terminal cluster symbol for each of variable length states is the same. We can use the resulting forward evaluation trellis to find the fit between any observation data series $x_1, x_2, ...x_T$ using the standard ideas of log likelihood per symbol. We can also use scaling to prevent the trellis co-efficients geometrically tending towards ∞.

The use of the terminal symbol means there is a 'many to one' mapping between states $\in Q$ and the mean and covariance value sets defined by the original N Gaussian mixtures. Although this is fine for learning the set of variable length states that describe any particular training domain, it does place limitations on the generative properties of the the resulting model overall. Consider the training set extract shown in Figure 2 that shows 2 Gaussian mixtures, M1 and M2, fitted to the training data. The size, and thus the mean and covariance value sets, for M1 and M2 will be a function of the number of mixtures fitted to the data. For a VLMM model derived from this example, there would likely be states of length from 1 to 3. A length 3 state corresponding to 3 consecutive points in mixture M1 would be strong evidence for a length 1 state corresponding to M2. Whilst this would be fine for classifying other trajectories with a similar structure, consider what might happen if we try to use the same model in a generative mode. As M1 and M2 have only one mean and covariance parameter set each, every time we want to stochastically generate a novel data point for a trajectory and specify the terminal state we will generate a value about the means of the Gaussian mixtures. However, we have lost valuable information about the distribution of mean and covariance values as we progress through the Gaussian feature space. What is really required is that for each variable length state $q \in Q$, we want an independent mean and covariance parameter set so that in the generative mode we create trajectories that are more akin to the original training data.

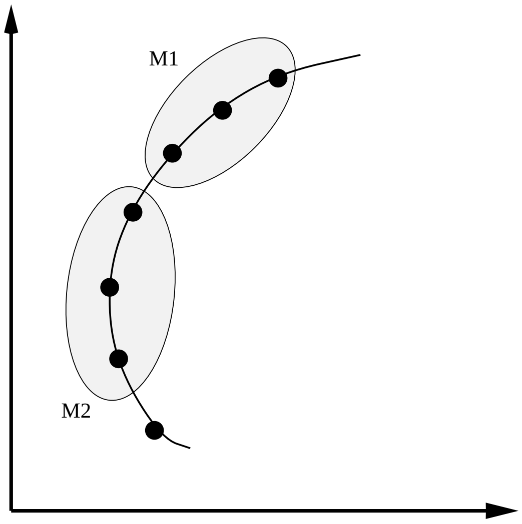


Figure 2: Training set extract consisting of 7 2-D data points along a continuous arc of motion

To establish independent mean and covariance parameter sets for each variable length state we used a single iteration of the Forward Backward algorithm as typically used in the Baum Welch iterative learning procedure for HMMs. We defined an alpha (forward evaluation) trellis $P_\alpha(Q,T)$ with a uniform distribution over $P_\alpha(Q,t=1)=1/M$:

$$p_\alpha(q_t) = \mathcal{N}(o_t;\mu_{ts(q)},\phi_{ts(q)}) \sum_{q' \in Q: \sigma = ts(q): \exists\tau(q',\sigma\rightarrow q)} p_\alpha(q',t-1)\gamma(q',\sigma) \tag{8}$$

and a beta (backwards evaluation) trellis $P_\beta(Q,T)$ with a uniform distribution over $P_\beta(Q,t=T)=1$:

$$p_\beta(q_{t-1}) = \sum_{q' \in Q: \sigma \in \Sigma: \exists\tau(q_{t-1},\sigma\rightarrow q')} p_\beta(q',t)\gamma(q_{t-1},\sigma)\mathcal{N}(o_t;\mu_{ts(q')},\phi_{ts(q')}) \tag{9}$$

and a Ω distribution ('counts' of how often a state $q \in Q$ was visited over any individual training exemplar): [1]

$$P_\Omega(Q,T) = P_\alpha(Q,T) \otimes P_\beta(Q,T) \tag{10}$$

where $\otimes$ denotes the element wise product. We used the well published methods for scaling coefficients in the distributions to prevent numerical underflow. The independent mean and covariance value sets for each $q \in Q$ were then defined as:

$$\mu_q = \frac{\sum_{t=1}^{t=T} o_t p_\Omega(q,t)}{\sum_{t=1}^{t=T} p_\Omega(q,t)} \tag{11}$$

$$\phi_q = \frac{\sum_{t=1}^{t=T} (o_t-\mu_q)p_\Omega(q,t)(o_t-\mu_q)^T}{\sum_{t=1}^{t=T} p_\Omega(q,t)} \tag{12}$$

[1] In other literature the counts are usually denoted by γ. We use Ω here to prevent confusion with the output probability function of the VLMM.

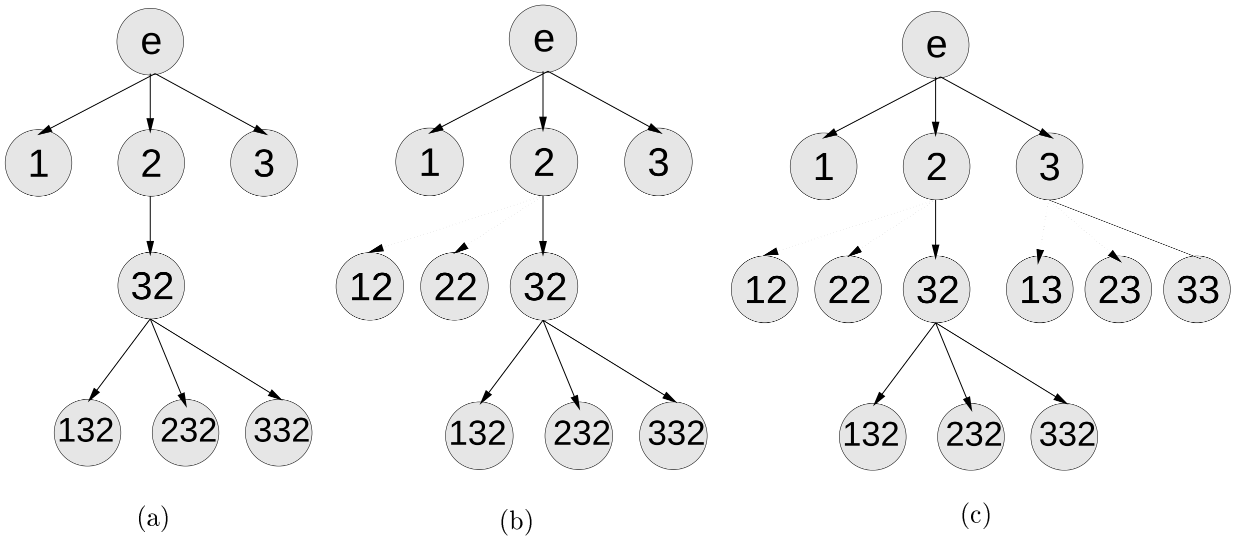


Figure 3: Prediction Suffix Trees with leaf sets (a) {1,3,132,232,332}, (b) {1,3,12,22,132,232,332} and (c) {1,12,22,13,23,33,132,232,332}

where ^{T} denotes matrix transpose.

We modified Ron's original algorithm for learning the Prediction Suffix Tree to facilitate easier conversion between the PST and the full PFA forms. By 'full', we mean that the resulting PFA is guaranteed always to be in a valid state $q \in Q$ and that τ is well defined for all symbols $\sigma \in \Sigma$ regardless of whether the symbol sequence $q\sigma$ appears in the training data. This means that the PFA can deal with unseen data without requiring any special modification of the forward evaluation process. These additional nodes will not be further expanded as they do not meet the KL criteria, but will appear in the set of PST leaf nodes as the first step of its conversion to the PFA.

To ensure that we can generate a full PFA, whenever a node k is added at some depth $l \leq L$ in the PST, we also add, $\forall s \in \Sigma$, $\{s, suffix(k)\}$ to the PST regardless of whether $\{s, suffix(k)\}$ meets the KL criteria or not. For PSTs with a maximum tree depth > 2, we need to ensure that the resulting PFA has the means to make a transmission between nodes that vary in length by more than 1 observation symbol. So, as well as including all of the PST leaf nodes in the PFA, we may need additional internal nodes in order to ensure that $\forall q \in Q$, then $\forall \sigma \in \Sigma$ then $\exists q' \in Q$: $somesuffix\{q\sigma\} = q'$ where $\{(somesuffix)\}$ constitutes any consecutive end set of symbols in a sequence.

To illustrate this, for an alphabet $\Sigma = \{1, 2, 3\}$, consider a discrete PST with depth $L = 3$ as shown in Fig. 3(a). Taking the leaf set of PST nodes {1,3,132,232,332}, it is not possible to draw a full PFA. For example, if the PFA is in states 1,132,232 or 332 and receives the next symbol 2, there are no states in Q to make a transition to. Also as the leaf set only contains nodes of length 1 and 3, there is no single symbol transition possible between the length 1 nodes and the length 3 nodes.

Our example has assumed that symbol 32 met the KL criteria but that 12 and 22 did not. If we add 12 and 22 anyway (because we added 32) then we have the PST as shown at Fig. 3(b) with the two additional nodes shown as dotted links. Now we have the PST leaf set {1,3,12,22,132,232,332}. Next, if the PFA was in state 1 then we can observe the symbol 2 and move to state 12, and if the PFA was in states 132,232 or 332 we would move to state 22.

However, we still have a problem with the length 3 states. Whilst we can define exit transitions for these states there are no transitions that cause us to enter them. This is because the prefixes for these states (i.e. 13,23 and 33) do not appear in the leaf set. To solve this problem we need to expand the node 3 to provide the additional nodes 13, 23 and 33 as shown in Fig. 3(c). This now gives us an augmented leaf set of {1,13,23,33,132,232,332} from which we can derive a complete PFA as summarised in Table 1.

The PST learning process is easily extended to multiple exemplars by taking average statistics for

Table 1: Tabular enumeration of state transitions for PFA built from augmented PST leaf set {1,12,22,13,23,33,132,232,332}.

	Next observed symbol		
Current state	1	2	3
1	1	12	13
12	1	22	23
22	1	22	23
13	1	132	33
23	1	232	33
33	1	332	33
132	1	22	23
232	1	22	23
332	1	22	23

$Err(\sigma s, s)$ over all exemplars.

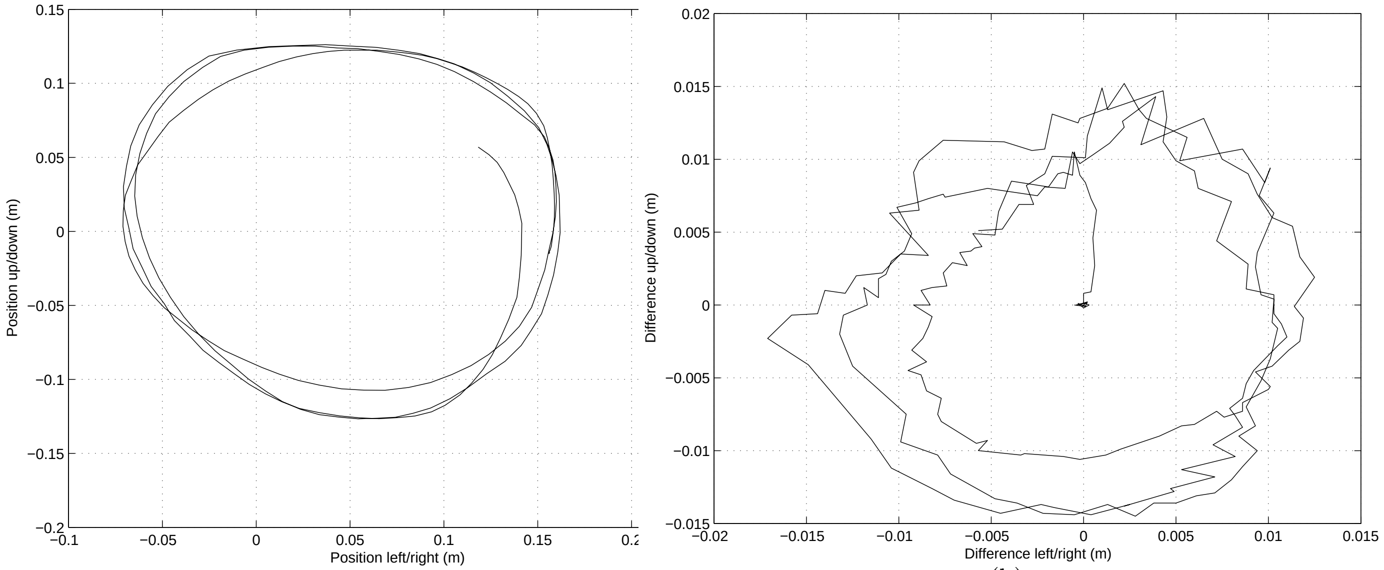
3 Tracking experiments

We present four experiments that deal with the learning of temporal structure for hand trajectories captured by hand trackers. The first three use 2-D hand positional data generated by FORTH's HT. The fourth uses 3-D data derived from the *Terminal Hand Orientation and Effort Reach Study* Database created by Human Motion Simulation (HUMOSIM) at the Center for Ergonomics, University of Michigan, USA. It is important to re-iterate here that these experiments are not learning about discrete transitions between a fixed set of prototypes. We are learning on continuous valued vectors where each vector at time t describes the fit between the training data and N Gaussian mixtures fitted over that data.

3.1 Experiments 1, 2 and 3: 2-D examples

The gesture training data for this experiment was collected using FORTH's HT. 2-D hand centroid positional data (hand in a constant z-plane position) was collected for examples of the hand moving in an approximately circular motion. Each example consisted of between 200-250 equal temporally spaced timesteps. An example data set exemplar is shown below in Fig. 4. The training set consisted of a total of 7 complete exemplars and each exemplar consisted of positional data for 1 hand with no occlusion. We took the first derivative of the training set data to give velocity data.

We first trained a continuous valued first-order HMM over the velocity training set with 10 hidden nodes which gave us all of the parameters for 10 Gaussian mixtures. The number of hidden nodes chosen was an arbitrary choice intended to balance over-generalisation and over-fitting. The number of hidden nodes determines the size of the VLMM alphabet. We then generated the distribution $P(\Sigma, T)$ for each velocity training set exemplar and trained the VLMM over those distributions with a maximum prediction suffix tree depth $L = 10$ using a minimum statistical surprise parameter min $\epsilon = 10^{-5}$. The resulting augmented Prediction Suffix Tree (PST) had 2821 nodes, which was then converted into a PFA which had 2548 nodes with node lengths in the range $2 - 8$. The maximum node length discovered was less than the maximum permitted showing that, for that value of statistical surprise, the model achieved the best result that was possible. Smaller values of min ϵ will result in larger PSTs. In general we found that values between 10^{-3} and 10^{-5} give good results over a range of applications.



(a) (b)

Figure 4: 2-D hand tracking training exemplar 01 as (a) positional data, (b) first derivative

We refined the Gaussian mixture representation so that there were independent parameter sets μ, ϕ for each of the M variable length states in Q. So we started out with N mixtures, one for each Gaussian mixture fitted to the training data and ended up with M mixtures, one for each of the variable length states. Taking $\Gamma = \{\mu_1, \mu_2, ..., \mu_M\}$ and $\Phi = \{\phi_1, \phi_2, ..., \phi_M\}$, we now define the full continuous valued VLMM, V, as the 7-tuple $(Q, \Sigma, \tau, \gamma, \pi, \Gamma, \Phi)$. As previously described, we can generate a forward evaluation matrix either as a $M \times T$ matrix (full forward evaluation trellis) or as a $N \times T$ matrix (compact forward evaluation matrix) by combining states that end in the same observation symbol. The compact trellis has the advantage that matrix entries for symbol $n \in N$ at time t represent the total probability associated with that observation symbol rather than the isolated probability of being in any particular state $q \in Q$.

We then defined a forward evaluation procedure similar to the standard approach [15] for calculating $p(O|\lambda)$ for a first-order HMM. This can be used to measure the log likelihood fit between any testing data and V, but we are more interested in the generative properties of V and using these properties to deal with problems such as a mid-term occlusion. We use the forward evaluation procedure to create a VLMM tracker using stochastic sampling to generate states data in the absence of observation data. This VLMM tracker is distinct to the FORTH tracker. The FORTH tracker produced the positional hand data that is used to train the VLMM. The VLMM tracker uses the learned PFA to deal with mid-order temporal occlusion and can 'fill-in' the trajectory. The key elements of the VLMM tracker procedure can be summarised by the algorithm shown in Fig. 5.

For experiments 1 through 4, the setup was slightly more complicated. We trained the model on first derivative data. Subsequent testing data was positional data (like the original form of the training data), so we had to design a VLMM tracker routine that took this into account. When observation data is not present (cf. occlusion) then we stochastically generate states that tell us the relative motion so we can produce a cumulative estimate of where we believe the hand to be.

The original training set was modified to provide a testing set. To simulate the effect of occlusion, we deleted a number of consecutive timesteps from each of the exemplars. This was achieved by adding a flag in the testing set to indicate whether the positional data was to be made available to the VLMM tracker at any timestep. The tracker then processed each timestep in turn. As long as data was available, it calculated the relative motion from the last timestep to the current one and used the VLMM model to update the full forward evaluation trellis. When data was not available

Note:

$ts(s)$ means the terminal symbol of a string s e.g. $ts(123) = 3$
We are given O although it may be incomplete. The output from this algorithm is the matrix $P_e(Q, T)$ and (vitally) generated values for missing O steps

Algorithm:
LET Observation sequence $O = \{o_1, o_2, ..., o_T\}$ where some elements in O are known and others are not (were not observed or are missing)
LET $V = \{Q, \Sigma, \tau, \gamma, \pi, \Gamma, \Phi\}$
LET Full forward evaluation trellis $P_e(Q, T) = 0$
LET π vector in the PFA initialised to $1/M$ (uniform prior assumption)

$P_e(Q, t = 1) = \pi * \mathcal{N}(o_1; \Gamma, \Phi)$
FOR $t = 2$ to T
  IF (observation data o_t is defined)
    $\forall q \in Q : p_e(q, t) = \mathcal{N}(o_t; \mu_q, \phi_q) \sum_{q' \in Q: \sigma = ts(q): \exists \tau(q', \sigma \to q)} p_e(q', t-1)\gamma(q', \sigma)$
  ELSE
    $\forall q \in Q : p_e(q, t) = \sum_{q' \in Q: \sigma = ts(q): \exists \tau(q', \sigma \to q)} p_e(q', t-1)\gamma(q', \sigma)$
    Choose a state q_s by sampling from the distribution $P_e(Q, t)$
    Set All values of $P_e(Q, t) = 0$ except for $p_e(q_s, t)$ which is set $= 1.0$
    Generate normally distributed random values for o_t according to μ_{q_s} and ϕ_{q_s}
  END_IF
END_FOR

Figure 5: Pseudo-code of tracking algorithm used in experiments 1 through to 4

(because the occlusion flag was set), the tracker used stochastic sampling to determine its next state and produce a relative motion estimate. We compared the generative capabilities under occlusion of the VLMM tracker with a first-order HMM [2] describing relative motion, together with structurally naive estimators based on constant velocity and constant acceleration assumptions. Where the data was available to the VLMM tracker, it is shown in black. Where data was not available to the VLMM tracker it is shown in yellow. Thus, for evaluating the performance of the VLMM tracker, the black and yellow points form the ground truth. Whilst in the black region, the VLMM tracker builds its internal distribution $P_e(Q, T)$. When data suddenly is not available, the VLMM tracker uses $P_e(Q, T)$ and the VLMM parameters to start generating predicted values to fill in the missing steps in O. These predicted values are shown in magenta. For comparison, predicted values for a first-order HMM are shown in red.

Figs. 6, 7 and 8 shows examples of 2-D generative trajectories created during times of occlusion during hand tracking for experiments 1, 2 and 3 respectively. In each case, the occlusion was for 25 timesteps. In Fig. 6 the occlusion started at step 37, in Fig. 7 at step 64 and in Fig. 8 at step 6. Two standard, learning-free, tracking methods, constant velocity (shown in green) and constant acceleration (blue, this is effectively a Kalman Filter without noise), are shown to perform, as expected, fairly well for short periods of occlusion - up to 3 or 4 timesteps. Fig. 6(a) shows that their trajectory predictions rapidly diverge from the true trajectory for longer periods of occlusion. Figs. 6(b), 7(b) and 8(b) show the actual timestep wise Euclidean distance of each generated trajectory to the original ground truth data, in other words, how close the prediction is to the actual hand movement. Figs. 6(c), 7(c) and

[2] The HMM was based on the same set of N Gaussian mixtures from which the VLMM was first built (before the construction of the M mixtures, one per variable length state).

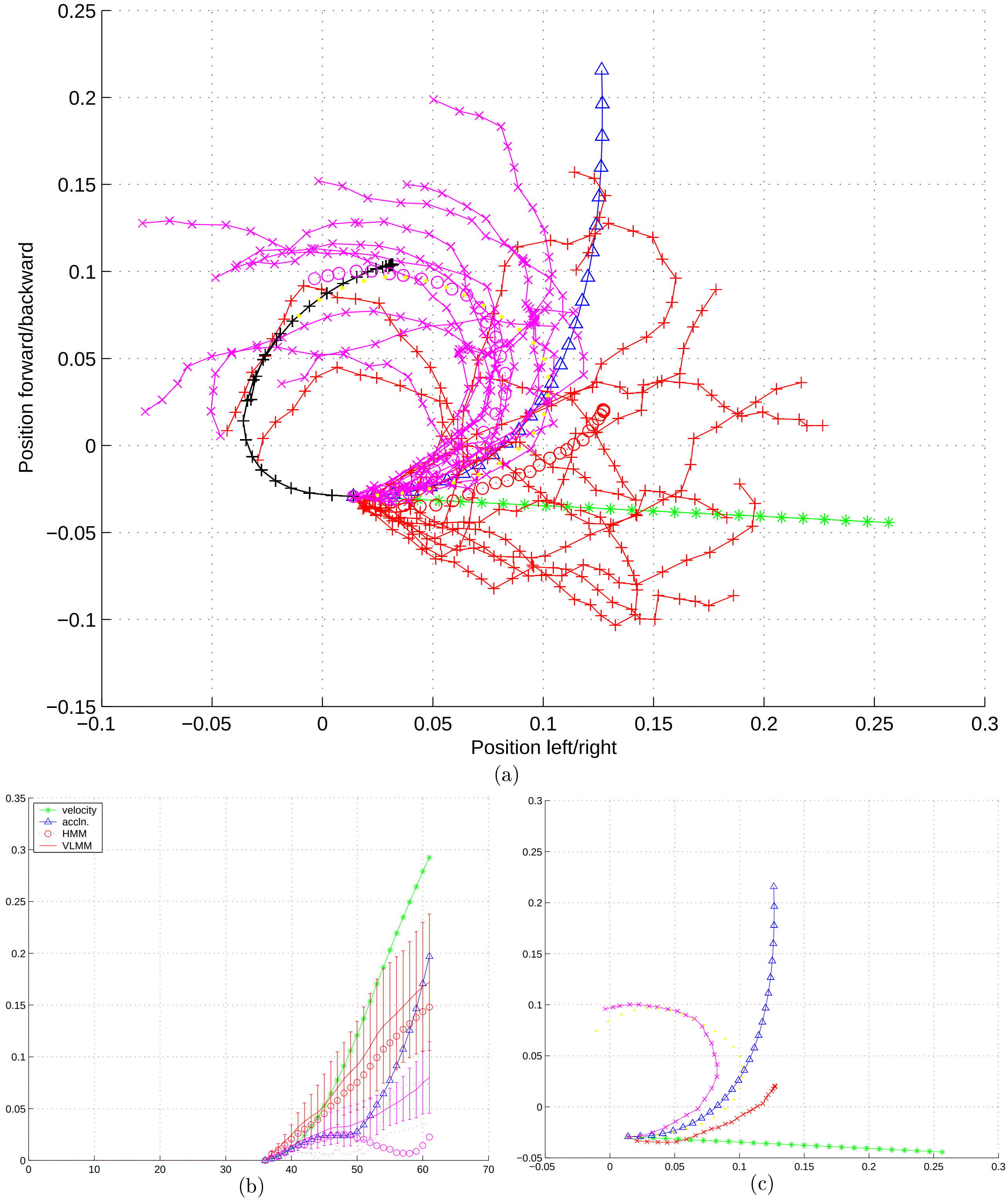
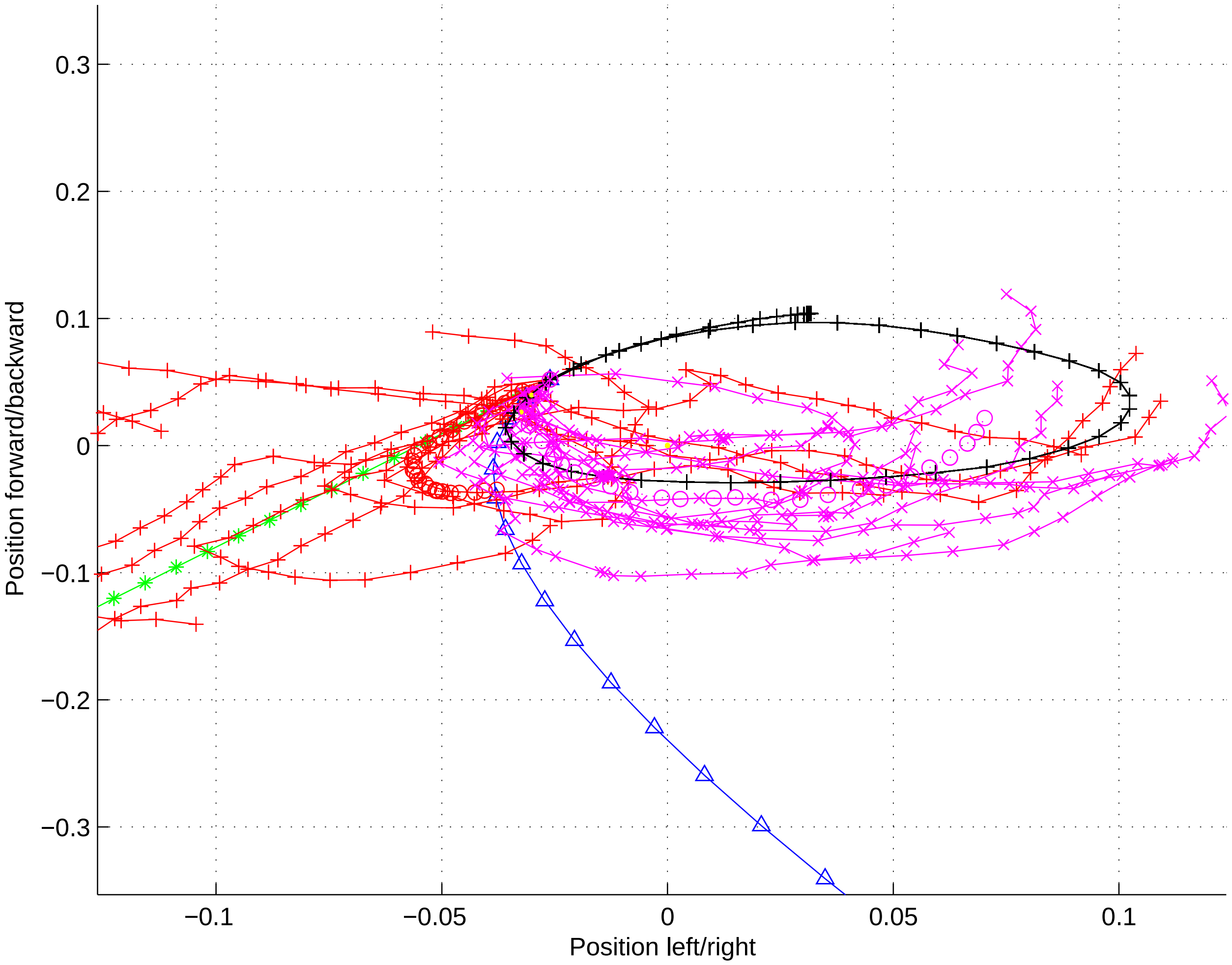
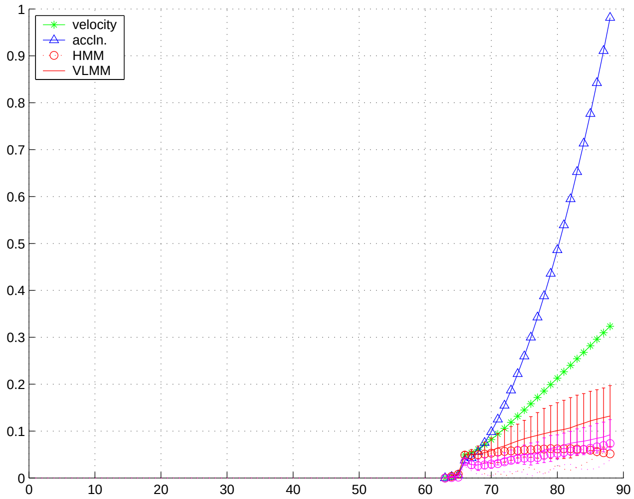


Figure 6: 2-D tracker experiment 1: constant velocity (green), constant acceleration (blue), first-order HMM (red) and VLMM (magenta), ground truth (unoccluded section in, black and occluded section in yellow), (a) as reconstructed 2-D trajectories, (b) the Euclidean distance between the trajectory of each method and the actual original hand trajectory and (c) the mean reconstructed trajectories.

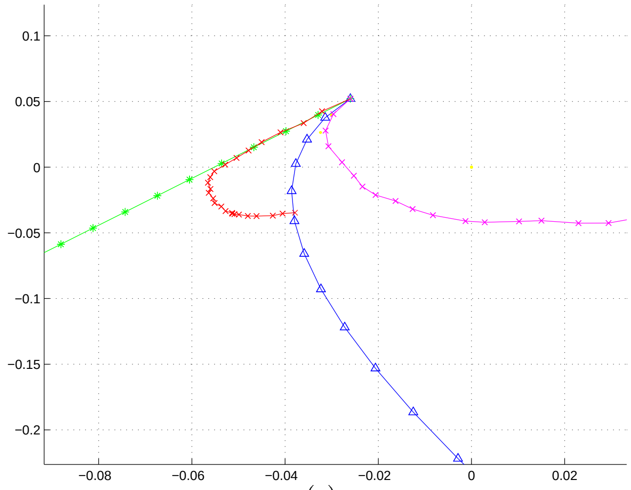
8(c) show the timestep-wise mean of each set of generated trajectories and provide a simple overall measure of the performance of the tracker methods.



(a)



(b)



(c)

Figure 7: 2-D tracker experiment 2: constant velocity (green), constant acceleration (blue), first-order HMM (red) and VLMM (magenta), ground truth (unoccluded section in, black and occluded section in yellow), (a) as reconstructed 2-D trajectories, (b) the Euclidean distance between the trajectory of each method and the actual original hand trajectory and (c) the mean reconstructed trajectories.

The trajectories of main interest are for the first-order HMM (red) and VLMM (magenta) methods. It can be seen in Figs. 6(a) and 7(a) that both methods attempt to reproduce the curving movement

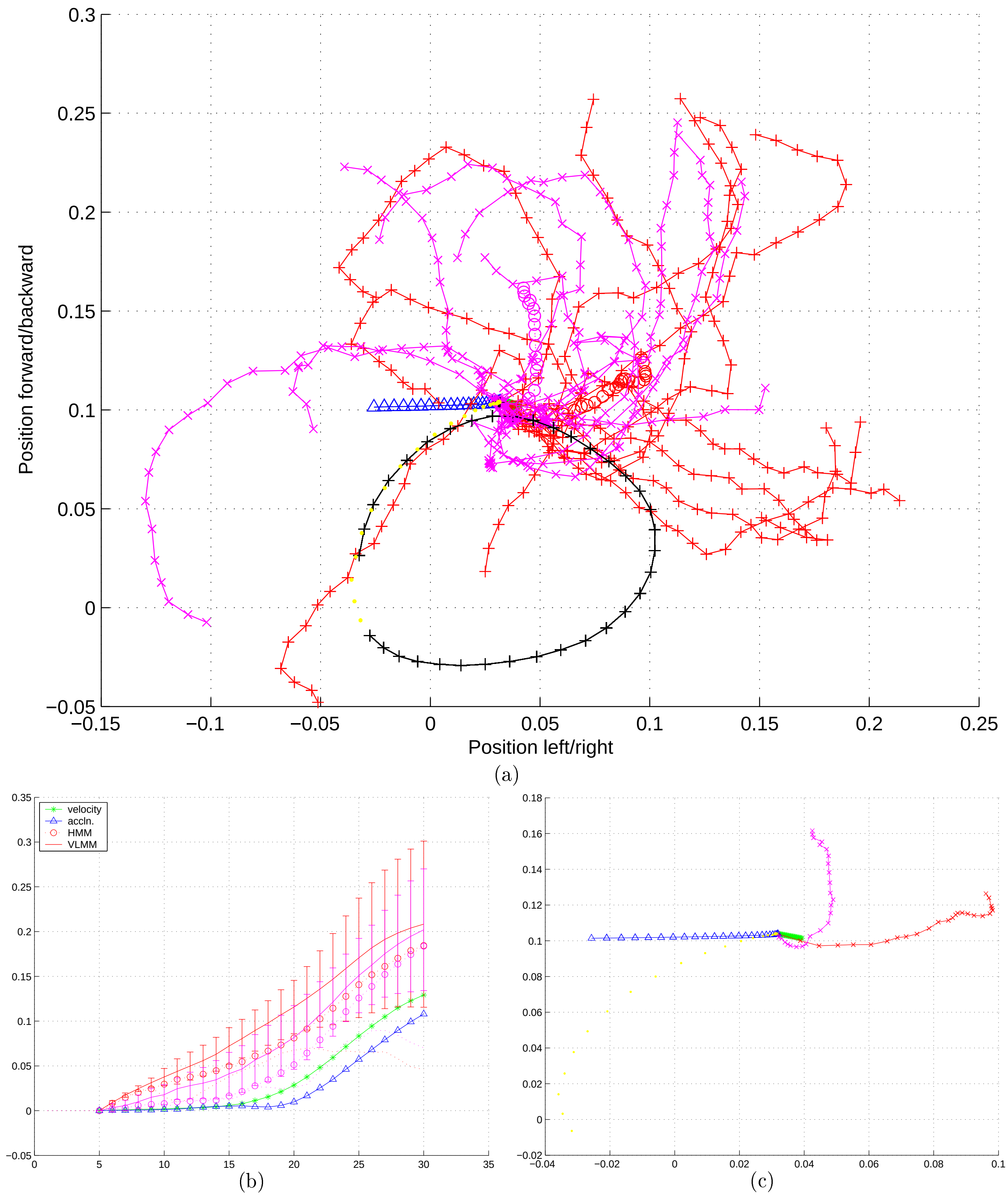
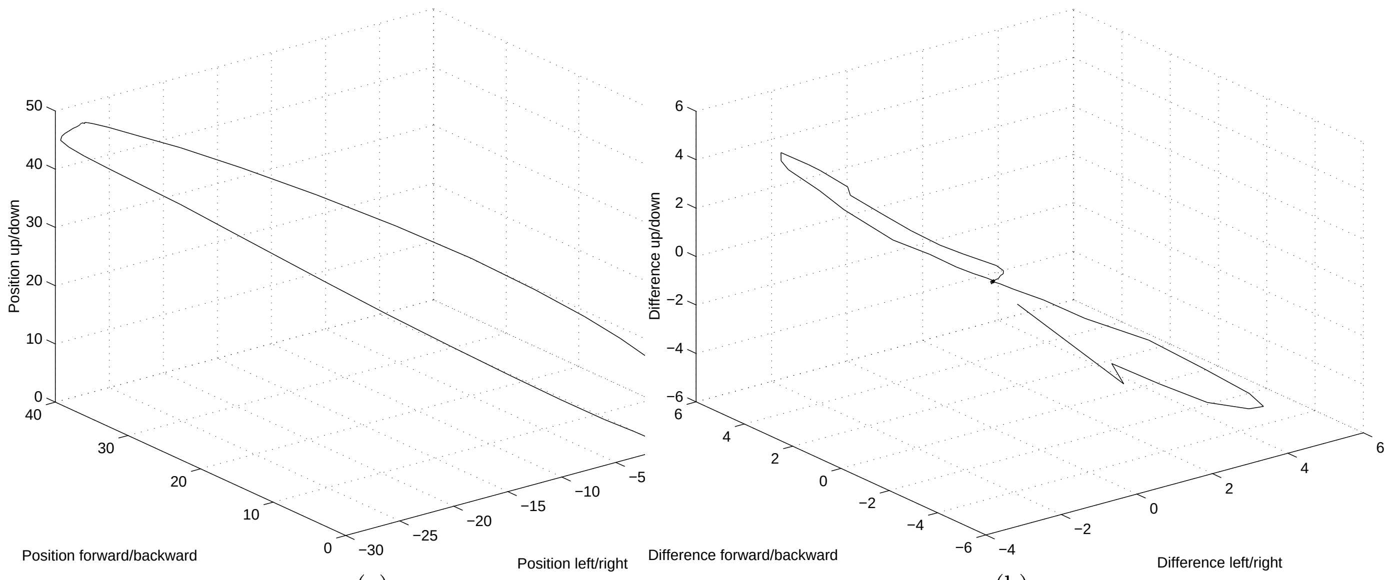


Figure 8: 2-D tracker experiment 3: constant velocity (green), constant acceleration (blue), first-order HMM (red) and VLMM (magenta), ground truth (unoccluded section in, black and occluded section in yellow), (a) as reconstructed 2-D trajectories, (b) the Euclidean distance between the trajectory of each method and the actual original hand trajectory and (c) the mean reconstructed trajectories.

learnt from the training data, though the first-order HMM produces more variability and exhibits less structure in its predictions. For Figs. 6 and 7 overall, the VLMM provides the closest fit to the



(a) (b)

Figure 9: Experiment 4: 3-D hand tracking training exemplar 01 as (a) positional data, (b) first derivative

original hand trajectory, reflecting the fact that the VLMM is able to use the longer histories to make quantitatively better future predictions than the first-order HMM. The only drawback to the VLMM approach can be seen in Fig. 8 where the occlusion has started very early in the exemplar (at step 6). At this early stage of the motion, the VLMM has not yet observed sufficient track history to determine the internal state in the space of transitions between variable length states. Prior to starting tracking, the VLMM forward evaluation trellis was initialised with a uniform prior over all its states (uniform π). Thus the VLMM is acting as a 'probe' to generate a number of plausible different possible paths when the occlusion occurs. This is not a problem unique to our VLMM implementation, it is common to any high order temporal model. We have since refined our learning method to include an estimate of the prior over the M variable length states at time $t = 1$ using $P_\Omega(Q, t = 1)$ from the application of the forward backward algorithm.

3.2 Experiment 4: 3-D example

The gesture training data used for this experiment was taken from the HUMOSIM database. The database contains 3-D hand trajectory data collected from 22 subjects of varying gender, age, and height. 210 target locations and hand orientations were used, giving a total number of 4,410 trials and the 8,820 reach movements.

We only used a small subset (17 exemplars) of the wealth of available training data relating to a simple reach out/pickup and return type series of gestures. An example of a typical 3D HUMOSIM training exemplar is shown at Fig. 9. We pre-processed the data so that consecutive timesteps with a Euclidean distance of less than 0.5cm were removed so that the resulting motion did not contain periods of stasis. It is not unusual for some of the HUMOSIM exemplars to have periods of 60-70 timesteps of stasis. To model this un-processed data would require a VLMM with an equivalent depth, but the essence of the interesting motion is captured in a considerably smaller model. Without the pre-processing, this imbalance in temporal model would give rise to a PST with some very deep trees sections to capture the lengthy periods of stasis, with other less deep sections that captured, for example, most of the inwards and outwards motion segments. Such a tree would be very unbalanced (i.e. have significant variations in root to leaf node path lengths). We are currently researching more efficient methods for representing such unbalanced problems using hierarchical approaches.

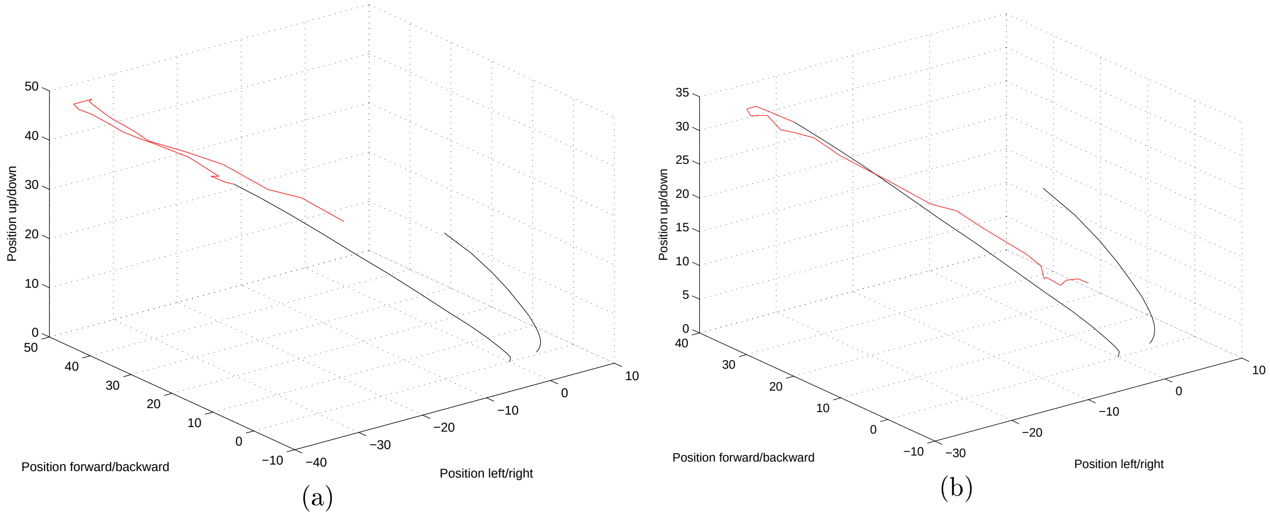


Figure 10: Example 3-D generative trajectories using the VLMM model in experiment 4.

We again used an alphabet of 10 Gaussian mixtures describing the 17 training exemplars as first derivative velocity data. As for the 2-D case, we then trained a VLMM with a maximum depth $L = 10$ and $\epsilon = 10^{-5}$. The resulting PST had 3731 nodes with a corresponding PFA with 3358 nodes. As before, we further refined the mean and co-variance parameters to have one per variable length state. Once again we created testing data by taking training data exemplars and setting our occlusion flag over some time interval. This test used a very much larger period of occlusion of 21 consecutive timesteps. We arranged the occlusion such that it started roughly half way through the 'reach out' phase and ended roughly halfway through the 'reach in' phase. This level of occlusion represents a serious challenge to any tracker not only due to the relatively extended period of occlusion but also because the predictive element has to deal with the reversal of motion at the maximal out reach point. To deal with such occlusion requires a higher order knowledge of the underlying temporal process than would be afforded by either naive trackers or first-order HMMs. In fact, these trackers performed so poorly in this task that the results are not presented here. Instead we show some of the VLMM tracker results at Fig. 10. Here we can see that the VLMM tracker has predicted the motion reversal during the occlusion period. The deeper structured representation of the underlying motion allows the VLMM to considerably out-perform the other models in the previous experiment in generative extrapolation.

4 Joint learning of temporal and spatial structure experiments

Learning in task based visual control, and computer vision more generally, spans two separate (but related) problems. On the one hand, we need to learn the spatial structure of task relevant objects and, on the other, we need to learn the temporal dynamics (behaviour) of those task relevant objects. Classical approaches to computer vision have viewed these as separate problems (e.g. determination of spatial structure as a segmentation task, or analysing behaviour as a statistically time dependent process). Recent developments have included combining spatial and temporal learning into a single inter-dependent process (such as Frey and Jojic's transform invariant mixture of Gaussians, [13, 7]) and the application of cognitive principles to exploit dependencies between the two modes of learning (Moore, Essa and Hayes [14]).

A further key aspect related specifically to the notion of control is the use of models that have generative properties. So, for spatial structure we can generate pixel values (visualise structure prototypes) and for temporal dynamics we can use constraints on the step changes in the hidden variables to predict future observation data.

4.1 Learning spatial structure

If there is no prior information about the spatial structure then the problem is one of discovering a set of prototypes (a model λ) that best fits the training (observation) data O, i.e. we need to maximise $P(\lambda|O)$. There is usually no computationally tractable globally optimal or closed form solution for this task and the problem falls to gradient ascent techniques such as Dempster's Expectation Maximisation (EM) approach [5], genetic algorithms, random search and so on (typically, any method that uses a likelihood function to model $L(O|\lambda)$). Additional variables may be incorporated into the EM learning. Frey and Jojic [8] use transformation indices (i.e. cluster transformation pairings) to create transform invariant mixture of Gaussian models. Further work by Frey and Jojic [6] showed how this process could be implemented computationally efficiently for all possible X and Y translations by modelling transformation as a convolution of the Fourier transforms of both the prototypes and O. Other types of invariance can then be achieved through appropriate co-ordinate system transformations (e.g. the log-polar form for arbitrary scale and rotations).

4.2 Joint learning of spatial & temporal dynamics

Jojic et al. [13] demonstrated that the learning of spatial and temporal structure can be usefully unified in their transformed HMMs. Their approach combines the learning of transform invariant Gaussian mixture models with Baum Welch re-estimation. The HMM parameters provide an informed prior to the prototype/transformation index that enables the mixture modelling to reach a more optimal solution than with a uniform prior distribution (i.e. ML) alone. Next we show, in a further extension of this work, how variable length Markov models can be combined with the Transform Mixture of Gaussian (TMG) models.

To demonstrate the various approaches we use a simple visual task summarised in Fig. 11 inspired by Frey and Jojic's pacman toy domain. In this task a 5x5 pixel 'pacman' shape follows a path defined by a cross on 25x25 grid. This path consists of 56 steps and can be summarised by a chain code. Each step of the path defines a 25x25 image with the appropriate pacman shape (always facing in the direction of travel) superimposed on a uniformly distributed noise background. The image sequence for the task then consists of 10 serial repetitions of the sequence giving a total of 560 frames. We constructed this task so that it contains higher order (and variable order) Markov temporal dependencies.

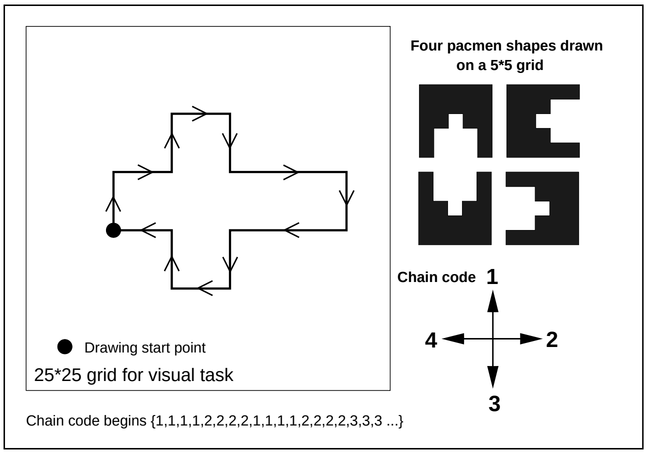


Figure 11: Summary of simple visual task with a chain code defining the direction of movement along the path for the pacman shape.

The TMG method learns a mixture of Gaussians model for a number C of different visual cluster prototypes normalised over a set of spatial transformations. Each cluster $c \in C$ has a latent image z_c

formed by mean (μ) and co-variance (Φ) parameters:

$$p(z_c) = p(z|c) = \mathcal{N}(z; \mu_c, \Phi_c) \tag{13}$$

The joint distribution of a training set image $x_t \in \mathbf{X}$, a transform l drawn from a set of L spatial transformation matrices $\Gamma = l_1, ...l_L$ and the latent image corresponding to cluster c with post-transformation noise Ψ can be defined as:

$$p(x_t|l, z_c) = \mathcal{N}(x_t; \Gamma_l z, \Psi) \tag{14}$$

$p(x_t)$ can then be defined by marginalising the expression for $p(x_t|l, z_c)$ over all $z_{c \in C}$ and Γ. Frey and Jojic then cast the learning of the cluster parameters μ_c and Φ_c as well as a post-transformation noise parameter Ψ as an Expectation Maximisation (EM) problem by setting the derivative of $\log p(x)$ with respect to the model parameters equals 0 and deriving the corresponding E and M step update equations. Frey and Jojic have produced an efficient implementation of this EM process for all possible $X \times Y$ translations using a convolution of the Fourier transform of the clusters with the training data. [3]

Other types of transformations can be accounted for by pre-processing the training set data by an appropriate co-ordinate transformation, such as the log polar transform for dealing with scale and rotation. For multiple transformations, such as X and Y translation and scale and rotation simultaneously, Frey and Jojic further developed a variational technique for decoupling transformation sets to reduce the computational complexity associated with multiple transformations.

4.3 Experiment 5: Applying VLMMs to the example task

4.4 Integrating spatial and temporal learning

Frey and Jojic's original published MATLAB code is concerned with learning spatial structure only [8]. Starting with the assumption that $P(C, L) = 1/CL$ for all timesteps T (uniform prior) and random values for the mean and variance values of the C cluster prototypes, it computes a post transformation probability estimate term $P(C)$ and then uses a ML estimate derived from that to inform the $P(C, L)$ prior at the next iteration ($P(c)/L$). The prior is assumed to be uniform across transformations, i.e. $p(c, L) = k$ where k is a constant.

We first modified their approach so that the $P(C, L)$ prior could be re-estimated independently for each frame so that we could use temporal models to constrain the prior estimates over time giving us $P(C_t, L_t)$. Jojic et al. [13] used this idea to build a first-order Markov model onto their original work. The distribution of $P(C_t, L_t)$ depends not only on the individual post transformation probability $P(C_t)$, but on $P(C_{t-N}, L_{t-N})$ for an N-th order Markov process. Having computed $P(C_t)$ for each frame independently, we were then able to use variable length Markov modelling to determine $P(C_t, L_t)$ for the next iteration. This has the effect of improving the cluster prototypes by ensuring that they are estimated from the members of the training set that it is believed correspond to that cluster structure. Each training image contributes to the post transformation probability estimate term according to how likely it was based on the previous N training set images.

For our application Σ corresponds to the cluster indices and we model timestep to timestep cluster indices. More sophisticated modelling would expand Σ to model the cluster index plus other parameters such as relative motion, but there is a trade-off between improving the spatial description of the derived clusters and the additional computational complexity of the VLMM.

We then use the PFA to estimate $P(C_t, L_t)$ prior for the next iteration. To do this we generate a forward evaluation trellis matrix of size $C * T$ which specifies how likely each cluster is for each frame in the training image set. We first create a forward evaluation trellis of size $M * T$.

[3]The source code for this implementation can be found at `http://www.psi.toronto.edu/~frey/tmgEM.m`

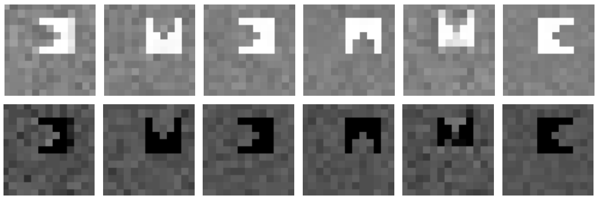


Figure 12: (a) Mean and (b) variance maps for experiment 5 learning the spatial structure for the pacman visual task. The spatial structure was constrained by the temporal structure provided by a VLMM

During the initial few iterations, the VLMM generates C states (one for each cluster) and a first-order Markov model with each state having transition probabilities $\approxeq 1/C$. As the learning process continues and spatial structure starts to emerge, typically one of the cluster prototypes will acquire a value $p(c_t)$ significantly greater than the $1/C$ value (until the spatial learning has fully specialised the learned structure). This causes an attractor effect in the VLMM learning as the statistically most likely explanation for the temporal ordering will be a first-order VLMM with the transition probability from the attractor state to itself for all timesteps. To overcome this attractor, we normalise the prior $P(C, L, T)$ prior to each EM iteration such that $\Sigma_T P(C_t, L_t) = k$.

4.5 Results

We trained our combined algorithm for 100 iterations with 6 clusters, allowing the VLMM to extend up to a depth of 5. The resulting mean and variance maps from the 6 clusters are shown in Fig. 12. Dark areas in the variance maps correspond to low variance. In contrast, a N-th order Markov model would require $N * N^k$ parameters for its state transition matrix. So a third order HMM with equivalent representational power would require 1,296 parameters for its state transition matrix compared to a VLMM with 211 nodes in its PFA. This reduction in the number of transitions required to model the domain reflects knowledge about the temporal structure of the task.

The resulting VLMM prediction suffix tree consisted of 253 nodes (including the root node) and the PFA had 211 nodes of length either 2 or 3. The shorter length nodes tended to correspond to portions of the task concerned with 'going straight ahead' and the longer ones with 'turning corners'. Clusters 2,3,4, and 6 captured most of the spatial structure for the task.

5 Conclusions

We have extended Ron's original VLMM formulation to deal with continuous data representing probabilistic fit with an alphabet at each time step rather than just discrete valued transition. We have shown that this version of the VLMM can be used to learn structured motion (i.e. gestures) in a hand tracking task. The resulting model has improved generative properties over the first-order HMM that are a useful tool in dealing with tracking issues such as target labelling consistency in the face of mid-term occlusion.

We have also shown that variable order temporal modelling can be applied to a joint spatial and temporal learning task. The resulting model exhibits both spatial and temporal generative properties that have applications in the learning of visual control strategies in cognitive vision systems. Given the current state of the joint model, we may stochastically predict the next. We plan to extend our experiments with real world images and to demonstrate an on-line control application as well as investigate other methods for model trimming. Model trimming is especially important in ensuring that the solution found is the most naturally interpretable one [3].

One major challenge to learning for task-based control is to learn hierarchical representations 1) for the case where the data has multiple levels of granularity, e.g. trajectories with long intermediate static frames, but also, more importantly, 2) for the case where the task requires context of different types to determine future processing, e.g. activity models which require hand state and gesture phase in the higher levels of the ActIPret system. Extensions are being analysed for the former using 2 strategies which avoid pre-segmentation of the data, one relies on context variables (see [18]) and the other involves extended unsupervised learning of the hierarchical organisation of data clusters. For the latter, we are currently using partially hand-coded models with supervised data learning where required on the dynamic links (see [11]). Further work on this approach for the ActIPret project is described in our upcoming paper [19].

Acknowledgements

This work was supported by the EU-Project ActIPret under grant IST-2001-32184. The authors gratefully acknowledge the invaluable help provided by the Laboratory for Human Motion Simulation (HUMOSIM) at the University of Michigan, USA in allowing us access to their 'Terminal Hand Orientation and Effort Reach Study, 2000' hand trajectory database.

References

[1] A. Argyros and M. Lourakis. Real-time tracking of multiple skin coloured objects. In *Proceedings of European Conference on Computer Vision*, pages 368–379, Prague, Czech Republic, 2004.

[2] D. Ballard. Animate vision. *Artificial Intelligence*, 41:57–86, 1991.

[3] M. Brand. Structure learning in conditional probability models via an entropic prior and parameter extinction. *Neural Computation*, 11(5):1155–1182, 1999.

[4] D. S. Clouse, C. L. Giles, B. G. Horne, and G. W. Cottrell. Time-delay neural networks: Representation and induction of finite-state machines. *IEEE Transactions on Neural Networks*, 8(5):1065–1070, 1997.

[5] A. P Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39:1–38, 1977.

[6] B. Frey and N. Jojic. Fast, large-scale transformation-invariant clustering. In *Advances in Neural Information Processing Systems*, Cambridge, MA, 2002.

[7] B. Frey and N. Jojic. Transformation-invariant clustering using the EM algorithm. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002.

[8] B.J. Frey and N. Jojic. Estimating mixture models of images and inferring spatial transformations using the EM algorithm. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 416–422, Fort Collins, Colorado, USA, 1999.

[9] A. Galata, N. Johnson, and D. Hogg. Learning variable length markov models of behaviour. *Computer Vision and Image Understanding*, 81(3):398–413, 2001.

[10] S. Gong and H. Buxton. On the expectations of moving objects. In *European Conference on Artificial Intelligence*, pages 781–785, Vienna, Austria, 1992.

[11] R. Howarth. Interpreting a dynamic and uncertain world: Task-based control. *Artificial Intelligence*, 100:5–85, 1998.

[12] N. Johnson and D. Hogg. Representation and synthesis of behaviour using gaussian mixtures. *Image & Vision Computing*, 20(12), 2002.

[13] N. Jojic, N. Petrovic, B. Frey, and T. Huang. Transformed hidden markov models: Estimating mixture models of images and inferring spatial transformations in video sequences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, South Carolina, USA, 2000.

[14] D. J. Moore, I. A. Essa, and M. H. Hayes. Exploiting human actions and object context for recognition tasks. In *Proceedings of International Conference on Computer Vision*, pages 80–86, 1999.

[15] L.R. Rabiner. A tutorial on hidden Markov models. *IEEE*, 77:257–286, 1989.

[16] D. Ron, Y. Singer, and N. Tishby. The power of amnesia. In *Advances in Neural Information Processing Systems*, volume 6, pages 176–183. Morgan Kaufmann Publishers Inc., 1994.

[17] K. Sage and H. Buxton. Joint spatial and temporal structure learning for task based control. In *International Conference on Pattern Recognition*, Cambridge, UK, 2004.

[18] K. Sage, A. J. Howell, and H. Buxton. Developing context sensitive HMM gesture recognition. In *Gesture-based communication in Human Computer Interaction*, volume 2915, pages 277–287. Springer LNAI, 2003.

[19] K. Sage, A. J. Howell, and H. Buxton. Temporal bayesian belief networks for the recognition of action, activity and behaviour. *submitted to Computer Vision and Image Understanding*, in press.

[20] T. Starner and A. Pentland. Visual recognition of American sign language using hidden markov models. In *International Workshop on Automatic Face and Gesture Recognition*, pages 189–194, 1995.

[21] C. Vogler and D. Metaxas. ASL recognition based on a coupling between HMMs and 3D motion analysis. In *International Conference on Computer Vision*, pages 363–369, 1998.