The Role of Task Control and Context in Learning to Recognise Gesture

Hilary Buxton, A.Jonathan Howell and Kingsley Sage
School of Cognitive and Computing Sciences,
University of Sussex, Brighton BN1 9QH.

Abstract

We are interested in methods for building more intelligent cognitive vision systems in our ActIPret project. The aim of this project is understanding the activities of expert operators for teaching and education. Our approach is to learn models for the components and later the task and context of the visual processing in the ActIPret system. The paper first introduces general issues and some approaches for the example of gesture learning and recognition. Second, aspects of our cognitive vision framework are described as they are relevant to the evaluation of the two approaches tested here. Third, the computational models for the time delay RBF (TDRBF) network and Hidden Markov Model (HMM) are described and results given. Finally, extensions of this work and conclusions for system integration of the results are discussed in the light of task-based control and contextual processing.

1. Introduction

What do cognitive vision (CV) systems entail? The basic approaches combine techniques from symbolic or subsymbolic AI with computer vision techniques in some way. Naturally, we then encounter many of the major issues in AI such as knowledge representation and reasoning, control and the handling of uncertainty, as well as machine learning. Much of the work assumes that knowledge drives reasoning in visual interpretation (*seeing as*), thus visual context is seen as essential for understanding what is depicted in images or image sequences. If we are to build efficient systems that can tackle many different tasks, high-level attention and control (*seeing for*) is also seen as essential. In addition, if we are to incorporate scene and task knowledge, we have to address the question of how such knowledge can be acquired (*learning*). These issues are illustrated in this paper using prototype components for gesture analysis in the ActIPret Project.

The ActIPret framework is designed to support the essential elements of CV: *memory* organisation of representations for objects, actions, behaviour strategies etc.; *reasoning* about these representations to support flexible decisions and actions in the system; *learning* of both the task-relevant representations and how to use them; and *control* of both viewing geometry and selective visual processing. A schematic diagram is given in Fig. 1 and briefly described in the next section to give a context to the work reported here. We are taking a system-based approach to development of these capabilities that includes embodiment with movable cameras. There are several approaches possible for each of the subproblems: task-based control, cognitive learning and interpretation of actions and activity sequences and learning to detect and react to early visual cues. Our objectives are to realise a CV system that can scale up in complexity through learning, with distributed control and robust performance through information integration.

The general problem of *task based control* involves selecting processes and setting tuneable parameters to get the required information from images for a particular application. Such 'intelligent image processing' has received attention over the past decade: in Japan research has focussed on this problem [21], in the US [8] and in Europe general tools have been developed [6, 7]. For example, a real-time, knowledge-based program supervision approach has been applied [32]. The advantage of this approach is an explicit, declarative description of the relationships between program modules, their parameters and operating conditions etc. at the conceptual level. However, it is not clear how to distribute processing and integrate the information propagation in this scheme. The proposed task-based control for ActIPret adapts prior research on a Dynamic Decision Network (DDN) approach [14, 15], which uses probabilistic reasoning models with reactive planning for the Activity Reasoning Engine. The advantages are an evolving probabilistic interpretation that can be demanded at any time in the processing and control based on utility/priority of the results with respect to the task.

Generative graphical models such as the Bayesian Belief Network (BBN) or Hidden Markov Model (HMM) are widely used at a more *cognitive level in visual processing* since they support not only learning but also some kinds of contextual processing and task control, eg. [5]. For an introduction to probabilistic reasoning in these models see [25] and for more variational learning methods see [20]. Here

we focus on the HMM for gesture analysis, which can be made sensitive to the detailed task context. One advantage of HMMs is that the 'hidden' purposes of regular behaviour patterns can be easily learned from examples, i.e. the structure of the model as well as the parameters are easily learned [30]. For example, in early work [13] the movement patterns of vehicles on an airport ground-plane were learned to support predictive tracking. The Gaussian is usually assumed as underlying measurement model so HMM models can be regarded as extending Gaussian mixture models by having learnt dynamic dependencies between states. These have a chain of simple dependencies on the immediately previous state but can be extended to coupled dependencies with states in another HMM to form a Coupled Hidden Markov Model (CHMM) [4] or to longer term temporal dependencies with previous states in the same HMM to form a Variable Length Markov Model (VLMM) [11].

Artificial Neural Network (ANN) techniques are a powerful, general approach to pattern recognition tasks and will work robustly for *adaptive recognition and reaction to behaviour cues*. There are a wide variety of different statistically motivated learning methods for such models, as seen in [2]. Classical networks do not include a time dimension so they have to be adapted to deal with dynamic scene analysis. Some extended models have implicit time like the partially recurrent networks of Elman [10] and Jordan [19], which represent temporal context by copying back the hidden or output node states. However, these networks are hard to train due to poor convergence [29]. Time can also be explicitly represented in the architecture at the network level using the connections or can be represented at the neuron level as in 'spiking networks' (for review see [12]). However, they have yet to be widely applied in visual processing as there is ongoing debate about how best to propagate information to support different tasks. Here we focus on time-delay RBF networks, which do not suffer from such problems, for application to our gesture analysis.

In what follows we first set the scene by describing the ActIPret framework to support cognitive vision (CV) tasks. This is followed by descriptions of the computational models for the prototype gesture recognition components and the 3D hand trajectory data sets used in generating results. The performance for both the learning and recognition phase using these models is then contrasted in terms of generalisation ability. Extensions of the methods to allow task control and more context sensitive processing are then discussed with conclusions and suggestions for further work.

2. ActIPret Framework

The *goal-oriented* nature of CV can be implemented by dynamic selection of the components that are best suited to solve the immediate task. The major task-based control

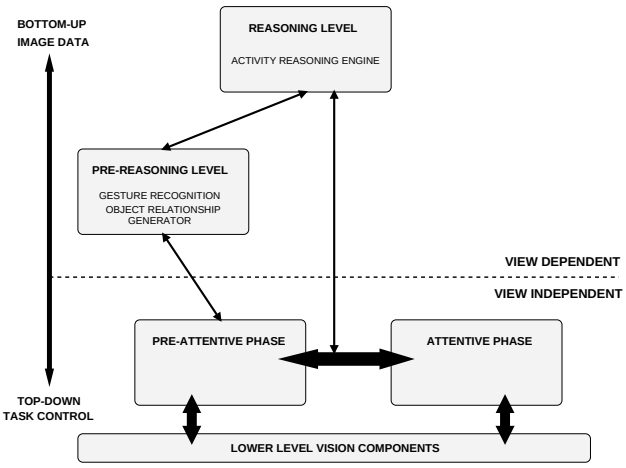


Figure 1. A block diagram outlining the integrated system.

is at the synthesis and reasoning level, that is, the 'control policy' makes the overall probabilistic decisions to decide what processing is next for the system. However, all levels provide some task-based control for lower level processes, from components responsible for recognition and tracking down to the selection of the most relevant views or features. This task-based control is most flexible if these components have some reasoning capability. In our framework, belief values will form the formal basis of probabilistic reasoning processes. In the longer term, the task-based control strategies will be learned in the context of the complete system. However, at first they will consist of hand-coded utility/priority estimates together with appropriate matrices of conditional probabilities for requests to the lower level components.

The *distributed* nature of our framework is supported by a 'service principle' to allow the system to perform quickly when the complexity of the tasks is high. This enables handling of the results of external processing from a service that has been requested together with internal processing. The information integration offered by probablistic reasoning methods will ensure that knowledge available in each component can be exploited in the current context. This also means that short-term memory is distributed to each component as it is related to the reasoning and control. However, there can also be learned models in long-term memory which persist and are handled by a model server.

The *extendable* nature of our framework will also be supported by learning, both 'off-line' and 'on-line'. Initially learning takes place off-line ('learning phase') on a component by component basis, eg. learning to detect and recognise gestures or learning of BBNs for activity interpretation. For gestures, 3D hand trajectories from the hand tracking component are the input for learning to discriminate

between task-related gesture classes for a variety of users. Later, in the context of the full system, on-line learning can improve this processing in a top-down manner by exploiting information at the interpretation level. Some kind of refinement in the context of a particular task scenario demonstrated by an ideal user ('expert phase') is envisaged. It is also possible that we could adapt the system to specific users when it is used for instructing trainees ('tutor phase'). Here, we first compare two approaches to the off-line gesture analysis.

3. Computational Models for Gesture

3.1. Time Delay RBF Network

The RBF network is a two-layer, hybrid learning network [22, 23], which combines a supervised layer from the hidden to the output units with an unsupervised layer from the input to the hidden units. The network model is characterised by individual radial Gaussian functions for each hidden unit, which simulate the effect of overlapping and locally tuned receptive fields. It is characterised by computational simplicity, supported by well-developed mathematical theory, and robust generalisation, powerful enough for real-time, real-life tasks [28, 31]. The nonlinear decision boundaries of RBF networks make better general function approximations than the hyperplanes created by the multilayer perceptron (MLP) with sigmoid units [26], and they provide a guaranteed, globally optimal solution via simple, linear optimisation. One advantage of the RBF network, compared to the MLP, is that it gives low false-positive rates in classification problems as it will not extrapolate beyond its learnt example set. This is because its basis functions cover only small localised regions, unlike sigmoidal basis functions which are nonzero over an arbitrarily large region of the input space.

Once training examples have been collected as input-output pairs, with the target class attached to each image, tasks can be learned directly by the system. This type of supervised learning can be seen in mathematical terms as approximating a multivariate function, so that estimations of function values can be made for previously unseen test data where actual values are not known. This process can be undertaken by the RBF network using a linear combination of basis functions, one for every training example, because of the smoothness of the manifold formed by the example views of objects in a space of all possible views of that object [27]. This underlies successful previous work with RBF nets for face recognition from video sequences [17], which uses an RBF unit for each training example, and rapid pseudo-inverse calculation of weights. An important factor in this approach is the flexibility of the RBF network learning approach, which allows formulation of the training in terms of the specific classes of data to be distinguished. For example, extraction of identity, head pose and expression information can be performed separately on the same face training data to learn a computationally cheap RBF classifier for each separate recognition task [9, 18].

To extend this research to support *visual interaction*, generic gesture models are developed here for the control of attention in gesture recognition. In previous work a time-delay variant of the Radial Basis Function (TDRBF) network recognised pointing and waving hand gestures in image sequences [16]. Characteristic visual evidence is automatically selected during the adaptive learning phase, depending on the task demands. A set of interaction-relevant gestures were modelled and exploited for reactive on-line visual control. These were then interpreted as user intentions for live control of an active camera with adaptive view direction and attentional focus. For ActIPret, some of the ideas for zooming in on activities can still be exploited. Also the gesture recognition is an excellent predictive cue for many of the actions and activities in our ActIPret scenarios. At the earlier levels of processing, but particularly in the gesture recognition, reactive behaviour is important for both camera movement and invoking further 'attentional' processing. The scheme is adapted here to accept 3D hand trajectories for predictive gesture recognition. The gesture recognition uses tri-phasic gesture detectors as in our previous work on predictive control [18].

3.2. Hidden Markov Model

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 [30]. 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 (at time $t = 0$) and B is a $N * M$ matrix accounting for the mapping between the N hidden states and the M output (observable) symbols.

Whilst the internal N hidden states are always discrete, the M output states may be discrete (in which case B is a probabilistic confusion matrix) or continuous. Where the output states are continuous symbols, or more generally, continuous vectors, the B probability density generally takes the form of a measure of probability that the vector will be between x and dx. The most commonly used form if this density is the Gaussian Y-component mixture density. Then, observation symbols are modelled as mixtures of Y Gaussian components (Y is the dimensionality of the observation feature space). B then accounts for the relationship between the hidden states and the parameters of the Gaussian components. For a good account of parameter estimation for continuous densities see Bilmes [1].

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, and most relevant for what follows, given O we can estimate model parameters λ that maximise the probability of O. The most common form of HMM model parameter estimation is the Baum-Welch algorithm (described in [30]) which is an iterative non-globally optimal procedure for maximum likelihood estimation.

To train a HMM using Baum-Welch, a set x of training observation sequences $O_1, O_2, ..., O_x$ are presented to the iterative procedure. This is an unsupervised learning process as there is no annotation or notion of correct/incorrect observations; the procedure simply finds the best possible model λ that it can. The user specifies the number of hidden states to be used. The resulting λ can then be used to estimate the probability of previously unseen observations (i.e. used as a classifier), or used to probabilistically generate exemplars based on the model. This presents a challenge in determining the size of the generalised representation (hidden internal structure) necessary to capture the full N-dimensional dynamics of the training set. Too minimal a structure will result in over generalisation and poor generative properties, too extensive a structure will result in over-fitting and loss of generalisation.

To capture gesture models, we use a continuous output HMM with training observation sequences represented as 6-valued vectors (2 sets of 3-D hand velocities – see next section) with the observation symbols modelled as 6-component mixture of Gaussian functions. We then vary the number of internal discrete hidden states to explore the underlying dimensionality of the training set (which corresponds approximately to the number of distinct gesture phases) and to demonstrate the ability of the HMM to distinguish the learned gesture from other gestures.

3.3. Gesture Data

The gesture data used for the experiments in this paper was the *Terminal Hand Orientation and Effort Reach Study* Database created by Human Motion Simulation at the Center for Ergonomics, University of Michigan, USA. 3-D hand trajectory data was collected from 22 subjects of varying gender, age, and height. Nineteen of the subjects were right-handed and two were left-handed. 210 target locations and hand orientations were used, giving a total number of 4,410 trials and the 8,820 reach movements.

Fig. 2 shows the target system for the HUMOSIM hand trajectory data. Four towers were used, from 45° left of the subject to 90° right, each of which had three 'pods' as targets. There is further variation in the targets, as each of the pods has five cubes, each of which can use four hand orientations. For the experiments in this paper, we consider only tower/pod combinations (12 in all). Each trial produced a file of 3-D locations for two points on the subject's hand. For each trial, data was collected at 25Hz for a sequence consisting of five distinct phases:

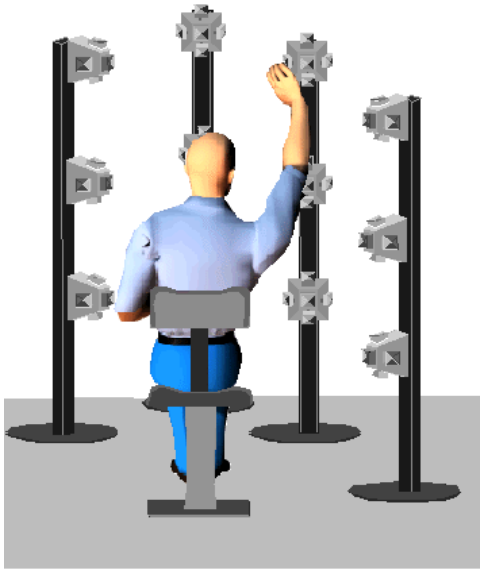


Figure 2. The target system for the HUMOSIM hand trajectory data.

- Start with a static hand placed at a 'home location' on the subject's leg, followed by:
- A movement toward the target, which we term *get*;
- A static phase while the hand is at the target;
- A second movement, away from the target, which we term *return*;
- A final static phase at the home location.

Each resulting datafile contained 80–135 timesteps.

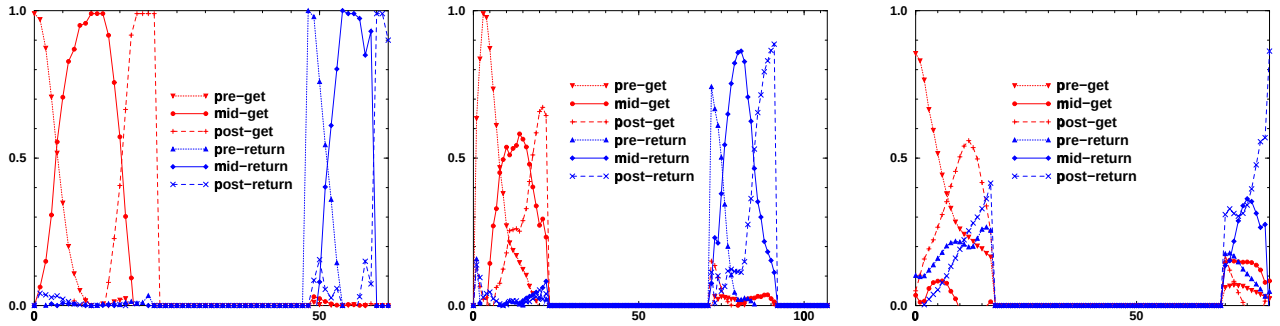
4. Results

To analyse and compare the TDRBF and HMM methods, we consider one specific learning task: learning a subset of trajectories for Tower 0, and testing generalisation by varying the test trajectories for all four tower positions.

For both methods, the 3-D location data was preprocessed by differencing it from one time step to the next (relative motion or velocity data).

4.1. TDRBF Performance

To train the TDRBF network, we used a fixed time delay length of six time steps, and segmented the training data automatically according to the level of relative motion within successive time delay segments. Based on the definition of



(a) (b) (c)

Figure 3. Generalisation for TDRBF network trained with targets in Tower 0 (45° left), when tested with complete hand trajectories from (a) Tower 0, (b) Tower 1 and (c) Tower 2. Values for output units for each gesture phase class (y axis) are shown for each time step (x axis).

the trial data above, we assume two distinct movements are contained in each trial data file, with static periods in between. We impose three phases within each of these movements: a *pre-phase*, at the start of movement a *mid-phase*, at the midpoint between start and end of movement and a *post-phase*, at the end of movement. If we add an extra class for stasis, or no movement, this gives seven classes:

- *pre-get, mid-get, post-get*
- *pre-return, mid-return, post-return*
- *stasis*

The three-phase structure for gesture classification is based on previous work [18], where we found it allowed more reliable recognition as well as supporting prediction. The strategy was to only accept specific sequences of phases as real gestures, eg. the pre-phase needed to be observed before the mid-phase, and confirmed by the post-phase to support appropriate attention frame shifts for visual interaction. Time delay segments with very low levels of relative motion are ignored by the TDRBF network and immediately classified as static.

To test the trained TDRBF network, we presented complete trajectory files from targets not used for training. Fig. 3 shows the results for a TDRBF network trained with 19 trajectories from target 3, which is on Tower 0 (45° left), when tested with another trajectory on Tower 0, for target 212 on Tower 1 (0°) and target 321 on Tower 2 (45° right). Static time steps are denoted by all outputs set to zero.

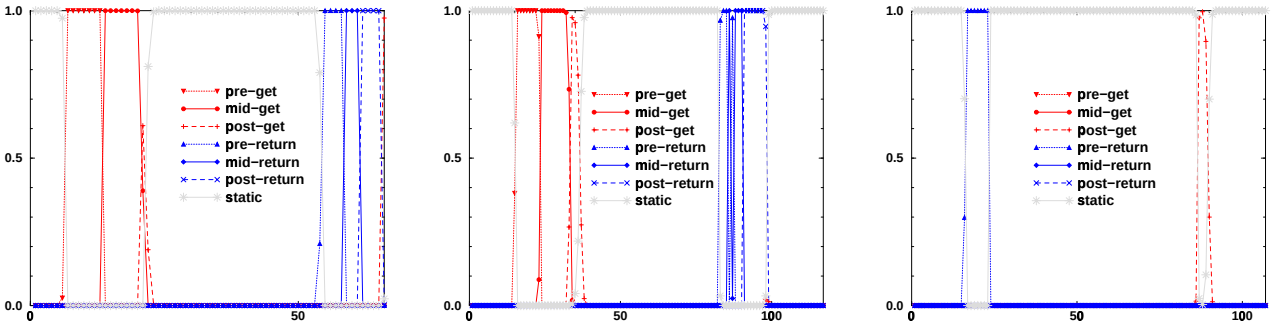
Smooth transitions can be seen between phase classes, and all time steps are correctly classified, even for specific people and timesteps not included in the training set. A gradual degradation in generalisation is seen as the angle between train and test data increases. A further test to classify data from Tower 3 (90° right) was attempted, but the network was not able to classify any part except the static phases.

4.2. HMM Performance

The HMM was trained using a fixed number (typically 1000) iterations of the Baum-Welch algorithm with a variable number of hidden states. Baum Welch produces a non-globally optimal solution to maximise $p(\lambda|O)$. We then used the resulting model λ to 'classify' examples. Classification is normally taken to mean estimating $p(O_{novel}|\lambda)$ by forward evaluation, but here we mean it to refer to Viterbi decoding where we wish to find the most probable internal sequence of hidden states for O_{novel}. This is because we wanted to compare the transitions between the hidden states with the functional gesture phases defined for the TDRBF model.

Viterbi decoding first requires the forward evaluation procedure. The probability values in the forward evaluation trellis tend geometrically toward zero (as we are always multiplying together values that are less than 0). In order to avoid mathematical underflow, we normalised each trellis column. This does not affect the states transitions, but does re-scale the terminal $p(O_{novel}|\lambda)$. This normalised data effectively represents the *relative* contribution made by each Gaussian function at each time step. This makes the data directly comparable with TDRBF model. All we then needed to do was to decode the relationship between the abstract numbered hidden states and the functional gesture phases. This was achieved by comparing the hidden state transition sequence for an observation sequence that had already been trained (on benchmark sequence) with the functional gesture phases.

A 'good classification' of a novel observation sequence is the defined as one where the hidden state transitions are qualitatively the same as for the benchmark sequence. A formal classifier metric can then be defined as the fit between the two sets of transitions. The number of hidden states in the HMM was varied between three and thirty,



(a) (b) (c)

Figure 4. Test generalisation for an HMM trained with targets in Tower 0 (45° left), when tested with complete hand trajectories from (a) Tower 0, (b) Tower 1 and (c) Tower 2. Details as for Fig. 3, except for extra *static* class.

with low numbers being unable to adequately reconstruct the data, and higher numbers tending to overfit the data.

Analysis of the fit between the probability of the model parameters given the training data and the number of hidden states shows that the fit reaches an optimal point where low numbers of hidden states are matched against ability to generalise. In our particular task, this point was typically where the HMM had seven hidden states (similar to [24]. This supports the seven-class system imposed during the TDRBF network training since the HMM will cluster the to maximise the probabilities of correctly classifying the training data. Also, the solution found in this case can be interpreted in the same context as the RBF model.

To test the trained HMM, as with the TDRBF networks tests above, we presented complete trajectory files from targets not used for training. Fig. 4 shows the results for an HMM trained with 19 trajectories from target 3, which is on Tower 0 (45° left), when tested with another trajectory on Tower 0, for target 212 on Tower 1 (0°) and target 321 on Tower 2 (45° right). The seventh hidden state explicitly represents timesteps of minimal motion, interpreted as the 'static' gesture class. As with the TDRBF tests, generalisation gradually decreases as the angle between train and test data increases.

5. Discussion

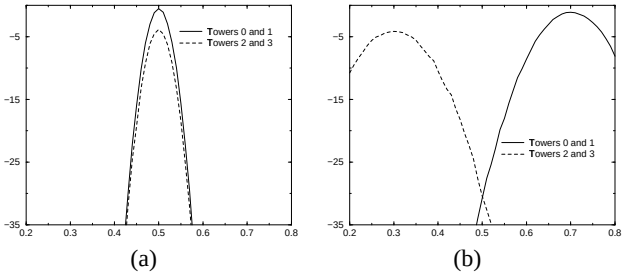
It can be seen that both the TDRBF and HMM approaches model the hand trajectory data efficiently, capturing the seven-state structure Fig. 6. In both, the temporal context was successfully captured during training and used in recognition. They can also both generalise over moderate variations in motion and position, at least one tower position (45° variation) and, in some circumstances, two tower positions (90° variation). The solution to handling a complete range of tower positions would be to provide training data covering a larger range of targets over several towers, as in previous RBF studies [17].

5.1. Task Control

Within the ActIPret system, the gesture recognition is done on demand from the reasoning level. It is part of prereasoning, i.e. gathering relevant evidence to support the interpretation of activity. The service principle within our framework is indifferent to the method actually embedded within the system components. Thus, we can consider either or both of these two approaches to gesture recognition but need further information about 'QoS', quality of service, and computation time. These measures affect how appropriate the service is for the immediate task as well as placing it somewhere along a continuum of attention (preattentive to attentive phase, as shown in Fig. 1). This, in turn can affect processing further down the system as the hand tracking services are also envisaged to operate with different QoS. In the more attentive phase there is more contextual information available and higher computational costs are acceptable since the number of possible interpretations should be low. This makes the HMM approach a more natural candidate for attentive processing and TDRBF for more reactive processing.

5.2. Using Context

In previous work [13], HMM trajectory prediction from entry regions through intermediate states to the re-fuelling or baggage-handling regions was augmented by updates on the position of vehicles from lower level vision for a known vehicle type. In general, scene context and aspects of the top-down interpretation or bottom-up visual information from moment to moment can be used to augment processing in an HMM without going to a full hierarchical BBN or DBN. In our case, additional context variables



(a) (b)

Figure 5. Classification of tower 0 and 1 trajectories as a function of context where training context (a) $context_1 = context_2 = 0.5$**, (b)** $context_1 = 0.7$ **and** $context_2 = 0.3$**.**

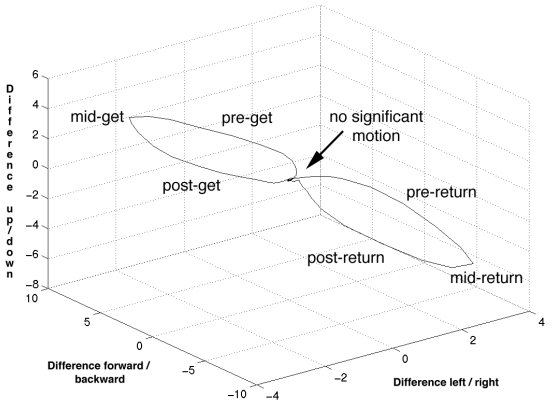


Figure 6. Example hand trajectory from the database: the line represents relative motion for *get* **and** *return* **movements, the centre point represents statis.**

could be introduced into the training data (for example, to indicate the target tower) [3]. These additional variables would cause separate Gaussian components to be generated in feature space such that each context would have an independent representation.

In order to demonstrate context control using HMMs, we selected four trajectory data sets, one each for the four towers with constant values for pod and cube. We then augmented the six value vectors generated for relative position with a single context value (a pseudo probability value). This single context value represents a gesture-context relationship with two context classes. $context_1$ is the value specified in each vector and $context_2$ is implied as $1 - context_1$. We assume that this context is provided by an external agent (perhaps an object classifier) but for this experiment the training context value is generated directly from a normal distribution about a mean, with a single context value generated for each time step.

We then grouped towers 0 and 1, and 2 and 3 together and trained 2 composite HMMs using the same seven hidden state structure as before. We pre-processed the data further slightly by removing timesteps where the sum of the absolute values across the six difference values was less than 1 cm. We then classified the towers 0 and 1 data using the two models and plotted a measure of model fit as a function of the context value. The measure of model fit used was the log_{10} of the mean (non-log) likelihood per observation symbol, or *lmlpos*. This measure varies in the range $[0, -\infty)$ where 0 represents a perfect fit between the model and the testing data at each time step and $-\infty$ represents a zero fit.

For the first test, we set the mean $context_1$ = $context_2$ = 0.5. The results are shown in Fig. 5(a), the y-axis showing the *lmlpos* value and the x-axis $context_1$. We see that for all test values of context that the model for towers 0 and 1 is preferred over the model for towers 2 and 3 but that the confidence of that fit is maximised where the testing context matches the training context and falls away either side of that value. To see the real value of context control, we then set the mean $context_1$ = 0.7 and $context_2$ = 0.3. The results for this are shown in Fig. 5(b). This time we see that when classifying the tower 0 and 1 examples, the towers 0 and 1 model is preferred when the context is around 0.7, but that as the context approaches 0.3, the model for towers 2 and 3 is preferred, albeit at a lower level of confidence. In other words, an extended generalisation (in trajectory terms) of the towers 2 and 3 model is being preferred over the towers 0 and 1 model around that value of context.

6. Conclusion

We have contrasted two prototype Gesture Recognition components above and shown that both approaches yield promising results, the HMM in a more unsupervised manner than the TDRBF. Although the first layer of weights learned during training are unsupervised in the TDRBF, the mapping of class prototypes onto the task-relevant classes needs to be supervised and a seven phase structure was imposed. The HMM could discover this structure from data clustering. Performance on the learning and generalisation tasks was broadly similar, although training the HMM with the Baum-Welch algorithm takes longer than weight training in the RBF network. The TDRBF was coded in C and adapted from previous work in the ISCANIT project [18], while the HMM was just developed here in Matlab for comparison. Thus, it is premature to give full QoS and computational costs but these will be established in future work. As in the discussion above, there is greater potential for contextual processing using the HMM for attentive processing and

it is likely that the TDRBF could supply initial fast, reactive results.

We also introduced the proposed approach to task control within the ActIPret system using a Dynamic Decision Network (DDN) of some kind, eg. [15], in the Activity Reasoning Engine, see Fig. 1. However, we also want distributed control in the lower levels and one way of imposing this is by conditional probability matrices to activate the services within each lower component. A service call in the system requires at least QoS, computational cost and priority metrics. Initially, it is proposed to hand code utility/task relevance nodes (eg. watch/ignore) that determine the priority metric. In the longer term, in the context of a complete system, we hope to learn these dynamic dependencies. It may be that in order to determine task-relevance automatically in this way, a uniform Bayesian approach using probability estimates is preferred. However, it may be that other probabilistic evidence measures for current task hypotheses such as the confidence measures available from RBF nets are equally learnable. This, together with optimal methods of exploiting task and scene context, are issues for further research.

Acknowledgements

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; also framework concepts and funding from the EU ActIPret project.

References

[1] J. Bilmes. A gentle tutorial on the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technical report, ICSI-TR-97-021, University of Berkeley, CA, 1998.

[2] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[3] D. M. Blei and P. J. Moreno. Topic segmentation with an aspect hidden Markov model. In *Int. Conf. Research and Dev. Inf. Retrieval*, pp. 343–348, New York, 2001.

[4] M. Brand, N. Oliver, and A. Pentland. Coupled hidden Markov models for complex action recognition. In *IEEE Conf. CVPR*, Puerto Rico, 1997.

[5] H. Buxton and S. Gong. Visual surveillance in a dynamic and uncertain world. *Art. Intelligence*, 78:431–459, 1995.

[6] S. Chien and H. Mortensen. Automating image processing for scientific data analysis of a large image database. *IEEE Trans. PAMI*, 18:854–859, 1996.

[7] V. Clement and M. Thonnat. Integration of image processing procedures: Ocapi, a knowledge based approach. *CVGIP: Image Understanding*, 57:166–184, 1993.

[8] B. Draper, A. Hanson, and E. Riseman. Knowledge-directed vision: Control, learning, and integration. *IEEE Trans. Signals and Symbols*, 84(11):1625–1637, 1996.

[9] S. Duvdevani-Bar, S. Edelman, A. J. Howell, and H. Buxton. A similarity-based method for the generalization of face recognition over pose and expression. In *IEEE Conf. FG*, pp. 118–123, Nara, Japan, 1998.

[10] J. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

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

[12] W. Gerstner. Time structure of the activity in neural networks. *Physical Review*, E 51:738–758, 1995.

[13] S. Gong and H. Buxton. On the visual expectations of moving objects: A probabilistic approach with augmented hidden Markov model. In *ECAI*, pp. 781–785, Vienna, 1992.

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

[15] R. Howarth and H. Buxton. Conceptual descriptions from monitoring and watching image sequences. *Image and Vision Computing*, 18:105–135, 2000.

[16] A. Howell and H. Buxton. Learning gestures for visually mediated interaction. In *BMVC*, pp. 507–517, 1998.

[17] A. Howell and H. Buxton. Learning identity with radial basis function networks. *Neurocomputing*, 20:15–34, 1998.

[18] A. Howell and H. Buxton. Time-delay RBF networks for attentional frames in visually mediated interaction. *Neural Processing Letters*, 15:197–211, 2002.

[19] M. Jordan. Serial order: A parallel, distributed processing approach. In *Advances in Connectionist Theory: Speech*. Lawrence Erlbaum, 1989.

[20] M. Jordan. *Learning in Graphical Models*. NATO Science Series, 1998.

[21] T. Matsuyama. Expert systems for image processing: Knowledge based composition. *CVGIP: Image Understanding*, 48:22–49, 1989.

[22] J. Moody and C. Darken. Learning with localized receptive fields. In *Conn. Models Summer School*, pp. 133–143, 1988.

[23] J. Moody and C. Darken. Fast learning in networks of locally tuned processing units. *Neural Comp.*, 1:281–294, 1989.

[24] D. J. Moore, I. A. Essa, and M. H. Hayes. Exploiting human actions and object context for recognition tasks. In *Proc. ICCV*, pp. 80–86, Vancouver, 1999.

[25] J. Pearl. *Probabilistic Reasoning in Intelligent Systems, Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[26] T. Poggio and S. Edelman. A network that learns to recognise three-dimensional objects. *Nature*, 343:263–266, 1990.

[27] T. Poggio and F. Girosi. Regularisation algorithms for learning that are equivalent to multilayer networks. *Science*, 247:978–982, 1990.

[28] D. A. Pomerleau. ALVINN: An autonomous land vehicle in a neural network. In *NIPS*, vol. 1, pp. 305–313, 1989.

[29] A. Psarrou and H. Buxton. Motion analysis with recurrent neural nets. In *ICANN*, pp. 54–57, Sorrento, 1994.

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

[31] M. Rosenblum, Y. Yacoob, and L. Davis. Human emotion recognition from motion using a radial basis function network architecture. *IEEE Trans. NN*, 7:1121–1138, 1996.

[32] M. Thonnat, S. Moisan, and M. Crubezy. Experience in integrating image processing programs. In *ICVS*, pp. 200–215, 1999.