Action Reaction Learning:
Automatic Visual Analysis and Synthesis of Interactive Behaviour

Tony Jebara Alex Pentland
Perceptual Computing - M.I.T. Media Laboratory
Massachusetts Institute of Technology
20 Ames Street, Cambridge, MA, 02139
{ jebara, sandy }@media.mit.edu
http://www.media.mit.edu/~jebara/arl

Abstract

We propose Action-Reaction Learning as an approach for analyzing and synthesizing human behaviour. This paradigm uncovers causal mappings between past and future events or between an action and its reaction by observing time sequences. We apply this method to analyze human interaction and to subsequently synthesize human behaviour. Using a time series of perceptual measurements, a system automatically uncovers correlations between past gestures from one human participant (an action) and a subsequent gesture (a reaction) from another participant. A probabilistic model is trained from data of the human interaction using a novel estimation technique, Conditional Expectation Maximization (CEM). The estimation uses general bounding and maximization to monotonically find the maximum conditional likelihood solution. The learning system drives a graphical interactive character which probabilistically predicts a likely response to a user's behaviour and performs it interactively. Thus, after analyzing human interaction in a pair of participants, the system is able to replace one of them and interact with a single remaining user.

1 Introduction

The Action Reaction Learning (ARL) framework is an automatic perceptual machine learning system. It autonomously studies the natural interactions of two humans to learn their behaviours and later engage a single human in a synthesized real-time interaction. The model is fundamentally empirical and is derived from what humans do externally, not from underlying behavioural architectures or hard wired cognitive knowledge and models.

Earlier models of human behaviour proposed by cognitive scientists analyzed humans as an input-output or stimulus-response system [26] [23]. The models were based on observation and empirical studies. These *behaviourists* came under criticism as cognitive science evolved beyond their over-simplified model and struggled with higher order issues (i.e. language, creativity, and attention) [14]. Nevertheless, much of the lower-order reactionary behaviour was still well modeled by the stimulus-response paradigm. To a casual observer, these simple models seem fine and it is only after closer examination that one realizes that far more complex underlying processes must be taking place.

We propose Action-Reaction Learning for the recovery of human behaviour by making an appeal to the behaviourists' stimulus response (input-output) model. By learning correlations between gestures that have been observed perceptually (i.e. using a vision system), it is possible to imitate simple human behaviours. This is facilitated by the evolution of computer vision beyond static measurements to temporal analysis and dynamic models.

For instance, Blake and others [3] discuss active vision beyond static imagery using includes Kalman filters and dynamical systems. More recently, visual tracking of human activity and other complex actions has included some learning algorithms and behavioural machinery that describe higher order control structures. Isaard describes how multiple hypothesis dynamical models can learn complex hand dynamics and exhibit better tracking [9]. Bobick and Wilson discuss learning hand dynamics using hidden Markov models in a state space [27] to learn complex gestures. Models which combine dynamics with learned Markov models are discussed by Pentland [19], and Bregler [4] for predicting and classifying human behaviour. Johnson [17] utilizes neural learning techniques to predict walking behaviours and discusses interactive behaviour synthesis as well. Thus, an important transition is taking place as automatic vision and perception allow the acquisition of behavioural models from observations.

Once behavioural models are acquired, the ARL framework uses them to synthesize interactive behaviour with humans (again using real-time visual tracking). Important contributions in behaviour synthesis arise in robotics and animation. In the ALIVE system[15], body tracking allows users to interact with Silas, a graphical dog based on ethological models and competing behaviours. Terzopolous [22] describes an animated environment of synthetic fish based on dynamical models. In robotics, Brooks [5] points out the need for bottom-up robotics behaviour with perceptually grounded systems. Pirjanian [20] discusses objectives and decision making in robotic behaviour. Uchibe [24] trains robots to acquire soccer playing interactions using reinforcement learning. Mataric [16] presents interacting multi-agent robots inspired from biology, cognitive models and neuroscience. Large [13] describes multiple competing dynamic models for synthesizing complex behaviour synthesis.

We consider the *integration* of both behaviour acquisition and interactive synthesis. The Action-Reaction Learning framework is initially presented. The approach treats past activity as an input and future activity as an output and attempts to uncover a probabilistic mapping between them (i.e. a prediction). The system performs imitation learning by observing humans and does not require manual segmentation, supervised training or classification. In particular, by learning from a time series of interactions, one can treat the past interaction of two individuals as input and predict a likely output reaction of the participants. The probabilistic model is estimated using, *Conditional Expectation Maximization* which recovers a conditional density of the input-output relationship between the two participants.

We also discuss the details of some of the perceptual inputs[1] into the learning system. Subsequently, there is a description of the processing of temporal information and the use of a probabilistic model for inferring reactions to a past interaction. This drives the output of the system which is realized as a graphical character. An example application as well as some results illustrating the technique are then shown as the system learns to behave with

[1]Only constrained visual behaviours and gestures will be considered. It is not essential that the input be visual or even perceptual. However, perceptual modalities are rich, expressive, intuitive and non-obtrusive. One could take other measurements if they help infer behaviour, internal state or intentionality.

simple gestural interactions. Effectively, the system learns to play or behave not by being explicitly programmed or supervised but simply by observing human participants.

2 System Architecture

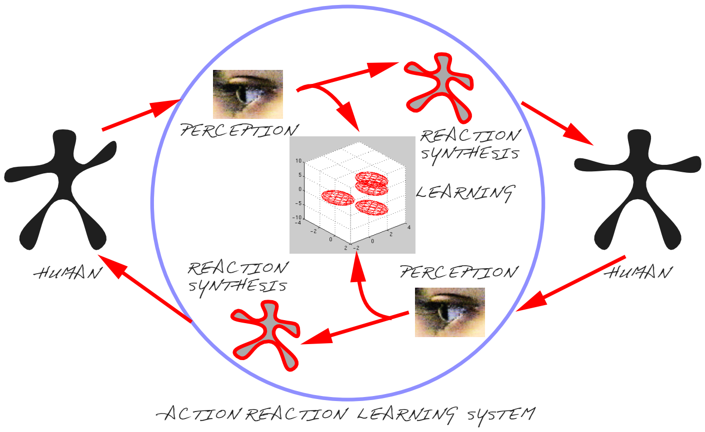


Figure 1: Offline: Learning from Human Interaction

The system is depicted in Figure 1. Three different types of processes exist: perceptual, synthesis and learning engines interlinked in real-time with asynchronous RPC data paths. Figure 1 shows the system being presented with a series of interactions between two individuals in a constrained context (i.e. a simple children's game) [2]. The system collects live perceptual measurements using a vision subsystem for each of the humans. The temporal sequences obtained are then analyzed by a machine learning subsystem to determine predictive mappings and associations between pieces of the sequences and their consequences.

On the left of the figure, a human user (represented as a black figure) is being monitored using a perceptual system. The perceptual system feeds a learning system with measurements which are stored as a time series within. Simultaneously, these measurements also drive a virtual character in a one-to-one sense (the gray figure) which mirrors the left human's actions as a graphical output for the human user on the right. A similar input and output is generated in parallel from the activity of the human on the right. Thus, the users interact with each other through the vision-to-graphics interface and use this virtual channel to visualize and constrain their interaction. Meanwhile, the learning system is 'spying' on the interaction and forming a time series of the measurements. This time series is training data for the system which is attempting to learn about this ongoing interaction in hopes of modeling and synthesizing similar behaviour itself.

In Figure 2, the system has collected and assimilated the data. At this point it can computationally infer appropriate responses to the single remaining human user. Here, the perceptual system only needs to track the activity of the one human (black figure on the left) to stimulate the learning or estimation system for real-time

[2] Of course, the individuals need not be in the same physical space and could be interacting through a virtual environment.

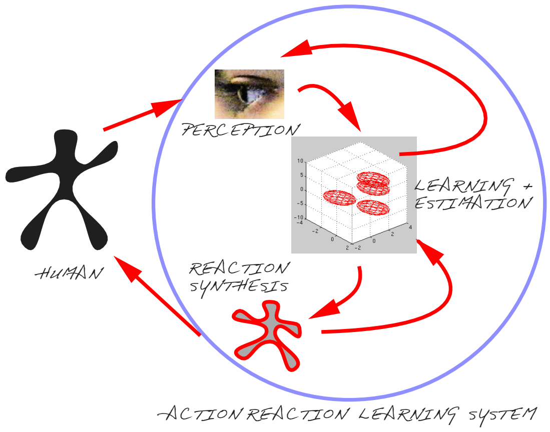


Figure 2: Online: Interaction with Single User

interaction purposes (as opposed to learning). The learning system performs an estimation to generate a likely response to the user's behaviour. This is manifested by animating a computer graphics character (gray figure) in the synthesis subsystem. This is the main output of the ARL engine. It is fed back recursively into the learning subsystem so that it can remember its own actions and generate self-consistent behaviour. This is indicated by the arrow depicting flow from the reaction synthesis to the learning + estimation stage. Thus, there is a continuous feedback of self-observation in the learning system which can recall its own actions. In addition, the system determines a likely action of the remaining user and transmits it as a prior to assist tracking in the vision subsystem. This flow from the learning system to the perception system (the eye) contains behavioural and dynamic predictions of the single user that is being observed and should help improve perception

2.1 A Typical Scenario

Action-Reaction Learning (ARL) involves temporal analysis of a (usually multi-dimensional) data stream. Figure 3 displays such a stream (or time series). Let us assume that the stream is being generated by a vision algorithm which measures the openness of the mouth [18]. Two such algorithms are being run simultaneously on two different people. One person generates the dashed line and the other generates the solid line.

Now, imagine that these two individuals are engaged in a conversation. Let us also name them Mr. Solid (the fellow generating the solid line) and Mrs. Dash (the lady generating the dashed line). Initially (A-B), Mr. Solid is talking while Mrs. Dash remains silent. He has an oscillatory mouth signal while she has a very low value on the openness of the mouth. Then, Mr. Solid says something shocking, pauses (B-C), and then Mrs. Dash responds with a discrete 'oh, I see' (C-D). She too then pauses (D-E) and waits to see if Mr. Solid has more to say. He takes the initiative and continues to speak (E). However, Mr. Solid continues talking non-stop for just too long (E-G). So, Mrs. Dash feels the need to interrupt (F) with a counter-argument and simply starts talking. Mr. Solid notes that she has taken the floor and stops to hear her out.

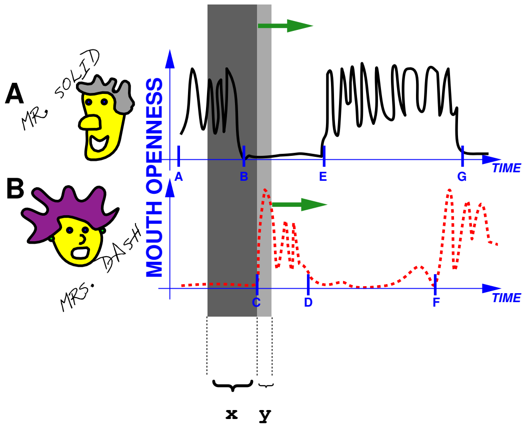


Figure 3: Dialog Interaction and Analysis Window

What Action-Reaction Learning seeks to do is discover the coupling between the past interaction and the next immediate reaction of the participants. For example, the system may learn a model of the behaviour of Mrs. Dash so that it can imitate her idiosyncrasies. The process begins by sliding a window over the temporal interaction as in Figure 3. The window looks at a small piece of the interaction and the immediate reaction of Mrs. Dash. This piece is the short term or iconic memory the system will have of the interaction and it is highlighted with a dark rectangular patch. The consequent reaction of Mrs. Dash and Mr. Solid is highlighted with the lighter and smaller rectangular strip. The first strip will be treated as an input $\mathbf{x}$ and the second strip will be the desired subsequent behavioural output of both Mr. Solid and Mrs. Dash ($\mathbf{y}$). As the windows slide across the interaction, many such $(\mathbf{x}, \mathbf{y})$ pairs are generated and presented to a machine learning system. The task of the learning algorithm is to learn from these pairs to later generate predicted $\hat{\mathbf{y}}$ sequences which can be used to compute and play out the future actions of one of the users (i.e. Mrs. Dash) when only the past interaction $\mathbf{x}$ of the participants is visible.

Thus, the learning algorithm should discover some mouth openness behavioural properties. For example, Mrs. Dash usually remains quiet (closed mouth) while Mr. Solid is talking. However, after Solid has talked and then stopped briefly, Mrs. Dash should respond with some oscillatory signal. In addition, if Mr. Solid has been talking continuously for a significant amount of time, it is more likely that Mrs. Dash will interrupt assertively. A simple learning algorithm could be used to detect similar $\mathbf{x}$ data in another situation and then predict the appropriate $\mathbf{y}$ response that seems to agrees with the system's past learning experiences.

Note now that we are dealing with a somewhat supervised learning system because the data has been split into input $\mathbf{x}$ and output $\mathbf{y}$. The system is given a target goal: to predict $\mathbf{y}$ from $\mathbf{x}$. However, this process is done automatically without significant manual data engineering. One only specifies a-priori a constant width for the sliding windows that form $\mathbf{x}$ and $\mathbf{y}$ (usually, the $\mathbf{y}$ covers only 1 frame). The system then operates in an unsupervised manner as it slides these windows across the data stream. Essentially, the learning uncovers a mapping between *past and future* to later generate its best possible prediction. Interaction can be learned from a variety of approaches including reinforcement learning [24]. The objective here is is primarily an *imitation*-type learning of interaction.

3 Perceptual System

A primary concern in the visual perceptual system is the recovery of action parameters which are particularly expressive and interactive. In addition, to maintain real-time interactiveness and fast training, the input/output parameters should be compact (low dimensional). The tracking system used is a head and hand tracking system which models the three objects (head, left and right hand) as 2D blobs with 5 parameters each. With these features alone, it is possible to engage in simple gestural games and interactions.

The vision algorithm begins by forming a probabilistic model of skin colored regions [1]. During an offline process, a variety of skin-colored pixels are selected manually, forming a distribution in *rgb* space. This distribution can be described by a probability density function (pdf) which is used to estimate the likelihood of any subsequent pixel ($\mathbf{x}_{rgb}$) being a skin colored pixel. The pdf used is a 3D Gaussian mixture model as shown in Equation 1 (with $M = 3$ individual Gaussians typically).

$$p(\mathbf{x}_{rgb}) = \sum_{i=1}^{M} \frac{p(i)}{(2\pi)^{\frac{3}{2}}\sqrt{|\Sigma_i|}} \; e^{-\frac{1}{2}(\mathbf{x}_{rgb}-\mu_i)^T \Sigma_i^{-1} (\mathbf{x}_{rgb}-\mu_i)} \tag{1}$$

The parameters of the pdf ($p(i)$,μ_i and Σ_i) are estimated using the Expectation Maximization [6] algorithm to maximize the likelihood of the training *rgb* skin samples. This pdf forms a classifier and every pixel in an image is filtered through it. If the probability is above a threshold, the pixel belongs to the skin class, otherwise, it is considered non-skin. Figures 4(a) and (d) depict the classification process.

To clean up some of the spurious pixels misclassified as skin, a connected components algorithm is performed on the region to find the top 4 regions in the image, see Figure 4(b). This increases the robustness of the EM based blob tracking. We choose to process the top 4 regions since sometimes the face is accidentally split into two regions by the connected components algorithm. In addition, if the head and hands are touching, there may only be one non-spurious connected region as in Figure 4(e).

Since we are always interested in tracking three objects (head and hands) even if they touch and form a single connected region, it is necessary to invoke a more sophisticated pixel grouping technique. Once again, we use the EM algorithm to find 3 Gaussians that this time maximize the likelihood of the *spatially* distributed (in xy) skin pixels. Note that the implementation of the EM algorithm here has been heavily optimized to require less than 50ms to perform each iteration for an image of size 320 by 240 pixels. This Gaussian mixture model is shown in Equation 2.

$$p(\mathbf{x}_{xy}) = \sum_{j=1}^{3} \frac{p(j)}{2\pi\sqrt{|\Sigma_j|}} \; e^{-\frac{1}{2}(\mathbf{x}_{xy}-\mu_j)^T \Sigma_j^{-1} (\mathbf{x}_{xy}-\mu_j)} \tag{2}$$

The update or estimation of the parameters is done in real-time by iteratively maximizing the likelihood over

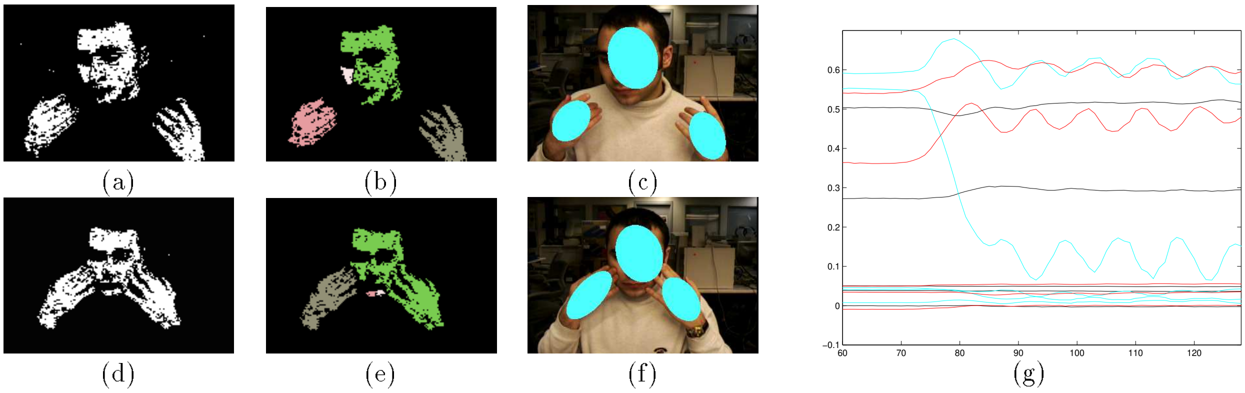


Figure 4: Head and Hand Blob Tracking with Time Series Data

each image. The resulting 3 Gaussians have 5 parameters each (from the 2D mean and the 2D symmetric covariance matrix) and are shown rendered on the image in Figures 4(c) and (f). The covariance (Σ) is actually represented in terms of its square root matrix, Γ where $\Gamma \times \Gamma = \Sigma$. Like Σ, the Γ matrix has 3 free parameters $(\Gamma_{xx}, \Gamma_{xy}, \Gamma_{yy})$ however these latter variables are closer to the dynamic range of the 2D blob means and are therefore preferred for representation. The 5 parameters describing the head and hands are based on first and second order statistics which can be reliably estimated from the data in real-time. In addition, they are well behaved and do not exhibit wild non-linearities. More complex measurements could be added but if they are stably estimable and don't exhibit excessive non-linearities. The 15 recovered parameters from a single person are shown as a well behaved, smooth time series in Figure 4(g). These define the 3 Gaussian blobs (head, left hand and right hand).

The parameters of the blobs are also processed in real-time via a Kalman Filter which smoothes and predicts their values for the next frame. The KF model assumes constant velocity to predict the next observation and maintain tracking. In addition, the same system can be used to track multiple colored objects and if colored gloves are used, the system handles occlusion and tracking more robustly.

4 Graphical System

At each time frame, the 15 estimated parameters for the Gaussians can be rendered for viewing as the stick figure in Figure 5. This is also the display provided to each user so that he may view the gestures of other human (or computer) players through his personal computer screen. The output is kept simple to avoid confusing users into believing that more sophisticated perception is taking place.

5 Temporal Modeling in the ARL System

The Action-Reaction Learning system functions as a server which receives real-time multi-dimensional data from the vision systems and re-distributes it to the graphical systems for rendering. Typically during training, two vision

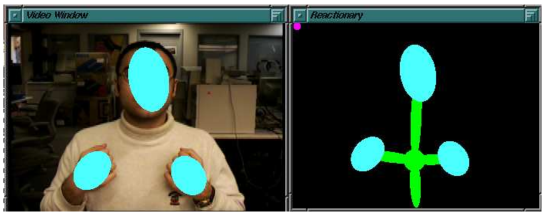


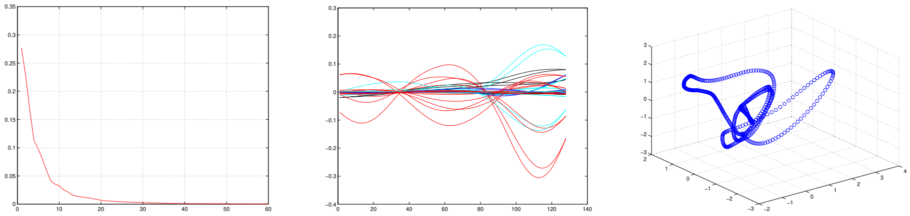
Figure 5: Graphical Rendering of Perceptual Measurements

systems and two graphics systems are connected to the ARL server. Thus, it is natural to consider the signals and their propagation as multiple temporal data streams. Within the ARL server, perceptual data or tracked motions from the vision systems are accumulated and stored explicitly into a finite length time series of measurements.

For head and hand tracking , two triples of Gaussian blobs are generated (one triple for each human) by the vision systems and form 30 continuous scalar parameters. These evolve as a multi-dimensional time series. Each set of 30 scalar parameters can be considered as a 30 dimensional vector $\mathbf{y}(t)$ arriving into the ARL engine from the vision systems at a given time t. The ARL system preprocesses then trains from this temporal series of vectors. We discuss the representation of the multidimensional time series data which will be analyzed to predict and forecast its evolution, or equivalently, estimate the parameters of the 6 blobs in near future.

An account of the Santa Fe competition is presented in [8] where issues in time series modeling and prediction are discussed. We consider the connectionist representation due to its explicit non-linear optimization of prediction accuracy and its promising performance against hidden Markov models, dynamic models, etc in the competition. One of its proponents, Wan [25], describes a nonlinear time series auto regression which computes an output vector $\mathbf{y}(t)$ from T previous input instances of the vector $\mathbf{y}(t-1), \mathbf{y}(t-2), ..., \mathbf{y}(t-T)$. The mapping is approximated via a neural network function $g()$ as in $\mathbf{y}(t) = g\left(\mathbf{y}(t-1), ..., \mathbf{y}(t-T)\right)$. In our case, each $\mathbf{y}$ is a 30 dimensional vector of the current perceptual parameters from two humans.

However, since T previous samples are considered, the function to be approximated has a high dimensional domain. For head and hand tracking data (15Hz tracking), values of $T \approx 120$ are required to form a short term memory of a few seconds ($\approx$ 6 seconds). Thus, the dimensionality ($T \times 30$) is in the thousands. A dimensionality reduction of the input space is accomplished via Principal Components Analysis (PCA) [2]. Consider the input space as a large vector $Y(t)$ composed of the concatenation of all the T vectors that were just observed $\mathbf{y}(t-1), ...\mathbf{y}(t-T)$. Each vector Y represents a short term memory of the past (a 6 second chunk). If we consider the training data as a whole, many such large vectors Y are observed and form a distribution. The few most energetic eigenvectors and eigenvalues of the covariance of this distribution span a compact subspace of the Y space. The eigenvectors and eigenvalues are ranked in decreasing order with the top few eigenvalues shown in Figure 6(a). Over 95% of the energy of the Y vectors (i.e. the short term memories) can be represented using linear combinations of the



(a) Largest 60 Eigenvalues (b) Top Eigenvector (c) Projection onto Top 3

Figure 6: Top Eigenvectors and Eigenvalues and 3D Projection onto Eigenspace

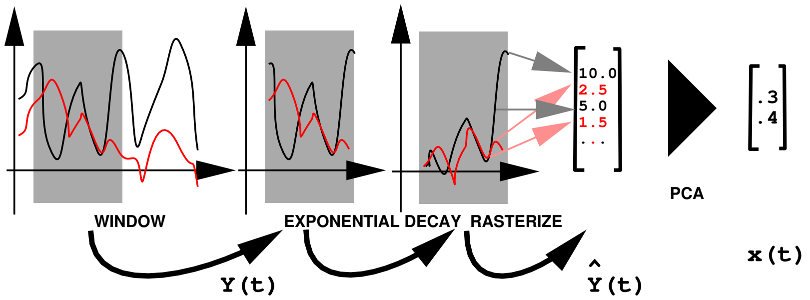


Figure 7: Exponential Decay and Pre-Processing

first 40 eigenvectors. The distribution of Y occupies only a small sub manifold of the original 3600 dimensional embedding space and 40 dimensions span it sufficiently. We call the low-dimensional subspace representation of $Y(t)$ the immediate past short term memory of interactions and denote it with $\mathbf{x}(t)$. In Figure 6(b) the first mode (the most dominant eigenvector) of the short term memory is rendered as a 6 second evolution of the 30 head and hand parameters of two interacting humans. Interestingly, the shape of the eigenvector is not exactly sinusoidal nor is it a wavelet or other typical basis function since it is specialized to the training data.

It should be noted that the above analysis actually used weighted versions of the Y vectors to include a soft memory decay process. An exponential decay scales down $\mathbf{y}$ vectors that constitute the big $Y(t)$ vector. The further back in time a $\mathbf{y}$ component is, the more its amplitude is attenuated. Thus, an exponential ramp function is multiplied with each Y window (i.e. a few seconds of each of the 30 time series). This reflects our intuition that more temporally distant elements in the time series are less relevant for prediction. This decay agrees with some aspects of cognitive models obtained from psychological studies [7]. Once the vectors have been attenuated, they form a new 'exponentially decayed' short term memory window $\hat{Y}(t)$. The process is shown in Figure 7 where a window is placed over the time series, generating a short term memory Y. An exponential decay function is used to decay it and generates the $\hat{Y}$ version. The eigenspace previously discussed is really formed over the $\hat{Y}$ distribution $\hat{Y}$ is represented in a subspace with a compact $\mathbf{x}(t)$. This is the final, low dimensional representation of the gestural interaction between the two humans over the past few seconds.

5.1 Probabilistic Time Series Modeling

Of course, immediately after the time window over the past, another observation $\mathbf{y}(t)$ (of the near future) is also obtained from the training data. One may again simply vectorize the parameters of the perceptual system (the Gaussian tracking blobs) into yet another $\mathbf{y}$ vector (of dimensionality $\mathcal{R}^{30}$). The $\mathbf{x}(t)$ vector represents the past action and the $\mathbf{y}(t)$ represents the consequent reaction exactly at time t. For a few minutes of data, we can obtain thousands of pairs of $\mathbf{x}$ and $\mathbf{y}$ vectors (i.e. action-reaction pairs) by sliding the attentional window over the training time series. Figure 6(c) shows the evolution of the dominant 3 dimensions of the $\mathbf{x}(t)$ vectors as we consider an involved interaction between two participants over time t of roughly half a minute. This represents the evolution of the short term memory of the learning system during half a minute.

Given sufficient pairs of the vectors ($\mathbf{x}$(t),$\mathbf{y}$(t)) from training data, it is possible to start seeing patterns between a short term memory of the past interaction of two humans and the immediate subsequent future reaction. A system which can forecast this behaviour could predict what to do next and engage with a single human. However, instead of learning an exact deterministic mapping between $\mathbf{x}$ and $\mathbf{y}$, as is done in a predictive neural network, we will discuss a more probabilistic approach. This involves estimating a probability density denoted as $p(\mathbf{y}|\mathbf{x})$ which yields the probability of a reaction *given* a short history of past action. We will always be observing the past ($\mathbf{x}$) but the future ($\mathbf{y}$) is what we are trying to predict. We are not, for instance, interested in the conditional pdf $p(\mathbf{x}|\mathbf{y})$, which computes the probability of the past ($\mathbf{x}$) given the future. Mostly, we will query the system about what future result should follow the actions it just saw. The use of probabilistic techniques here allows the notion of randomness and stochasticity which is appropriate for behaviour modeling. In essence, they make the system generate behaviour that is interesting and correlated with the past and the user's stimulating actions but is *also* not entirely predictable and contains some pseudo random choices in its space of valid responses.

6 Conditional Expectation Maximization

To model the action-reaction space or the mapping between $\mathbf{x}$ and $\mathbf{y}$ we estimate a conditional probability density function. This non-deterministic mapping is appropriate due to the randomness of the interactive behaviour in humans, the noise in the perceptual systems and the sparseness of the observations. In addition, since we will always observe $\mathbf{x}$ and want to predict the subsequent reaction $\mathbf{y}$, a conditional density of the form $p(\mathbf{y}|\mathbf{x})$ is required.

The conditioned mixture of Gaussians is selected for its ability to model non-linear phenomena and its ease of use. The model can be interpreted as a mixture of experts with multiple linear regressors and ellipsoidal basis gating functions [12]. Equation 3 depicts the model where $\mathcal{N}$ represents a normal distribution (Gaussian).

$$p(\mathbf{y}|\mathbf{x}) = \frac{p(\mathbf{x},\mathbf{y})}{p(\mathbf{x})} = \frac{\sum_m^M p(\mathbf{x},\mathbf{y},m)}{\sum_m p(\mathbf{x},m)} = \frac{\sum_m^M p(m)\mathcal{N}(\mathbf{x},\mathbf{y}|\mu_m^x,\mu_m^y,\Sigma_m^{xx},\Sigma_m^{yy},\Sigma_m^{xy})}{\sum_m^M p(m)\mathcal{N}(\mathbf{x}|\mu_m^x,\Sigma_m^{xx})} \quad (3)$$

Traditionally, estimating probabilistic models is done by maximizing the likelihood (L) of a model (Θ) given the data as shown in Equation 4. Techniques such as Expectation Maximization [6] can be used to optimize the parameters of a probability density function such that its joint density is a good model of the data. In clustering, for instance, data is treated homogeneously without special considerations for the distinction between input $\mathbf{x}$ and output $\mathbf{y}$. If the data is split as aforementioned into response ($\mathbf{y}$) and covariate ($\mathbf{x}$) components, this indicates that the covariate components will always be available to the system. Thus, when fitting a probabilistic model to the data, we should optimize it only to predict $\mathbf{y}$ using $\mathbf{x}$ ($\mathbf{x}$ is always measured). This forms a more discriminative model that concentrates modeling resources for the task at hand.

$$L = \prod_{i=1}^{N} p(\mathbf{x}_i, \mathbf{y}_i | \Theta) \tag{4}$$

We recently developed a variant of the EM algorithm called Conditional Expectation Maximization (CEM) for specifically optimizing conditional likelihood [10]. It essentially fits a probability density function (pdf) that maximizes the conditional likelihood of the response given the covariates. CEM is an iterative technique which uses fixed point solutions (i.e. as opposed to gradient descent) to converge the parameters of a conditional density to a local maximum of conditional likelihood (L_c) as described by Equation 5. The fixed point solutions are computed by forming a lower bound on conditional log-likelihood and maximizing the lower bound iteratively. The CEM algorithm has also been extended to use priors for Maximum A Posteriori or MAP estimation and deterministic annealing for more global solutions.

$$L_c = \prod_{i=1}^{N} p(\mathbf{y}_i | \mathbf{x}_i, \Theta) \tag{5}$$

Applying CEM to the pdf optimizes its $p(\mathbf{y}|\mathbf{x})$ over the data. EM, on the other hand, typically optimizes $p(\mathbf{x}, \mathbf{y})$, the ability to model the data as a whole. Since resources (i.e. memory, complexity) are sparse and training examples are finite, it is preferable here to directly optimize the model's conditional likelihood [21] using CEM. In other words, we want the learning system to be good at figuring out what Mrs. Dash will do next (i.e. use $\mathbf{x}$ to predict $\mathbf{y}$). We are not as interested in asking the system what past event would have provoked Mrs. Dash to do what she just did (i.e. use $\mathbf{y}$ to get $\mathbf{x}$).

Consider the 4-cluster (x, y) data in Figure 8(a). The data is modeled with a conditional density $p(y|x)$ using *only 2* Gaussian models. Estimating the density with CEM yields the $p(y|x)$ shown in Figure 8(b). CEM exhibits monotonic conditional likelihood growth (Figure 8(c)) and obtains a more conditionally likely model. In the EM case, a joint $p(x, y)$ clusters the data as in Figure 8(d). Conditioning it yields the $p(y|x)$ in Figure 8(e). Figure 8(f) depicts EM's non-monotonic evolution of conditional log-likelihood. EM produces a superior joint likelihood (L) but an inferior conditional likelihood (L_c). Note how the CEM algorithm utilized limited resources to capture the multimodal nature of the distribution in y and ignored spurious bimodal clustering in the x feature space. These

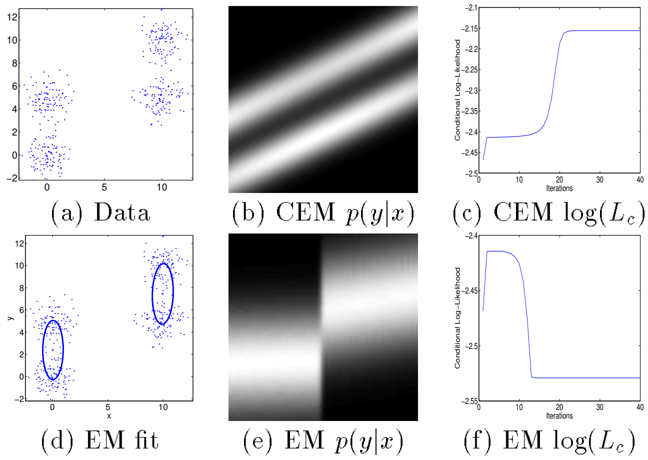


Figure 8: Conditional Density Estimation for CEM and EM

properties are critical for a good conditional density $p(y|x)$. In regression experiments on standardized databases, mixture models trained with CEM outperformed those trained with EM as well as conventional neural network architectures [10].

Thus, the CEM algorithm is used to estimate the conditional probability density (cpdf) relating past time series sequences ($\mathbf{x}$) to their immediate future values ($\mathbf{y}$) from training data (thousands of $\mathbf{x}, \mathbf{y}$ pairs). A total of M Gaussians are fit to the data as a conditioned mixture model. This is ultimately used to regress (predict) the future values of a time series for a single forward step in time. Once the probabilistic behavioural model is formed from training data, it is possible to estimate an unknown $\hat{\mathbf{y}}$ from observed $\mathbf{x}$. When $\hat{\mathbf{x}}$ is measured from the past time series activity and inserted into the conditional probability density, it yields a marginal density exclusively over the variable $\mathbf{y}$ (the prediction or reaction to the past stimulus sequence). This density becomes a 30 dimensional, M-component Gaussian mixture model.

However, we need to select a single reaction, $\hat{\mathbf{y}}$ from the space of possible reactions over $\mathbf{y}$. It is customary in Bayesian inference to use the expectation of a distribution as its representative. Using the pdf over $\mathbf{y}$, we integrate as in Equation 6 to obtain the predicted $\hat{\mathbf{y}}$, a likely reaction according to the model (we have also considered arg maximization and sampling methods [10] for obtaining $\hat{\mathbf{y}}$ candidates).

$$\hat{\mathbf{y}} = \int \mathbf{y} p(\mathbf{y}|\hat{\mathbf{x}}) d\mathbf{y} = \frac{\sum_m^M \hat{\mathbf{y}}_m p(\hat{\mathbf{y}}_m|\hat{\mathbf{x}})}{\sum_m^M p(\hat{\mathbf{y}}_m|\hat{\mathbf{x}})} \qquad \text{where} \quad \hat{\mathbf{y}}_{\mathrm{m}} = \mu_{\mathrm{m}}^{\mathrm{y}} + \Sigma_{\mathrm{m}}^{\mathrm{yx}} \Sigma_{\mathrm{m}}^{\mathrm{xx}\,-1} (\hat{\mathbf{x}} - \mu_{\mathrm{m}}^{\mathrm{x}}) \tag{6}$$

7 Integration

At this point, we discuss the integrated system. The flow between perceptual input, the graphical output, the time series processing and the learning system are presented as well as some of the different modes of operation they can encompass.

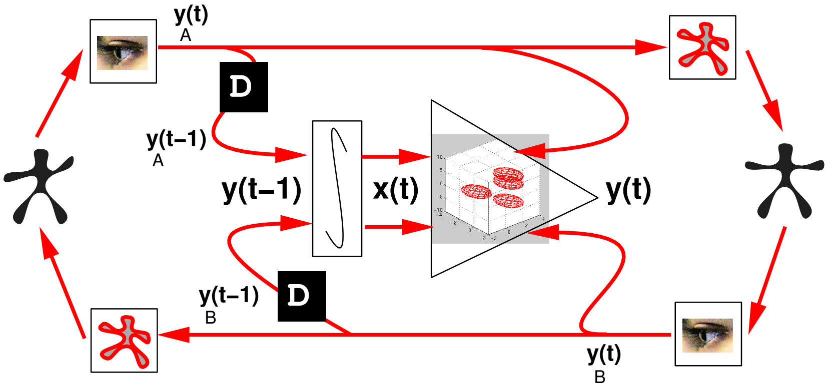


Figure 9: Training Mode

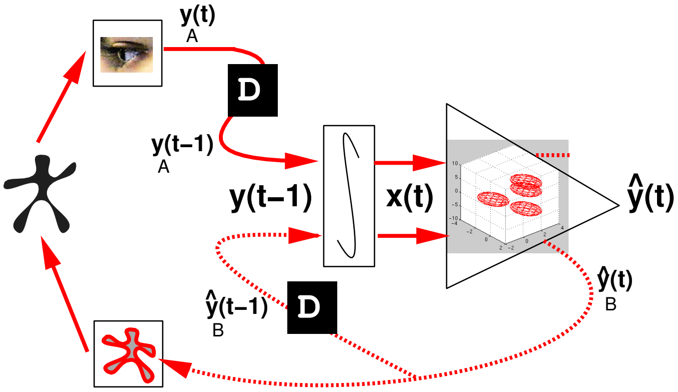


Figure 10: Interaction Mode

7.1 Training Mode

For training, two humans interact while the system accumulates information about the actions and reactions (see Figure 9). The learning system is being fed $\mathbf{x}(t)$ on one end and $\mathbf{y}(t)$ on the other. Once many pairs of data are accumulated, the system uses CEM to optimize a conditioned Gaussian mixture model which represents $p(\mathbf{y}|\mathbf{x})$. We note the role of the integration symbol which indicates the pre-processing of the past time-series via an attentional window over the past T samples of measurements. This window can be represented compactly in an eigenspace with $\mathbf{x}(t)$. Note, that the $\mathbf{y}$ vector can be split into $\mathbf{y}_A(t)$ and $\mathbf{y}_B(t)$, where each half the vector corresponds to a user.

7.2 Interaction Mode

In Figure 10 one can see the system as it synthesizes interactive behaviour with a single user. User A is given the illusion of interacting with user B through the synthesis of the ARL system. The vision system on A still takes measurements and these integrate and feed the learning system. However, the output of the learning system is *also* fed back into the short term memory. It fills in the missing component (user B) inside $\mathbf{x}$. Thus, not only does user A see synthesized output, continuity is maintained by feeding back synthetic measurements. This gives the system the ability to see its own actions and maintain self-consistent behaviour. The half of the time series that

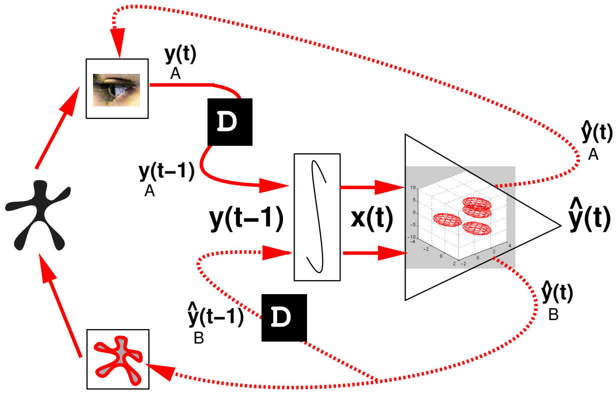


Figure 11: Perceptual Mode

used to be generated by B is now being synthesized by the ARL system. The $\mathbf{x}(t)$ is continuously updated allowing good estimates of $\hat{\mathbf{y}}$. In fact, the probabilistic model trained by CEM only predicts a small steps into the future and these 'deltas' do not amount to a full gesture on their own unless they are integrated and accumulated. Thus, the feedback path is necessary for the system to make continuous predictions. Since the attentional window which integrates the $\mathbf{y}$ measurements is longer than a few seconds, this gives the system enough short term memory to maintain consistency over a wide range of gestures and avoids instability [3]. Simultaneously, the real-time graphical blob representation is used to un-map the predicted perceptual (the $\hat{\mathbf{y}}_B(t)$ action) for the visual display. It is through this display that the human user receives feedback in real-time from the system's reactions.

7.3 Perceptual Feedback Mode

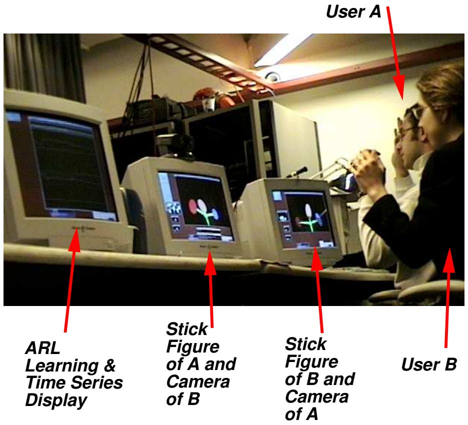
Of course, the CEM learning system generates *both* a $\hat{\mathbf{y}}_B(t)$ and a $\hat{\mathbf{y}}_A(t)$. Therefore, it would be of no extra cost to utilize the information in $\hat{\mathbf{y}}_A(t)$ in some way while the system is interacting with the user. Instead of explicitly using Kalman filters in the vision systems (as described earlier), we also consider using the predicted $\hat{\mathbf{y}}_A(t)$ as an alternative to filtering and smoothing. The ARL system then emulates a non-linear dynamical filter and helps resolve some vision tracking errors.

Typically, tracking algorithms use a variety of temporal dynamic models to assist the frame by frame vision computations. The most trivial of these is to use the last estimate in a nearest neighbour approach to initialize the next vision iteration. Kalman filtering and other dynamic models [3] involve more sophistication ranging from constant velocity models to very complex control systems. Here, the the feedback being used to constrain the vision system results from dynamics *and* behaviour modeling. This is similar in spirit to the mixed dynamic and behaviour models in [19]. In the head and hand tracking case, the system continuously feeds back behavioural prediction estimates of the 15 tracked parameters (3 Gaussians) in the vision system for improved results.

[3] For the initial seconds of interaction, the system has not yet synthesized any actions and user A still has to begin gesturing but the feedback loop automatically bootstraps and stabilizes.

Interaction	User	Corresponding Action
1	A	Scare B by raising arms
	B	Fearfully crouch down
2	A	Wave hello
	B	Wave back accordingly
3	A	Circle stomach & tap head
	B	Clap enthusiastically
4	A	Idle or Small Gestures
	B	Idle or Small Gestures

(a) Instructions



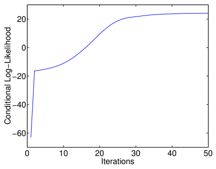
(b) Perceptual Training

Figure 12: Instructions and Perceptual Training

More significant vision errors can also be handled. Consider the specific case of head and hand tracking with skin blobs. For initial training, colored gloves were used to overcome some correspondence problems when heads and hands touched and moved by each other. However, once appropriately trained, the probabilistic model described above feeds back the positions of the Gaussians to the vision. This prevents blob mislabeling by using the whole gesture as a predictor instead of short range dynamics. Thus, it is possible to recognize a blob as a hand from its role in a gesture and to maintain proper tracking. In addition, a coarse model of $p(\mathbf{x})$ is available and can be evaluated to determine the likelihood of any past interaction (short term memory). If different permutations of the tracked blobs are occasionally tested with $p(\mathbf{x})$, any mislabeling of the blob features can be detected and corrected. The system merely tests each of the 6 permutations of 3 blobs to find the one that maximizes $p(\mathbf{x})$ (the most likely past gesture). This permutation is then fed back in $\hat{\mathbf{y}}$ to resolve the correspondence problem in the vision module. Instead of using complex static computations to resolve these ambiguities, a reliable correspondence between the blobs is computed from temporal information.

8 Interaction Results

It is prudent to train the ARL system in a constrained context to achieve learning convergence from limited data and modeling resources. Thus, users involved in training are given loose instructions on the nature of the interactions they will be performing (as in Figure 12(a)). The humans (A and B) randomly play out these multiple interactions. The learning algorithm is given measurements of the head and hand positions of both users over several minutes of interaction. Once the training is complete, the B gesturer leaves and the single user remaining is A. The screen display for A still shows the same graphical character except now user B is impersonated by synthetic reactions in the ARL system (or, by symmetry, the system can instead impersonate user A).



(a) Conditional Log Likelihood

Nearest Neighbour	Constant Velocity	ARL
1.57 %	0.85 %	0.62 %

(b) RMS Errors

Figure 13: Conditional Log Likelihood on Training and RMS Errors on Testing ARL Data

More specifically, the training process involved between 5 to 10 of each of the above interactions and lasted roughly 5 minutes. This accounts for slightly over 5000 observations of the 30 dimensional $\mathbf{y}(t)$ vectors. Each of these form an $(\mathbf{x}, \mathbf{y})$ where the $\mathbf{x}$ was the eigen-representation of the past short term memory over $T = 120$ exponentially decayed samples (covering 6 seconds). The dimensionality of $\mathbf{x}$ was reduced to only 22 dimensions and the system used $M = 25$ Gaussians for the pdf (these limitations were mainly for real-time speed considerations). The learning (CEM algorithm with annealing) took approximately 2 hours to converge on an SGI OCTANE for the 5 minute training sequence of interactions (convergence is shown in Figure 13(a)).

8.1 Quantitative Prediction

For a quantitative measure, the system was trained as usual on interactions between the two individuals and learned the usual predictive mapping $p(\mathbf{y}, \mathbf{x})$. Since real-time is not an issue for this kind of test, the system was permitted to use more Gaussian models and more dimensions to learn $p(\mathbf{y}, \mathbf{x})$. Once trained on a portion of the data, the system's ability to perform prediction was tested on the remainder of the sequence. Once again, the pdf allows us to compute an estimated $\hat{\mathbf{y}}$ for any given $\mathbf{x}$ short term memory (i.e.$\mathbf{y}(t-1), ..., \mathbf{y}(t-T)$) . The expectation was used to predict $\hat{\mathbf{y}}$ and was compared to the true $\mathbf{y}$ result in the future of the time series. For comparison, RMS errors are shown against the nearest neighbour and constant velocity estimates. The nearest neighbour estimate merely assumes that $\mathbf{y}(t) = \mathbf{y}(t-1)$ and the constant velocity assumes that $\mathbf{y}(t) = \mathbf{y}(t-1) + \Delta_t \dot{\mathbf{y}}(t-1)$. Figure 13(b) depicts the RMS errors on the test interaction and these suggest that the system is a better instantaneous predictor than the above two methods and could be useful in Kalman filter-type prediction applications.

8.2 Qualitative Interaction

In addition, real-time online testing of the system's interaction abilities was performed. A human player performed the gestures of user A and checked for the system's response. Whenever the user performed one of the gestures in Table 12, the system responded with a (qualitatively) appropriate animation of the synthetic character (the gesture of the missing user B). By symmetry, the roles could be reversed such that the system impersonates user A and a human acts as user B.

In Figure 14, a sample interaction where the user 'scares' the system is depicted. Approximately 500ms elapse between each frame and the frames are arranged lexicographically in temporal order. The user begins in a relaxed

Figure 14: Scare Interaction

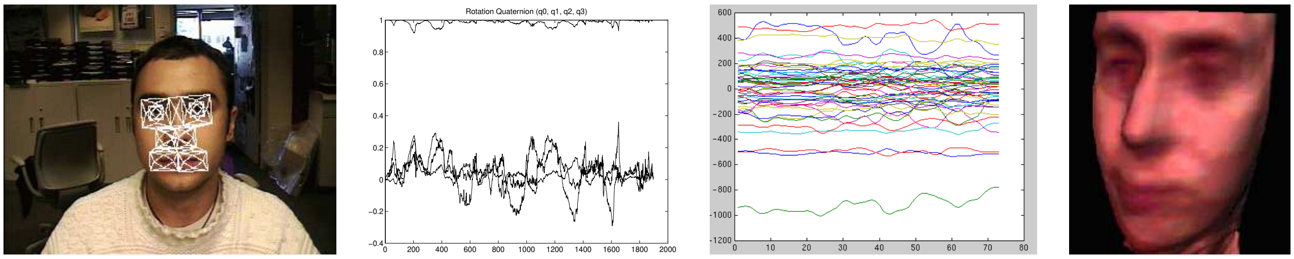
rest state and the synthetic character as well. Then, the user begins performing a menacing gesture, raising both arms in the air and then lowering them. The synthetic character responds by first being taken aback and then crouching down in momentary fear. This is the behaviour that was indicated by examples from the human-to-human training. Moreover, the responses from the system contain some pseudo random variations giving them a more compelling nature.

A more involved form of interaction is depicted in Figure 15. Here, the user stimulates the character by circling his stomach while patting his head. The system's reaction is to clap enthusiastically for this slightly tricky and playful gesture. Once again, the system stops gesturing when the user is still (as is the case at the beginning and at the end of the sequence). The oscillatory gesture the user is performing is rather different from the system's response. Thus, there is a higher-level mapping: oscillatory gesture to different oscillatory gesture[4].

In the above examples, the user is *delegating* tasks to the animated character since it is not a simple 1-to-1 mapping between the current measurements to output. The user produces a complex action and the system responds with a complex reaction. The response depends on user input as well as on the system's previous internal state. The mapping thus associates measurements over time to measurements over time which is fundamentally a higher dimensional problem.

[4]Please consult our web page for the video animation and for other interaction examples.

Figure 15: Clapping Interaction



(a) Tracking (b) Quaternions (c) 3D Eigen Deformations (d) 3D Synthesis

Figure 16: 3D Face Modeling and Tracking

9 Current and Future Work - Continuous Online Learning and Face Modeling

It is also feasible to continue the training of the CEM algorithm while the system acquires more data and performs synthesis. Thus, it performs online learning and updates its mixture of conditional models dynamically as it obtains new samples. This and the fact that the ARL system interacts with a user allow it to continuously learn new behaviours and interaction skills. The system merely looks at the reactions produced by the user from the past interaction he had with the system's synthesized character. The window of mutual interaction and the immediate consequence form the same input-output data pair $(\mathbf{x}, \mathbf{y})$ as was initially processed offline. The system could thus dynamically learn new responses to stimuli and include these in its dictionary of things to do. This makes it adaptive and its behaviour will be further tuned by the engagement with the single remaining user. This mode of operation is currently under investigation.

Face modeling is also being considered as an alternate perceptual modality. A system which automatically detects the face and tracks it has been implemented [11]. It is capable of tracking the 3D rotations and movements of a face using normalized correlation coupled with structure from motion. In addition, at each moment in time, it computes an eigenspace model of the face's texture which is used to infer 3D deformations. This system generates a real-time temporal sequence including XYZ translations, 3D rotations and texture/deformation coefficients (see Figure 16). To synthesize an output, a 3D renderer reconstructs a facial model in real-time using the recover deformation, texture and pose. The sample output is shown in Figure 16(d). The data representing each static frame is again a time series ($\approx$50 dimensional) permitting the ARL system analysis to extend to this platform.

10 Conclusions

We have demonstrated a perceptual real-time system which learns two-person interactive behaviour automatically by modeling the probabilistic relationship between a past action and its consequent reaction. The system is then able to engage in real-time interaction with a single user, impersonating the missing person by estimating and synthesizing likely reactions. The ARL system is data driven, autonomous, and perceptually grounded and it learns its behaviour by looking at humans.

11 Acknowledgements

The authors thank Nuria Oliver for help with RPC data communications.

References

[1] A. Azarbayejani and A. Pentland. Real-time self-calibrating stereo person tracking using 3-d shape estimation from blob features. In *International Conference on Pattern Recognition (ICPR)*, 1996.

[2] C. Bishop. *Neural Networks for Pattern Recognition.* Oxford Press, 1996.

[3] A. Blake and A. Yuille. *Active vision.* MIT Press, 1992.

[4] C. Bregler. Learning and recognizing human dynamics in video sequences. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 1997.

[5] R.A. Brooks. From earwigs to humans. *Robotics and Autonomous Systems*, 20(2-4), 1997.

[6] 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*, B39, 1977.

[7] S. Elliott and J. R. Anderson. The effect of memory decay on predictions from changing categories. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 1995.

[8] N. Gershenfeld and A. Weigend. *Time Series Prediction: Forecasting the Future and Understanding the Past.* Addison-Wesley, 1993.

[9] M. Isaard and A. Blake. A mixed-state condensation tracker with automatic model-switching. In *Sixth International Conference on Computer Vision*, 1998.

[10] T. Jebara. Action-reaction learning: Analysis and synthesis of human behaviour. Master's thesis, Massachusetts Institute of Technology, May 1998.

[11] T. Jebara, K. Russel, and A. Pentland. Mixtures of eigenfeatures for real-time structure from texture. In *Proceedings of the International Conference on Computer Vision*, 1998.

[12] M.I. Jordan and R.A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6:181–214, 1994.

[13] E. W. Large, H. I. Christensen, and R. Bajcsy. Scaling dynamic planning and control: Cooperation through competition. In *IEEE International Conference on Robotics and Automation*, 1997.

[14] K.S. Lashley. The problem of serial order in behavior. In L.A. Jefress, editor, *Cerebral Mechanisms in Behavior*, pages 112–136, New York, 1951. The Hixon Symposium, John Wiley.

[15] P. Maes, T. Darrel, B. Blumberg, and A. Pentland. The alive system: Wireless, full-body interaction with autonomous agents. *Special Issue on Multimedia and Multisensory Virtual Worlds, ACM Multimedia Systems*, 1996.

[16] M.J. Mataric. Behavior-based robotics as a tool for synthesis of artificial behavior and analysis of natural behavior. *Trends in Cognitive Science*, 2(3), 1998.

[17] Johnson N. and Hogg D. C. Learning the distribution of object trajectories for event recognition. *Image and Vision Computing*, 14(8), 1996.

[18] N. Oliver, A. Pentland, F. Berard, and J. Coutaz. Lafter: Lips and face tracker. In *Computer Vision and Pattern Recognition Conference '97*, 1997.

[19] A. Pentland and A. Liu. Modeling and prediction of human behavior. In *IEEE Intelligent Vehicles 95*, 1995.

[20] P. Pirjanian and H.I. Christensen. Behavior coordination using multiple-objective decision making. In *SPIE Conf. on Intelligent Systems and Advanced Manufacturing*, 1997.

[21] A.C. Popat. Conjoint probabilistic subband modeling (phd. thesis). Technical Report 461, M.I.T. Media Laboratory, 1997.

[22] D. Terzopoulos, X. Tu, and Grzeszczukm R. Artificial fishes: Autonomous locomotion, perception, behavior, and learning in a simulated physical world. *Artificial Life*, 1(4):327–351, 1994.

[23] E.L. Thorndike. Animal intelligence. an experimental study of the associative process in animals. *Psychological Review, Monograph Supplements*, 2(4):109, 1898.

[24] E. Uchibe, M. Asada, and K. Hosoda. State space construction for behaviour acquisition in multi agent environments with vision and action. In *Proceedings of the International Conference on Computer Vision*, 1998.

[25] E.A. Wan. Time series prediction by using a connectionist network with internal delay lines. In A.S. Weigend and N.A. Gershenfeld, editors, *Time Series Prediction*, 1993.

[26] J.B. Watson. Psychology as the behaviorist views it. *Psychological Review*, 20:158–17, 1913.

[27] A. Wilson and A. Bobick. Recognition and interpretation of parametric gesture. In *International Conference on Computer Vision*, 1998.