Action Reaction Learning: Analysis and Synthesis of Human 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://jebara.www.media.mit.edu/people/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 a mapping between 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 system drives a graphical interactive character which probabilistically predicts the most likely response to the 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

With advances in computation, the simulation and the analysis of behaviour has become a feasible proposition. In simulation domains, dynamics, kinematics, animal behaviour, rule based systems and reinforcement learning have been proposed to synthesize compelling interaction with artificial characters [3] [22] [20] [2]. Simultaneously, computer analysis and automatic learning of behaviour and dynamics from perceptual measurements has also strongly developed [26] [17] [18] [5] [4] [21]. Of particular relevance is the ability to predict regularities in human behaviour using computational models trained with machine learning [18]. We propose the combination of the effects of both behaviour simulation and perceptually driven behaviour analysis into a common automatic framework. The Action-Reaction learning approach acquires models of human behaviour from video and controls synthetic characters. Driven by these models and perceptual measurements, these characters are capable of interacting with humans in real-time. Ultimately, the user need not specify behaviour directly (and tediously) but teaches the system merely by interacting with another individual.

Earlier models of human behaviour proposed by cognitive scientists analyzed humans as an input-output or stimulus-response system [25] [23]. 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.

Of particular relevance is the close similarity of the stimulus-response behaviourist model to input-output learning algorithms. The Action-Reaction learning system is a probabilistic algorithm that uncovers a mapping between the stimulus and the response from interaction data. The goal of the model is not to classify behaviour into a variety of categories or for surveillance [21]. Typically, these classifications involve manual supervised segmentation and identification of specific types of behaviour. Rather, the model will be used for unsupervised analysis and its ultimate goal is the synthesis of such human behaviour with minimal artificial constraints, hand-wired knowledge and zero user intervention. The behaviour in question is limited to physical activities which can be measured by the system.[1]

The Action-Reaction Learning framework is initially discussed. The approach treats present activity as an input and future activity as an output and attempts to uncover a probabilistic mapping between them (i.e. a prediction). In particular, by learning from a series of human interactions, one can treat the past interaction of two individuals as an input and try to predict the most likely reaction of the participants. The probabilistic model is estimated using, *Conditional Expectation Maximization* which recovers a conditional density describing the input-output relationship between the two participants.

We also discuss the details of some of the perceptual inputs into the learning system. Subsequently, there is a description of the performance of the probabilistic model as it infers the optimal reaction to a past interaction. This drives the output of the system which is realized as a graphical character. A typical application as

[1] It is not essential that the input be 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.

well as some results illustrating the technique are then shown as the system learns to behave in a simple gestural interaction. Effectively, the system learns to play not by being explicitly programmed or supervised but simply by observing other human participants. Finally, current extensions to the formulation are described.

1.1 System Architecture

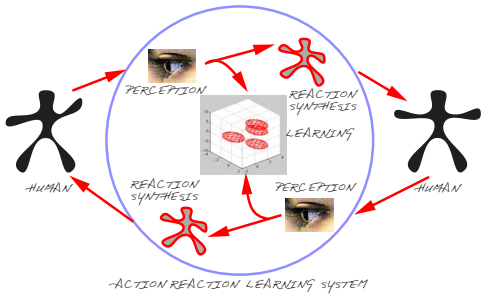


Figure 1: Offline: Learning from Human Interaction

The system is depicted in Figure 1 and in Figure 2. 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 real perceptual measurements using a vision subsystem for each of the humans. The temporal sequences obtained are then analyzed by a learning subsystem to determine predictive mappings between pieces of the sequences and their consequences.

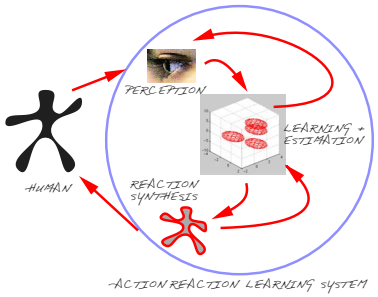


Figure 2: Online: Interaction with Single User

In Figure 2, the system has collected and assimilated the data and can now infer the maximum likelihood response to the actions of each individual. The perceptual system then only tracks the activity of the single remaining user and feeds it the learning system. The learning model performs an estimation and generates the most likely response to the user's behaviour by animating a computer graphics character in the synthesis subsystem. This is the main output of the ARL engine. It is fed back recursively into the learning subsystem

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

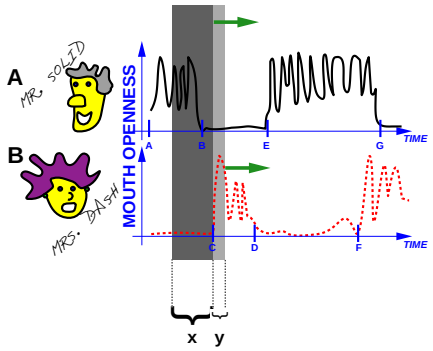


Figure 3: Dialog Interaction and Analysis Window

so that it can remember its own actions and generate self-consistent behaviour. In addition, the system determines the most likely action of the remaining user and transmits it as a prior to assist tracking in the vision subsystem.

1.2 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 [16]. 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 nonstop 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.

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 $\mathbf{y}^*$ 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 any 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.

2 Perceptual System

Of 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 parameters should be compact. 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 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.

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

The parameters of the pdf ($p(i)$,μ_i and Σ_i) are estimated using the Expectation Maximization algorithm

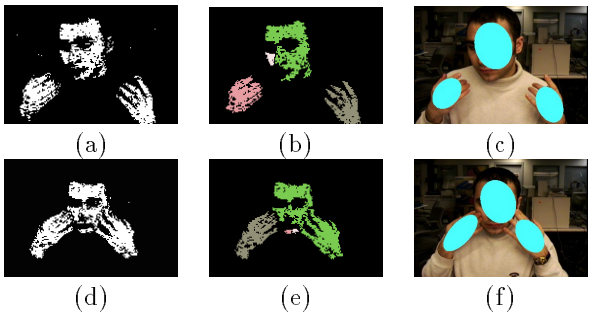


Figure 4: Head and Hand Blob Tracking

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 (as in Figures 4(a) and (d)).

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). 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 which 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. The resulting 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 of the parameters is done in real-time by iteratively maximizing the likelihood over each image. The resulting 3 Gaussians have 5 parameters each (from the 2D mean and the symmetric covariance) 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 $\Gamma \times \Gamma = \Sigma$. The Γ matrix has 3 free parameters ($\Gamma_{xx}, \Gamma_{xy}, \Gamma_{yy}$) which are closer to the dynamic range of the 2D blob means. 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. Consequently they are adequate for temporal modeling. Higher order measurements could be added in the future but these might be more unstable to estimate and might have non-linear phenomena associated with them. The

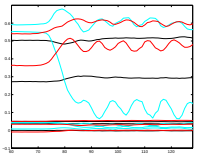


Figure 5: Time Series Data of 3 Blobs (1 User)

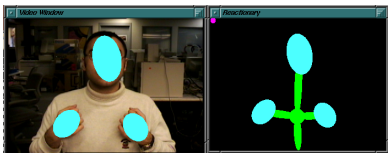


Figure 6: Perceptual Measurements and Graphics

15 recovered parameters from a single person are shown as a well behaved, smooth time series in Figure 5. 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.

3 Graphical System

At each time frame, the 15 estimated parameters for the Gaussians can be rendered for viewing as in Figure 6. This is also the display provided to each user so that he may view the gestures of the other human (or computer) player through his personal computer screen.

4 ARL System

The Action-Reaction Learning system functions as a server which receives time series data from the vision systems and re-distributes it to the graphical systems for rendering. Typically, two vision systems and two graphics systems are connected to the ARL server. Within the ARL server, tracking data for the vision systems is stored as a long time series. Both sets of Gaussians from each vision system are concatenated to form 30 parameters that evolve as a multi-dimensional time series (6 Gaussian blobs over time). The ARL system preprocesses and trains from this time series data. Once converged, it can begin predicting the evolution of the time series and, equivalently, estimate the parameters of the 6 blobs in the near future.

5 Time Series Pre-Processing

Time series techniques for predicting the evolution of many variables have been well documented in [9]. The techniques typically recover relationships between past time sequences and their future consequences. These range from neural networks [15] to HMMs [8]. For a neural approach, there exist many different possibilities for processing and representing temporal data. The simplest one is to merely rasterize each time series into a vector. The window of 128 samples is placed over the

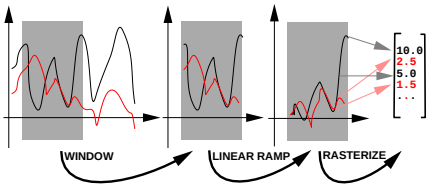
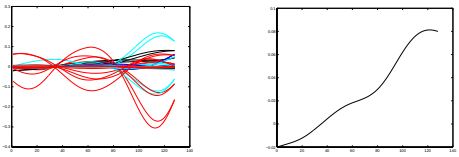


Figure 7: Pre-Processing the Past Sequence



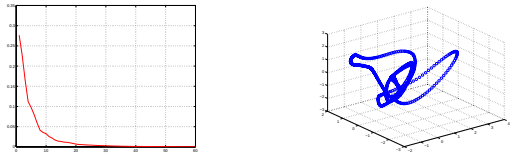
(a) For all 30 Features (b) For Feature 1

Figure 8: First Eigenvector of Past Interaction

time series data and the 30 different streams are rasterized into a 30×128 element vector (i.e. in $\mathcal{R}^{3840}$). This window effectively covers $128 \times 50ms = 6.5$ seconds of temporal data. This data is weighted linearly by its distance from time zero (i.e. the right side of the sliding window). Thus, a linear ramp function is multiplied with each time series. This reflects our intuition that the more temporally distant the elements in the time series the less relevant they are for prediction [7]. The linear ramp multiplication and rasterization process is depicted in Figure 7.

Each 50ms, the window slides 50ms to the right, and we re-rasterize the data into another vector in $\mathcal{R}^{3840}$. From a total of a few minutes of such time series data, thousands of these vectors are accumulated in an offline process. A Principal Components Analysis is then performed (i.e. computation of the top eigenvectors of a Gaussian estimate of the $\mathcal{R}^{3840}$ data). The PCA analysis obtains the top 40 most energetic modes of variation of the $\mathcal{R}^{3840}$ which account for more than 95% of the signal energy. Thus, we can represent each vector $\mathcal{R}^{3840}$ by its coefficients in the subspace spanned by these modes and effectively reduce its dimensionality to $\mathcal{R}^{40}$. The past 6 second interaction between two individuals (6 blobs) is represented with these 40 parameters. In addition, we concatenate to these parameters the most recent values of the time series (the right most frame in the window) as well as the time series velocity. This increases the total dimensionality to $40 + 30 + 30 = 100$ parameters for the sliding window. In Figure 8 the first mode (eigenvector) is shown for the time window as well as the piece corresponding to the x coordinate of the head. This demonstrates, interestingly, that the eigenvector's shape is not a sinusoid, wavelet or other standard basis function since it is specialized to the training data.

We wish to use this $\mathcal{R}^{100}$ vector (call it $\mathbf{x}$) of past interaction to predict the 30 parameters of the 6 blobs in the next 50 ms video frame (right after the attentional window). Thus, immediately after the time window,



(a) Top 60 Eigenvalues (b) Projection onto top 3

Figure 9: Eigenspace Properties for $\mathbf{x}$

one may simply vectorize the parameters of the 6 blobs into a $\mathcal{R}^{30}$ vector, $\mathbf{y}$. The $\mathbf{x}$ vector represents the past action and the $\mathbf{y}$ represents the consequent reaction. 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 time series and processing it as above. Figure 9 shows the first 60 eigenvalues and the evolution of $\mathbf{x}$ in its first 3 coefficients in the eigenspace as we slide the window over ≈30 seconds. Given sufficient pairs of vectors, it is possible to learn how to predict the interaction of two users in a constrained context.

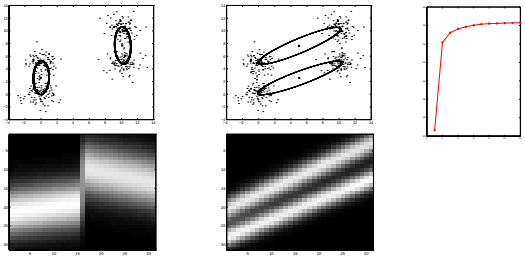
6 Conditional Expectation Maximization

To model the action-reaction space we shall use a probabilistic approach. Due to the randomness of the interactive behaviour in humans, the noise in the perceptual systems and the sparseness of the observations, an exact deterministic mapping would be difficult to compute. 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. In other words, the density will usually be used to predict the 30 dimensional reaction from the past action and is not just a model of the full $100 + 30$ dimensional space.

Due to its flexibility to model non-linear phenomena and its ease of use, we use a conditioned mixture of Gaussians [13]. This model is depicted in Equation 3 with $\mathcal{N}$ representing a normal distribution or Gaussian model.

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

Traditionally, training 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 $\mathbf{y}$ ($\mathbf{x}$ is always measured).



(a) $\max L$ (b) $\max L_c$ (c)

Figure 10: Conditional Density Estimation

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

We recently developed a variant of the EM algorithm called Conditional Expectation Maximization (CEM) for specifically optimizing conditional likelihood [11]. 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.

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

Applying CEM to the pdf optimizes its $p(\mathbf{y}|\mathbf{x})$ over the data [3]. This is exactly the intended use of the learning system. 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 [13] [19] 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 $\mathbf{x}$).

Figure 10(a) and (b) depicts a situation where the CEM algorithm (in (b)) outperforms the EM algorithm (in (a)) when fitting a pair of Gaussians. In this example, the EM algorithm optimized L and was then used to compute L_c. The CEM algorithm directly optimized L_c and has therefore obtained a superior conditional density estimate. The top figures display the positions of the Gaussians in the joint density and the bottom figures show the estimated conditional density $p(y|x)$, reflecting the fact that x will be given.

For the ARL system, training data consisted of approximately 6000 $(\mathbf{x},\mathbf{y})$ pairs occupying 100 and 30 dimensions respectively. The number of Gaussians used in the conditional model was $M = 25$. Figure 10(c)

[3] The ability to predict $\mathbf{y}$ if $\mathbf{x}$ is observed.

displays the conditional log likelihood (L_c) as the CEM algorithm optimizes and monotonically converges to the maximum conditional likelihood solution (from an initial random state). The system took slightly under one hour on an SGI OCTANE to achieve the displayed convergence for a 5 minute long training sequence of interactions.

7 Prediction

Assume that the system has just come online after an extended offline examination of two human participants. Now, the system is alone with a single human participant and it is obtaining some perceptual measurements of the user's actions over the past time window. This data is presented to the probabilistic model $p(\mathbf{y}|\mathbf{x})$ which now gives us a distribution over possible reactions. Using the estimated pdf, $p(\mathbf{y}|\mathbf{x})$, we can predict the future reaction given any past interaction. This is done by again obtaining an attentional window over a few seconds of past data and pre-processing it into a vector $\mathbf{x}^*$. The values of $\mathbf{x}^*$ are inserted into the conditional density generating a density exclusively over the variable $\mathbf{y}$. This density is effectively a 30 dimensional, M-component Gaussian mixture model.

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 $\mathbf{y}^*$, the most likely reaction according to the model.

$$\begin{aligned} \mathbf{y}^* &= \int \mathbf{y} p(\mathbf{y}|\mathbf{x}^*) d\mathbf{y} \;=\; \frac{\sum_m^M \mathbf{y}^*_m p(\mathbf{y}^*_m|\mathbf{x}^*)}{\sum_m^M p(\mathbf{y}^*_m|\mathbf{x}^*)} \\ \text{where} & \\ \mathbf{y}^*_m &= \mu^y_m + \Sigma^{yx}_m \Sigma^{xx\,-1}_m (\mathbf{x}^* - \mu^x_m) \end{aligned} \tag{6}$$

8 Synthesis and Feedback

Thus, the system has predicted the reaction to what the user has done. This reaction is represented in the same way as the signals generated by the perceptual system. So, if it is possible to undo the work of the perceptual system at this stage (i.e. with graphics) by inverting the perceptual measurement and realizing a physical manifestation of it (i.e. an animated blob and stick figure display), the system can interact with the user in a compelling manner.

In addition, the reaction (in the form of perceptual measurements) that the ARL system has just synthesized can be fed back in as a component of the next $\mathbf{x}$ input (along with the perceptual measurements taken off of the user). Thus, the system has a memory of its own past actions or an internal state. The process continues and time evolves, sliding the temporal perceptual measurements taken off the observed user into the ARL window dynamically. The ARL system simultaneously generates a stream of perceptual measurements for its own character and displays the graphical output associated with them.

More specifically, the half of the predicted $\mathbf{y}^*$ vector that will be used to animate the system's virtual character will be denoted $\mathbf{y}^*_B$ and the half that predicts what the current user should do next will be called $\mathbf{y}^*_A$. Figure 2 demonstrates the system's continuous self feedback process. Half of the multi-dimensional time series evolves due to the user and the other half is the accumulated history of the predictions the system has performed. This allows the system to have a memory of its own synthesized actions and maintain self-consistent behaviour. Of course, for the initial few seconds of interaction, the system has not synthesized any actions (i.e. no $\mathbf{y}^*$ vectors have yet been computed). Therefore, the first few seconds of the time series that should correspond to the synthesized character are merely set to arbitrary default values, effectively bootstrapping the system.

Simultaneously, the real-time graphical blob representation is used to unmap the probabilistically predicted perceptual measurement (the $\mathbf{y}^*_B$ action) for the visual display. It is through this display that the human user receives feedback in real-time from the system's reactions and interaction attempts. This is necessary to maintain the interaction which requires the human user to pay attention and respond appropriately. The graphics system is kept simple and merely renders blobs in a 1 to 1 mapping since it displays to the user exactly what is perceived (merely three blobs) by the system. The ARL system's primary concern is head and hand position and this becomes clear from the coarse display. In addition, the user is not as misled into expecting human-level intelligence from such a simple output.

9 Perceptual Feedback

Evidently, the predicted user's action $\mathbf{y}^*_A$ could also be fed back into the time series doing away with any human input altogether. However, the system often locks up into some looping meaningless behaviour when both halves of the time series are completely synthesized with no human tracking. Some modifications are being investigated for zero-user, two-computer configurations.

More importantly, however, the $\mathbf{y}^*_A$ vector does have two critical applications. It can be used as a predictor of the user's actions to help the tracking since it is somewhat of a sophisticated dynamical system and it can be used to resolve some vision system errors.

9.1 Beyond Dynamics

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 estimate. Kalman filtering and other dynamic models involve more sophistication ranging from constant velocity models to very complex dynamic systems [10]. Here, the dynamics and the feedback being used to constrain the vision system are the results of not only dynamics but also behaviour modeling. This is similar in spirit to the mixed dynamic and behaviour models in [18]. The system thus continuously feeds back prediction estimates of the 15 parameters corresponding to the 3 Gaussians being tracked in the vision system for improved results.

9.2 Blob Correspondence

Initially, colored gloves were used to overcome some of the blob correspondence problems that would occur when heads and hands touched and moved by each other. The first few training sequences involved no blob miscorrespondence due to explicit labeling of the head, left hand and right hand. However, once appropriately

Interaction	User	Corresponding Action
1	A	Scare B by moving towards camera
	B	Fearfully crouch down & bring hands in
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

Table 1: Interaction Instructions

trained, the probabilistic model described above feeds back the positions of the Gaussians to the vision and 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. This permits us to reliably do away with colored gloves. A coarse model of $p(\mathbf{x})$ is available and can be evaluated to determine the likelihood of any particular past interaction. If permutations of the blobs being tracked by the computer vision are occasionally tested with $p(\mathbf{x})$, any mislabeling of the blob features can be detected and corrected. The system merely selects one of the 6 permutations of 3 blobs that maximizes $p(\mathbf{x})$ and then feeds back the appropriate $\mathbf{y}^*$ estimate to the computer vision. Instead of using complex static computations, a reliable correspondence between the blobs is computed from the temporal information.

10 Interaction Results

It is prudent to train the ARL system in a constrained context to achieve some kind of learning convergence from limited data and limited modeling resources. Thus, the users involved in training the system initially are given some loose instructions on the nature of the interactions they will be performing. The users were given the instructions listed in Table 1.

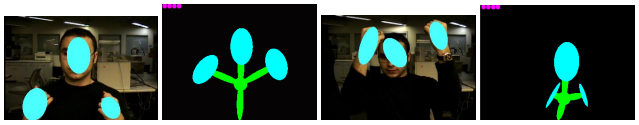
The two users (A and B) begin by playing the above game and A gestures while B responds appropriately. The users are physically separated from each other and only see graphical representations of each other on their screens. The learning algorithm is given measurements of the head and hand positions of both users. These measurements are taken off of two players for several minutes of interaction. These sequences generate many input-output pairs when the fixed size analysis windows slide over them. These pairs are used to train the system which is then able to impersonate player B. 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 the actions of the character are synthesized by the ARL system as opposed to the other player.

Two evaluation methods were used. The system was required to evaluate a real interaction between two individuals and predict the reactions of both by only viewing the past interaction. The RMS errors are shown in Table 2 and compared with nearest neighbour estimates and constant velocity estimates.

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 1, the system responded with a (qualitatively) appropriate animation of the synthetic character (the gesture of the missing user B). Figure 11 shows a sample interaction. In (a) the user is in a normal state ready to start the gesture and the synthetic character is in a relaxed state in (b). In (c) the user is performing a menacing gesture and the system responds by crouching down in fear and bringing its hands close as in (d). Moreover, the responses from the system contain some default pseudo random variations giving them a more compelling nature.

Nearest Neighbour	Constant Velocity	ARL
1.57 %	0.85 %	0.64 %

Table 2: RMS Errors



(a) (b) (c) (d)

Figure 11: Real-Time Interaction with Character

The user is *delegating* tasks to the animated character since it is not a simple 1-to-1 mapping between current measurements to output. The user produces a complex action and the system responds with a complex action that depends on that input as well as on its own previous internal state.

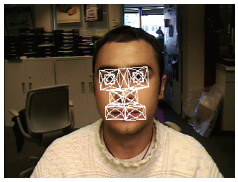
11 Current and Future Work

11.1 Continuous Online Learning

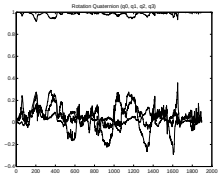
It is also feasible to continue the training of the CEM algorithm while the system acquires more data and performs synthesis. Thus, it performs online learning and updates its mixture of conditional models dynamically as it obtains new samples. This and the fact that the ARL system interacts with a user 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 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.

11.2 Face Modeling for Interaction

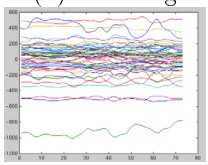
Face modeling can also be used to recover more perceptual details for a convincing interaction. A system which automatically detects the face and tracks it has been implemented [12]. It is capable of tracking the 3D rotations and movements of a face using normalized correlation coupled with structure from motion. In addition, at each moment in time, it can compute



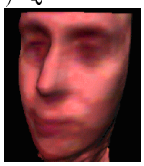
(a) Tracking



(b) Quaternions



(c) 3D Eigen Deformations



(d) 3D Synthesis

Figure 12: 3D Face Modeling and Tracking

an eigenspace model of the face's texture which is then used to infer some 3D deformations. This system generates a real-time temporal sequence which includes XYZ translations, 3D rotations and texture and deformation coefficients (see Figure 12).

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 12(d). The data representing each static frame is again a time series (≈50 dimensional) permitting the ARL system analysis to extend to this platform.

12 Conclusions

We have demonstrated a real-time system which learns two-person interactive behaviour fully 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 and impersonate the missing person by estimating and simulating the most likely action to take.

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] N.I. Badler, C. Phillips, and B.L. Webber. *Simulating Humans: Computer Graphics, Animation and Control*. Oxford University Press, 1993.

[3] B. Blumberg, P. Todd, and P. Maes. No bad dogs: Ethological lessons for learning. In *From Animals To Animats, Proceedings of the Fourth International Conference on the Simulation of Adaptive Behavior*, 1996.

[4] A. Bobick. Movement, activity, and action: The role of knowledge in the perception of motion. In *Proceedings of Royal Society: Special Issue on Knowledge-based Vision in Man and Machine*, 1997.

[5] C. Bregler. Learning and recognizing human dynamics in video sequences. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 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] A.M. Fraser and A. Dimitriadis. Forecasting probability densities by using hidden markov models with mixed states. In A.S. Weigend and N.A. Gershenfeld, editors, *Time Series Prediction*, 1993.

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

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

[11] T. Jebara and A. Pentland. Beyond em: General bound maximization, conditional densities and the cem algorithm. Technical report, M.I.T. Media Laboratory - Vision and Modeling, Forthcoming.

[12] 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.

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

[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] M.C. Mozer. Neural net architectures for temporal sequence processing. In A.S. Weigend and N.A. Gershenfeld, editors, *Time Series Prediction*, 1993.

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

[17] A. Pentland. Machine understanding of human action. Technical Report 350, M.I.T. Media Laboratory - Vision and Modeling, 1995.

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

[19] A.C. Popat. *Conjoint Probabilistic Subband Modeling*. PhD thesis, M.I.T. Media Laboratory, 1997.

[20] K. Sims. Evolving virtual creatures. In *Proceedings of SIGGRAPH '94*, volume 26, 1994.

[21] T. Starner and A. Pentland. Visual recognition of american sign language using hidden markov models. In *International Workshop on Automatic Face and Gesture Recognition*, 1995.

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

[26] Y. Yacoob and L. Davis. Learned temporal models of image motion. In *Proceedings of the International Conference on Computer Vision*, 1998.