Responsive Listening Behavior

M. Gillies[1], X. Pan[2], M. Slater[2] & J. Shawe-Taylor[2]

[1]Department of Computing, Goldsmiths, University of London

New Cross, London SE14 6NW, UK

email: m.gillies@gold.ac.uk

[2]Department of Computer Science, University College London

Malet Place, London WC1E 6BT, UK

Abstract

Humans use their bodies in a highly expressive way during conversation, and animated characters that lack this form of non-verbal expression can seem stiff and unemotional. An important aspect of non-verbal expression is that people respond to each other's behavior and are highly attuned to picking up this type of response. This is particularly important for the feedback given while listening to some one speak. However, automatically generating this type of behavior is difficult as it is highly complex and subtle. This paper takes a data driven approach to generating interactive social behavior. Listening behavior is motion captured, together with the audio being listened to. This

data is used to learn an animation model of the responses of one person to the other. This allows us to create characters that respond in real-time during a conversation with a real human.

Keywords: computer animation, non-verbal behavior, motion capture, machine learning

1 Introduction

A key element of social interaction is that people respond to each other. The meaning of a person's behavior is not solely contained in that behavior but how that behavior responds to others. This responsiveness is vital for creating characters that can interact believably with real people. Without it, interactive media and games cannot fully achieve realistic social interaction. This responsiveness is what makes a character into something truly interactive rather than merely being pre-scripted. It is thus, not enough for characters' movements to look right, if they do not respond appropriately to real people's actions in real time. This is particularly true of feedback given while listening to somebody speaking. People feed back non-verbally whether they are interested; whether they agree or disagree with the person, and other reactions to the speech. It was shown in Pertaub, Slater and Barker [1] that people have a strong emotional response when talking to an audience of animated characters, if the characters display strongly interested or uninterested behavior. Creating believable, real-time and responsive behavior is one of the key challenges for computer animation.

The primary contribution of this paper is a method of learning responsive behavior from motion capture data. This method can create characters that are able to interact in real-time with real people, responding to a number of different user inputs. In this paper the characters respond to the person's voice. The first aspect of the contribution is the method by which data from a conversation between two people can be captured and used for creating interactive behavior. We motion capture the listener's responses and either simultaneously

record the voice of the speaker or, alternatively, we can capture responses to a pre-recorded speech. The motion capture data is used to create a motion graph data structure that can be used to generate new motion in real time. The main technical contribution is a novel method that uses the relationship between the inputs and motion data to learn a value function for edges in the motion graph. This value function is used to drive the motion generation in response to new inputs. Therefore, the method results in real-time prediction and realization of the behavior of a virtual character as a function of the behavior of a real tracked person. This is important because, for the first time, it provides the possibility of highly realistic, but data driven interaction between real and virtual people.

2 Related Work

2.1 Data Driven Animation

There are two basic classes of techniques for character animation, procedural and data driven. Procedural methods (for example, Perlin [2], Tu and Terzolpoulos [3], Neff and Fiume [4] or Kopp and Wachsmuth [5]) generate motion entirely from algorithmic or mathematical models, for example, physical simulations. Data driven techniques on the other hand use data from human movement, normally from motion capture. The advantage of data driven techniques is that they make it easy to create highly realistic animation, as they are based on real human motion. They are also able to easily model the movements of a

particular individual. The major research area for data driven animation is the task of motion editing, taking one or more pieces of motion data and adapting or combining them to produce a new animation that is suited to a new circumstance. This basic problem takes many forms. Sometimes new constraints need to be added to a motion. Generally these are inverse kinematics constraints such as reaching for an object or walking over rough terrain (e.g. Gleicher[6] or Lee and Shin [7]). Constraints are also important when retargetting motion to new characters (e.g. Gleicher [8] or Tak and Ko [9]). It is also useful to be able to play a number of different motions in sequence, transitioning from one to the other. Since finding transitions between arbitrary motions is a very difficult problem, most research has focused on ways of finding suitable points at which a transition can easily be made, for example graph based techniques [10, 11, 12], which are used in this paper. More generally data driven techniques can be used to create simplified interfaces for controlling characters, for example the work of Lee et al. [12]. Finally, there has been considerable interest in changing the style of a motion or applying the style of one motion to a different motion (e.g. Hsu et al. [13] or Brand and Hertzmann [14]). Techniques range from interpolating multiple motions [15, 16] to using physical simulation to alter the details of a motion [17, 18]. Taken together the methods described form a powerful tool-set for creating highly realistic and flexible animation.

An important aspect of data driven techniques is the method used for data capture. This paper proposes a capture methodology and a method for refining a model once the data has been captured. In this the work is similar to that of Cooper *et al.*[19], who propose a capture

and interactive refinement method for data driven models. Their model is different from ours, however, in that their refinement step occurs during the capture session while ours occurs afterwards.

2.2 Expressive Behavior

Animated characters that can engage in realistic, responsive social behavior and express emotion can add far greater depth to interactive media than has previously been possible. There has been considerable work on creating characters with this type of expressive behavior [20], for example Cassell *et al.*'s [21] and Garau *et al.*'s [22] work on social interaction, or Gratch and Marsella's [23] and Poggi, Pelachaud *et al.*'s [24] work on emotion. The closest work to our own application has been Maatman, Gratch and Marsella [25], who created a character that listens responsively to a speaker. However, this work has been primarily knowledge driven, whereas we take a data driven approach that we believe is better able to capture the subtleties of human conversational behavior. Some researchers such as Nakano *et al.* [26] and Lee, Badler and Badler [27] have created models of expressive behavior based on data, but they used hand annotation of video sequences to extract the data as well hand analysis to design the model. This means that creating the behavior model was a very labor intensive process. Using motion capture and machine learning techniques we show that it is possible to create this type of interactive behavior automatically. Kipp et al.[28] have presented a machine learning method for reproducing the gestural style of particular indi-

viduals, though their input data still requires manual annotation of video sequences. Egges *et al.* [29] learned a model of expressive posture from motion capture data using Principal Component Analysis, however, their aim was not to produce responsive behavior. Stone *et al.* [30] created a conversational character based on motion capture data. Unlike our work the responsiveness of their character was based largely on a hand authored language model rather than being wholly learned from data.

2.3 Responsive Behavior

There has been some work on data driven methods for achieving responsive behavior, though not applied to expressive social interaction. Kim, Park and Shin [31] and Shiratori, Nakazawa and Ikeuchi [32] have characters that are able to respond to the rhythm of music. Hsu, Gentry and Popović [33] demonstrated an animated dancer that was able to respond to the moves of its partner. They assume that the movements of the partner are known in advance and do a search for an optimal solution to the entire sequence. Though this gives good results for off-line responsiveness, and is likely to out-perform our method in this context, it is less suited to real time interaction. Doing a complete search is expensive, and is likely to be impossible if inputs are received one at a time. When producing responsive behavior it is important to choose motion clips based on the quality of match to the user input both of the clips themselves, and also of future clips that they make available. In order to respond in real time, we adapt methods from reinforcement learning [34] to do this

choice without searching all possible future clips. Reinforcement learning is the process of learning by giving a measurable reward for correct behavior (and punishments for incorrect behavior). This has been applied very literally to animation by Blumberg *et al.* [35], who use the metaphor of "training" an animated dog by rewarding and punishing it. Lee, and Lee [36] use reinforcement learning to train a virtual boxer, giving rewards based on how close a punch is to the desired position. Similarly McCann and Polllard [37] use similar methods for training a character that can produce realistic responses to a navigation control signal from a user, while Treuille, Lee and Popović [38] have used reinforcement learning for autonomous navigation behaviour. This type of learning works well for goal directed behavior, for example, in boxing the goal is to hit the opponent. However, the type of social behavior we are simulating, does not have this type of explicit goal, so a reward cannot be easily formulated and reinforcement learning cannot be used. One of the main contributions of this paper is therefore to show how similar techniques to reinforcement learning can be adapted to a non-reinforcement learning context in which a reward function cannot be used. This paper presents a method for simulating small scale, two person interaction. At the other end of the scale, Lee *et al.* have proposed a data driven method for simulating large scale crowd behavior[39].

3 Learning Listening Behavior

The aim of this work is to produce realistic and responsive non-verbal behavior for conversations between a real person and an animated character in real-time, such as in an interaction within a virtual environment. Our character responds to an input signal from a real person. In our current example, these signal is the person's voice. Voice is a suitable signal as it is closely related to expressive behaviour, and it has been shown (e.g. a study by Kendon [40]) that the non-verbal behaviour of listener responds to elements of the speakers voice. It is also a useful input to use for real time interaction with an animated character, as it can be capture much more easily and unintrusively than, say, body movement. The novelty of our approach is that we use a wholly data driven method, using data captured from a real conversation to create responsive behavior. This method is completely automated, and therefore provides substantial labor saving over creating behavior models by hand.

3.1 Data Capture

The capture process is shown in figure 1. We motion capture a listener responding non-verbally to speech. This can either be speech from a pre-recorded audio file, or live conversation. In either case we make a recording of the speech that is being responded to. From these two sets of data we create a model of both the style of movement of the motion captured person and the style of behavior: how they respond to other people. The motion capture data is used to create a motion graph based data structure [10, 11, 12] that is used to

generate animation. The input signals from the conversational partner are quantized to form a set of discrete input features: $\mathbf{f} = \{f_0 \dots f_n\}$. The stream of features is then synchronized with the motion data. We a synchronization signal at the start and end of each data set. The actor being captured was asked to adopt a standard posture and rapidly move out of it at the end of a 3, 2, 1 count down heard in the audio signal. These two signals were used to align the start and end of the two data set, which could then be resampled to the same frame rate. The result is a set of frames containing both motion data and input features: $(\mathbf{m}, \mathbf{f})$. In order to use the motion graph to generate responsive behavior, we use the co-occurrence of input features and motion data to learn a value function that determines how appropriate each edge in the graph is to each combination of input features.

3.2 Handling Motion Data

For low level behavior generation we use a method based on Motion Graphs [10, 11, 12]. This gives a way of decomposing one or more long motion clips in such a way that they can be re-played in a different order. The key to the method is to find frames in the motion sequences where realistic looking transitions can be made to new places in the motion. A motion graph is a graph structure as shown in figure 2, in which nodes represent possible transition points and edges represent motion clips that join different transition points. In our implementation we make an explicit distinction between two types of edge:

Continuation Edges (shown in black in figure 2) represent continuing along the original

motion sequence from the current node without making a transition. The edge ends at the next transition point in the motion. If this end point results in a motion clip that is too short a later transition point is used to ensure that the resulting clip is longer than a minimum clip length.

Transition Edges (shown in grey in figure 2) represent a transition to a new position in the same or a different motion sequence.

Nodes can be either the start of transitions in which case they will have one or more outgoing transition edges, or the end of a transition in which case they will have an incoming transition edge but no outgoing ones. Continuation edges use the original motion but for transition edges we need to create a new motion clip that smoothly interpolates the start and end points of the transition. To do this we use a variant of Kovar and Gleicher's formula [11]:

$$p(t) = \alpha(t)p_0(t_0 + t) + [1 - \alpha(t)]p_1(t_1 - k + t)$$

$$q(t) = slerp(q_0(t_0 + t), q_1(t_1 + t), \alpha(t))$$

In which $p(t)$ and $q(t)$ are the root position and quaternion joint orientations of the new motions, p_0, q_0 and p_1, q_1 are the original motion sequences and t_0 and t_1 are the times of the transitions points. k is the length of the transition and $\alpha(t)$ is a function used to ensure C^1 continuity:

$$\alpha(t) = 3\left(\frac{t}{k}\right)^2 - 2\left(\frac{t}{k}\right)^3$$

New motion can be generated by performing a graph walk and at each node choosing one outgoing edge. We use the value function defined in the next section to choose the edges.

We construct the motion graph using the method suggested by Kovar and Gleicher [11], with the exception that we use Lee et al.'s [12] function for determining the difference between frames in a motion. This uses instantaneous joint rotations and angular velocities at the frame, in contrast to Kovar and Gleicher's function that is based on a set of surface points on the character summed over a time window to get velocity information. Lee et al.'s method was used as it is more readily and efficiently calculated from the standard data available in an animation system, and there is little evidence that Kovar and Gleicher's method gives superior results.

3.3 Handling Voice Data

In order to be able to drive character behavior from voice, we must extract a suitable set of features from the voice. These features should be good predictors of the character's behavior. The first step is to extract the pitch (fundamental frequency) and intensity (loudness) of the voice using Boersma's periodicity analysis method from the Praat tool[41, 42]. We then calculate the mean and variance of these features over a 1 second window of each frame. Our method (described in the next section) requires discrete input features. We use a quantization method that aims to maximize the information that the discrete audio variable gives us about the character's motion. We therefore choose a discretization that maximizes the conditional entropy:

$$H[a|\mathbf{M}] = \sum_{a} \int p(a, \mathbf{M}) \ln p(\mathbf{M}|a) d\mathbf{M} \tag{1}$$

Where a is the discretized audio variable and $\mathbf{M}$ is the motion. $H[a|\mathbf{M}]$ measures the average information required to specify the value of $\mathbf{M}$ if a is already known. Therefore minimizing this quantity will give a discretization that best predicts $\mathbf{M}$. Applying equation 1 requires conditional and joint probability distributions for a and $\mathbf{M}$. Fitting a continuous distribution to $\mathbf{M}$ can be problematic if the distribution is not a good model of the movement. For example, an obvious choice, a multivariate Gaussian performs particularly badly, as the distribution across the motion is multi-peaked. We avoid the problem by discretizing $\mathbf{M}$. We perform a principal component analysis on the frames of the motion, using both position and velocity information, and then perform vector quantization on the result. The vector quantization is performed finely (number of quanta 100) to obtain a discrete variable m. We can then model two discrete distributions $p(a)$ and $p(m|a)$ whose values can be learned by counting occurances in the data set. Rewriting equation 1 we obtain the following equation for the conditional entropy:

$$H[m|a] = \sum_{a} p(a) \sum_{m} p(m|a) \ln p(m|a) \tag{2}$$

As this value is relatively quick to compute we can maximize $H[m|a]$ in an unbiased way by performing an exhaustive search. Testing each data point as a possible threshold for discretization. As is it also desirable for the inputs to have low dimensionality we also use the conditional entropy to choose the best input dimensions to use. Either the best overall dimension (from the mean and variance of both pitch and intensity) is used if it is a sufficiently good predictor, otherwise we choose the best pitch measure together with

the best intensity measure (as the two measure from pitch/intensity are generally highly correlated).

3.4 Generating Behavior

Generating animation in response to new input features means finding a path through the motion graph, whose edges are most appropriate to the current input features. This is done by assigning a value $V_{e,\mathbf{f}}$ to each edge e for each combination of possible input features $\mathbf{f} = \{f_0 \dots f_n\}$. $V_{e,\mathbf{f}}$ represents how well the edge fits the input features. Given values for each edge, a path can be found by choosing, at each successive node, an outgoing edge based on its value. The generation process is shown in figure 3. The quantized input features (see previous section) are used to evaluate the value of each outgoing edge of the current node. Once an outgoing edge is chosen the corresponding motion clip is played. When the clip has finished, the process is repeated using the end node of the chosen edge.

Before discussing the exact form of $V_{e,\mathbf{f}}$ we will explain how it is used, which will make the choice of $V_{e,\mathbf{f}}$ clearer. At each node we must choose one outgoing edge. This edge may be either a continuation edge or a transition edge. If a continuation edge is chosen, the motion clip attached to that edge is played and then the next edge is chosen. However, it is important not to play more than one transition in a row or the motion will become notably unrealistic. For transition edges we therefore play the transition clip attached to the edge, followed by the continuation clip of the edge's end node. Only then do we choose a new

edge. It is generally a good idea to avoid taking too many transition edges. If the value of a continuation edge is high it is often better to choose the continuation edge, rather than the optimal edge, in order to ensure greater continuity in the generated motion. We manage the trade off between using the best edge according to $V_{e,\mathbf{f}}$ and using the continuation edge by choosing one or the other at random. The probabilities we use for this selection depend on two factors. Firstly the value, $V_{e,\mathbf{f}}$, measuring how good a fit the edge is to the current input features. The second factor is a penalty based on the quality of the transition. For this we use the function suggested by Lee et al [12]:

$$p_t = \exp(\frac{-D_{i,j}}{\sigma})$$

where $D_{i,j}$ is the distance measure described above in equation **??** and σ is a parameter for controlling the degree to which poor transitions are penalized. In our method this becomes a useful trade off parameter that controls how likely the system is to choose a poor transition that is a good match to the current input features. For continuation edges we define $D_{i,j}$ to be 0 and so $p_t = 1$ and we can ignore it. Therefore we choose the edge with the maximum value with probability:

$$P_{\max} = \frac{p_t V_{\max(n),\mathbf{f}}}{p_t V_{\max n,\mathbf{f}} + V_{\mathrm{cont}(n),\mathbf{f}}} \tag{3}$$

Where $\max(n)$ is the outgoing edge of node n with the highest value, and $\mathrm{cont}(n)$ is the continuation edge of n. The probability of choosing the continuation edge is $1 - P_{\max}$.

How do we find $V_{e,\mathbf{f}}$? As described in section 3.1, we capture data in the form of motion capture synchronized with input features. This gives us information about which pieces

of motion data co-occurred with which input features. $V_{e,\mathbf{f}}$ can therefore be chosen based on whether features $\mathbf{f}$ co-occurred with the motion clip $\mathbf{M}_e$ corresponding to edge e. We define $\mathbf{M}_e$ in the following way. If e is a continuation edge, then it directly corresponds to a motion clip from the original data set. However, transition edges have automatically generated transitions, and so do not correspond to motion from the original data set. We therefore define the motion corresponding to a transition edge to be the motion attached to the continuation edge of its end node. The simplest way to define $V_{e,\mathbf{f}}$ would be to make it equal 1 if $\mathbf{M}_e$ co-occurred with features $\mathbf{f}$ and 0 otherwise. However, there are two problems. Firstly, the input features might change over the length of $\mathbf{M}_e$. This problem can easily be overcome by averaging over all the frames in $\mathbf{M}_e$. Another problem is that we loose a lot of information if the set of features $\mathbf{f}$ is multidimensional. By using only direct equality as a measure we loose any information about similar feature sets, for example, sets of features that differ only in a small number of features. This again, can be easily solved by replacing simple equality with a count of the number of equal features in the the two feature sets. We can therefore define the value $V^0_{e,\mathbf{f}}$ for an edge:

$$V^0_{e,\mathbf{f}} = \frac{\sum_{i \in \text{frames}(M_e)} \sum_{j=1}^{N} \delta(f_j, f_j(i))}{\sum_{i \in \text{frames}(M_e)} 1}$$

Where f_j is the jth feature of $\mathbf{f}$, $f_j(i)$ is the jth feature of the feature set corresponding to frame i of $\mathbf{M}_e$ and δ is the Kronecker delta function. $V^0_{e,\mathbf{f}}$ contains all the information about how suitable the motion corresponding to the edge e is for the feature set $\mathbf{f}$. Unfortunately this is still not enough to choose an optimal path. A particular node may only have outgoing

edges with low or zero values for $V^0_{e,\mathbf{f}}$ and so not have much information about which one is the best to choose. In this case the best strategy would be to choose the edge that leads to a portion of the graph with many high valued edges. Another problem is that an edge might have a very high value but lead to a portion of the graph which only contains edges with very low values. The problem with using only $V^0_{e,\mathbf{f}}$ to choose a path is that it is a very greedy strategy, not taking into account long term factors, i.e. the values of edges that can be accessed after taking a particular edge. This could be solved by performing some form of heuristic search to find an optimal path in terms of $V^0_{e,\mathbf{f}}$. However, this approach is not very suitable for real time motion generation. Firstly, it is a slow process relative to greedily choosing the best edge. More important, much of the effort of finding a path may be wasted, as the input features may change rapidly. The path would need to be re-planned each time the input features change, and the time taken to plan the original path would be wasted. Rather than planning a path, our approach is to embed information about future edges into a new score $V^*_{e,\mathbf{f}}$ and use a greedy search based on $V^*_{e,\mathbf{f}}$.

In order to find a value for $V^*_{e,\mathbf{f}}$ that uses both the immediate value of an edge e and the values of future edges that are accessible after choosing e, we use a technique that is commonly used in Reinforcement Learning[34], finding the solutions to the Bellman Equations for the graph. The Bellman Equation expresses the value of an edge as the expected value of taking that edge, and subsequent edges that follow form it. It relates the value of choosing a particular edge with the values of outgoing edges of the end node of that edge (we call this end node n_e). The equation we use is the following:

$$V^*_{e,\mathbf{f}} = V^0_{e,\mathbf{f}} + \gamma \sum_{\mathbf{f}'} \sum_{e' \in \mathrm{ch}(n_e)} P_{\mathbf{f}\to\mathbf{f}'} P(e')(V^*_{e',\mathbf{f}'})$$

The value, $V^*_{e,\mathbf{f}}$ of edge e is the immediate value of the edge $V^0_{e,\mathbf{f}}$ as described above, plus a falloff value γ times the expected value of the next edge to be chosen after e. $\mathrm{ch}(n_e)$ are the children of node n_e. $P(e')$ is the probability of choosing edge e' and $P_{\mathbf{f}\to\mathbf{f}'}$ is the probability that the input features change from value $\mathbf{f}$ to $\mathbf{f}'$ in one time step (where $\mathbf{f}'$ ranges over all possible discretized input feature values). $P_{\mathbf{f}\to\mathbf{f}'}$ can easily be estimated from the transition frequencies in the training data. The recursion in this equation ensures that values of possible future paths are taken account of. The fall-off parameter γ ensures that the values of future edges are weighted lower than current edges. This is important as feature values may change before a future edge is reached. In our experiments we found that a value of 0.2 for γ worked well.

As described above, only the maximum valued edge and the continuation edge can be chosen, so this equation becomes:

$$V^*_{e,\mathbf{f}} = V^0_{e,\mathbf{f}} + \gamma \sum_{\mathbf{f}'} P_{\mathbf{f}\to\mathbf{f}'} \langle V^* \rangle \tag{4}$$

Where $\langle V^* \rangle$ is the expected future value of V^*:

$$\langle V^* \rangle = P_{\max(n_e)} V^*_{\max(n_e),\mathbf{f}'} + P_{\mathrm{cont}(n_e)} V^*_{\mathrm{cont}(n_e),\mathbf{f}'}$$

Substituting 3 into the above we get the following:

$$\langle V^* \rangle = \frac{\left(\max_{e' \in \mathrm{ch}(n_e)} (p_t V^*_{e',\mathbf{f}'})^2 + (V^*_{\mathrm{cont}(n_e),\mathbf{f}'})^2\right)}{\max_{j \in \mathrm{ch}(i)} (p_t V^*_{e',\mathbf{f}'}) + V^*_{\mathrm{cont}(n_e),\mathbf{f}'}}$$

The value for $V^*_{e,\mathbf{f}}$ that satisfies equation 4 provides an optimal policy for choosing which outgoing edges to take at each node. The equation can be solved iteratively. At each iteration we update $V^*_{e,\mathbf{f}}$ from the current value using the Bellman equation. The value for $V^*_{e,\mathbf{f}}$ at step $k+1$ is calculated in terms of the values of the children of n_e at step k:

$$V^{k+1}_{e,\mathbf{f}} = V^0_{e,\mathbf{f}} + \gamma \sum_{\mathbf{f}'} P_{\mathbf{f} \to \mathbf{f}'} \left\langle V^k \right\rangle$$

Where:

$$\left\langle V^k \right\rangle = \frac{\left(\max_{e' \in \mathrm{ch}(n_e)} (p_t V^k_{e',\mathbf{f}'})^2 + (V^k_{\mathrm{cont}(n_e),\mathbf{f}'})^2\right)}{\max_{j \in \mathrm{ch}(i)} (p_t V^k_{e',\mathbf{f}'}) + V^k_{\mathrm{cont}(n_e),\mathbf{f}'}}$$

By using $V^*_{e,\mathbf{f}}$ we can embed all our prior knowledge of the appropriateness of edges and of the connectivity of the graph into a single value that enables us to use a very simple, greedy, $O(1)$ strategy for choosing motion clips in response to new inputs. The algorithm can be summarized as follows:

1. initialize the current node to a starting value
2. select the child edge $e_{\max}$ that maximises the value $p_t(e)V^*_{e,\mathbf{f}}$
3. choose this edge with the probability given in equation 3, otherwise choose the continuation edge
4. play the motion clip associated with the chosen edge

5. once the motion clip has completed set the current node to the end node of the chosen edge and repeat from 2

3.5 Model Refinement

One potential disadvantage of a data driven approach is that if there are flaws in the final model due to problems with the data, it might be necessary to perform the capture again from scratch. Luckily our reinforcement learning based model makes it possible to correct errors in the control model while still using the original data set. We do this by changing the the base edge values $V^0_{e,\mathbf{f}}$. A user interface is provided to give feedback while watching the generated behavior. If a user sees a character performing a particularly appropriate or inappropriate action in a given context they can press buttons labeled "good", "bad" or "very bad" which result in $V^0_{e,\mathbf{f}}$ being set to 1, 0 or -1. After watching some behavior and giving a number of items of feedback, the value function is recalculated using the updated values of $V^0_{e,\mathbf{f}}$.

4 Results

For our experiments we captured 3 data sets. Each data sets consisted of a number of captures of the same actor responding to a male voice speaking in different styles (though different data sets used different actors). The first data set consisted of feeding back interest or boredom to a speaker who spoke in either a varied (interesting) or monotonous (boring)

voice. This data set consisted of 3 clips each of approximately 60 seconds. The second data set was an exercise class scenario, in which the actor had to move in response to motivational speech. There were 2 clips of a little more than 60 seconds. The speech was either energetic or calming and the actor produced corresponding movements. In the final data set the speaker spoke in either a calm or an angry voice and the actor with either a neutral or submissive posture. In this data sets the clips are shorter (approximately 20 seconds) but we used 5 clips. In all data sets the speaker changed style of speech during all of the audio clips. Unsurprisingly we found that variance in pitch was the best predictor of interest in the voice for the first data set. In the other two data sets the variance of pitch and mean intensity were the best predictors. We trained a model for each data set using, all of the clips in each set (in order to demonstrate the quality of purely learned motion, we did not use our model refinement method in any of the experimental test). We then synthesized new animation using new audio data. Our models were able to generate responses to the audio data in real time (for repeatability we used prerecorded audio, but it would be possible for the characters to respond to live audio). The results are shown in figure 4. For all data sets our models were able to produce similar responses to those in the original dataset, and with appropriate timings. The generated motion is reasonably smooth and realistic.

5 Conclusion and Further Work

This paper has presented a method for creating responsive and expressive listening behavior for interactive characters based on data from an actor's performance. The learnt value function means that characters' responses run comfortably at real-time speeds and our method manages to respond appropriately to its input while maintaining the characteristic style of the actor's performance. Responding to a user is ultimately what makes a character interactive, and we have demonstrated that it is possible to use data driven techniques to learn this type of behavior, even in non-goal driven contexts. One area for future investigation is scaling our method to very complex input feature spaces. Our method currently relies on on tabulating the value function $V^*_{e,\mathbf{f}}$ for all possible combinations of input features. This causes two problems. Firstly, the storage could become large for a large number of inputs, however, the input feature space would have to be very large for these storage requirements to outway the size of the motion data. The other problem is that as the number of inputs rise then the number of possible combinations rises exponentially, and the number of times each combination is seen in the data set is likely to become very small. In this case, to learn effectively, we need to exploit similarities between combinations of input features so that data from all similar combinations can be used together. In Reinforcement Learning this is known as the generalization problem [34]. Of course, our method already generalizes by using a value function based on the sum of similar input features, and so could probably scale well to relatively large numbers of input features. For very large or continuous input

feature spaces we would have to learn the values as a parametric function rather than as a table. Traditionally, in reinforcement learning neural networks have been used [34], though contemporary techniques such as support vector machines may give better results [43]. With these techniques our method could well scale to very complex inputs. In particular we would like to investigate more sophisticated vocal features which would allow for greater across speaker robustness and also make it possible to respond to more complex information in the voice.

This work has many possible applications. Improving social responsiveness of characters would greatly improve the social and emotional depth that can be portrayed in computer games and widen the range of themes that could be dealt with. This is particularly true of multiplayer on-line computer games and virtual worlds in which are major part of the social interaction. As Vihljálmsson and Cassell [44] have noted adding responsive non-verbal behavior to user avatars in an on-line world can greatly improve the quality of social interaction. While we have modeled listener behavior our method could also be used to model speaker behavior based on their own speech signal, thus simulating both sides of a conversation. Finally, it is important to note that our method is not only applicable to social interaction but could be used for any form of interaction we would like to have with animated characters. It could be used for sports simulations, for example, we could model the distinctive way in which a baseball batter swings in response to the different ways a ball can be pitched.

Acknowledgements

This work was supported by the Empathic Avatars project funded by the UK Engineering and Physical Sciences Research Council. We would like to thank the member of the UCL Department of Computer Science Virtual Environments and Computer Graphics research group and the member of the UCL Centre for Computational Statistics and Machine Learning for their help and support. We would also like to thanks the anonymous reviewers for their comments and suggestions that have improved the paper.

References

[1] D. P. Pertaub, M. Slater, and C. Barker. An experiment on public speaking anxiety in response to three different types of virtual audience. *Presence-Teleoperators and Virtual Environments*, 11(1):68–78, 2002.

[2] K. Perlin. Real time responsive animation with personality. *IEEE transactions on visualization and Computer Graphics*, 1(1):5–15, March 1995.

[3] X. Tu and D. Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. In *ACM SIGGRAPH*, pages 43–49, 1994.

[4] M. Neff and E. Fiume. Aer: Aesthetic exploration and refinement for expressive character animation. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2005.

[5] S. Kopp and I. Wachsmuth. Synthesizing multimodal utterances for conversational agents. *The Journal Computer Animation and Virtual Worlds*, 15(1):39–52, 2004.

[6] Michael Gleicher. Motion editing with space time constraints. In *symposium on interactive 3D graphics*, pages 139–148, 1997.

[7] Jehee Lee and Sung Yong Shin. A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, pages 39–48, August 1999.

[8] Michael Gleicher. Retargetting motion to new characters. In *ACM SIGGRAPH*, pages 33–42, 1998.

[9] Seyoon Tak and Hyeong-Seok Ko. A physically-based motion retargeting filter. *ACM Trans. Graph.*, 24(1):98–117, 2005.

[10] Okan Arikan and D. A. Forsyth. Interactive motion generation from examples. *ACM Transactions on Graphics*, 21(3):483–490, July 2002.

[11] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. *ACM Transactions on Graphics*, 21(3):473–482, July 2002.

[12] Jehee Lee, Jinxiang Chai, Paul S. A. Reitsma, Jessica K. Hodgins, and Nancy S. Pollard. Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics*, 21(3):491–500, July 2002.

[13] Eugene Hsu, Kari Pulli, and Jovan Popović. Style translation for human motion. *ACM Transactions on Graphics*, 24(3):1082–1089, August 2005.

[14] Matthew Brand and Aaron Hertzmann. Style machines. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 183–192, July 2000.

[15] Charles Rose, Michael F. Cohen, and Bobby Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics & Applications*, 18(5):32–40, September - October 1998.

[16] Tomohiko Mukai and Shigeru Kuriyama. Geostatistical motion interpolation. *ACM Transactions on Graphics*, 24(3):1062–1070, August 2005.

[17] Zoran Popović and Andrew P. Witkin. Physically based motion transformation. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, pages 11–20, August 1999.

[18] C. Karen Liu, Aaron Hertzmann, and Zoran Popović. Learning physics-based motion style with nonlinear inverse optimization. *ACM Transactions on Graphics*, 24(3):1071–1081, August 2005.

[19] Seth Cooper, Aaron Hertzmann, and Zoran Popović. Active learning for real-time motion controllers. *ACM Transactions on Graphics*, 26(3), August 2007.

[20] V. Vinayagamoorthy, M. Gillies, A. Steed, E. Tanguy, X. Pan, C. Loscos, and M. Slater. Building expression into virtual characters. In *Eurographics Conference State of the Art Reports*, 2006.

[21] J. Cassell, T. Bickmore, L. Campbell, K. Chang, H. Vilhjálmsson, and H. Yan. Embodiment in conversational interfaces: Rea. In *ACM SIGCHI*, pages 520–527. ACM Press, 1999.

[22] M. Garau, M. Slater, V. Vinayagamoorhty, A. Brogni, A. Steed, and M.A. Sasse. The impact of avatar realism and eye gaze control on perceived quality of communication in a shared immersive virtual environment. In *Proceedings of the ACM SIG-CHI conference on Human factors in computing systems*, 2003.

[23] Jonathan Gratch and Stacy Marsella. Tears and fears: modeling emotions and emotional behaviors in synthetic agents. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 278–285, New York, NY, USA, 2001. ACM Press.

[24] I. Poggi, C. Pelachaud, F. De Rosis, V. Carofiglio, and B De Carolis. Greta. a believable embodied conversational agent. In Stock O. and Zancanaro M., editors, *Multimodal Intelligent Information Presentation (Text, Speech and Language Technology Vol. 27)*. Kluwer, 2005.

[25] R. M. Maatman, J. Gratch, and S. Marsella. Natural behavior of a listening agent. In T. Panayiotopoulos, J. Gratch, R. Aylett, D. Ballin, P. Olivier, and T. Rist, editors, *Intelligent Virtual Agents, 5th International Working Conference*, Kos, Greece, September 2005.

[26] Y. Nakano, G. Reinstein, T. Stocky, and J. Cassell. Towards a model of face-to-face grounding. In *Annual Meeting of the Association for Computational Linguistics*, 2003.

[27] Sooha Park Lee, Jeremy B. Badler, and Norman I. Badler. Eyes alive. *ACM Transactions on Graphics*, 21(3):637–644, July 2002.

[28] Michael Kipp, Michael Neff, Kerstin H. Kipp, and Irene Albrecht. Towards natural gesture synthesis: Evaluating gesture units in a data-driven approach to gesture synthesis. In *IVA*, pages 15–28, 2007.

[29] A. Egges, T. Molet, and N. Magnenat-Thalmann. Personalised real-time idle motion synthesis. In *12th Pacific Conference on Computer Graphics and Applications*, pages 121–130, October 2004.

[30] Matthew Stone, Doug DeCarlo, Insuk Oh, Christian Rodriguez, Adrian Stere, Alyssa Lees, and Chris Bregler. Speaking with hands: creating animated conversational characters from recordings of human performance. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 506–513, New York, NY, USA, 2004. ACM Press.

[31] Tae hoon Kim, Sang Il Park, and Sung Yong Shin. Rhythmic-motion synthesis based on motion-beat analysis. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 392–401, New York, NY, USA, 2003. ACM.

[32] Takaaki Shiratori, Atsushi Nakazawa, and Katsushi Ikeuchi. Dancing-to-music character animation. *Comput. Graph. Forum*, 25(3):449–458, 2006.

[33] Eugene Hsu, Sommer Gentry, and Jovan Popović. Example-based control of human motion. In *2004 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 69–77, July 2004.

[34] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[35] Bruce Blumberg, Marc Downie, Yuri Ivanov, Matt Berlin, Michael Patrick Johnson, and Bill Tomlinson. Integrated learning for interactive synthetic characters. *ACM Transactions on Graphics*, 21(3):417–426, July 2002.

[36] Jehee Lee and Kang Hoon Lee. Precomputing avatar behavior from human motion data. In *2004 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 79–87, July 2004.

[37] James McCann and Nancy Pollard. Responsive characters from motion fragments. *ACM Transactions on Graphics*, 26(3), August 2007.

[38] Adrien Treuille, Yongjoon Lee, and Zoran Popović. Near-optimal character animation with continuous control. *ACM Trans. Graph.*, 26(3):7, 2007.

[39] Kang Hoon Lee, Myung Geol Choi, and Jehee Lee. Group behavior from video: A data-driven approach to crowd simulation. In *2004 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, July 2007.

[40] A. Kendon. Movement coordination in social interaction. *Acta Psychologica*, 32:1–25, 1970.

[41] P. Boersma. Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound. *Proceeding of the Institute of Phonetic Sciences*, 17:97–110, 1993.

[42] P. Boersma. Praat, a system for doing phonetics by computer. *Glot International*, 5(9/10):341–345, 2001.

[43] N. Christianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[44] H. H. Vilhjálmsson and J. Cassell. Bodychat: Autonomous communicative behaviors in avatars. In *second ACM international conference on autonomous agents*, 1998.

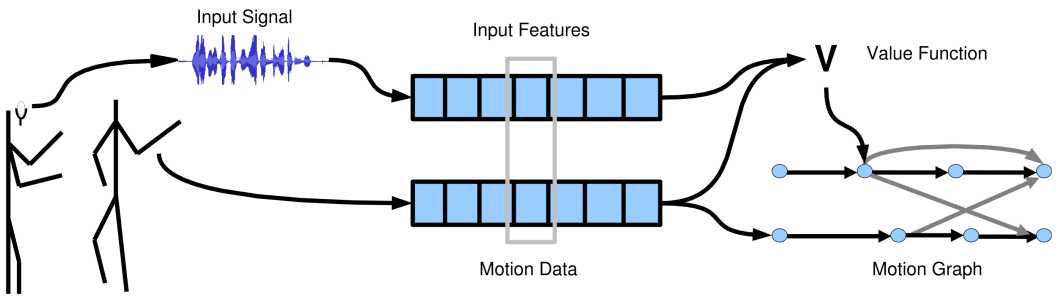

Figure 1: The Capture Process. (BW)

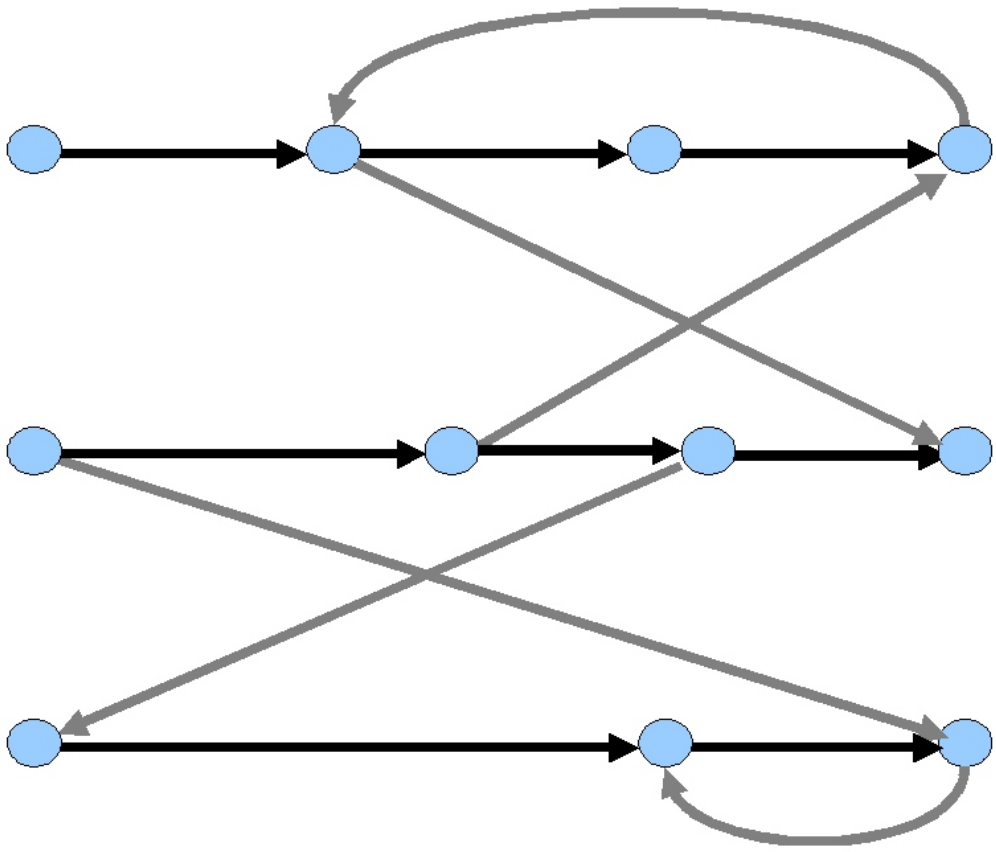

Figure 2: Motion graphs. Motion graphs consists of a number of motion sequences (shown horizontally from left to right) and transitions between them. Continuation nodes are shown in black and transition nodes in grey. (BW)

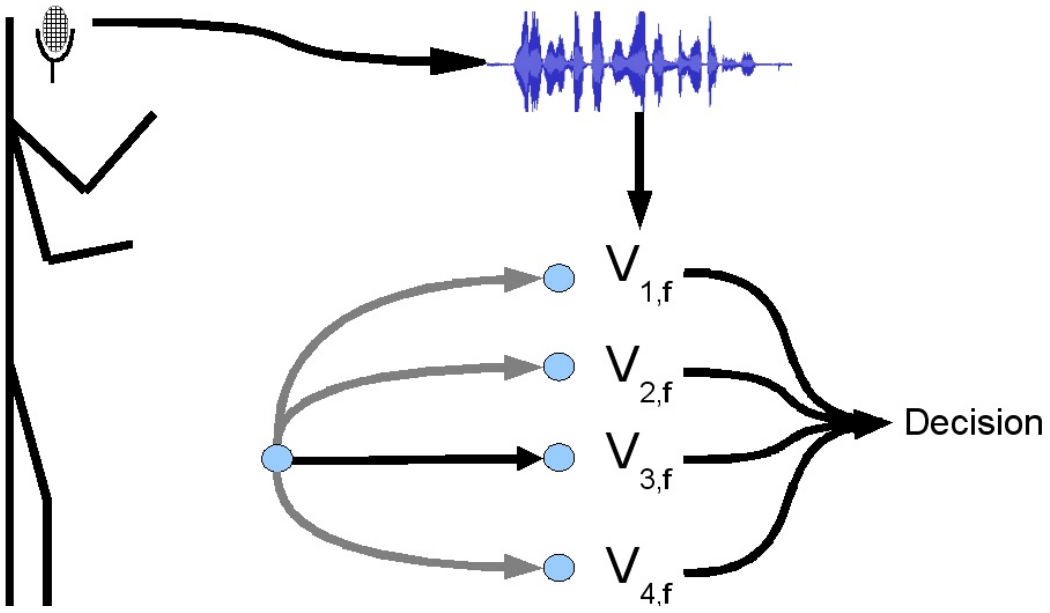

Figure 3: The Generation Process: at each node in the graph there are several possible options for the next edge to take. The input signals from the user are used to evaluate a value function for each edge, which is then used to choose which edge to take. (BW)

Figure 4: Behavior generated by our models. First row: the interest model, responding to interesting (left) and boring (right) speech. Second row: the exercise class model, responding to an energizing (left) and calming (right) voice. Bottom row: the angry voice, responding to a neutral (left) and angry (right) voice. (COLOR)