Real-Time Motion Control Using Pose Space Probability Density Estimation

Dumebi Okwechime, Eng-Jon Ong, and Richard Bowden
CVSSP, University of Surrey, Guildford, Surrey, GU17XH, UK
{d.okwechime,e.ong,r.bowden}@surrey.ac.uk

Abstract

The ability to control the movements of an object or person in a video sequence has applications in the movie and animation industries, and in HCI. In this paper, we introduce a new algorithm for real-time motion control and demonstrate its application to pre-recorded video clips and HCI. Firstly, a dataset of video frames are projected into a lower dimension space. A k -mediod clustering algorithm with a distance metric is used to determine groups of similar frames which operate as cut points, segmenting the data into smaller subsequences. A multivariate probability distribution is learnt and probability density estimation is used to determine transitions between the subsequences to develop novel motion. To facilitate real-time control, conditional probabilities are used to derive motion given user commands. The motion controller is extended to HCI using speech Mel-Frequency Cepstral Coefficients (MFCCs) to trigger movement from an input speech signal. We demonstrate the flexibility of the model by presenting results ranging from datasets composed of both vectorised images and 2D point representation. Results show plausible motion generation and lifelike blends between different types of movement.

1. Introduction

Video synthesis has been an emerging topic in recent years, highly applicable to the movie and games industries. When filming a movie, certain elements in a video scene such as the movement of trees blowing in the wind, do not perform on cue. It may not always be cost effective, safe or even possible to control the surroundings to match the directors intentions. In this paper, an approach to controlling pre-recorded motions in a video clip is presented giving the user real-time control in creating novel video sequences. This is achieved by modelling the motion in the video sequence as a pose space probability density function (PDF). The model can synthesize novel sequences whilst retaining the natural variances inherent to the original data. By learning the mapping between motion subspaces and external stimu-

Figure 1. Images showing candle and plasma beam videos.

Figure 2. Images showing tracked face data used in modelling conversational cues.

lus, the user can drive the motion at an intuitive level.

As an extension to this work, the system is also demonstrated in modelling human conversational cues in a social context. During a one to one conversation, listeners tend to infer their engagement by nodding or performing some sort of non-verbal behaviour. These responses are in most cases encouraged by the *speaker's* tone. When a speaker attempts to make a point, they put emphasis on certain words which is a cue for the *listener* to acknowledge with some visual gesture. Here, having learnt their relationship, we take visual and audio features captured in a social context and synthesize appropriate, novel responses to audio stimulus.

Our *Motion Control Model* has four stages. First, given a dataset of video frames, the data is projection to a lower dimension pose space using Principal Component Analysis (PCA). Next, the space is segmented into cut points using

an unsupervised k-mediod clustering algorithm. These cut points represent points in the video sequence where a transition to a different subsequence is possible. We model the data in pose space using a PDF where motion is generated as the likelihood of a transition given a cut point. Finally, a real-time motion control system is developed mapping a user interface to the desired movement. Figure 1 shows the candle flame and plasma beam sequences used for video textures. The purpose of the *Motion Control Model* is to control the direction which the flame and beam move in real-time whilst generating a novel animation with plausible transitions between different types of movement. Figure 2 shows a 2D tracked contour of a face generated from a video sequence of a person listening to a speaker. Mapping the audio MFCC features of the speaker to the 2D face, we generate appropriate non-verbal responses triggered by novel audio input.

The paper is divided into the following sections. Section 2, briefly details related works in the field of video data modelling and motion synthesis. Section 3 presents an overview of the *Motion Control System*. Section 4 details the technique used for learning the model. Section 5 presents the real-time motion control system and the remainder of the paper describes the results and conclusions are drawn.

2. Related Work

There are several techniques to synthesizing video. Early approaches were based on parametric [6] [18] and non-parametric [4] [21] texture synthesis methods which create novel textures from example inputs. Kwatra *et al.* [9] generate perceptually similar patterns from smaller training data using a graph cut technique based on Markov Random Fields (MRF). Bhat *et al.* [2] presents an alternative approach to synthesize and edit videos by capturing dynamics and texture variation using texture particles traveling along user defined flow lines. Schödl *et al.* [16] introduces *Video Textures* which computes the distances between frames to derive appropriate transition points to generate a continuous stream of video images from a small amount of training video. The limitation to this approach is, to determine transition points, frame-to-frame distances are calculated for all frames which can be computationally expensive for large data sets. Also, their dynamic programming algorithm only allows for backward transitions limiting the scope of potential transitions.

In this work, we look at video synthesis as a continuous motion estimation problem, inspired by techniques used in modelling motion captured data. Troje [20] was one of the first to use a simple sine functions to model motion. Pullen and Bregler [14] used a kernel based probability distribution to extract 'motion texture' i.e. the personality and realism, from the motion capture data to synthesize novel motion with the same style and realism of the original data. Ok-

wechime and Bowden [12] extended this work using a multivariate probability distribution to blend between different types of cyclic motion to create novel movement. Arikan *et al.* [1] allow a user to synthesize motion by creating a timeline with annotated instructions such as walk, run or jump. For this approach to be successful, a relatively large database is needed to work with their optimisation technique otherwise it may result in highly repetitive motion generation.

Treuille *et al.* [19] developed a system that synthesizes kinematic controllers which blend subsequences of pre-captured motion clips to achieve a desired animation in real-time. The limitation to this approach is it requires manual segmentation of motion subsequences to a rigid design in order to define appropriate transition points. Our motion control model uses an unsupervised k-mediod algorithm for sequence segmentation and extends the application of motion modelling to generate novel motion sequences in video.

In Human Computer Interaction (HCI), using audio features to drive a model can be a useful tool in developing effective user interfaces. Brand [3] learns a facial control model driven by audio speech signals to produce realistic facial movement including lip-syncing and upper-face expressions. Stone *et al.* [17] uses context dependent scripts to derive an animation's body language including hand gestures. In these examples, audio signals are used to animate a speaker but little work has looked at using audio signals to animate a listener in a conversational context in real-time. Jebara and Pentland [7] touched on modelling conversational cues and proposed Action Reaction Learning (ARL), a system that generates an animation of appropriate hand and head poses in response to a users hand and head movement in real-time. However, this does not incorporate audio. In this paper, we demonstrate the flexibility of the motion controller, modelling conversation cues based on audio input. Audio features of the speaker, represented by MFCCs, are used to derive appropriate visual responses of the listener in real-time from a 2D face contour modelled with the motion controller. This results in plausible head movement and facial expressions in response to the speaker's audio.

3. System Overview

The purpose of the *Motion Control System* is to learn a model of a single agent undergoing motion. The model will allow a user to reproduce motion in a novel way by specifying in real-time which type of motion to perform inherent in the original sequence. This is achieved by learning a *pose space PDF*.

We start by creating a first order dynamic representation of the motion in pose space. Then, using conditional probabilities we are able to dynamically change likelihoods of the pose to give the user control over the final animation. This is demonstrated in three examples:

Figure 3. (A) Plot of eigen projection of first 2 dimensions of a candle video sequence, (B) Image showing k-mediod points numbered, (C) Image showing cut point groups in eigenspace, (D) Image showing transitions between cut points.

- **Candle Flame.** We synthesize the movement of a candle flame where the user has control over three discrete states: *ambient flame*, *flame blow left*, *flame blow right*, and can blend between them.
- **Plasma Beam.** The user controls the movement of a plasma beam using conditional probability to provide a continuous user interface.
- **Tracked 2D Face Contour.** An animation of a 2D face is driven directly from an audio signal, displaying appropriate visual responses for an avid listener based on audio stimulus.

4. Motion Control Model

There are several stages involved in developing the *Motion Control Model*. Given a data set of an agent undergoing motion, the frames are vectorised and dimensionally reduced for efficiency. An unsupervised clustering algorithm is then used to segment the data into smaller segments. This leads to the computation of the *pose space PDF* on the data on which the motion model is derived. A kd-tree is used for rapid density estimation and dynamic programming is used to estimate novel transitions between segments in the data, based upon the influence of a user in terms of probability of desired appearance. Conditional Probabilities map from the input space to that of synthesis, providing the user with control over the final animation.

4.1. Data

Given the motion sequence $\mathbf{X}$, each frame is represented as a vector $\mathbf{x}_i$ where $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_F\}$ and F is the number of frames. Various motion data formats can be used in

this system. So far, both 2D tracked points and rgb pixels have been successfully implemented. In both cases, each time step i of the data to be modelled is vectorised as $\mathbf{x}_i = (x_{i1}, y_{i1}, \dots, x_{in}, y_{in}) \in \mathbb{R}^{2n}$ for a 2D contour of n points or $\mathbf{x}_i = (r_{11}, g_{11}, b_{11}, \dots, r_{xy}, g_{xy}, b_{xy}) \in \mathbb{R}^{xy}$ for an $x \times y$ image.

4.2. Eigenspace Representation

To reduce the complexity of building a generative model of motion, Principal Component Analysis (PCA) [15] is used for dimensionality reduction.

For a given D -dimension data set $\mathbf{X}$, as defined in Section 4.1, the D principal axes $\mathbf{T}_1, \mathbf{T}_2, \dots, \mathbf{T}_D$ are given by the D leading eigenvectors of the sample covariance matrix $\mathbf{S}$. An eigen decomposition gives $\mathbf{S} = \sum \lambda_i \mathbf{T}_i$, $i \in \{1, \dots, D\}$, where λ_i is the i th largest eigenvalue of $\mathbf{S}$.

The dimension of the feature space is reduced by projecting into the eigenspace

$$\mathbf{y}_i = \mathbf{V}^T(\mathbf{x}_i - \boldsymbol{\mu}) \quad (1)$$

where $\boldsymbol{\mu}$ is the sample mean $\boldsymbol{\mu} = \frac{1}{F} \sum_{i=1}^F \mathbf{x}_i$, $\mathbf{V}$ are the eigenvectors $\mathbf{V} = [\mathbf{T}_1, \dots, \mathbf{T}_d]$ and d is the chosen lower dimension $d \leq D$ such that $\sum_{i=1}^d \frac{\lambda_i}{\sum \lambda_i} \geq .95$ or 95% of the energy is retained. $\mathbf{Y}$ is defined as a set of all points in the dimensionally reduced data where $\mathbf{Y} = \{\mathbf{y}_1, \dots, \mathbf{y}_F\}$ and $\mathbf{y}_i \in \mathbb{R}^d$. This results in a d -dimension representation of each frame in the sequence. This representation reduces the computational and storage complexity of the data whilst still retaining the time varying relationships between each frame. Figure 3 (A) shows a plot of the candle data set projected onto the first two eigenvectors. It produces a non-linear but continuous sub-space characteristic of continuous motion.

4.3. Unsupervised Segmentation

As explained in the introduction to Section 4, this lower dimension data is segmented into several short subsequences. A single subsequence is represented as a set of consecutive frames between a start and end transition point. The idea is to connect various subsequences together to create a novel sequence.

Figure 4 shows an example of the process of unsupervised segmentation. To cut the data into smaller segments and derive subsequences with appropriate start and end transition points, various points of intersection in the motion trajectory, which we refer to as *cut points*, need to be determined. Given a motion sample, shown by the two dimension motion trajectory in Figure 4 (A), a k-mediod clustering algorithm is used to find N_c k-mediod points. K-mediod points are data points derived as cluster centre in regions of high density. In Figure 4 (B), $N_c = 3$ and are shown as the three red crosses which we refer to as a , b , and c . Using a

Figure 4. (A) Trajectory of the original motion sequence. Arrows indicate the direction of motion. (B) $N_c = 3$ k-mediod points derived using the unsupervised k-mediod clustering algorithm. The three red crosses are the three k-mediod points a, b, and c. (C) The small green dots are the cut points derived as the nearest neighbouring points to a k-mediod point less than a user defined threshold θ . The three gray circles represent cut point groups a, b and c. (D) Cut points act as start and end transition points segmenting the data into shorter segments. The orange dots are start transition points and the purple dots are end transition points. (E) Diagram of possible transitions within group c. For simplicity only a few transitions are displayed.

distance matrix, the nearest neighbouring points to each k-mediod points are derived and make up groups of cut points. The cut points are represented by the small green dots in 4 (C), and the groups of cut points are represented by the gray circles. Cut point groups consists of discrete frames which are not directly linked, however smooth transitions can be made between them. Similar to Motion Graphs [8], an approach used in character animation, assuming frames close to each in eigenspace are similar, a simple blending techniques such as linear interpolation can reliably generate a transition. As shown in Figure 4 (D) the cut point are used to segment the data to smaller subsequences with start and end transition points. 4 (E) shows a few of the possible transitions between various subsequences in group c.

Firstly, we define a database distance matrix $\mathbf{D}$ to determine the similarity between all frames in the video sequence.

$$\mathbf{D}_{(i,j)} = d(\mathbf{y}_i, \mathbf{y}_j) | i = \{1, \dots, F\}, j = \{1, \dots, F\} \quad (2)$$

The distance between the frames can be calculated by computing the L2 distance between their corresponding points in eigenspace.

The next step is to cluster the data to estimate suitable cut points using an unsupervised k-mediod clustering algorithm. The k-mediod clustering algorithm works similarly to the k-means clustering algorithm in that it associates to each cluster the set of images in the database that closely resembles the cluster centre. The main difference is that the cluster centre is represented as a data point median as opposed to a mean in the eigenspace. This ensures that no blurring occurs and all points are embedded within the data.

We define N_c k-mediod cluster centres as μ_i^c given by the k-mediod method whereby $\mu_i^c \in \mathbf{Y}$ and $0 < i < N_c$. The set containing the members of the n^{th} cluster is defined as $\mathbf{Y}_n^c = \{\mathbf{y}_{n,1}^c, \dots, \mathbf{y}_{n,c_n}^c\}$, where the number of members

of the n^{th} cluster is denoted as C_n .

Figure 3 (B) shows a set of $N_c = 36$ k-mediod points used for the candle video synthesis. N_c is chosen based on the number of clusters that best defines the distribution of poses in pose space. If N_c is too high, the model will generate unrealistic motion and if N_c is too low, not enough cut points will be available to provide transitions to varying motion types reducing the novelty of animation.

Cut points are data points where the distance to μ_i^c is less than threshold θ . The threshold is determined experimentally whereby if it is set to high, it becomes more challenging to produce plausible blends when making transitions, and if its too low, potential cut points are ignored and we are limited to points that overlap, which is an unlikely occurrence in a multi-dimensional space.

As shown in Figure 3 (C), we are able to represent the data as cut point groups where transitions between subsequences are generated by branching cut points, as demonstrated in Figure 3 (D). Cut point groups represent groups of frames where, via simple linear interpolation, a transition to a different subsequence can occur. Taking $\mathbf{Y}_i^c = \{\mathbf{y}_{i,1}^c, \mathbf{y}_{i,2}^c, \mathbf{y}_{i,3}^c\}$ as a set of neighbouring cut points as illustrated in Figure 5. If a transition is made to $\mathbf{y}_{i,1}^c$, since $\mathbf{y}_{i,1}^c$ is in the same group as $\mathbf{y}_{i,2}^c$ and $\mathbf{y}_{i,3}^c$, they are consider close enough in pose space to create a visually plausible blend. If the next desired subsequence is a transition from $\mathbf{y}_{i,3}^c$ to $\mathbf{y}_{a,\alpha}^c$, the model blends from pose $\mathbf{y}_{i,1}^c$ to $\mathbf{y}_{i,3}^c$ using linear interpolation, then the transition is made to $\mathbf{y}_{a,\alpha}^c$.

4.4. Appearance Model

4.4.1 Pose Space PDF

A PDF of appearance is created using kernel estimation where each kernel $p(\mathbf{y}_i)$ is effectively a Gaussian centred on a data example $p(\mathbf{y}_i) = G(\mathbf{y}_i, \Sigma)$. Since we want our

Figure 5. Image showing transitions between cut points in different clusters where α is any cut point member in its respective group.

probability distribution to represent the dimensionally reduced data set $\mathbf{Y}$ of d dimensions as noted in Section 4.2, the likelihood of a pose in pose space is modelled as a mixture of Gaussians using multivariate normal distributions. We will refer to this Gaussian mixture model as the *pose space PDF*.

$$P(\mathbf{y}) = \frac{1}{F} \sum_{i=1}^F p(\mathbf{y}_i) \quad (3)$$

where the covariance of the Gaussian is:

$$\Sigma = \alpha \begin{pmatrix} \sqrt{\lambda_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sqrt{\lambda_d} \end{pmatrix} \quad (4)$$

For these experiments $\alpha = 0.5$.

4.4.2 Fast Gaussian Approximation

As can be seen from Equation 3, the computation required for the probability density estimation is high since it requires an exhaustive search through the entire set of data examples. This causes slow computation for real time implementation. As a result, a fast approximation method based on kd-trees [11] is used to reduce the estimation time without sacrificing accuracy.

Instead of computing kernel estimations based on all data points, with the kd-tree we can localise our query to neighbouring kernels, assuming the kernel estimations outside a local region contribute nominally to the local density estimation. We are now able to specify N_n nearest neighbours to represent the model, where $N_n < N_T$. This significantly reduces the amount of computation required. Equation 3 is simplified to:

$$P(\mathbf{y}) = \frac{1}{N_n} \sum_{i=1}^{N_n} p(\mathbf{y}_i) \quad (5)$$

where $\mathbf{y}_i$ are the N_n nearest neighbouring kernels found efficiently with the kd-tree, and $\mathbf{y}_i \in \mathbf{Y}$.

4.4.3 Markov Transition Matrix

When generating novel motion sequences, we are not only interested in generating the most likely pose/frame but also the mostly probable path leading to it. A first order Markov Transition Matrix [5] is used to discourage movements that are not inherent in the training data. As an approach formally used with time-homogeneous Markov chains to define transition between states, by treating our clusters of cut points $\mathbf{Y}_i^c$ as states, this approach can be used to apply further constraints and increase the accuracy of the transition between sequences.

We define $\mathbf{P}_{k,l} = \{p_{kl}\}$ as the transition matrix whereby $\mathbf{P}_{k,l}$ denotes the fixed probability of going from cluster k to cluster l , and $\sum_l \mathbf{P}_{k,l} = 1$. We are now able to represent the conditional probability of moving from one cluster to another as $P(C_t|C_{t-1}) = \mathbf{P}_{t-1,t}$ where C_t is defined as the index for a cluster/state at time t . This transition matrix is constructed using the cut points within the sequence to identify the start and end transitions within the data.

The transition matrix probability values act as weighting variables, giving higher influence to transitions that occur more frequently in the original data. To account for situations where the only possible transition is highly improbable, a uniform normalisation value $v = 0.5$ is added to all elements in the transition matrix before normalisation. This allows the transition model to move between states not represented as transitions in the original sequence.

4.4.4 Generating Novel Video Sequences

To generate novel motion sequences the procedure is:

1. Given the current position in pose space $\mathbf{y}_t$, find all adjacent cut points and associate transition sequences. The end points of these sequences gives a set of M possible modes in pose space $\{\mathbf{y}_{t+1}^1, \dots, \mathbf{y}_{t+1}^M\}$
2. Derive cluster index $C_{i,t}$ for $\mathbf{y}_t$ (where $\mathbf{y}_t \in \mathbf{Y}_{i,t}^c$), and its corresponding future cluster indices $(C_{i,t+1}^m) \forall_m$ where $\mathbf{y}_{t+1}^m \in \mathbf{Y}_{i,t+1}^c$.
3. Calculate the likelihood of each mode as:

$$\phi_m = P(C_{i,t+1}^m | C_{i,t}) P(\mathbf{y}_{t+1}^m) \quad (6)$$

where $\Phi = \{\phi_1, \dots, \phi_M\}$.

4. Randomly select a sequence from Φ based upon its likelihood.

5. Using Linear interpolation, blend $\mathbf{y}_t$ to the new sequence start.
6. The frames associated to transition sequence are reconstructed for rendering as $\mathbf{x}_t = (\mu + \mathbf{V}\mathbf{y}_t)$.
7. The process then repeats from step (1).

4.4.5 Dynamic Programming

Similar to a first order markov chain, our goals are currently in terms of the next step in the transition. In most cases, motion requires sacrificing short term objectives for the longer term goal of producing a smooth and realistic sequence. As a result, dynamic programming is used for forward planning. Formally used in Hidden Markov Models to determine the most likely sequence of hidden states, it is applied to the *pose space PDF* to observe the likelihoods for $m = 3$ steps in the future.

Treating our clusters as states, a trellis is built m steps in the future effectively predicting all possible transitions m levels ahead. Dynamic programming is then used to find the most probable path for animation.

5. Real-Time Motion Control System

Thus far, the *Motion Control Model* randomly generates the mostly likely set of motion sequences given a starting configuration. To allow real-time control of the motions we use a Projection Mapping method.

5.1. Projection Mapping

The Projection Mapping approach is used when wanting to enable motion control using an interface such as a mouse cursor, touch screen monitor or even on audio features.

The input space is quantised into an appropriate number of symbols. Using the direct mapping within the training data of input symbols and the clusters of the pose space, a condition probability is calculated that maps from the input space to pose space $P(C|input)$. This is used at run-time to weight the pose space PDF in the motion controller by altering Equation 5 to:

$$P(\mathbf{y}) = \frac{1}{N_n} \sum_{i=1}^{N_n} p(\mathbf{y}_i) \omega \quad (7)$$

where

$$\omega = \begin{cases} P(C_i|input) & \text{if } \mathbf{y}_i \in \mathbf{Y}_i^c \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

6. Experimental Results

To produce our results we performed tests on three different data sets. Two data sets consist of video sequences, and the other consists of a video and audio recording of two people in a conversation with each other.

6.1. Experiments with Video Data

Two video sequences were recorded using a webcam of a single agent performing various types of motion. One was of a candle flame and the other of plasma beams from a plasma ball. The candle flame sequence was captured using a webcam (185×140 pixels, 15 frames per second). The recording was 3:20 minutes long containing 3000 frames. Similarly the plasma beam sequence was captured with a webcam (180×180 pixels, 15 frames per second). The recording was 3:19 minutes long containing 2985 frames. The candle flame sequence consists of the candle performing 3 different motions, blowing left, blowing right and burning in a stationary position. The plasma beam sequence however has more varying movement, from multiple random plasma beams too a concentrated beam from a point of contact around the edges of the ball.

The *Motion Control Model* is able to produce novel sequence at real-time intervention and generate plausible video sequences with the same authenticity as the original data. As shown in Figure 6, the user can control the three discrete states of the candle flame motion. If the animation is at a blow right state, it has to travel to the stationary state before a blow left state can be reached expressed by the transition matrix and determined through dynamic programming. The modelled plasma beam offers more varying degrees of control. Dividing the different state of the plasma beam motion round the edges of the plasma ball into ten discrete states, using a mouse cursor or touch screen, the user can control the movement of the plasma beam around the edges of the plasma ball. As shown in Figure 7 (A), unlike the candle flame, the plasma beam can always return to a random beam state, however, when a concentrated beam is in motion, it can only move in a continuous manner inherent in the original data.

6.2. Experiments with Conversation Data

Two people conversing with each other were recorded using two SD (Standard Definition) cameras (720×576 pixels, 25 frames per second) and a microphone (48kHz). They sat face to face at a comfortable distance apart. The frontal view of each face was captured whilst they conversed for 12 minutes. One of the subjects was male and the other female. They spoke in fluent english and considered themselves friends.

The data was analysed and one of the subjects was chosen to be the expressive *listener* whilst the other the *speaker*. The conversational sequence was segmented to periods when the *listener* is clearly engaged in listening to the *speaker* with no co-occurring speech. This produced 10 audio-visual segments with a total of 2:30 minutes duration.

The facial features of the listener including head poses were tracked using a Linear Predictor tracker [13]. 44 2D

Figure 6. Image showing candle flame synthesis

Figure 7. (A) Image showing plasma beam synthesis. Arrows indicate direction of sythesis flow. (B) Image showing synthesis of a nod as a response to audio stimulus

points were used to cover the contour of the face including the eye pupils. As shown in Figure 7 (B), the movements of these 2D points are dynamically generated from the motion controller in real-time. The audio stimulus uses the conditional probability to derive various visual responses based on its content. The most prominent visual responses are head nods, though other expressions like smiles, eye brow lifts, blinks are generated when appropriate.

The audio stream is represented using 12 MFCCs and a single energy feature of the standard HTK setup, a configuration commonly used in speech analysis and recognition [10]. The frame rate of 100 frames per second (fps) was selected with 50% overlap, i.e., the window size is 20ms and the step size 10ms. A k-means algorithm is used to cluster the MFCC into 25 classes. The number of classes are chosen experimentally to represent an even distribution of the MFCCs. The classes are projected with a conditional probability, as explained in Section 5.1, to map the audio features to visual motion sequence based on the original data.

Person	Audio-Model	Random Play	$\frac{\text{Model}}{\text{Random}}$
1	0.66	0.32	2.1
2	0.56	0.43	1.3
3	0.86	0.11	7.8
4	0.45	0.56	0.8
5	0.45	0.56	0.8
6	0.9	0.1	9
7	0.27	0.7	0.4
8	0.78	0.18	4.3
9	0.55	0.43	1.3
10	0.78	0.2	3.9
11	0.65	0.33	2
12	0.7	0.3	2.3
13	0.5	0.5	1
14	0.57	0.42	1.3
Aver.	0.62	0.37	2.7
Std. Dev.	0.18	0.18	2.66

Table 1. Scores for visual responses to audio. Column 1 is the numerical index of people giving scores. Column 2 and 3 is the normalised and averaged scores for visual responses that are audio driven and randomly playing respectively. Column 4 is the audio driven scores divided by the random play scores

Person	Replay	Random Play	$\frac{\text{Replay}}{\text{Random}}$
Aver.	0.57	0.45	3.5
Std. Dev.	0.3	0.33	3.8

Table 2. Average and standard deviation of scores for visual responses to audio based on replay of original data and random play

For testing, another set of audio sequences were captured from the same speaker having a casual conversation with someone else. 15 segments were captured totaling 2:31 minutes. When combined with the visual model of the listener, we were able to generate a synthetic listener with plausible visual responses to the audio segments.

To validate results, 14 people were asked to listen to the 15 test audio segments and to score between 1 and 10 how well the visual model responded to the audio as a synthetic listener in the conversation. They were unaware that approximately half of the visual responses to the audio segments were playing randomly regardless of the audio input whilst the other half were generated from the audio input to the model. The results are listed in Table 1. We normalised

each person's score and took the average for both audio-model generation and random play. As shown in the fourth column of Table 1 entitled '*Model Random*', 11 out of 14 generated a score greater than or equal to 1, showing preference to the visual responses generated by the audio input than by the random play. Although the majority could tell the difference, the margins of success are not considerably high producing an average of 0.62. Several assumptions may be drawn from this. As nods are the most effective non-verbal responses of an engaged listener, random nods can be an acceptable response to a *speaker*. To try to validate these tests, the same 14 people were asked to repeat the test but this time on the 10 audio segments used in training the model. 5 out of 10 of the audio segments were randomly playing visual response and the other 5 were replays of the original audio-visual pairing. Results in Table 2 show that most people prefer the replay to the random play visual responses but with a similar margin of success when compared to our model generation. The test with the real data also produced a higher standard deviation than the test with our model. It would appear that whilst some people can tell the difference confidently, some can not tell the difference at all.

7. Conclusion

The *Motion Control Model* can generate novel motion sequences in real-time with the same realism inherent in the original data. It has been shown that motion modelling approaches traditionally used for motion capture data can be applied to video. The model can work with data sets containing both 2D points and rgb pixels, applying principals of novel motion sequence generation to different data formats. We have demonstrated that it is also possible to learn a conversational cue model using the motion controller to derive appropriate responses using audio features. This approach can be extended to model other aspect of non-verbal communication such as hand gestures, though further study is needed in understanding different aspects of what a persons is attempting to express when using their hands.

8. Acknowledgements

This work is supported by the FP7 project DICTASIGN and the EPSRC project LILiR.

References

- [1] O. Arikan, D. Forsyth, and J. O'Brien. Motion synthesis from annotation. In *ACM Trans. on Graphics*, 22, 3, July, (SIGGRAPH 2003), pages 402–408.
- [2] K. Bhat, S. Seitz, J. Hodgins, and P. Khosla. Flow-based video synthesis and editing. In *ACM Trans. on Graphics*, SIGGRAPH 04.
- [3] M. Brand. Voice puppetry. In *Proc. of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH 1999, pages 21–28. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA, 1999.
- [4] A. Efros and T. Leung. Texture synthesis by non-parametric sampling. In *Int. Conf. on Computer Vision*, pages 1033–1038, 1999.
- [5] W. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.
- [6] D. Heeger and J. Bergen. Pyramid-based texture analysis. In *Proc. of SIGGRAPH 95, August*, pages 229–238, Los Angeles, California.
- [7] T. Jebara and A. Pentland. Action reaction learning: Analysis and synthesis of human behaviour. In *Workshop on the Interpretation of Visual Motion - Computer Vision and Pattern Recognition Conference*, 1998.
- [8] L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. In *Proc. of ACM SIGGRAPH*, 21, 3, Jul, pages 473–482, 2002.
- [9] V. Kwatra, A. Schodl, I. Essa, G. Turk, and A. Bobick. Graphcut textures. In *ACM Transactions on Graphics*, SIGGRAPH 2003, 22, 3, pages 277–286, 2003.
- [10] A. Mertins and J. Rademacher. Frequency-warping invariant features for automatic speech recognition. In *2006 IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2006. ICASSP 2006 Proceedings, volume 5, 2006.
- [11] A. Moore. A tutorial on kd-trees. Extract from PhD Thesis, 1991. Available from <http://www.cs.cmu.edu/simawm/papers.html>.
- [12] D. Okwechime and R. Bowden. A generative model for motion synthesis and blending using probability density estimation. In *Fifth Conf. on Articulated Motion and Deformable Objects*, 9-11 July, Mallorca, Spain, 2008.
- [13] E.-J. Ong and R. Bowden. Robust lip-tracking using rigid flocks of selected linear predictors. In *8th IEEE Int. Conf. on Automatic Face and Gesture Recognition*, Amsterdam, The Netherlands, 2008.
- [14] K. Pullen and C. Bregler. Synthesis of cyclic motions with texture, 2002.
- [15] E. Sahouria and A. Zakhov. Content analysis of video using principal components. In *IEEE Trans. on Circuits and Systems for Video Technology*, volume 9, 1999.
- [16] A. Schödl, R. Szeliski, D. Salesin, and I. Essa. Video textures. In *Proc. of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH 2000, pages 489–498. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA.
- [17] M. Stone, D. DeCarlo, I. Oh, C. Rodríguez, A. Stere, A. Lees, and C. Bregler. Speaking with hands: Creating animated conversational characters from recordings of human performance. In *ACM Trans. on Graphics (TOG)*, SIGGRAPH 2004, volume 23, pages 506–513. ACM New York, NY, USA, 2004.
- [18] M. Szmurmer and R. Picard. Temporal texture modeling. In *Proc. of IEEE Int. Conf. on Image Processing*, 1996, pages 823–826, 1996.
- [19] A. Treuille, Y. Lee, and Z. Popovic. Near-optimal character animation with continuous control. In *Proc. of SIGGRAPH 07* 26(3).
- [20] N. F. Troje. Decomposing biological motion: A framework for analysis and synthesis of human gait patterns. *J. Vis.*, 2(5):371–387, 9 2002.
- [21] L.-Y. Wei and M. Levoy. Fast texture synthesis using tree-structure vector quantization. In *Proc. of SIGGRAPH 2000, July*, pages 479–488, 2000.