Realistic Synthesis of Novel Human Movements from a Database of Motion Capture Examples

Appeared in Proceedings of the IEEE Workshop on Human Motion HUMO 2000

Luis Molina Tanco Adrian Hilton
Centre for Vision, Speech and Signal Processing
School of Electronical Engineering, Information Technology and Maths
University of Surrey, Guildford, Surrey GU2 7XH, UK
{L.Molina-Tanco, A.Hilton}@eim.surrey.ac.uk

Abstract

In this paper we present a system that can synthesise novel motion sequences from a database of motion capture examples. This is achieved through learning a statistical model from the captured data which enables realistic synthesis of new movements by sampling the original captured sequences. New movements are synthesised by specifying the start and end keyframes. The statistical model identifies segments of the original motion capture data to generate novel motion sequences between the keyframes. The advantage of this approach is that it combines the flexibility of keyframe animation with the realism of motion capture data.

1. Introduction

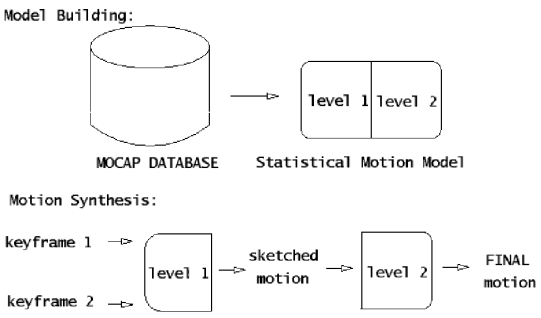

Figure 1. Overview of the system

The automatic synthesis of realistic human motion is one of the most difficult topics in computer graphics. An experienced animator is challenged when trying to create some apparently simple motion like walking. The production of only a few seconds of animation can take weeks, and requires a skill that can take years to acquire. Motion capture (*Mocap*) seems to be the most suitable solution to bring realistic, natural motion into the computer when a skilled animator is not available. However, motion capture has an important weakness: it lacks flexibility. It is very difficult to edit the captured motion without degrading its quality. Commercially available motion capture libraries are difficult to use as they can have hundreds of examples which are normally browsed by the name of the actions they contain. In this paper we present a system that combines the advantages of motion capture data with those of keyframing, which is a natural and widely used interface for computer animation.

We apply statistical learning techniques to build a synthetic model from a database of motion capture examples. The model is built with two levels. In the first level, a Markov chain of the joint trajectories is built. This level on its own can generate a coarse motion by traversing states of the Markov chain. However this does not suffice if the goal is high quality, realistic animation, as all the nuances in the motion are lost in the compression performed by the statistical model. The main novelty of our approach is the second level of the model, which relates the states of the Markov chain with segments of the original motions in the database, and generates a realistic synthetic motion based on these segments.

In section 2 we compare our approach with the work of other researchers. In section 3 we explain how to build the two levels of the model. Finally we present our results and discuss the future work in sections 4 and 5.

2. Related Work

Our method derives from those employed by researchers that use state-space models to “see” human activities, fol-

lowing the classification provided by Aggarwal and Cai [1].

Johnson and Hogg [5] use vector quantisation and Markov chains to build a statistical model of the silhouettes of two persons shaking hands in order to synthesise one of the silhouettes, resulting in a virtual interaction. Brand [3] builds a HMM of three-dimensional positions and velocities of motion capture markers on a human model. This model is used to infer 3D information from noisy 2D silhouettes. Both are tracking methods and therefore need a cue signal which is not available for animation. These models are not built on the joint space but on some work space of 2D or 3D points. This requires strong efforts in order to make the models invariant with respect point of view or scale.

More recently, in [4], Galata extends [5] using Variable Length HMM to model longer memory dependencies in time sequences. Also Bowden [2] uses K-means clustering and hierarchical PCA to build a Markov chain that models a motion example. These methods of motion synthesis would not be useful for animation as once a first state (keyframe) is chosen, the action that results is either fixed or random, but cannot be controlled nor can have new combinations of actions.

Pullen and Bregler [9] also use statistical models of the motion capture data, in particular a Kernel based modelling of the joint probability of features of the motion at different frequencies. They show its application to the generation of physically plausible random variations of a 2D character with 3 degrees-of-freedom performing one action. It is difficult to predict how well this method will scale for full body 3D motion.

Stuart and Bradley describe a corpus-based motion interpolation in [12]. The motion of each joint is independently learnt and represented. This allows for innovation in the synthesis, but fails to capture the correlations between joints, and very awkward poses can result.

Most of previous approaches have in common that the synthesis is an intermediate step for tracking or gesture recognition and is not a goal in itself. Hence the quality of the motion is often too poor to be of interest for character animation. However the statistical framework gives support for generalisation and handling of high dimensionality. In this paper we introduce a statistical framework which addresses the problem of "realistic" synthesis of novel human movements from a set of capture data examples.

3. Model Building

The statistical model has two levels, each having responsibility for a stage of the motion synthesis. The first level produces a sketch of the motion synthesis ("keyframes" the animation) and the second uses this sketch to sample the original motion capture data and generate a novel motion sequence.

3.1. Representation of Motion Capture Data

The raw motion capture data provides measurements of joints angles at discrete time intervals $t_i = i * \Delta t$ for $i = 0....N-1$. A minimal **angle-axis** representation [7] is used to represent the rotation for the j^{th} joint as a three-component vector $\boldsymbol{r}_i^{\,j} = \theta(t_i)\boldsymbol{n}(t_i)$ where $\boldsymbol{n}$ is a unit length axis and θ the angle about the axis. This representation gives a minimal parameterisation which readily allows the distance between two rotations and the mean of a set of rotations to be computed. For similar rotations the distance between two rotations $\boldsymbol{r_a}$ and $\boldsymbol{r_b}$ is approximated by the Euclidean distance between the vectors: $d_{ab} \approx ||(\boldsymbol{r_a} - \boldsymbol{r_b})||$. The mean rotation of a set of N similar rotations can be then estimated as the barycentre $\boldsymbol{r_m} \approx \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{r_i}$.

These properties of the angle-axis rotation enable construction of an efficient statistical representation of the motion capture data. Alternative minimal representation such as Euler angles do not allow mean rotation or distance between rotations to be directly computed. Equivalent redundant representation such as 3×3 rotation matrix or 4-component quarternion representations could be used but require additional constraints to be imposed to compute distances and to interpolate between rotations. The captured joint-angle motion data at each time instant can therefore be represented as a vector $\boldsymbol{\phi}$(t) which concatenates the individual joint angle rotations: $\boldsymbol{\phi}(t_i) = [\boldsymbol{r}_i^0, \boldsymbol{r}_i^1, ..., \boldsymbol{r}_i^{n-1}]^T$ where n is the number of joints. Translation of the root segment is ignored for the statistical model.

3.2. Preprocessing

The preprocessing comprises alignment and reduction of dimensionality of the data. Each of the motion examples is transformed by the inverse of the mean orientation of the root segment along the vertical axes to make the model **invariant** with respect to where the action is **facing** (e.g. we would like to have the same representation for a walking action from right to left as for one from left to right, where "left" and "right" are two opposite directions on the ground plane). We can represent the translation and rotation that a motion example applies to the root segment in the hierarchy of the human model via an homogeneous transformation matrix H_i^0 for each frame i. Before alignment, this transformation can be decomposed as follows[11].

$$H_i^0 = H(s_i)\ H(R_y(\theta_{3i}))\ H(R_x(\theta_{2i}))\ H(R_z(\theta_{1i})) \quad (1)$$

s_i is the translation applied to the model at frame i and $\theta_{1,2,3\ i}$ are the Euler Angles around axes Z, X, Y. Suppose Y is the vertical axes. The mean orientation around this vertical axis can be defined as $\widehat{\theta_3} = \frac{1}{N}\sum_{i=1}^{N} \theta_{3i}$. This mean is computed for each motion example. The new transforma-

tion applied to the root segment becomes

$$H_i^{0\prime} = H(R_y(-\widehat{\theta}_3))\ H_i^0 \tag{2}$$

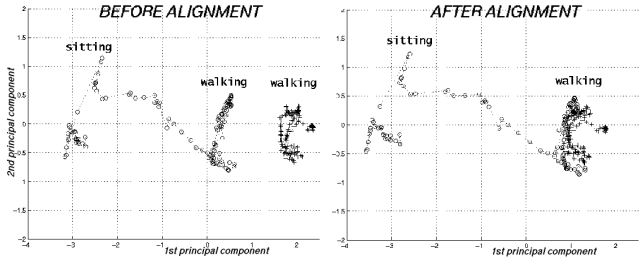

Figure 2. The two principal components of two motion examples are shown before and after alignment.

The human models used in motion capture to represent complex movements have a large number of degrees-of-freedom (DOF) (on the order of 70) corresponding to joint rotations. This representation contains considerable redundancy where joints such as hinge joints like the knee and elbow with one or two DOF are modelled as a revolute joint with 3 DOF. Also in a typical movement sequence not all physically plausible DOF are used. Motion data from optical or electromagnetic capture devices will also contain noise with a variance in the individual joint angle estimates of a few degrees. Principal Component Analysis (PCA) (see for instance [8]) is employed as a data compression technique to identify the significant variations in the data and eliminate the redundancy in the representation. The aim is to approximate N vectors $\boldsymbol{\phi}$ in a d-dimensional space by vectors in a M-dimensional space where $M < d$. Through principal component analysis we minimise the error introduced by the dimensionality reduction by choosing the vectors to be approximated with the expression

$$\tilde{\boldsymbol{\phi}} = \bar{\boldsymbol{\phi}} + \sum_{i=1}^{M} a_i \mathbf{u}_i \quad with \quad \bar{\boldsymbol{\phi}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{\phi} \tag{3}$$

where the $\mathbf{u}_i$ are the eigenvectors of the covariance matrix of the data $\mathbf{C}$. $\mathbf{C}\mathbf{u}_i = \lambda_i \mathbf{u}_i$ The eigenvectors are arranged in descending order according to the eigenvalues. The magnitude of the eigenvalues λ_i identify the components in the data which are most significant. Taking the M eigenvectors with the largest eigenvalues we can approximate the input data with a reduced basis. In practice taking approximately 15 eigenvalues represents $> 97\%$ of the variation in the mocap data. This reduces the complexity of deriving a statistical motion model from trajectories in a 70-dimensional space to a 15-dimensional space.

3.3. Level 1

After the preprocessing stage, the first level of the model building starts by dividing the joint space in clusters using a K-means classifier [6]. This is effectively a **vector quantisation** which is preferred to a multidimensional grid that divides the space into regular cells; through vector quantisation the shape of the cells can vary to adapt the density of the point distribution of the input signal. This allows us to better minimise the quantisation error [6]. A **Markov chain** is then built to recover temporal behaviour in the examples. Each state corresponds to one of the regions found in the previously computed clustering. The transition probabilities between these states are estimated by frequency counting on the training data. That is, for two consecutive input vectors found in the training data, $\boldsymbol{\phi_a}$ and $\boldsymbol{\phi_b}$, first the quantised versions are computed using the K-means classifier $\boldsymbol{q_a} = q(\boldsymbol{\phi_a})$ and $\boldsymbol{q_b} = q(\boldsymbol{\phi_b})$. The total number of times that a transition between $\boldsymbol{q_a}$ and $\boldsymbol{q_b}$ occurs, divided by the total number of times a position is quantised as $\boldsymbol{q_a}$, gives the estimated transition probability $P(\boldsymbol{q_b}|\boldsymbol{q_a})$.

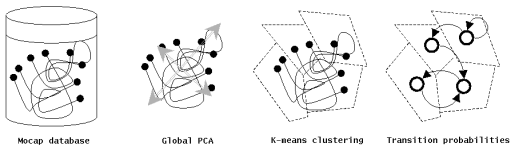

Figure 3. The first level of the model

3.4. Level 2

The second level of the model is a discrete output **Hidden Markov Model** [10] in which the hidden "states" are the motion examples. We choose the observable output probability distribution b of each of these states as being on the labels of the clusters found on the previous stage of the model, i.e. the states of the Markov chain ($b = b(cluster|example)$). This can be estimated using Bayes' relationship

$$b(cluster|example) \propto b(example|cluster)b(cluster)$$

We set the same a-priori probability for all K clusters $b(cluster) = \frac{1}{K}$. The conditional probability $b(example|cluster)$ is estimated as the ratio between the number of samples of that example that fall into that cluster divided by the total number of samples in the cluster. The result is normalised by the marginal probability

$$b(example) = \sum_{cluster=1}^{K} b(example|cluster)b(cluster)$$

The only thing that is needed to have a complete HMM is the transition probabilities between motion segments. These should be inversely related to the cost of jumping from one motion segment to another. In a first approximation we choose these transition probabilities to be fixed for any two examples m and n. Our basic assumption is that the longer the synthetic motion resembles a motion capture example, the more realistic it will appear. This implies choosing the transition probabilities so that we discourage jumping between motion examples, i.e. choosing $P(m|n)$ small and $P(m|m)$ large.

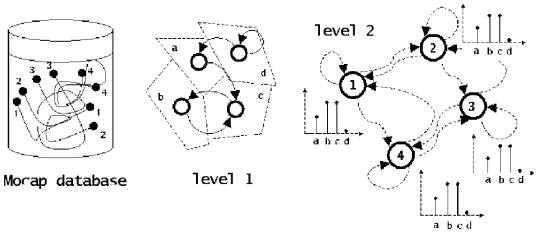

Figure 4. The second level of the model identifies motion examples with hidden states of a HMM.

3.5. Motion Synthesis

The synthesis is performed in two stages. The first stage starts by quantising the two user-specified keyframes using the K-means classifier. This results in an initial and final state in the associated Markov chain. A sequence of jumps between different states is needed to travel from the initial to the final state. We formulate this state sequence search as a **synchronous sequential decision** problem [10], which is solved using dynamic programming. Only transitions between different states have to be considered to find the most likely state sequence given the model.

A motion example may intersect several of the regions in the clustered joint space. The frontiers between these regions divide the motion examples into motion "segments". The second stage of the synthesis takes the generated state sequence as an input and solves for the most likely sequence of motion segments that could have generated that state sequence. The parameters of the HMM were chosen in such a way that the solution can be evaluated using the **Viterbi** algorithm [10] with known initial and final hidden states. The initial and final motion segments are chosen to be the motion trajectories closest to the arbitrary start and end keyframes specified by the user.

The resulting motion consists of a series of segments of the original captured motion sequences. To smoothly link these segments, the last position of the root segment in each motion segment serves as a world coordinate system for the following motion segment. The orientations of all joints are **blended** over a small interval of frames around the point at which the two motion segments are closest. Two rotations $\boldsymbol{r_a}$ and $\boldsymbol{r_b}$ can be interpolated by taking the rotation that transforms one into the other $\boldsymbol{r_{ab}} = \boldsymbol{r_b} \circ \boldsymbol{r_a^{-1}} = \theta_{ab}\boldsymbol{n_{ab}}$. $\circ$ is the composition of rotations and $\boldsymbol{r_b^{-1}}$ is the inverse rotation of $\boldsymbol{r_b}$ [7]. Interpolated rotations $\boldsymbol{b_{ab}}$ can be found by composing a fraction $\lambda \in [0, 1]$ of this rotation $\boldsymbol{r_{ab}}$ with $\boldsymbol{r_a}$ ($\boldsymbol{b_{ab}} = \lambda \boldsymbol{r_{ab}} \circ \boldsymbol{r_a}$). The motions of a joint in two examples can be blended over an interval of time $[n, n + T - 1]$ by calculating a sequence of T interpolated rotations with some function $\lambda(t_i)$ defined in that interval.

$$\boldsymbol{b_{ab}}(t_i) = \begin{cases} \boldsymbol{r_a}(t_i) & i \in [0, ..., n - T - 1] \\ \lambda(t_i)\boldsymbol{r_{ab}}(t_i) \circ \boldsymbol{r_a}(t_i) & i \in [n - T, ..., n] \\ \boldsymbol{r_b}(t_i) & i \in [n + 1, ..., N - 1] \end{cases}$$

n and $N - n + T$ are the number of frames of examples a and b respectively.

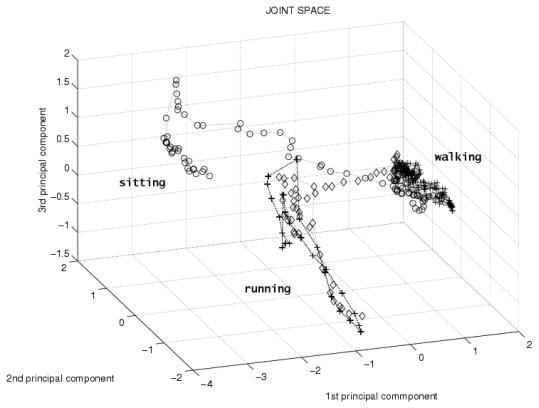

Figure 5. Three principal components of the motion capture database joint space.

4. Results

We show here an example of the functionality of the model on a small motion capture database of four motion sequences. The database contains a total of 465 frames and three different actions: standing up from the floor, walking and running. The original examples combine these actions as follows (see Fig. 5). 1: Standing up and then walking. 2: Running. 3: Walking and then running. 4: Walking.

We choose the number of clusters for the model to be 15, which represents a good compromise between accumulated quantisation error and complexity of the model. In figure 6 the error is plotted in 110 runs of the K-means algorithm as a function of K. The random initialisation of the K-means algorithm can produce big errors when K is small.

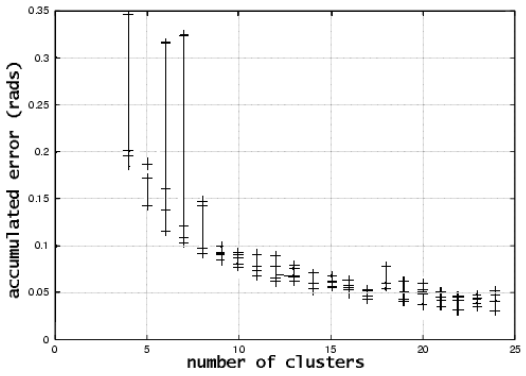

Figure 6. Accumulated quantisation errors.

A possibly desired synthetic motion would have the model standing up and then running, however his series of actions is not available in the captured examples. To synthesise the novel motion sequence we present the model with the keyframes of Fig. 7.

Figure 7. Keyframes for sitting and running.

The resulting motion has gone through a total of ten clusters and is composed of 71 frames (Fig 8). Two examples were used by the system to build the motion, number 1 and number 3. In figure 9 we show the smooth transition between the two actions.

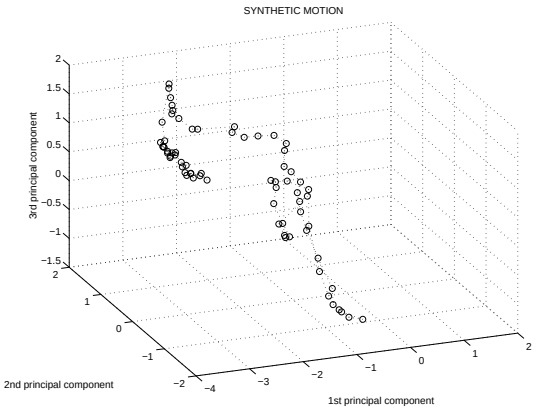

Figure 8. Synthetic motion in the joint space

5. Discussion and Future Work

We have developed a statistical framework which addresses the problem of "realistic" synthesis of novel human movements form a set of capture data examples. Our goal is to use this framework to build a natural interface between an animator and a motion capture database. For this reason we have chosen to have the system driven by user-specified keyframes.

The initial result presented in this paper for a small database of motion capture examples demonstrate the feasibility of realistic motion synthesis using a statistical model which generates novel sequences based on the original motion capture data. Results show that for this scale of database the reconstructed movement is realistic. The next stage is to scale the framework to a database of 50-100 movement examples to evaluate the performance.

Acknowledgements

The authors would like to thank Joel Mitchelson and Charnwood Dynamics for kindly providing the motion capture data. Also they would like to thank Andrew Stoddart, who initially supervised this project. This work has been funded by EPSRC grant GR/L78772 "Synthesising Human Motion for Virtual Performance".

References

[1] J. K. Aggarwal and Q. Cai. Human motion analysis: A review. *Computer Vision and Image Understanding*, 73(3):428–440, Mar. 1999.

[2] R. Bowden. Learning statistical models of human motion. In *Proceedings of the IEEE Workshop on Human Modeling, Analysis and Synthesis*, pages 10–17, June 2000.

[3] M. Brand. Shadow puppetry. In *ICCV'99. Proceedings of the International Conference on Computer Vision*, 1999.

[4] A. Galata, N. Johnson, and D. Hogg. Learning structured behaviour models using variable length markov models. In *Proceedings of the IEEE International Workshop on Modelling People (mPeople)*.

[5] N. Johnson, A. Galata, and D. Hogg. The acquisition and use of interaction behaviour models. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 866–871, June 1998.

[6] J. Makhoul, S. Roucos, and H. Gish. Vector quantization in speech coding. *Proceedings of the IEEE*, 73(11):1551–1588, Nov. 1985.

[7] X. Pennec and J. Thirion. A framework for uncertainty and validation of 3D registration methods based on points and frames. *Int. Journal of Computer Vision*, 25(3):203–229, 1997.

[8] M. Petrou and P. Bosdogianni. *Image Processing. The Fundamentals.*

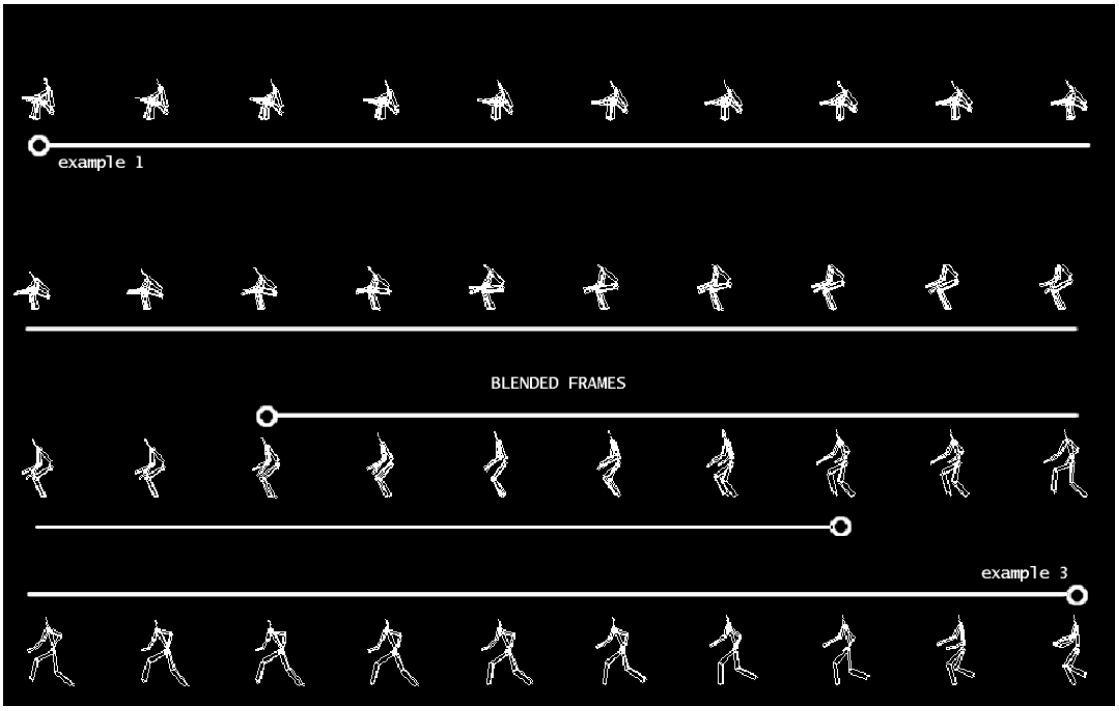

Figure 9. Transition between sitting and running

[9] K. Pullen and C. Bregler. Animating by multilevel sampling. In *IEEE Computer Animation Conference 2000*, pages 40–48, 2000.

[10] L. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition*. Prentice-Hall, 1993.

[11] K. Shoemake. *Graphics Gems IV*, chapter Euler Angle Conversion, pages 222–229. Academic Press, 1994.

[12] J. M. Stuart and E. Bradley. Learning the grammar of dance. In *Proc. 15th International Conf. on Machine Learning*, pages 547–555. Morgan Kaufmann, San Francisco, CA, 1998.