A. Egges, T. Molet, N. Magnenat-Thalmann. Personalised Real-time Idle Motion Synthesis. Pacific Graphics 2004, Seoul, Korea, pp. 121-130. October, 2004.

Personalised Real-time Idle Motion Synthesis

Arjan Egges
MIRALab - University of Geneva
Switzerland
egges@miralab.unige.ch

Tom Molet
MIRALab - University of Geneva
Switzerland
molet@miralab.unige.ch

Nadia Magnenat-Thalmann
MIRALab - University of Geneva
Switzerland
thalmann@miralab.unige.ch

Abstract

In this paper, we propose a novel animation approach based on Principal Component Analysis that allows generating two layers of subtle motions: small posture variations and personalised change of balance. Such a motion generator is needed in many cases when one attempts to create an animation sequence out of a set of existing clips. In nature there exists no motionless character, while in computer animation we often encounter cases where no planned actions, such as waiting for another actor finishing his/her part, is implemented as a stop/frozen animation. We identify many situations where a flexible idle motion generator can help: from synchronisation of speech/body animation duration, to dynamic creation of stand still variations in between two active plays. Our approach overcomes the limitations of using a small set of existing clips as a basis for synthesizing idle motions, such as unnatural repetition of movements and difficulties to insert idle motions into an animation without breaking its continuity. A realistic animation is obtained by blending small posture variations with personalised balance shifting animations.

1. Introduction

Although virtual models of humans continue to improve both in real-time and non-real-time applications, controlling and animating them realistically still remains a difficult task. Human beings are moving *all the time* and in a *unique way*. When animating a Virtual Human, one first has to solve the problem of the lack of animation between different animation clips, to avoid an unnatural-looking frozen posture between motions. Additionally, as every person has a unique way of moving and standing, an individualised motor control system is required to transfer this property to Virtual Human motions.

From now on, we will call the motions that occur in waiting/idle states between animation clips **idle motions**. These motions include changing balance because of fatigue, small variations in body posture caused by small muscle contractions or eye blinking. There is very little work done to realistically simulate idle motions, with an exception to the work by Perlin [18], that describes a motion generator (based on a noise function) for a few joints. However, human idle motions affect *all joints*, which cannot easily be solved by using noise functions on all joints, because of dependencies between joints. Apart from generating random movements for separate joints, another possibility is to take captured animation files as a basis for generating idle motions. This results in idle motions that affect all joints, but unfortunately, they are very repetitive and inflexible. Secondly, this approach generates transition problems between the animation clips.

In this paper, we propose a system for real-time virtual human idle motion synthesis for the body. The generated idle motions are based on statistical data obtained from a large set of recorded animations. As a basis for the motion generation, we use a Principal Component Analysis (PCA) to determine dependencies between joints in the various postures. This results in animations that affect all main joints in a coherent way. Furthermore, the Principal Components allow performing many other operations such as easy blending and fitting of motions, while working with virtually independent variables in linear space, resulting in flexible and realistic animations. With the proposed technique, we can also emulate idle behaviour of a specific individual by using animation segments of the recordings of this person, and thus make a Virtual Human move like this person would.

In the next section, we will discuss some related work in this area. After that, we discuss the PCA of animations. In Section 4, we present the main architecture of our idle motion synthesizer. Section 5 and Section 6 discuss the different parts of the system and the techniques behind them in more detail. In Section 7 we discuss the integration of the idle motion engine in a real-time animation framework. Section 8 shows some results of the idle motion synthesis and Section 9 presents our conclusions and indications of future work.

2. Related Work

There is a broad number of algorithms for motion synthesis from motion data, although they are seldom directed to addressing the idle-motion concerns. Kovar et al. [13] proposed a technique called motion graphs for generating animations and transitions based on a motion database. It could possibly be applied to other specific problems than the demonstrated locomotion; however this technique relies on a closed database of motions, going through a preprocessing stage, whereas our technique smoothly adapts to any animation schemes, being motion captured or computed on the fly. This is a key aspect since the idle-motion is rarely the end goal of animators, but rather a convenient way for building a transition between different active movements. It is therefore desirable to have activate/deactivate functionalities that can be triggered at AI or script level.

Li et al. [15] divided the motion into textons modelled using linear dynamic system. Additionally, Kim et al. [12] propose a method that analyses sound and that can generate rhythmic motions based on the beats that are recognised in the audio. These techniques are best suited for motions consisting of rhythmic patterns such as disco dance. The method might eventually be applicable for idle motions, but this paper proposes a much simpler solution.

Pullen and Bregler [19] proposed to help the process of building key-frame animation by an automatic generation of the overall motion of the character based on a subset of joints animated by the user. There the motion capture data is used to synthesize motion for joints not animated by the user and to extract texture (similar to noise) that can be applied on the user controlled joints. Our method tries to minimise user intervention, and it is specifically oriented towards real-time applications.

Lee et al. [14] propose a motion synthesis method based on example motions that can be obtained through motion capture. Also relevant work has been done by Arikan et al. [5, 4] that define motion graphs based on an annotated motion database. They present an automatic annotation method, which could be applied also to the balance shifting motions as an extension. However, since balance shifting motion clips often contain unwanted joint movements (such as gestures or other arm and head movements during the balance shift), clear constraints have to be defined during the annotation process.

3. Principal Components of Body Animations

There exist many techniques for animating virtual characters. Two very commonly used techniques are:

- **Key-frames:** an animation is constructed from a set of key-frames (manually designed by an animator or generated automatically) by using interpolation techniques. Although this method results in very flexible animations, the realism of the animations is low, unless a lot of time is invested.
- **Pre-recorded animations:** an animation is recorded using a motion capture/tracking system (such as Vicon or MotionStar). The animation realism is high, but the resulting animation is usually not very flexible, although methods have been developed to overcome part of this problem [6].

A method like PCA can determine dependencies between variables in a data set. The result of PCA is a matrix (constructed of a set of eigenvectors) that converts a set of partially dependent variables into another set of variables that have a maximum independency. The PC variables are ordered corresponding to their occurrence in the dataset. Low PC indices indicate a high occurrence in the dataset; higher PC indices indicate a lower occurrence in the dataset. As such, PCA is also used to reduce the dimension of a set of variables, by removing the higher PC indices from the variable set.

A PCA on animation data has already been successfully attempted by Alexa and Müller [3]. In their work, they perform a PCA on animations represented by relative deformations of the model, thus creating a ‘shape space’. As we are focusing on humanoid animations and we want to be as independent as possible from the geometry itself, we perform the PCA on the underlying joint skeleton, according to the H-Anim standard [10]. For our analysis, we use a subset of the H-Anim joints. This is done not only to improve the real-time performance, but also because we believe that some joint sets can better be treated separately to increase flexibility. For example, finger animations need to be very detailed for gesture animations, and they would probably not benefit much from an integrated PC analysis on both body and finger joints.

3.1. Motion Capture

The clips that we use to produce new motions, are recorded using the Vicon motion capture system [1]. The

marker positions are converted into an H-Anim skeleton animation represented by joint angle rotations and a global (root) translation. Clips recorded by motion capture are usually bound to specific coordinate frame of the tracking system and/or start each at different global position and orientation. In order to remove those irrelevant initial conditions, we apply a data alignment procedure on the root joint. Depending on the clip, we align position and orientation (2 + 3 degrees of freedom DOF), position and turn (2 + 1 DOF), or only position (2 DOF). In all of these procedures the vertical displacement is kept unchanged because it is valuable information. The first solution is the most severe since it resets the initial global posture back to the world's origin. This is the best choice solution as long as no relevant initial orientation must be preserved. However, in cases where the clips start with meaningful tilt, roll orientation, such as in a laying down posture, the second correction—align position and turn—is preferred. These two alignment procedures cover most of the situations because the initial heading posture is rarely relevant unless being part of the design of the motion clip, in which circumstance we can either align only the position or use no correction at all.

3.2. Conversion from and to Principal Components

In order to perform a PCA, we need to convert each frame of the animation sequences in the data set into an N-dimensional vector. In our case, one posture/key-frame is represented by 25 joint rotations (including one *root* joint) and one root joint translation.

Directly applying the PCA on the rotation matrix values does not give a desirable result, since the rotation space has a non-Euclidian geometry. However, a 'linearised' version of the (orthogonal) rotation matrix can be constructed, as is explained in the following paragraph.

For representing rotations, we use a skew-symmetric matrix. For every real skew-symmetric matrix, its exponential map is always a rotation matrix (see for example Chevalley [7]). Conversely, given a rotation matrix R, there exists some skew-symmetric matrix B such that $R = e^B$. The skew-symmetric matrix representation of a rotation is very useful for motion interpolation [16, 2], because it allows to perform linear operations on rotations. A three-dimensional real skew-symmetric matrix has the following form:

$$B = \begin{bmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{bmatrix} \tag{1}$$

Such an element can also be represented as a vector $r \in \mathbb{R}^3$

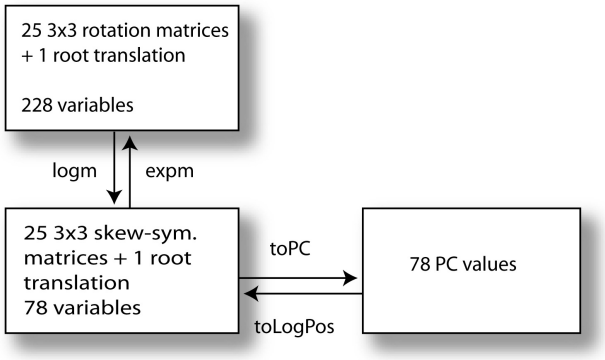

Figure 1. Overview of the steps involved for converting transformation matrices into Principal Components and back. First we apply the matrix logarithm to go to linearised transformation matrices, and then convert the data to PC space by using the PC matrix.

where:

$$r = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{2}$$

Given that $\theta = \sqrt{a^2 + b^2 + c^2}$, the exponential of a matrix B is defined by Rodrigues' formula:

$$e^B = I_3 + \frac{\sin\theta}{\theta} B + \frac{(1 - \cos\theta)}{\theta^2} B^2 \tag{3}$$

Similarly, methods exist that define how to calculate a determination of the (multivalued) matrix logarithm. For example, Gallier and Xu [9] present methods to calculate exponentials and logarithms, also for matrices with higher dimensions.

Using the vector representation shown in Equation 2 for a joint rotation, a posture consisting of m joint rotations and a global root translation can be represented by a vector $v \in \mathbb{R}^{3m+3}$. In our case, we use 25 joints, so a posture is defined by a vector of dimension 78. We apply a Principal Component Analysis on these posture vectors, resulting in a PC space of equal dimension (see Figure 1).

4. Multi-level Idle Motions

We have separately recorded the motions of ten people of both genders while they were in a conversation. This provides us with motion data of both gestures and idle motions. In the recorded data, we have observed three common types of idle behaviour:

Posture shifts This kind of idle behaviour concerns the shifting from one resting posture to another one. For example, shifting balance while standing, or go to a different lying or sitting position.

Continuous small posture variations Because of breathing, maintaining equilibrium, and so on, the human body constantly makes small movements. When such movements are lacking in virtual characters, they look significantly less lively.

Supplemental idle motions These kinds of motions generally concern interacting with ones own body, for example touching of the face or hair, or putting a hand in a pocket.

In this paper, we focus on the first two types of variations. The supplemental idle motions often involve an interaction with one's own body. This makes it difficult to simulate them in a generic way without using complex retargeting techniques [21, 22], that are difficult to integrate into a real-time framework. In our system, we use pre-recorded sequences for these kinds of motions, but this is not a very flexible solution and it is part of our future work to extend the system so that it can also generate these kinds of motions automatically. In the following sections, we will discuss generating both generic posture shifts and small posture variations for the case of idle motions while standing. Additionally, in Section 7, we will show how the idle motions can be integrated within a real-time animation system.

5. Posture Changing

Every person has a distinctive way of standing, sitting, lying and moving. In fact, from only the animations, in most of the cases people (family, friends or colleagues) can determine which person they belong to. Our goal is to capture (at least some of) this personalised way of moving. In this section we will propose an approach that allows creating an animation database from recorded clips. From this database, we can generate fully automatically new realistic animations. This results in a Virtual Human that mimics the postures and the way of moving of the person that was recorded. As an example to illustrate our techniques, we use recordings of people standing.

5.1. The Basic Animation Structure

When a human is standing, he/she needs to change posture once in a while due to factors such as fatigue [20]. Between these posture changes, he/she is in a resting posture. We can identify different categories of resting postures, such as: balance on the left foot, balance on the right foot or rest on both feet. As such, given a recording of someone standing, we can extract the animation segments that form the transitions between each of these categories. These animation segments together form a **database** that is used to synthesize balancing animations. In order for the database to be usable, at least one animation is needed for every possible category transition. However, more than one animation for each transition is better, since this creates more variation in the motions later on. The database of animation segments is called A from now on. In order to generate new animations, recorded clips from the database are blended and modified to ensure a smooth transition.

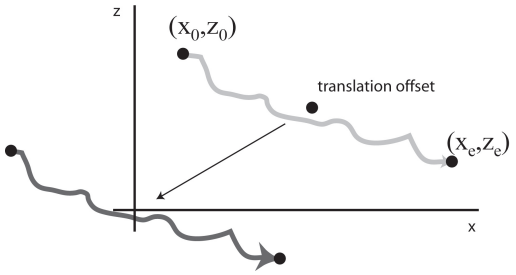

Figure 2. This figure illustrates what the estimated translation offset is in the (x, z)-plane (frontal-lateral plane) for an animation sequence.

5.2. Removing the Translation Offset

Since we will later on use the animations from the database to create balance-shifting sequences, it is necessary to apply a normalisation step so that the relative translations between the animations are minimised. Therefore we estimate the **translation offset** for each of the animation sequences to the origin $(0, 0, 0)$ by the mean value of the first and last frames of the animation (see Figure 2). In order to remove this offset from the animation, for each frame we subtract the translation offset from the root translation. There are other possibilities to estimate the translation offset, for example by searching the most central key-posture in the animation, or by using the timing information to determine the middle of the posture shift.

5.3. Transition between Categories

We denote the set of categories of resting postures as C. As each animation depicts a transition from one category to another, we can assign the first and last frame of each $a \in A$ to the category in C that they belong to. Therefore a category $c \in C$ corresponds to a set of postures $Q_c = q_1, \ldots, q_m$. Apart from these postures, we add standalone postures (that are not part of any animation in the database) in order to increase the variety of resting postures

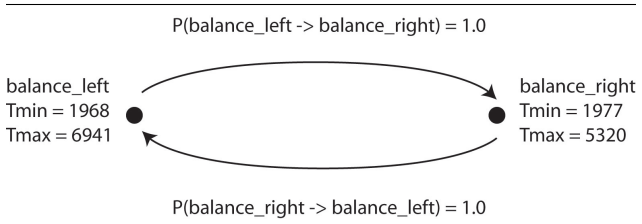

Figure 3. A simple example of a category transition graph. In this case, there are two categories: balance on the left foot and balance on the right foot. For both the categories, a minimum and maximum time is determined from the data (written in milliseconds).

in the category. However, these postures should be within a limited translation interval to avoid unnatural shifts during the transition. If necessary, such postures are normalised by setting the translation to the mean translation of the other postures that are already in the posture set for the corresponding category.

From the original animation data, we also determine what the probability is that a transition occurs from one category to another by counting the occurrences in the data. The probability for a transition between category c_1 and c_2 is denoted by $P(c_1 \Rightarrow c_2)$. Obviously, the sum of the probabilities of one category to others should be 1.

Furthermore, for each category c, we extract from the data the minimum and maximum amount of time (denoted by $T_{c,min}$ and $T_{c,max}$) that a person stays in one of the categories, before performing a transition to another category. Figure 3 shows a simple example of a probability distribution for two categories and their $T_{c,min}$ and $T_{c,max}$.

5.4. Synthesis of Balance Changes

5.4.1. First step As a first step, we choose a random category c and then a random posture $p \in Q_c$. Furthermore, we choose a random time value t, where $T_{c,min} \leq t \leq T_{c,max}$. This already allows to create an animation consisting of two key-frames containing posture p at time indices 0 and t.

5.4.2. Choosing a New Category After the posture at time index t, we need to construct a transition to another category. In order to define the next category, we again choose it randomly, but according to the probability distribution as defined in the category transition graph. This gives the successor category s. Within this category, we choose randomly a posture $q \in Q_s$. Now we have to retrieve the animation from posture p to posture q and add the key-frames of the animation to the animation that we are constructing.

5.4.3. Retrieving the Best-fitting Animation The earlier constructed database of animations A may not contain all possible animations for all the postures. Therefore, we have developed a technique that allows to map an animation onto a set of key-postures. In order for this technique to be as realistic as possible, we have to determine which animation in the database would fit best with the start posture p and end posture q. In order to determine this, the system must choose the animation that has its first frame and last frame as close as possible to respectively p and q. Suppose that the postures are represented by an N-dimensional vector of PC values. We then define the distance between two postures p and q as the weighted Euclidean distance between the two vectors (w_i is the weight of variable i):

$$d_{p,q} = \sum_{i=1}^{N} w_i \cdot (p_i - q_i)^2 \tag{4}$$

Using these weights allows us to assign more importance to parts of the posture that occur often in the data. As a result, less relevant parts of the posture are contributing less to the distance between postures. The problem of estimating the distance between postures has also been addressed by Kovar et al. [13], who use point clouds that are driven by the skeletons. As values for the weights, we use the normalised[1] values of the eigenvector of the PC matrix, since these values correspond to the occurrence in the data of each PC value.

In order to select the best fitting animation between p and q, we calculate for each animation $a \in A$ the maximum M_a of the distances $d_{a_0,p}$ and $d_{a_e,q}$. The animation that we will use, is the one that has a minimal M_a.

5.4.4. Fitting the Selected Animation to the Begin and End Posture The animation a (=$(a_0, a_1, \ldots, a_e)$) starts at frame 0 and ends at frame e. If we want to fit the animation so that it starts at posture p and ends at posture q, we have to transform a so that $a_0 = p$ and $a_e = q$ (see Figure 4). This problem can be solved as a specific case of a more general problem:

> Given an animation a and a list L of tuples (f, p) where f is a frame number, p is a posture represented by a vector of PC values, and $L_i = (f_i, p_i)$ is the i-th element in the list L. Furthermore, $\forall L_i \in L : f_i < f_{i+1}$ (i.e. the elements in the list are ordered according to increasing frame number). Finally, $\forall L_i \in L : 0 <= f_i <= e$. How can we modify the animation a so that is passes through all the postures in the list L at their specified frame numbers?

1 The eigenvector e of a PC matrix has the property that $e_i \geq e_{i+1}$ for $0 \leq i < N$ and also $e_i \geq 0$ for $0 \leq i \leq N$. The vector is normalised by applying a multiplication factor so that $e_0 = 1$.

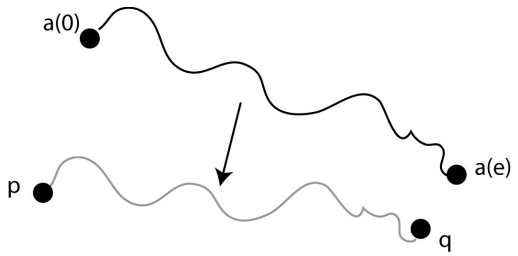

Figure 4. An example of fitting an animation from postures (a_0, a_e) to postures (p, q).

fitAnim(a, L)
$begin = 0, \;\; end = 0$

for (i=0; i <size(L); i++)
$begin = end$
$end = f_i$
if (end!=0)
fitAnim_part($begin, end, a_{f_i}, p_i, a$)

if (f_i != e)
fitAnim_part(f_i, e, p_i, a_e, a)

fitAnim_part(f_i, f_j, p_i, p_j, a)
$o_i = p_i - a_i$
$o_j = a_j - p_j$
for ($f = f_i$; $f \leq f_j$; f++)
$a_f = a_f + o_i \cdot \frac{f_j - f}{f_j - f_i} + o_j \cdot \frac{f_i - f}{f_i - f_j}$

Figure 5. Algorithm for fitting an animation *a* to a key-posture list *L*. The main function uses a subroutine `fitAnim_part` that fits an animation segment between two frames.

The animation fitting can be done by calculating the offset of the postures $(f_i, p_i) \in L$ with respect to a_{f_i} and then interpolating between the offsets of all pairs L_i and L_{i+1}. The algorithm that calculates the modified animation a passing through the key-posture list L is shown in Figure 5.

In order to change the best-fitting animation a so that it starts at posture p and end at posture q, we can simply use the function `fitAnim_part`(a_0,a_e,p,q,a). After this, the key-frames of the updated animation segment a' is appended to the sequence we are constructing. Next, a resting posture key-frame is added according to the method explained in Section 5.4.1. Then again a balance shift is added, and so on. Fitting an animation in this way only works if the difference in translation between p and q and the original starting- and end-point is small. As we are specifically

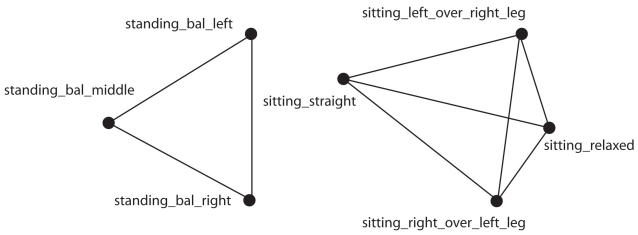

Figure 6. Example of a category transition graph with different types of idle motions.

dealing with balance changes that involve slight translation, this approach works well. However, when one wants to add more general categories of motions (such as walking to an object, sitting down on a chair), more complex methods have to be used, such as retargeting [21, 22].

5.5. Sitting, Standing and Lying Idle Motions

Although the PCA was performed on a database of standing people, the technique that is presented in this paper is not limited to only synthesizing standing humans. A more extensive database can be constructed that contains not only balance shifts of standing people, but also animations of sitting people that change their sitting posture or lying posture. Again, a Category Transition Graph can be constructed that contains animation sequences between different sitting postures and the animations between categories can be refitted just like the animations between standing postures. A problem that arises is the fact that such kinds of postures are relative to one or more objects (a chair or a bed for example). However, the idle motion can be played relative to the humans position in space, after performing the normalisation process as described in Section 5.2.

In Figure 6, an extended graph is shown that also contains sitting postures. In this case, the animations of a human sitting down or standing up are not in the database. This is done for several reasons. First, we feel that sitting down or standing up is not really an idle action, but an action that animators want to control independently from the idle motions. Secondly, sitting down or standing up means a possibly significant change of location of the virtual human. The animation fitting technique described in Section 5.4.4 works only for animations that are not too different in terms of displacement.

5.6. Personalised Idle Motions

The animation database, as described in Section 5.1, is constructed from one or more recorded animations of a

person. We feel that not only the basic postures of someone, but also the way a person changes posture, is unique for everyone. This feeling is strengthened by our findings that, in general, people were able to recognise whose animations were at the basis of the animations generated by the method described before. According to this subjective analysis, our technique allows grasping the way someone stands and moves. We have constructed personalised animation databases for several people, which allows one to choose which person a virtual human should mimic. Quantitative blind test studies should be performed to further assess these promising results.

6. Continuous Small Variations in Posture

Apart from the balance shifting postures that were discussed in the previous section, small posture variations also greatly improve the realism of animation. Due to factors such as breathing, small muscle contractions etc., humans can never maintain the exact same posture.

As a basis for the synthesis of these small posture variations, we use the Principal Component representation for each key-frame. Since the variations apply to the Principal Components and not directly to the joint parameters, this method generates randomised variations that still take into account the dependencies between joints. Additionally, because the PCs represent dependencies between variables in the data, the PCs are variables that have *maximum independency*. As such, we can treat them *separately* for generating posture variations.

One method to generate variations, is to use one-dimensional Perlin Noise [17] for each—or a subset—of the Principal Components. In this paper we will present another method that generates these small variations based on the shape of the curves in the motion capture data. This method applies a statistical model to generate similar (randomised) curves. This approach also allows keeping certain existing tendencies in the variations that are specific to certain persons (such as typical head movements, etc.). An additional advantage over using Perlin noise, is that this approach is fully automatic and it avoids the need to define frequencies and amplitudes to generate noise, although these parameters could eventually be determined automatically by analysing the signal.

In our method, we analyse animation segments that do not contain any balance shifts or other motions other than the small variations. The results of the analysis will then be used to generate similar variations on top of other postures.

6.1. Normalisation of the Data

The selected animation segments need to be normalised, so that only the variations persist and the base postures are removed. This is necessary, since these variations will be synthesized on *top* of the balance shifting postures (see Section 7). In order to normalise an animation segment, we calculate the mean value of each PC in the segment and subtract it from the PC values in each key-frame, therefore removing the base posture and keeping only the variations (see Figure 7 for an example). Clearly, this approach only works when dealing with animation sequences where the base posture does not change significantly during the sequence.

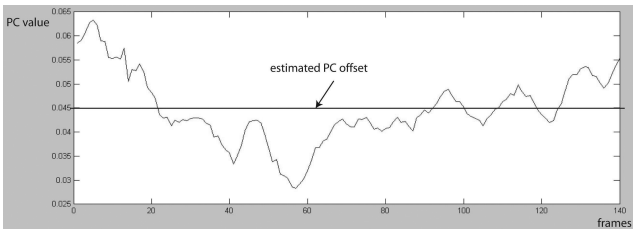

Figure 7. This example shows the values of a Principal Component for an animation segment. The offset is estimated by the mean value (which is around 0.045).

6.2. Variation Prediction

In this subsection, we describe the method we use to predict the variations according to motion capture data. For illustration of our method, we use the PC values of the animation segment in Figure 7. In order to reduce the number of points in the PC curve while retaining its global shape, we convert the PC values in a set of maxima and minima. Then, we will show a technique to predict the next maximum/minimum given its predecessor point.

For estimating the maximum and minimum, a simple algorithm can walk through the points and—given an error value ϵ—determine if a point is a maximum or a minimum. In order to remove the absolute timing from the data, we specify each maximum/minimum by defining its distance (in milliseconds) from the previous maximum/minimum and the corresponding PC value (see Figure 8 for an illustration).

The next step is to construct a system that can, given a $(distance, value)$ pair, predict the next $(distance, value)$ pair. The generated $(distance, value)$ pairs then form a new animation sequence. In order to add some randomness to the system, we do not use the points directly, but we apply a point-clustering algorithm. In our application we have applied a k-means clustering algorithm, but other clustering algorithms can also be used (for an overview of different clustering algorithms, see Jain et al. [11]). From the mo-

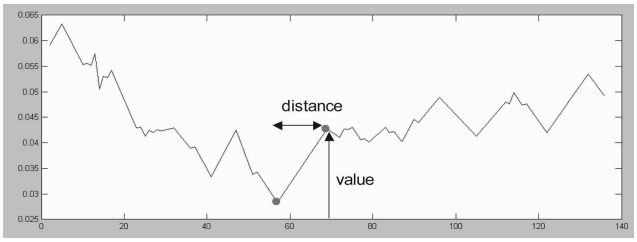

Figure 8. This figure shows a sequence of maxima and minima for a Principal Component. Each maximum and minimum is specified by defining the distance to its predecessor and its PC value.

tion data that we have, we can set up a probabilistic model that defines the probability of a transition from one cluster of points to another. The result is a Probability Transition Matrix that defines the probability for any transition between the clusters. This model can be used to generate new animations, according to the following algorithm:

1. add the neutral position at time $t = 0$;
2. select randomly a cluster of points;
3. choose randomly a point ($(distance, value)$ pair) within the cluster;
4. add a key-frame to the animation at time $t + distance$ with the specified PC value;
5. based on the last selected cluster, choose the next cluster according to the Probability Transition Matrix;
6. choose randomly a point ($(distance, value)$ pair) within the cluster;
7. go to step 4 until desired length of animation is reached.

Because the animations are not produced from the original points but from randomly chosen points in the clusters, the variations show very little repetition, but the PC value curve over the animation still has a very similar structure to the animation samples that were analysed.

In order to have a variation synthesis, the process of generation of animation segments, which is described in this section, is applied to a subset of PC values. For each Principal Component, the variations are generated independently. This does not give unrealistic results, since the dependency between Principal Components is relatively small. The final result of the variation synthesis shows little to no repetition, because of the random choice of points in each cluster, and because of the separate treatment of PCs.

7. Integration into a Real-Time Application

In order to integrate the idle motions into a real-time application, where other animations might be playing as well, an infrastructure is needed that governs how the animations are mixed with the already existing animations. The already existing animations can be key-frame gestures, head movements, walking, and so on. The idle motions are blended relative to the other animations, so that the idle motions can be played independently of the location and rotation of the corresponding virtual human. Every virtual human is linked with an idle motion engine that uses animation segments from different databases. This allows having several virtual humans in one scene, all displaying different personalised idle motion behaviour.

7.1. Inserting idle motions between two animation clips

When a virtual human is in a waiting state between animations, it is desirable to generate an idle animation that starts at the end posture p_0 of the finished animation and that ends at the begin posture p_1 of the next animation. Our system allows to do this, assuming that the differences between the two postures are not too big, especially on the level of the translation, to avoid unnatural shifts. The first step is to determine in which posture shifting category p_0 belongs. This is done by calculating the difference between p_0 and the postures in each of the categories. The category that has the minimum distance is then chosen. The approach described in Section 5.4 can then be used to generate the animation, but with p_0 as starting posture. As soon as the system receives a message that the next animation has to start, the idle animation has to end at posture p_1. First, the system determines in which category p_1 belongs (using the same method as for p_0). Then, the shortest path in the category transition graph from the current category to the end category is determined. If the animation is already in the end category, the shortest cycle in the category transition graph is determined[2]. If there is more than one shortest path/cycle, the system chooses randomly one. Arriving in the end posture p_1 is done by using the previously described animation fitting method for the last transition so that the animation ends at the p_1 posture. In case of time constraints, the final animation part is shortened by decreasing the time between balance shifts and by time scaling the balance shifting animations. A too rigorous time scaling or a too small waiting time between balance shifts will result in a less realistic animation. However, if the posture p_1 is very close to a posture in a certain category, and when the new animation has to start, the idle motion engine is already in that

2 For an explanation of shortest-path/cycle algorithms such as Dijkstra's and Floyd's, please see Cormen et al. [8].

category, a simple interpolation between the current posture and p_1 suffices.

7.2. Blending of idle motions with other animations

In addition to filling up the waiting/idle states between animations, idle motions can also be mixed with other animations, such as gestures, head movements and so on. Mixing idle motions with other animations happens in two steps (Figure 9). First, the **balance-shifting** motions are blended with the other animations, using a weight factor. This blending happens at joint level, which allows defining joint masks that specify which parts of the animations should be discarded in the final animation. Using these masks is crucial, since there are certain animations that should not be blended at all with balance shifting motions, such as walking or running. Also, the masks indicate which part of the motion should not be changed. For example, in a gesture animation the moving arms should not be blended with the static arm posture in the idle motions, as this would result in a less pronounced gesture.

After blending the balance-shifting motions with the other animations, the **small posture variations** are added to the animation. The addition of small posture variations happens in PC space. For each frame of the animation, the PC values of the blended animation and the variation animation are added directly, resulting in the final animation. This approach works, because although the variations add to the realism of the virtual human during static postures, their presence during the animation segments is hardly noticeable.

8. Results

The techniques that are described in this paper allow to create a system that, based on a personalised database of animations and postures, can realistically reconstruct idle motions that are coherent with the animations and postures in the database (see Figure 10). For evaluation purposes, we have created personalised databases for different people. From preliminary visual tests, we can conclude that the animations that are generated from these databases, capture—at least a part of—the individuality of someone's motions. In general, test subjects could recognise the person that is behind the set of animations used to automatically generate the idle motion.

The animations can be generated and played in real-time on a normal desktop PC (1.8GHz), as the most time-consuming calculations are pre-computed during the data analysis part. Additionally, the conversion between PC values and rotation matrices follows the real-time operation constraints, as it consists of applying a logarithm/exponential of a 3×3 matrix and a matrix multiplication for each key-frame. Note that quaternions could replace 3×3 rotation matrices since they do also allow for log/exp linearization.

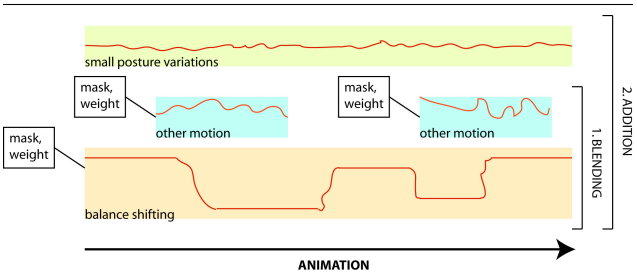

Figure 9. Integration of idle motions into a real-time animation framework. The automatic balance shifting motions and generic animations (gestures, head movements, etc.) are blended using a weight factor. Furthermore, joint masks can be used to remove certain parts of the animations to avoid conflicts and to improve flexibility. Finally, the small posture variations are added to the animation.

Figure 10. Some example postures from different databases.

9. Conclusions and Future Work

We have developed a flexible idle motion engine used for animation applications that require realistic motions of virtual humans during waiting/idle states. The motions are generated on the fly from a database of animations and pos-

tures, obtained by motion capture. This approach permits to specify different databases for different persons, thus introducing a personalised aspect to the animations that are generated.

In the near future, we will work on further improving the quality of the idle motions. First of all, we aim at enhancing the link between the small posture variation engine and the balance-shifting engine, in order to develop additional features, such as 'stabilising variations' just after the transition from one balance to another. Although there were no noticeable artefacts in the animations that we generated, it might be necessary to link the small variations to the current posture in the animations, since different postures could induce different kinds of small posture variations. Additionally, we would like to explore other possibilities that the Principal Components offer for improving the realism of other motions, such as body gestures. Finally, we would like to link the idle motion synthesis with more conceptual notions of personality and emotion based on psychological models.

10. Acknowledgements

This work has been developed through the support of the AIM@Shape Network of Excellence (Advanced and Innovative Models And Tools for the development of Semantic-based systems for Handling, Acquiring, and Processing Knowledge Embedded in multidimensional digital objects, IST-506766, http://www.aimatshape.net) funded under the Sixth Framework Programme and by the OFES. We would like to thank Dr. HyungSeok Kim for proofreading the paper.

References

[1] Vicon motion systems. http://www.vicon.com/. Accessed May 2004.

[2] M. Alexa. Linear combination of transformations. In *SIGGRAPH 2002*, pages 380–387, 2002.

[3] M. Alexa and W. Mueller. Representing animations by principal components. *Computer Graphics Forum*, 19(3):411–418, 2000.

[4] O. Arikan and D. Forsyth. Interactive motion generation from examples. In *Proceedings of ACM SIGGRAPH 2002*, 2002.

[5] O. Arikan, D. A. Forsyth, and J. F. O'Brien. Motion synthesis from annotations. *ACM Transactions on Graphics*, 22(3):392–401, 2003.

[6] A. Bruderlin and L. Williams. Motion signal processing. *Computer Graphics*, 29(Annual Conference Series):97–104, 1995.

[7] C. Chevalley. *Theory of Lie Groups*. Princeton University Press, New York, 1946.

[8] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge MA, 1990.

[9] J. Gallier and D. Xu. Computing exponentials of skew-symmetric matrices and logarithms of orthogonal matrices. *International Journal of Robotics and Automation*, 17(4), 2002.

[10] H-ANIM Humanoid Animation Working Group. Specification for a standard humanoid. http://www.h-anim.org/. Accessed May 2004.

[11] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[12] T. H. Kim, S. I. Park, and S. Y. Shin. Rhythmic-motion synthesis based on motion-beat analysis. *ACM Transactions on Graphics*, 22(3):392–401, 2003.

[13] L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. In *Proc. SIGGRAPH 2002*, 2002.

[14] J. Lee, J. Chai, P. Reitsma, J. K. Hodgins, and N. Pollard. Interactive control of avatars animated with human motion data. In *Proceedings of SIGGRAPH 2002*, July 2002.

[15] Y. Li, T. Wang, and H. Y. Shum. Motion texture: A two-level statistical model for character motion synthesis. In *Proc. SIGGRAPH 2002*, 2002.

[16] F. C. Park and B. Ravani. Smooth invariant interpolation of rotations. *ACM Transactions on Graphics*, 16(3):277–295, July 1997.

[17] K. Perlin. An image synthesizer. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, pages 287–296. ACM Press, 1985.

[18] K. Perlin. Real time responsive animation with personality. *IEEE Transactions on Visualization and Computer Graphics*, 1(1), 1995.

[19] K. Pullen and C. Bregler. Motion capture assisted animation: Texturing and synthesis. In *Proc. SIGGRAPH 2002*, 2002.

[20] I. Rodriguez, R. Boulic, and D. Meziat. A joint-level model of fatigue for the postural control of virtual humans. *Journal of 3D Images*, 16(4), December 2002.

[21] C. Rose, M. Cohen, and B. Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications*, 18(5):32–48, September/October 1998.

[22] A. Witkin and Z. Popovic. Motion warping. In *Proceedings of SIGGRAPH '95*, pages 105–108, 1995.