The International Journal of Virtual Reality, 2006, 5(2):15-24

Interactive Virtual Humans in Real-Time Virtual Environments

Nadia Magnenat-Thalmann and Arjan Egges

***Abstract*—In this paper, we will present an overview of existing research in the vast area of IVH systems. We will also present our ongoing work on improving the expressive capabilities of IVHs. Because of the complexity of interaction, a high level of control is required over the face and body motions of the virtual humans. In order to achieve this, current approaches try to generate face and body motions from a high-level description. Although this indeed allows for a precise control over the movement of the virtual human, it is difficult to generate a natural-looking motion from such a high-level description. Another problem that arises when animating IVHs is that motions are not generated all the time. Therefore a flexible animation scheme is required that ensures a natural posture even when no animation is playing. We will present MIRAnim, our animation engine, which uses a combination of motion synthesis from motion capture and a statistical analysis of prerecorded motion clips. As opposed to existing approaches that create new motions with limited flexibility, our model adapts existing motions, by automatically adding dependent joint motions. This renders the animation more natural, but since our model does not impose any conditions on the input motion, it can be linked easily with existing gesture synthesis techniques for IVHs. Because we use a linear representation for joint orientations, blending and interpolation is done very efficiently, resulting in an animation engine especially suitable for real-time applications.**

***Index Terms*—Real-time Animation, Interactive Virtual Humans, Motion Synthesis**

I. INTRODUCTION

Over the last years, there has been a lot of interest in the area of Interactive Virtual Humans (IVHs).Virtual characters who interact naturally with users in mixed realities have many different applications, such as interactive videogames, virtual training and rehabilitation, or virtual heritage. The main purpose of using interactive virtual humans in such applications is to increase the realism of the environment by adding life-like characters. The means by which these characters perceive their environment and how they express themselves greatly influences how convincing these characters are.

Next to the improved capability of computers to analyze and interpret information, computers can also better respond to what is perceived. This can be a response using text, speech, or more elaborate visual output such as controlling a character in 3D. The most evident area that profits from these techniques is the area of computer entertainment industry. In many games, interactive 3D characters are controlled by the player or they form an interactive component as a part of the 3D gaming environment. In most games, such characters are controlled using scripted animation systems that sequence and play different animation and sound clips. Regardless of the recent advancements in this research area, controlling and animation virtual character still remains a tedious task and it requires a lot of manual design and animation work.

On a conceptual level, the earlier mentioned perceptual techniques (speech recognition, facial expression recognition, etc.) and responsive techniques (speech synthesis, virtual character animation) form a part of a loop, called the Perception-Action loop [1][1] This loop describes the feedback mechanism that defines any interaction of an entity with its environment. By acting in an environment, an entity changes it. These changes are then perceived and a new action is commenced (or not). This paper focuses on the expressive part of this loop.

In this paper, we will present an overview of relevant research related to the development of Interactive Virtual Humans (IVHs).Virtual Humans can express themselves through speech, facial animation and body animation. Generally, these components automatically generate the speech and the motions from an abstract representation of the desired output. This abstract representation can be for example a text or a complex XML structure. A dialogue system outputs such an abstract representation depending on what the user is doing. Fig.1 shows an overview of the main components that together form an expressive IVH simulator.

The main objective of the work that will be shown in the following sections is to provide for an efficient method to control both face and body movements of virtual characters, as well as techniques that increase the realism of character motions without impinging the tractability of the character control method. In fact, the research that will be presented constitutes an elaborate *realism-adding filter* that can be placed between an automatically generated animation and the target character.

II. BACKGROUND IN REAL-TIME ANIMATION

There exist many different techniques to `animate` virtual humanoids. In this section, we will give an overview of the different approaches that exist. Both the architecture and the

Manuscript Received on August 20, 2006

Nadia Magnenat-Thalmann and Arjan Egges are at MIRALab -University of Geneva, 7 route de Drize , 1227 Geneva, Switzerland. Tel: +41223797769. Fax: +41223790079. Email: { thalmann , egges}@miralab.unige.ch

[1] This is a term originating from the Robotics and Sensor research area.

performance of any animation pipeline will be greatly influenced by the approaches that are chosen. The representation of an animation depends on its level of abstraction and the structure of the world that it is used in. At the most basic level, we consider an animation as a continuous function A: $t \mapsto F$, where t is a timekey $\in$ [ts, te] with $0 \leq ts < te < \infty$, and F is the corresponding frame of the animation. The continuity of this function is an important property, because it allows to display the object's current state at any desired time. Because in practice it is not possible to define an animation as a continuous function—one would need to define an infinite amount of frames—most of the animation approaches rely on key-frames. In this approach, an animation is defined by a discrete sequence of key-frames, related with a timekey. In order to go back to the original continuous function, an interpolation technique is required. The choice of the interpolation approach depends on the representation of key-frames. In the following paragraph, we will discuss a few different representations of animations. Most of the animation models are restricted to the **spatial domain**, but one could also imagine animations that change the color or texture of the 3D object (in the case of facial animation, think of blushing for example). In principle, any animation can be coded as a direct manipulation of the geometry of the target 3D object. However, there clearly are many disadvantages to this approach:

- it is a lot of work to develop low-level animations; this also makes it difficult to retain a reasonable level of realism;
- Animations are not easy to transfer from one model to another as the animations will be dependent on the 3D model;
- Editing the animations afterward is a difficult task; common operations on animations (scaling, masking, and so on) are difficult to apply.

Different animation techniques have been developed, that allow describing an animation in more abstract terms, while being easily adaptable to different 3D objects. Especially when one wants to develop a (semi-)automatic animation system, these techniques form a crucial part of the animation pipeline.

Another important point to consider when one conceives an animation system, is its **usability** from a designer point-of-view. When we look at the design process for animations, there are two commonly used techniques:

- **Manual approach**: an animation is constructed from a set of key-frames, manually designed by an animator. many commercial packages are available characters).since the animator has a complete control over the resulting animation, this is a very popular approach. However, unless a lot of time is invested, the realism of the resulting animation is rather low.
- **Pre-recorded animations**: an animation is recorded using a motion capture or tracking system (such as Vi-con [2] or MotionStar [3]). The MotionStar system uses magnetic tracking and it is much faster than the Vicon tracking system, which uses a multi-camera optical tracking approach. The Vicon however produces much more precise animations. This is why in many commercial applications, such as games or movies, an optical motion capture system is used to drive the character motions. A motion capture system is an excellent time-saver, because a lot less manual work is required to produce animations. A major disadvantage of using such a system is that the animations need to be adapted so that they look good on different characters. Also, when using recorded motion clips, one must address the problem of unnatural transitions between the different clips.

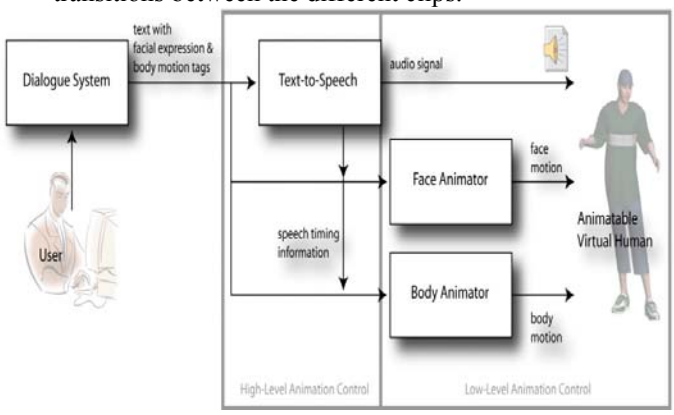

Fig. 1. Main components of an Interactive Virtual Human simulator.

For producing real-time body animations, it is common practice to work on the underlying skeleton instead of the model itself. When using a geometry-based approach, the animation model is more precise, but the animation is dependent on the model. A skeleton-based approach allows for model-and environment -independent animations as well as fewer parameters to manipulate. Our system is based on the H-Anim standard [4].

A. Body Animation Representation

Given that a 3D Humanoid model is being controlled through a skeleton motion, it is important to choose the right representation for the transformations of the skeleton joints. The most basic representation is to define a 4×4 transformation matrix that contains a translation and a rotation component, where the rotation is defined as a 3×3 matrix. Transforming a skeleton joint becomes tricky when looking at the rotation component. A rotation involves three degrees of freedom (DOF), around the x, y and z axis, whereas a rotation matrix defines 9 (as a 3×3matrix).A matrix can only represent a rotation if it is orthonormalised. This means that during animation, additional operations are required to ensure the orthonormality of the matrix, which is computationally intensive. Furthermore, rotation matrices are not very well suited for rotation interpolation. Therefore, other representations of joint rotations have been proposed. For a more detailed discussion of each of the representations, we refer to Grassia [5].

Our system uses the exponential map representation. Alexa [6] has described a method using the exponential map to allow

for transformations that are performed completely in the linear domain, thus solving a lot of problems that the previous methods are suffering from. Rotations are represented by 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 [8], because it allows to perform linear operations on rotations. A three-dimensional real skew-symmetric matrix has the following form:

$$B=\begin{bmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & q & 0 \end{bmatrix} \quad (1)$$

Such an element can also be represented as a vector $r \in R^3$ where:

$$r=\begin{bmatrix} a \\ b \\ c \end{bmatrix} \quad (2)$$

The exponential map representation r represents a rotation of $\theta=\sqrt{a^2+b^2+c^2}$ degrees around axis r. The exponential of a matrix B is defined by Rodrigues' formula:

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

Similarly, methods exist that define how to calculate a determination of the (multi-valued) matrix logarithm. For Example, Gallier and Xu [9] present methods to calculate exponential sand logarithms, also for matrices with higher dimensions.

Although the exponential map representation of orientation does allow for easy manipulation and interpolation, there are singularities in the domain. A log map can map each orientation to an infinite number of points, corresponding to rotations of $2n\pi+\theta$ about axis v and $2n\pi-\theta$ about axis −v for any $n \in N[5]$. Consequently, measures have to be taken to ensure that these singularities do not interfere with the interpolation.

B. Performance Animation

Defining animations as joint rotations and translations as opposed to direct geometry manipulation, already gives a good basis for a real-time animation system. However, a higher level control is needed, since animating all the joints by hand still is a lot of work. Furthermore, it is difficult to create natural motions using only procedural methods, unless a lot of time is invested. An interesting approach is to create new motions using recorded motion data. This approach is called performance animation and its main goal is to try to combine the naturalness of recorded motions with the flexibility of procedural motion synthesis. Although recorded motions are regularly used for facial animation [10], the performance animation technique is the most popular for controlling the body.

There is a broad number of algorithms for body motion synthesis from motion data. All of these approaches are very similar at the basis. Mainly three steps are involved:

1) The first step is the segmentation of motion captured data into separate motion segments. Different researchers use different methods and criteria of segmentation, and it is performed either manually or automatically. Generally, the graph is extended with new transitions, using existing motion segments in the graph.
2) The animation segments are stored in a graph-like structure. Again the connectivity and complexity of the resulting graph depends on the method that is used.
3) Finally, new animations are constructed by traversing a path through the motion graph.

Kovar et al. [11] proposed a technique called Motion Graphs for generating animations and transitions based on a motion database. The main contribution in their technique is the automatic construction of a motion graph from a collection of pre-recorded motion segments. The graph is constructed using two operations: splitting a motion segment, and adding a transition between two motion segments. Where a segment is split depends on whether or not a transition to another motion segment is favorable according to a distance criterion. This work therefore touches an important problem that arises when one wants to compare different animations: what criterion should be used to define the distance between two body postures? Most performance animation techniques use the (weighted) joint-angle distance between two postures as a criterion. Instead of using the joint orientation differences to estimate the pose, Kovar et al. propose another distance metric. They use a point cloud that assumes the shape of the body posture. Additionally, in order to ensure the coordinate system alignment, they propose the calculation of the distance as an minimal cost problem. Although this approach allows for an appropriate distance calculation, it is computationally expensive.

Li et al. [12] propose a method that learns a motion texture from sample data. Using the motion texture, they then generate new motions. This method only synthesizes motions that are similar to the original motion. The system is mostly useful for rhythmic motions with repeating segments, such as dance. Similarly, Kim et al. [13] propose a method of control for motions, by synthesizing motions based on rhythm. Pullen and Bregler [14] 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. Finally, relevant work has been done by Arikan et al. [15, 16] that defines motion graphs based on an annotated motion database. They also present an automatic annotation method that can be used to segment motion capture data automatically. However, since motion captured clips often contain unwanted joint movements, clear constraints have to be defined during the annotation process.

III. BACKGROUND IN INTERACTIVE VIRTUAL HUMANS

There are several research endeavors towards the development of an IVH simulator. The BEAT system [17] allows animators to input typed text that they wish to be spoken by an animated human figure, and to obtain as output speech and behavioral characteristics. Another project at MIT that concerns gesture synthesis is REA [18]. REA (or: the Real Estate Agent) is based on similar principles as the BEAT system. The MAX system, developed by Kopp and Wachsmuth [19], automatically generates face and gesture animations based on an XML specification of the output. This system is also integrated with a dialogue system, allowing users to interact with MAX in a construction scenario. Finally, we would like to mention the work of Hartmann et al. [20], which provides for a system—called GRETA—to automatically generate gestures from conversation transcripts using predefined key-frames.

These systems produce gesture procedurally. As such, the animator has a lot of control on the global motion of the gesture, but the resulting motions tend to look mechanic. This is the effect of an interface problem that exists between high-level gesture specifications and low-level animation. When humans move their arm joints, this also has an influence on other joints in the body. Methods that can calculate dependent joint motions already exist, such as the research done by Pullen and Bregler [14].In their work, they adapt key-framed motions with motion captured data, depending on the specification of which degrees of freedom are to be used as the basis for comparison with motion capture data. However the method they propose is not fully automatic: the animator still needs to define the degrees of freedom of the dependent joints.

Recent work from Stone et al. [21] describes a system that uses motion capture data to produce new gesture animations. The system is based on communicative units that combine both speech and gestures. The existing combinations in the motion capture database are used to construct new animations from new utterances. This method does result in natural-looking animations, but it is only capable of sequencing pre-recorded speech and gesture motions. Many IVH systems use a text-to-speech engine, for which this approach is not suitable. Also, the style and shape of the motions are not directly controllable, contrary to procedural animation methods.

In the following sections, we will present our approach to modeling and animating Interactive Virtual Humans. We will discuss our MIRAnim animation engine, which is capable of handling any type of motion. Then we will show a selection of techniques for realistic motion synthesis from motion capture data, while retaining much of the flexibility of procedural motion synthesis methods.

IV. THE MIRANIM ANIMATION ENGINE

In this section, we will present our animation engine, called MIRAnim [22]. The main architecture of the animation engine is a multi-track approach, where several animation streams need to be blended into a final animation. There has been quite some research in motion blending. Perlin [23] was one of the first to describe a full animation system with blending capabilities based on procedurally generated motions. There are several researchers who have used weight-based general blending to create new animations [24, 25]. There have also been several efforts to apply motion blending not directly on the joint orientation domain. For example, Unuma et al. [26] perform the motion blending in the Fourier domain and Rose et al. [27] used space time optimization to create transitions that minimize joint torque. Kovar et al. [28] use registration curves to perform blending. Their approach automatically determines relationships involving the timing, local coordinate frame, and constraints of the input motions. Blend-based transitions have been incorporated into various systems for graph-based motion synthesis [25，11].

A. Animation Representation

The basic animation representation in our system is based on a Principal Component Analysis (PCA) of existing motion data. 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. We will use the results of the PCA later on for synthesizing the dependent joint motions (see Section IV-D).For our analysis, we perform the PCA on a subset of H-Anim joints. In order to do that, we need to convert each frame of the animation sequences in the data set into an N-dimensional vector.

For representing rotations, we use the exponential map representation [5]. Using the exponential map representation for a joint rotation, a posture consisting of mjoint rotations and a global root translation can be represented by a vector $v \in R^{3m+3}$. In our case, one frame is represented by 25 joint rotations and one root joint translation, resulting in a vector of dimension 78. We have applied a PCA on a large set of motion captured postures, resulting in a PC space of equal dimension.

B. Animation Engine Structure

The goal of our animation engine is to provide for a generic structure that allows for the implementation of different blending strategies. This is especially important, since our animations use different representations, depending on the application. Additionally, in the final system, we will need to perform blending operations on both body and face animations, which are two completely different animation formats that require different blending strategies. The approach that we will present in this Section is suitable for any of the previously discussed blending approaches. A large set of blending tools,

for example time warping, splitting, fading, and so on are available. The advantage of using this generic approach is that once a blending tool has been defined, it can be used for any type of animation, regardless of its structure. In order to be able

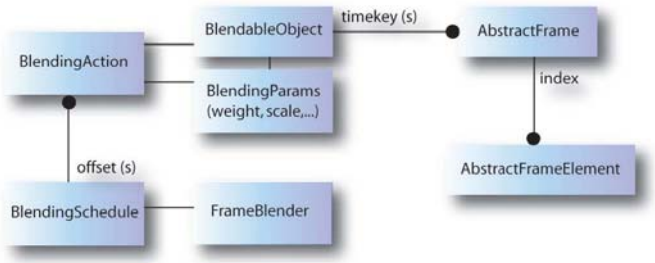

Fig. 2. Overview of the blending engine data structure.

to use these blending tools, only an interface needs to be provided between the data structure used for blending, and the original animation structure. An overview of the blending engine is provided in Fig. 2.

The basic structure used in the blending engine is the so-called Blendable Object interface. Ablendable object is the representation of an animation that can be blended with other animations. A blendable object is defined as a function A:t→ F, where t is a timekey ∈ [ts , te]with 0≤ ts <te < ∞, and F is the corresponding key-frame of the animation. A frame in the blending engine is called an Abstract Frame. An abstract frame consists of a number of elements, called Abstract Frame Element objects. Each of these elements is a list of floating point values. For example, in the case of body animations, an Abstract Frame Element could be a list of 4 floating points, representing a quaternion rotation, or a list of 3 floating points, representing a 3D translation. An abstract frame could then consist of a combination of abstract frame elements that are either translations or rotations. In the case of facial animation, the abstract frame element could be a list of 1 floating point, representing a FAP value in the MPEG-4 standard [29].

In order to provide for a higher flexibility, the blending engine accepts blendable objects with or without a fixed duration. The latter type is especially practical in the case of playing an animation controlled by a motion synthesizer. Since these animations are generated on-the-fly, the duration of the animation may not be known at run-time.

A structure is now required that can take a number of such 'animation tracks' and blend them according to different parameters. The parameters are defined through the BlendingParams interface. The most basic parameter used for blending is a weight value to be used for blending. In addition to a list of weights, the BlendingParams interface also provides for a list of scalings. The evolution of the weights and scalings over time is governed by a parametrizable curve. Fig. 3 shows some examples of curves that can be chosen. Different types of BlendingParams objects can be multiplexed, and custom blending parameter objects can be de-fined. For example, a custom BlendingParams object can be created for body animations, that defines a joint mask. When multiplexed with a weight curve BlendingParams object, this results in a set of curves defined for each joint in the mask. Any arbitrary combination of such blending parameters is possible, allowing for a flexible blending parameterization scheme. Here, also the advantage of the independency of the blending strategy comes forward. Once a blending parameter feature such as curve base

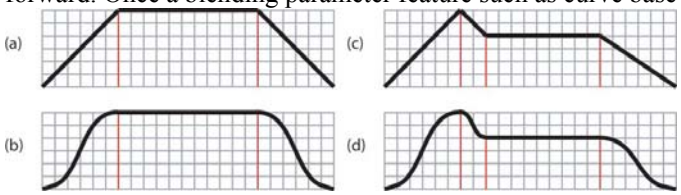

Fig.3. Various basic curve structures areavailable, such as(a)linear or(b) cubicfading, and (c) linear or (d) cubic attack-decay-sustain-release.

-d blending is implemented, it can be used for any type of animation.

The BlendingParams object, together with the Blendable Object, form a BlendingAction object. This object has a flag indicating if it should be rendered outside the timekey domain of the blendable object source. This flag is useful when linking the system with a motion synthesizer, where frames are created during run-time, independent of the current length of the animation.

The final step in obtaining a mix of different BlendingAction objects requires a structure that allows for activating and deactivating different animations according to the blending parameters. This structure is called a BlendingSchedule. A blending schedule consists of a list of BlendingAction objects. Each blending action is associated with a timekey, which defines the time that the blending action should start.

The actual blending itself happens in the Frame Blender object. This object is by default a linear blender, but it can be replaced by a more complex blender, that allows for example blending of other—non-linear—data structures, such as quaternions. This blender can also be replaced if there are different types of frame elements in the same frame, like for example translations (linear) and rotations (non-linear).

The Blending Schedule is again a blendable object. This allows for performing animation blending on different levels, with local blending parameters. When key-frames are obtained from the blending schedule, they are rendered in real-time as a result of the different activated actions and their blending parameters. So, blending actions can be added and removed from the blending schedule during runtime, resulting in a flexible animation blender, adaptable in real-time. In order to optimize performance, a cache is maintained of previously rendered frames.

In addition to the basic data structures and tools used for blending animations, the blending engine also provides for a few extensions that allow to further parameterize the animation and blending process. For example, modifiers can be defined which act as a wrapper around blendable objects. Examples of such modifiers are time stretching, flipping, or looping of animations. Again, custom modifiers can be defined for different animation types. To give an example in the case of body animations: a modifier is available that performs a global transformation on the whole animation. Any sequence of modifiers can be used, since modifiers are again blendable objects.

C. Automatic Idle Motion Synthesis

An important aspect of an animation system is how to deal with a scenario where several animations are played sequentially for various actors. 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. A flexible idle motion generator [30] is required to provide for realistic motions even when no action is planned. In the recorded data, we have observed two important types of idle behaviors:

1) Posture shifts: this kind of idle behavior 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.
2) Continuous small posture variations: because of breath -ing, 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.

1) Balance Shifting: Humans needs to change posture once in a while due to factors such as fatigue. Between these posture changes, he/she is in a resting posture. We can identify different categories of resting postures, such as in the case of standing: balance on the left foot, balance on the right foot or rest on both feet. Given a recording of someone standing, we can extract the animation segments that form the transitions between each of these categories[2]. 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. In order to generate new animations, recorded clips from the data base are blended and modified to ensure a smooth transition. For selecting compatible animation segments, we define a distance criterion as a weighted distance between PC vectors [30]:

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

The weight values wi are chosen as the eigenvalues found during the PCA. Because the PC space is linear, calculating this distance can be done as fast (or faster) as the previously mentioned joint-based methods. However, the use of the PC space has another property that will allow for a significant speedup of the distance calculation: the dimension reduction. Since higher PCs represent lesser occurring body postures, they are mostly 0and therefore they do not contribute significantly to the distance factor. This means that by varying the amount of PCs used, we can look for a reasonable trade-off between speedup and precision.

Once the transitions between the different postures have been calculated, the creation of new animations consists of simply requesting the correct key frame from the database during the animation. This means that the database can be used to control many different virtual humans at the same time. For each virtual human, a different motion program is defined that describes the sequence of animation segments that are to be played. This motion program does not contain any real motion data but only references to transitions in the database. Therefore it can be constructed and updated on-the-fly.

2) Continuous Small Posture Variations: Apart from the balance shifting postures, small variations in posture also greatly improve the realism of animations. 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 randomized 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. The variations can be generated either by applying a Perlin noise function [32] on the PCs or by applying the method that is described in our previous work [30].

D .Automatic Dependent Joint Motion Synthesis

As discussed in Section III, body gesture synthesis sys-tems often generate gestures that are defined as specific arm movements coming from a more conceptual representation of gesture. Examples are: 'raise left arm', 'point at an object', and so on. Translating such higher level specifications of gestures into animations often results in motions that look mechanic, since the motions are only defined for a few joints, whereas in motion captured animations, each joint motion also has an influence on other joints. For example, by moving the head from left to right, some shoulder and spine movements normally occur as well. However, motion captured animations generally do not provide for the flexibility that is required by gesture synthesis systems.

Such systems would greatly benefit from a method that can automatically and in real-time calculate believable movements for the joints that are dependent on the gesture. We will present a method that uses the Principal Components to create more natural looking motions, in real-time [33].

The Principal Components are ordered in such a way that lower PC indices indicate high occurrence in the data and higher PC indices indicate low occurrence in the data. This allows for example to compress animations by only retaining the lower PC indices. Animations that are close to the ones that are in the database that was used for the PCA, will have higher PC indices that are mostly zero (see Fig. 4) for an example. An animation that is very different from what is in the database, will have more noise in the higher PC indices to compensate for

[2] In the current configuration this segmentation is done manually, however automatic segmentation methods also exist [31].

The International Journal of Virtual Reality, 2006, 5(2):15-24

the difference (see Fig. 5).

If one assumes that the database that is used for the PCA is representative for general motions that are expressed by humans during communication, then the higher PC indices represent the part of the animation that is 'unnatural' (or: not frequently occurring in the animation database). When we remove these higher PC indices or apply a scaling filter (such as the one displayed in Fig. 6), this generates an error in the final animation. However, since the scaling filter removes the 'unnatural' part of the animation, the result is a motion that actually contains the movements of dependent joints. By varying the PC index where the scaling filter starts, one can define how close the resulting animation should be to the original key-framed animation.

To calculate the motions of dependent joints, only a scaling function has to be applied. Therefore this method is very well suited for real-time applications. A disadvantage is that when

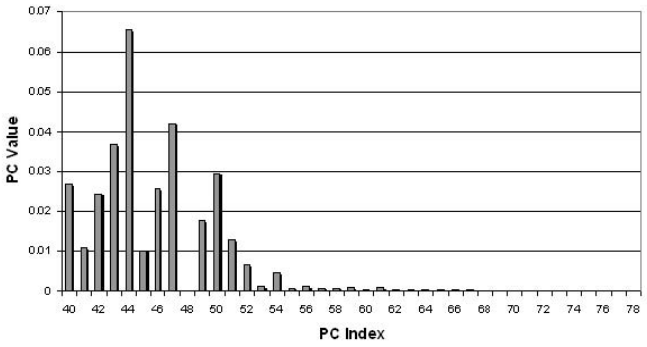

Fig.4.(Absolute)PC values of a posture extracted from amotion captured animation sequence.

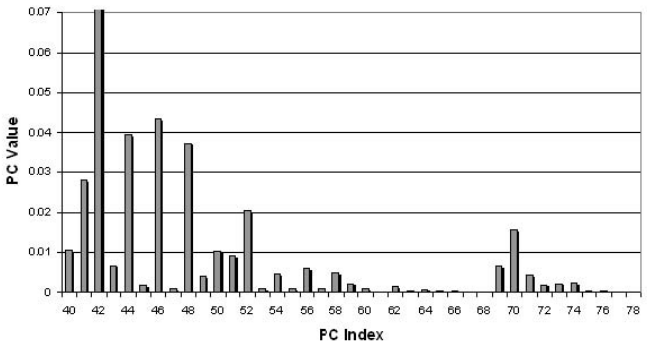

Fig. 5. (Absolute) PC values of a posture modelled by hand for a few joints.

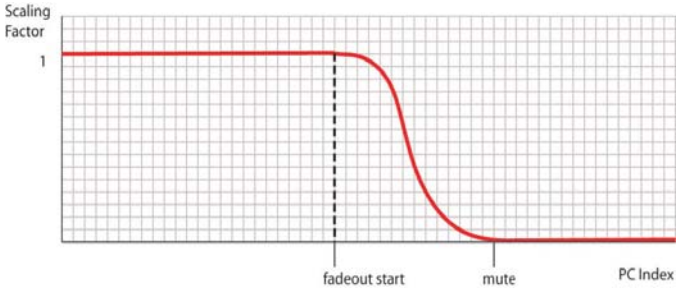

Fig.6. An example of a scaling filter that can be applied to the PC vector representation of a posture.

applying the scaling function onto the global PC vectors, translation problems can occur. In order to eliminate these translation artefacts, we have also performed a PCA on the upper bodyjoints only (which does not contain the root joint translation). The scaling filter is then only applied on the upper body PC vector. This solution works very well since in our case, the dependent joint movements are calculated for upper body gestures only, whereas the rest of the body is animated using the idle motion engine. Fig.7 shows some examples of original frames versus frames where the PC scaling filter was applied.

V. INTERACTIVE VIRTUAL HUMANS

In many Interactive Virtual Humans systems, a dialogue manager generates responses that define the desired speech and face/body movement of the character on a high level. Our system produces output phrases that are tagged with XML. These tags indicate where a gesture should start and end. There are many different representation languages for multimodal

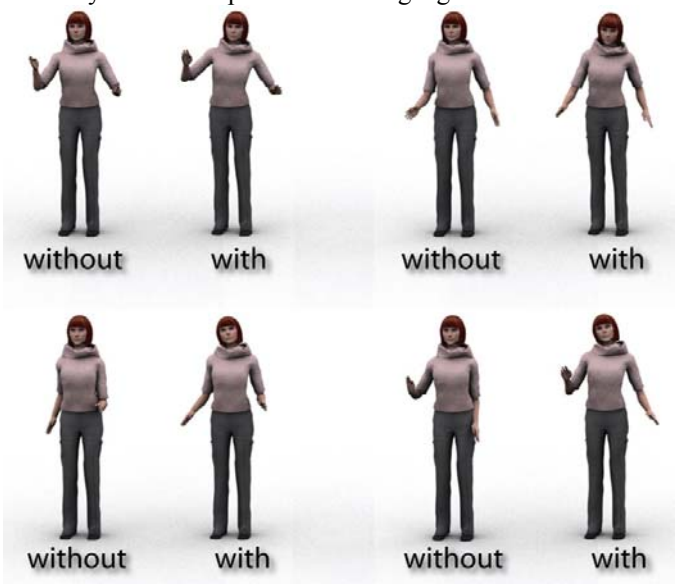

Fig.7. Someexamplesofkeyframe postures designed forafew joints and the same postures after application of the PC scaling filter. See Color Plate 6

content, for example the Rich Representation Language (RRL) [34] or the Virtual Human Mark-up Language (VHML)[35]. In this section, we will give an example of how such a representation language can be used to control gesture sequences in our system .For testing purposes, we have defined a simple tag structure that allows for the synchronized playback of speech and non-verbal behavior. An example of a tagged sentence looks like this:

<begin_gestureid="g1"anim="shake head"/>Unfortunately, I have <begin_gestureid="g2"anim="raise shoulders"/> no idea <end_gestureid="g2"/> what you are talking about .<end_gestureid="g1"/>

Within each gesture tag, an animation ID is provided. When the gesture animation is created, these animations are loaded from a database of gestures—also called a Gesticon [34] — and they are blended using the previously described blending

engine. The timing information is obtained from the text-to-speech system. Although this is a very rudimentary system, we believe that this way of generating gestures can easily be replaced with another, more elaborate gesture synthesizer, since the animation system is completely independent of what happens on the gesture construction level. The animation system only activates actions at given times with specified animation lengths and

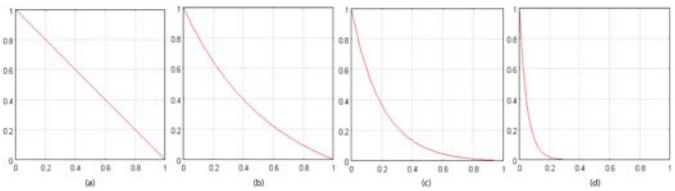

Fig. 8. Coarticulation base function with different α values: (a) α =0(b) α =2(c) α =5and (d) α =10.

blending parameters. Although our current testing system only generates gestures in synchrony with speech, this is not a limitation of the animation system. The animation system is capable of handling any set of actions at any time, even without speech.

A. Creating Facial Animation

In this section, we will shortly explain the techniques used to create the facial animation from the output text and speech. The output text is first converted into a speech signal by the text-to-speech engine. At the basic level, speech consists of different phonemes. These phonemes can be used to generate the accompanying face motions, since every phoneme corresponds to a different lip position. The lip positions related to the phonemes are called visemes. There are not as many visemes as phonemes, because some phonemes revert to the same mouth position. For example, the Microsoft Speech SDK defines 49 phonemes, but only 21 different visemes.

For each viseme, the mouth position is designed using the MPEG-4Face Animation Parameters (FAPs). Constructing the facial motion is achieved by sequencing the different mouth position, taking into account the speech timing obtained from the TTS engine. An important issue to take into consideration when creating facial speech is coarticulation, or: the overlapping of phonemes/visemes. Generally, coarticulation is handled by defining a dominance function for each viseme. For example, Cohen and Massaro [36] use this technique and they define an exponential dominance function. Similarly, we use the following base function to construct the coarticulation curve:

$$f(x) = e^{-\alpha x} - x \cdot e^{-\alpha} \qquad (5)$$

where $0 < \alpha < \infty$. The parameter α governs the shape of the curve. Fig.8 shows the curve for different values of α . Using this base function, the final coarticulation dominance function is defined as follows:

$$C_\alpha(x) = e^{-\alpha 2|x-0.5|} - 2|x-0.5| \cdot e^{-\alpha} \qquad (6)$$

Two different examples of the $C_\alpha(x)$ dominance function are given in Fig. 9. For each viseme, the value of the α parameter can be chosen. Also, the weight of each function, as well as its spread (overlap) can be defined for each viseme.

Because of the generic structure of the MIRAnim engine (see Section IV), it is a simple task to create the facial animation from the viseme timing information. We define an area. By

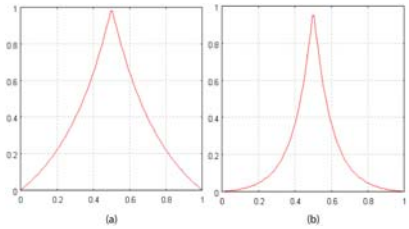

Fig. 9. Example of complete coarticulation functions with different α parameters: (a) α =2and (b) α =5.

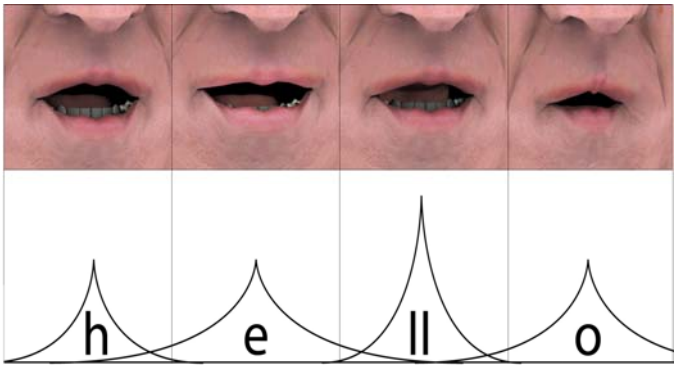

Fig.10. Facial animation for hello that takes tongue movement into account. See Color Plate 7

blending the different facial animation tracks, the final animation is obtained.

B. Creating Body Animation

Very similar to facial animation synthesis, the body animation is also partly generated from the tagged text. The same definition is employed for the body animation tags. An example of a body animation in combination with speech is given as follows:

<begin_gesture id="g1" anim="hello"/>Hello,
<end_gesture id="g1"/> how are you doing today?

Similar to the facial animation synthesis, the TTS timing information is used to plan the length of the gesture motions. A blending fade-in and fade-out is applied on the gesture motions in order to avoid unnatural transitions. Also, the automatic dependent joint motion filter explained in Section IV-D is applied on the gesture signal. The filter is applied on the upper body and starts fading out at PC index 20 with a mute for PCs >30(of a total of 48).

Next to the gesture motions, the idle motion engine is

The International Journal of Virtual Reality, 2006, 5(2):15-24

running continuously, therefore providing the IVH with continuous idle motions. In order to obtain the final animation, the resulting idle motions and the gesture motions with dependent joint movements are blended on-the-fly by the MIRAnim engine. Fig. 11 shows some examples of full body postures obtained using our approach.

VI. CONCLUSION

We have presented MIRAnim, a robust and flexible animation engine for controlling both the face and body of a virtual character. We have shown an efficient animation repre -sentation, in combination with a set of techniques for improving the realism of gesture motions, by applying a scaling filter as well as a blending operation with full body postures obtained from motion capture data. We have shown how the animation engine is controlled in order to produce natural face and body motions in real-time. The range of possible applications for our system is not limited to Interactive Virtual Human simulation. The generic animation system could for example also be employed to control avatars. Most of the techniques presented in the paper can separately improve the efficiency of various animation tasks that needed to be done manually beforehand.

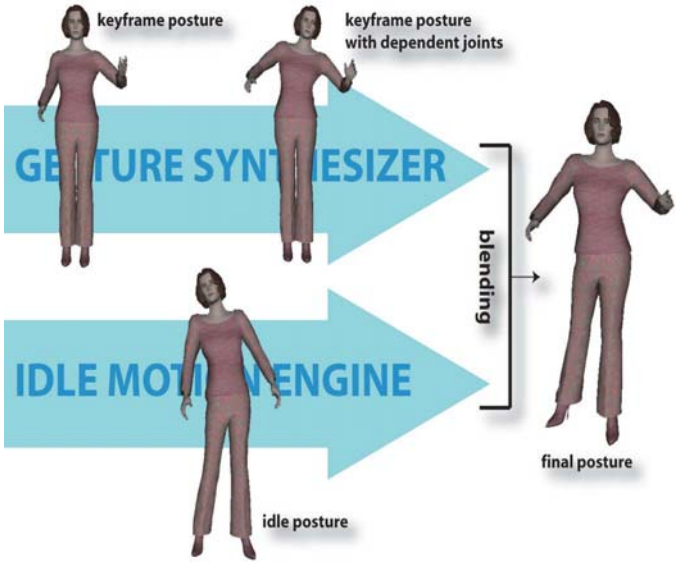

Fig.11. Integration of gesture animations, dependent joint motion synthesis and idle motion synthesis. See Color Plate 8

In the current implementation of our system, the output animation sequence is generated from tagged text. Although such a method is sufficient for testing the animation engine that control the virtual human motion, it is a rather time-consuming job if a complex interactive script needs to be developed. In order to allow for IVHs which are easy to develop, a system is required to do this automatically.

Although blending between different types of motions in our system is very easy to perform, the system does not rely on fixed physical parameters. As such, some foot skating or collisions can occur. We have proposed a few simple approximate solutions for these problems, but the interactive system does not yet have a complete control over the body. More complex actions, such as running, ducking, jumping or sitting down, are very difficult to control from tagged text. A more generic action selection mechanism needs to be developed that can control such types of motions, but that is still flexible enough.

Most virtual humans are 3D geometrical models, whose motion is controlled by manipulating a skeleton. The 'inside' of these virtual humans is empty. A real challenge would be to model this 'inside' part of a virtual human. Human beings adapt their behavior according to many different physiological signals. These signals do not only affect the behavior of humans, but it also affects our appearance, for example blushing, sweating, crying, and so on. These physiological processes form a key part of our expressions and behavioral system, but there is almost no research addressed to modeling these signals as an integral part of virtual human behavior. Additionally, more basic motivators such as hunger or sleepiness could be implemented. As a final result, a virtual human will not cease to exist when an application is terminated, but he/she will go to sleep or do other things, which in turn would inspire new conversations with the user when the application is launched again.

ACKNOWLEDGMENT

This research has been supported through the EU funded project HUMAINE(IST-507422).We would also like to thank Nedjma Cadi and Lionel Egger for designing the characters and the animations.

REFERENCES

[1] G. Dudek and M. Jenkin. Computational Principles of Mobile Robotics, *Cambridge University Press*, 2000.
[2] "Vicon motion systems," http://www.vicon.com/, accessed April 2006.
[3] "Motionstar," http://www.ascension-tech.com/, accessed May 2006.
[4] "H-Animspecification for a standard humanoid," http://www.h-anim.org/, accessed May 2006.
[5] F.S.Grassia. Practical parameterization of rotations using the exponen-tial map, *Journal of GraphicsTools*, vol. 3, no. 3, pp. 29–48, 1998.
[6] M.Alexa.Linear combination of transformations, in *Proceedings SIGGRAPH 2002*. ACM Press, pp. 380–387, 2002.
[7] C. Chevalley. Theory of Lie Groups, New York: Princeton University Press, 1946.
[8] F. C. Park, B. Ravani. Smooth invariant interpolation of rotations, ACMTransactions on Graphics, vol. 16, no. 3, pp. 277–295, July 1997.
[9] J. Gallier, D. Xu. Computing exponentials of skew-symmetric ma-trices and logarithms of orthogonal matrices. *International Journal of Robotics and Automation*, vol. 18, no. 1, pp. 10–20, 2003.
[10] S. K shirsagar, T. Molet, N. Magnenat-Thalmann. Principal components of expressive speech animation, in *Computer Graphics* International 2001. IEEE Computer Society, pp. 38–44, February, 2001.
[11] L. Kovar, M. Gleicher, F. Pighin. Motion graphs. in *Proceedings SIGGRAPH 2002*, ACM Press, pp. 473–482, 2002.
[12] Y. Li, T.Wang, H.Y. Shum. Motion texture:A two-level statistical model for character motion synthesis, in *Proceedings SIGGRAPH 2002*, ACM Press, pp. 465–472, 2002.
[13] T. H. Kim, S. I.Park, S.Y. Shin. Rhythmic-motion synthesis based on motion-beat analysis, *ACMTransactions on Graphics*, vol. 22, no. 3, pp. 392–401, 2003.
[14] K. Pullen, C. Bregler. Motion capture assisted animation: Texturing and synthesis, in *Proceedings SIGGRAPH 2002*. ACM Press, pp. 501– 508, 2002.
[15] O. Arikan, D. A. Forsyth. Interactive motion generation from examples, in *Proceedings SIGGRAPH 2002*. ACM Press, pp. 483–490, 2002.
[16] O.Arikan, D. A. Forsyth, J.F. O'Brien. Motion synthesis from annotations, *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 392–401, 2003.

[17] J.Cassell, H.Vilhj´almsson, T.Bickmore. BEAT: the expression animation toolkit. in *Proceedings SIGGRAPH 2001*. ACM Press, pp. 477–486, 2001.

[18] J.Cassell, T.Bickmore, M.Billinghurst, L.Campbell, K.Chang, H.Vilhj´almsson and H.Yan. Embodiment in conversational interfaces: Rea, in *Proceedings of the CHI'99 Conference*, pp. 520–527, 1999.

[19] S. Kopp, I. Wachsmuth. Synthesizing multimodal utterances for conversational agents, *Computer Animation and Virtual Worlds*, vol. 15, no. 1, pp. 39–52, 2004.

[20] B. Hartmann, M. Mancini and C. Pelachaud. Implementing expressive gesture synthesis for embodied conversational agents, in *Gesture in Human-Computer Interaction and Simulation: 6th International Gesture Workshop*, ser. Lecture Notes, in Computer Science. Berlin, Germany: Springer Verlag, pp. 188–199, May 2005.

[21] M. Stone, D. DeCarlo, I. Oh, C. Rodriguez, A. Stere, A. Lees and C. Bregler. Speaking with hands: Creating animated conversational characters from recordings of human performance, in *Proceedings SIGGRAPH 2004*. ACM Press, pp. 506–513, 2004.

[22] A. Egges, G. Papagiannakis and N. Magnenat-Thalmann. An interactive mixed reality framework for virtual humans, in *International Conference on Cyber worlds 2006*. IEEE Computer Society, to appear, Nov. 2006.

[23] K. Perlin. Real time responsive animation with personality, *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, no. 1, pp. 5–15, 1995.

[24] D. Wiley , J. Hahn. Interpolation synthesis of articulated figure motion, *IEEE Computer Graphics and Applications*, vol. 17, no. 6, pp. 39–45, 1997.

[25] C.Rose, M.Cohen and B.Bodenheimer. Verbs and adverbs: Multi -dimensional motion interpolation, *IEEE Computer Graphics and Applications*, vol. 18, no. 5, pp. 32–40, September/October 1998.

[26] M. Unuma, K. Anjyo, T. Tekeuchi. Fourier principles for emotion-based human figure animation, in *Proceedings SIGGRAPH 1995*. ACM Press, pp. 91–96, 1995.

[27] C. Rose, B. Guenter, B. Bodenheimer and M. Cohen. Efficient generation of motion transitions using spacetime constraints, in *Proceedings SIGGRAPH 1996*. ACM Press, August pp. 147–154, 1996.

[28] L. Kovarand M. Gleicher. Flexible automatic motion blending with registration curves, in *Proceedings Symposium on Computer Animation (SCA) 2003*. Eurographics Association, pp. 214–224, 2003.

[29] S. Garchery, R. Boulic, T. Capin. and P. Kalra. Standards for virtual humans, in *Handbook of Virtual Humans*, N. Magnenat-Thalmann and D. Thalmann, Eds. Hoboken, NJ, USA: John Wileyand Sons, ch. 16, pp. 373–391, Aug. 2004.

[30] A. Egges,T. Molet, N. Magnenat-Thalmann. Personalised real-time idle motion synthesis, in *Pacific Graphics 2004*, pp. 121–130, 2004.

[31] M. Mueller, T. Roeder, and M. Clausen. Efficient content-based retrieval of motion capture data, in *Proceedings SIGGRAPH 2005*, 2005.

[32] K. Perlin. Animage synthesizer, in *Proceedings SIGGRAPH 1985*. ACM Press, pp. 287–296, 1985.

[33] A. Egges and N. Magnenat-Thalmann. Emotional communicative body animation for multiple characters, in *First International Workshop on Crowd Simulation (V-Crowds)*. IEEE Computer Society, pp. 31–40, 2005.

[34] B. Krenn, H. Pirker. Defining the gesticon: Language and gesture coordination for interacting embodied agents, in *Proceedings of the AISB-2004 Symposium on Language, Speech and Gesture for Expressive Characters*, University of Leeds, UK, pp. 107–115, 2004.

[35] "Virtual human markup language (VHML)," http://www.vhml.org/, accessed November 2005.

[36] M. M. Cohen, D. W. Massaro. Modeling co articulation in synthetic visual speech. in *Models and Techniques in Computer Animation*, N. Magnenat-Thalmann and D. Thalmann, Eds. Berlin, Germany: Springer Verlag, pp. 139–156, 1993.

Nadia Magnenat-Thalmann has pioneered research into virtual humans over the last 25 years. She obtained several Bachelor's and Master's degrees in various disciplines (Psychology, Biology and Chemistry) and a PhD in Quantum Physics from the University of Geneva. From 1977 to 1989, she was a Professor at the University of Montreal where she founded the research lab MIRALab. She was elected Woman of the Year in the Grand Montreal for her pioneering work on virtual humans and her work was presented at the Modern Art Museum of New York in 1988.She moved to the University of Genevain1989, where she founded the Swiss MIRALab, an international interdisciplinary lab composed of about 30 researchers. She is author and coauthor of a very high number of research papers and books in the field of modeling virtual humans, interacting with them and living in augmented life. She has received several scientific and artistic awards for her work, mainly on the Virtual Marylin and the film RENDEZ-VOUS A MONTREAL, but more recently, in 1997, she has been elected to the Swiss Academy of Technical Sciences, and has been nominated as a Swiss personality who has contributed to the advance of science in the 150 years history CD-ROM produced by the Swiss Confederation Parliament. She has directed and produced several films and real-time mixed reality shows, among the latest are the UTOPIANS (2001), DREAMS OFAMANNEQUIN (2003) and THEAUGMENTED LIFE IN POMPEII (2004). She is editor-in-chief of the Visual Computer Journal published by Springer Verlag.

Arjan Egges has finished his Master's thesis in 2001 at the University of Twente, the Nether-lands on Artificial Intelligence, Agent Theory and Dialogue Modeling. He recently has obtained his PhD at MIRALab -University of Geneva, on the subject of Real-time Animation of Interactive Virtual Humans. His main interests involve the modeling of intelligent behavior, non-verbal communication, such as gestures and facial animation, real-time body animation, and personality and emotion modeling.