Visual Comput (2007) 23: 317–333
DOI 10.1007/s00371-007-0113-z

ORIGINAL ARTICLE

Arjan Egges
George Papagiannakis
Nadia Magnenat-Thalmann

Presence and interaction in mixed reality environments

Published online: 28 March 2007

A. Egges (✉)
Center for Advanced Gaming and Simulation, Department of Information and Computing Sciences
Utrecht University
The Netherlands
egges@cs.uu.nl

G. Papagiannakis · N. Magnenat-Thalmann
MIRALab – University of Geneva
Switzerland
{george, thalmann}@miralab.unige.ch

Abstract In this paper, we present a simple and robust mixed reality (MR) framework that allows for real-time interaction with virtual humans in mixed reality environments under consistent illumination. We will look at three crucial parts of this system: interaction, animation and global illumination of virtual humans for an integrated and enhanced presence. The interaction system comprises of a dialogue module, which is interfaced with a speech recognition and synthesis system. Next to speech output, the dialogue system generates face and body motions, which are in turn managed by the virtual human animation layer. Our fast animation engine can handle various types of motions, such as normal key-frame animations, or motions that are generated on-the-fly by adapting previously recorded clips. Real-time idle motions are an example of the latter category. All these different motions are generated and blended on-line, resulting in a flexible and realistic animation. Our robust rendering method operates in accordance with the previous animation layer, based on an extended for virtual humans precomputed radiance transfer (PRT) illumination model, resulting in a realistic rendition of such interactive virtual characters in mixed reality environments. Finally, we present a scenario that illustrates the interplay and application of our methods, glued under a unique framework for presence and interaction in MR.

Keywords Presence · Interaction · Animation · Real-time rendering · Mixed reality

1 Introduction

Over the last few years, many different systems have been developed to simulate scenarios with interactive virtual humans in virtual environments in real-time achieving various degrees of presence. The control of these virtual humans in such a system is a widely researched area, where many different types of problems are addressed, related to animation, speech, deformation, and interaction, to name a few research topics. The scope of applications for such systems is vast, ranging from virtual training or cultural heritage to virtual rehabilitation. Although there is a variety of systems available with many different features, we are still a long way from a completely integrated system that is adaptable for many types of applications. This is not only because of the amount of effort that is required to integrate different pieces of software, but also because of the real-time constraint. The latter especially becomes an issue when many different components need to work together aiming at a higher degree of presence in a mixed reality environment.

The true challenge of such systems is for a person to be able to feel both present and naturally *interact* with the virtual humans in the virtual, as well as the real scene. A lot of research has been done to develop chatbots that are

mostly focused on a stand-alone speech application with a cartoon face [1]; however, only a few systems succeed to link interaction with controlling 3D face and/or body motions played in synchrony with text-to-speech [5, 14, 18]. The main problem with such systems is that they are far from ready to be used in mixed reality applications, which are much more demanding than a stand-alone application. For example, a highly flexible animation engine is required that not only plays animations in combination with interaction, but that is also capable of playing simple key-frame animations as part of a predefined scenario. This also means that a dynamic mechanism is required that allows to switch between playing key-frame animations as part of a scenario, and animations related to the interaction (such as body gestures and facial expressions) without interrupting the animation cycle. Furthermore, mixed realities require a more elaborate rendering method for the virtual humans in the scene, which uses the global illumination information, so that virtual humans that augment a reality have lighting conditions that are consistent with the real environment. This is another challenge that this work examines, since consistent rendering in MR contributes to both feeling of presence as well as realism of interaction in the case of virtual characters in MR scenes. To the best knowledge of the authors – and as it is highlighted in Sect. 2 – there are no such systems appearing in the bibliography up to date.

In this paper, we propose simple and fast methods for three important components (interaction, animation and rendering), that elevate some of the presence issues discussed above. Our approaches are specifically tailored to work as components of a mixed reality real-time application. Our fully integrated system, includes speech recognition, speech synthesis, interaction, emotion and personality simulation, real-time face and body animation and synthesis, real-time camera tracking for AR and real-time virtual human precomputed radiance transfer (PRT) rendering, and is able to run at acceptable speeds for real-time (2030 fps) on a normal PC. As part of the work presented in this paper, we will show the system running different scenarios and interactions in MR applications.

2 Background

A growing number of projects are currently based on AR integrated platforms, exploring a variety of presence applications in different domains such as cultural heritage [26, 27, 42], training and maintenance [41] and edutainment-games [37, 38]. Special focus has recently been applied to system design and architecture in order to provide the various AR enabling technologies a framework for proper collaboration and interplay. Azuma [3] describes an extensive bibliography on current state-of-the-art AR systems and frameworks. However, few of these systems take the modern approach that a realistic mixed reality application, rich in AR virtual character experiences, should be based on a complete VR framework (featuring game engine-like components) with the addition of AR enabling technologies like real-time camera tracking, AR displays and interfaces, and registration and calibration. Virtual characters were also used in the MR-Project [37], where a complete framework for mixed reality applications has been created. Apart from the custom tracking/rendering modules, a specialized video and see-through HMD have been devised.

An important issue in adding virtual content to real scenery is the application of proper lighting and shading techniques on the virtual objects. Only if the environmental properties of both the virtual and real worlds match is a seamless transition between real and virtual content possible. Ren et al. [33] recently presented a fast method for real-time soft shadows in dynamic scenes illuminated by large, low-frequency light sources. This work is the closest to our goals as it is applicable to virtual characters in real-time VR. However, it is unclear how the transfer from multiple objects can be combined in the same hierarchy as in the case of multi-segmented VHs of H-Anim [13] and introduced in a mobile AR framework (performance critical). Nevertheless this would be the most appropriate alternative to our approach on evaluating the hypotheses of high-fidelity illumination registration of VH in MR. We believe that as PRT methods [17, 36] can produce the most realistic and physically principled real-time GI effects up to date, they are ideal for VHs as they allow a wealth of GI effects to be simulated: area lights, self-shadowing, interreflections, BSSRDF with subsurface scattering, all important elements for realistic depiction of VHs.

Mixing such aesthetic ambiences with virtual character augmentations [6] and adding dramatic tension has developed very recently these narrative patterns into an exciting new edutainment medium. Balcisoy [4] also presented a novel system (one of the first in the bibliography) to present interactive virtual humans in AR (game of checkers). In these systems, the animation of the humanoid is generally based on scripted animations. However, a truly *interactive* virtual human requires a high level of control over the body postures and gestures. There are several research efforts that try to control such types of character movements. The Greta system is an embodied conversational agent (ECA) simulator that includes a dialogue system, emotions and a facial expression synthesizer [30]. Hartmann et al.[14] present an extension of the Greta system that automatically generates hand and arm gestures from conversation transcripts using predefined key-frames. Another well-known system that can produce gesture animations from text, is BEAT [5]. BEAT allows animators to input typed text that they wish to be spoken by an animated human figure, and to obtain as output speech and behavioural characteristics. The MAX system, developed by Kopp and Wachsmuth[18], automat-

ically generates face and body (gesture) animations based on an XML specification of the output.

Most of these systems use procedural methods to generate the gesture motions, resulting in rather stiff motions. In order to overcome this problem, recent work starts to integrate motion capture-based techniques to generate new motions while still retaining the flexibility of procedural motion synthesizers [10]. This approach synthesizes full body motions as a blended combination of *idle motions* and *gesture motions*. The idle motions provide for a continuous motion for any character of any required length. These motions are generated by automatically adapting prerecorded motion segments, based on a technique similar to motion graphs [20].

In combination with these idle motions, gesture motions are blended in. We have developed a method for automatic dependent joint motion synthesis is real-time [9]. By using this method, existing gesture synthesizers, such as the previously mentioned systems, can be used to produce the basic gesture tracks for a few joints, and our method will automatically add the motions of dependent joints (such as head or spine joints).

Currently, no AR system exists that can handle such kinds of complex full body motions, interaction, speech recognition/synthesis, illumination registration, geometrical registration, skin deformation, facial animation, and more, all in a mobile setup. In this paper, we will present such a system, building on a previously developed framework called VHD++ [31].

In Sect. 3, we will give an overview of that system. Then, we will describe how this system was extended with animation and interaction capabilities in Sects. 4 and 5. Section 6 will show our global illumination model that is used to light the scene. Finally, in Sect. 7 we will show two different scenarios that our system can successfully handle: one in VR and one in AR.

3 VHD++

Our MR-rendering, animation and interaction system is incorporated in the VHD++ [31] component-based framework engine developed by VRLab and MIRALab. This framework allows quick prototyping of VR-AR applications featuring integrated real-time virtual character simulation technologies, as depicted in Fig. 1 and the OpenSceneGraph [25] real-time scenegraph rendering API. The key innovation of VHD++ is focused in the area of component-based framework that allows the plug-and-play of different heterogeneous human simulation technologies such as: real-time character rendering in AR (supporting real-virtual occlusions), real-time camera tracking, facial simulation and speech, body animation with skinning, 3D sound, cloth simulation and behavioural scripting of actions. The different components may be grouped into the two following main categories:

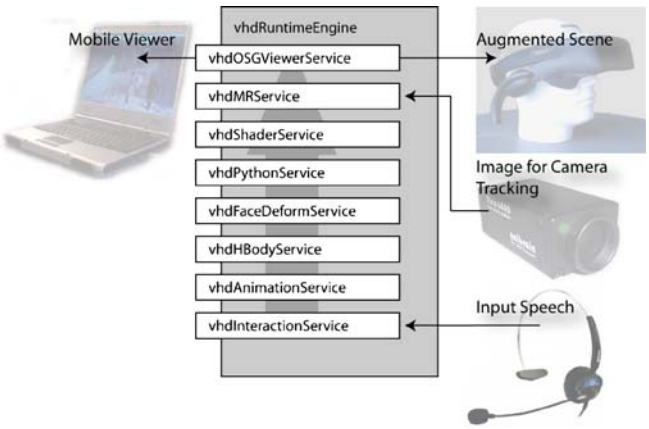

Fig. 1. Overview of the VHD++ MR framework

- System kernel components responsible for the interactive real-time simulation initialization and execution
- Interaction components driving external VR devices and providing various GUIs allowing for interactive scenario authoring, triggering and control

Finally the content to be created and used by the system was specified, which may be classified into the two following main categories: (a) static and (b) dynamic content building blocks such as shader nodes encapsulating rendering effects, models of the 3D scenes, virtual humans, objects, animations, behaviours, speech, sounds, python scripts, etc.

The software architecture is composed of multiple software components called services, as their responsibilities are clearly defined. They have to take care of rendering of 3D simulation scenes and sound, processing inputs from the external VR devices, animation of the 3D models and in particular complex animation of virtual human models including skeleton animation and respective skin and cloth deformation. They are also responsible for maintenance of the consistent simulation and interactive scenario state that can be modified with python scripts at run-time. To keep good performance, the system utilizes five threads. One thread is used to manage the updates of all the services that we need to compute, such as human animation, cloth simulation or voice (sound) management. A second thread is used for the 3D renderer, who obtains information from the current scenegraph about the objects that must be drawn as well as the image received from the camera. It will change the model view matrix accordingly to the value provided by the tracker. The third thread has the responsibility of capturing and tracking images. The fourth thread manages the update process of the interaction system, by pruning the script to see whether or not an action needs to be taken by the virtual human. The last thread is the python interpreter, which allows us to create scripts for manipulating our application at the

system level, such as generating scenario-based scripted behaviours for the human actions (key-frame animation, voice, navigation combined to form virtual short plays).

The AR system presented in Fig. 1 features immersive real-time interactive simulation supplied with proper information in the course of the simulation. That is why content components are very diversified and thus their development is an extremely laborious process involving long and complex data processing pipelines, multiple recording technologies, various design tools and custom-made software. The various 3D models to be included in the virtual environments like virtual humans or auxiliary objects have to be created manually by 3D digital artists. The creation of virtual humans requires the recording of motion captured data for realistic skeletal animations as well as a database of gestures for controlling face and body animations. Sound environments, including voice acting, need to be recorded in advance based on the story-board. For each particular scenario, dedicated system configuration data specifying system operational parameters, parameters of the physical environment and parameters of the VR devices used have to be defined as well as scripts defining behaviours of simulation elements, in particular virtual humans. These scripts can modify any data in use by the current simulation in real-time. This allows us to continue running the simulation whilst some modifications are performed. Finally our rendering algorithms are supplied in the form of shader effect nodes in the 3D rendering kernel, thus allowing for real-time application in multiple rendering hierarchies.

For animation and interaction, we have developed two services. The *vhdAnimationService* contains all the animation related components: motion synthesis, blending, loading and saving animations and so on. The *vhdInteractionService* uses the animation service to control a virtual character. The interaction service includes speech recognition and synthesis, a dialogue manager and emotion/personality simulation. For the global illumination and rendering, an *vhdOSGShaderService* has been developed, that implements the PRT shading approach. In the next sections, we will give an overview of each of these services.

4 The MIRAnim animation engine

In this section, we will present our animation engine, called MIRAnim. 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 [29] 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 [34, 43]. There have also been several efforts to apply motion blending not directly on the joint orientation domain. For example, Unuma et al. [39] perform the motion blending in the Fourier domain and Rose et al. [35] used spacetime optimization to create transitions that minimize joint torque. Kovar et al. [19] 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 [20, 34].

4.1 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 Sect. 4.4). 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 [12]. Using the exponential map representation 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, 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.

4.2 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

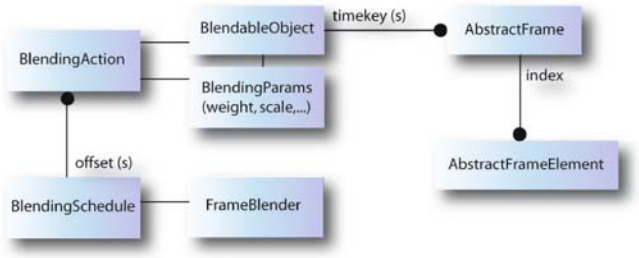

Fig. 2. Overview of the blending engine data structure

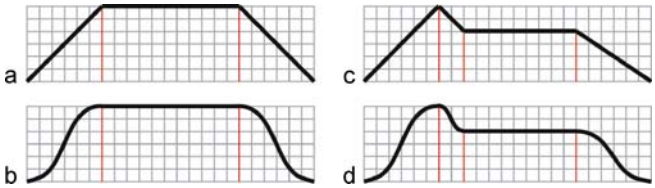

Fig. 3a–d. Various basic curve structures are available, such as **a** linear or **b** cubic fading, and **c** linear or **d** cubic attack-decay-sustain-release

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 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 `BlendableObject` interface. A blendable object is the representation of an animation that can be blended with other animations. A blendable object is defined as a function $A : t \rightarrow F$, where t is a timekey $\in [t_s, t_e]$ with $0 \leq t_s < t_e < \infty$, and F is the corresponding key-frame of the animation. A frame in the blending engine is called an `AbstractFrame`. An abstract frame consists of a number of elements, called `AbstractFrameElement` objects. Each of these elements is a list of floating point values. For example, in the case of body animations, an `AbstractFrameElement` 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 one floating point, representing a FAP value in the MPEG-4 standard [11].

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. Figure 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 defined. For example, a custom `BlendingParams` object can be created for body animations, which 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-based blending is implemented, it can be used for any type of animation.

The `BlendingParams` object, together with the `BlendableObject`, 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 `FrameBlender` 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 `BlendingSchedule` is again a blendable object. This allows for performing animation blending on different levels, with local blending parameters. When keyframes 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 run-time, 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.

4.3 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 [10] 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 behaviour:

1. Posture shifts: this kind of idle behaviour concerns the shifting from one resting posture to another one, for example, shifting balance while standing, or going to a different lying or sitting position.
2. 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.

4.3.1 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.[1] 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 database are blended and modified to ensure a smooth transition.

[1] In the current configuration this segmentation is done manually; however, automatic segmentation methods also exist [24].

For selecting compatible animation segments, we define a distance criterion as a weighted distance between PC vectors [10]:

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

The weight values w_i 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 0 and 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.

4.3.2 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 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. The variations can be generated either by applying a Perlin noise function [28] on the PCs or by applying the method that is described in our previous work [10].

4.4 Automatic dependent joint motion synthesis

As discussed in Sect. 2, body gesture synthesis systems 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 [9].

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 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

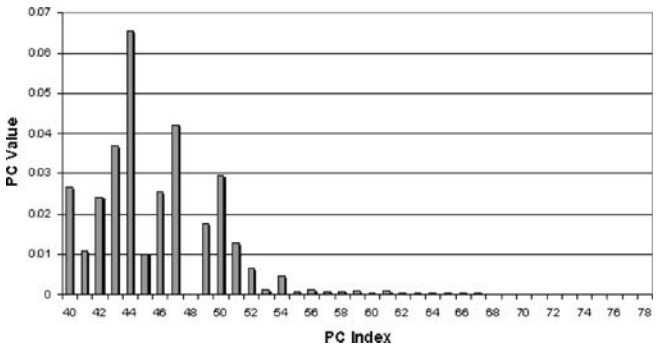

Fig. 4. (Absolute) PC values of a posture extracted from a motion captured animation sequence

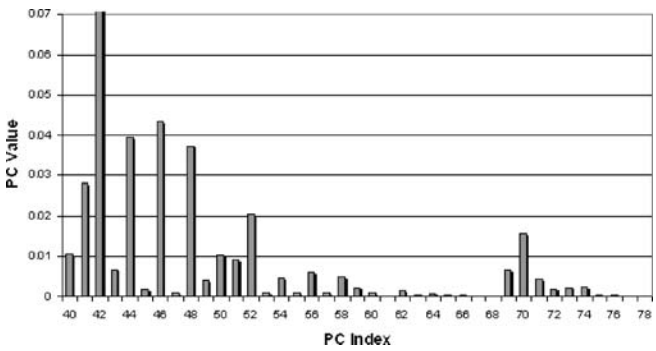

Fig. 5. (Absolute) PC values of a posture modelled by hand for a few joints

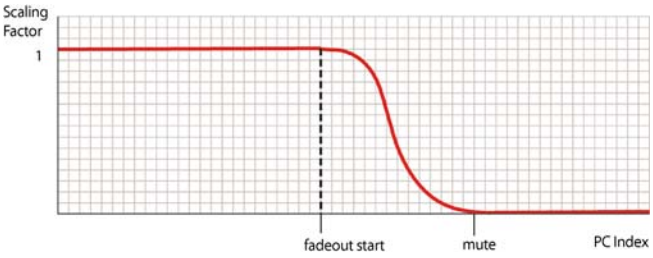

Fig. 6. An example of a scaling filter that can be applied to the PC vector representation of a posture

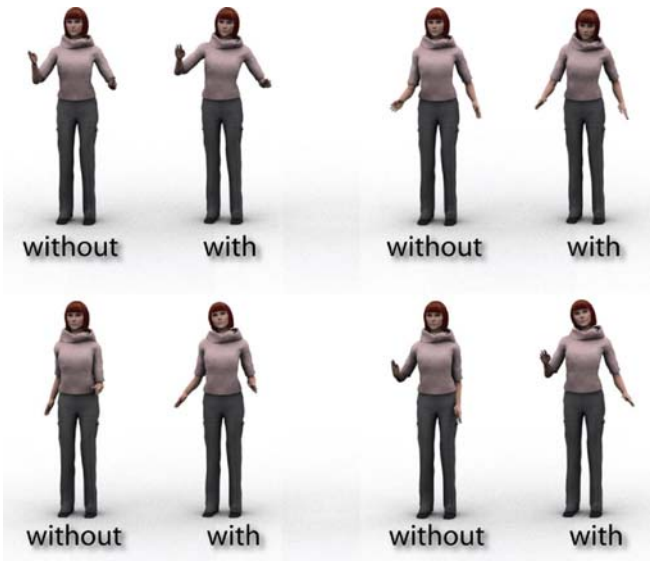

Fig. 7. Some examples of key frame postures designed for a few joints and the same postures after application of the PC scaling filter

is very well-suited for real-time applications. A disadvantage is that when 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 body joints 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. Figure 7 shows some examples of original frames versus frames where the PC scaling filter was applied.

4.5 Animation service

The animation service is built around the blending engine, and it contains specific implementations for controlling both face and body animation. The service can blend several different types of motions, including real-time idle motions and key-frame animations. For each virtual human in the scene, an XML file contains the actions that are available to the human, as well as the parameters for blending these different animations. Internally, a *vhdAnimationProperty* is defined for each virtual human, which contains a blending schedule, and some options, such as a choice whether or not facial and/or body animation should be played or if the translation/orientation of the virtual human was defined on the global (world) coordinate system or local coordinate system. The service also includes an integrated player, that plays and blends scheduled animations in a separate thread for all the humans in the scene.

The animation service is controlled either through a GUI, or through a Python script. The use of Python scripts allows for a complete control over many different parts of the environment, such as audio playing in synchrony with animations or camera motions. The following example shows a simple script that controls the body motions of two characters and activates several audio signals as well (crowd cheering, and prerecorded speech signals). The script activates prerecorded actions (such as "creon_wrk") and actions linked with the motion synthesizer (such as "creon_idle"). These different actions are blended on-the-fly and played on the characters. Both sound and body motions are played in synchrony.

```
# global variables
Antigone="Antigone_vhd_occ_rec"
Creon="Creon_vhd_occ_rec"
cam01Creon_crowd="root.Cam01Creon_crowd"

# start the sound
sndService.sndmpPlayMedia(cam01Creon_crowd)

# no facial animation
animService.initPlayerScope_face(False)

# start the player
animService.start_player()
animService.activateAction_body
 (Antigone,"antigone_idle")
# creon monologue
animService.activateAction_body(Creon,
 "creon_wrk")
voiceService.activateAction(Creon,
 "CreonSpeech.Cam01CreonP1",1.0)
animService
 .waitUntilActionFinished_body(Creon,
 "creon_wrk",-4.0)
animService.activateAction_body(Creon,
 "creon_idle")

# Antigone answers to Creon
animService.cutOffAction_body(Antigone,
 "antigone_idle",3.0)
animService.activateAction_body(Antigone,
 "antigone_wrk")
voiceService.activateAction(Antigone,
 "AntigoneSpeech.Cam01CreonP3",1.0)
animService
 .waitUntilActionFinished_body(Antigone,
 "antigone_wrk",-4.0)
animService.activateAction_body(Antigone,
 "antigone_idle")

sndService.sndmpStopMedia(cam01Creon_crowd)
```

5 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 multi-modal content, for example the Rich Representation Language (RRL)[21] or the Virtual Human Mark-up Language (VHML)[40]. 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 behaviour. An example of a tagged sentence looks like this:

```
<begin_gesture id="g1" anim="shakehead"/>
Unfortunately, I have
<begin_gesture id="g2" anim="raiseshoulders"/>
no idea<end_gesture id="g2"/> what you are
talking about.<end_gesture id="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 [21] – 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

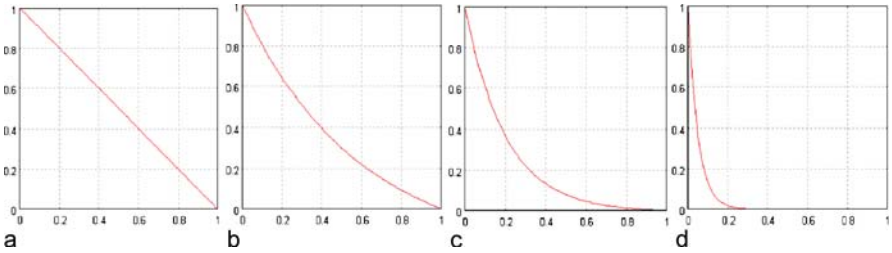

Fig. 8a–d. Coarticulation base function with different α values: **a** $\alpha = 0$ **b** $\alpha = 2$ **c** $\alpha = 5$ and **d** $\alpha = 10$

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 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.

5.1 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-4 Face 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 [7] use this technique and they define an exponential dominance function. Similarly, we use the following base function to construct the coarticulation curve:

$$f(x) = e^{-\alpha x} - x \cdot e^{-\alpha} \tag{2}$$

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

$$C_\alpha(x) = e^{-\alpha 2|x-0.5|} - 2|x - 0.5| \cdot e^{-\alpha}. \tag{3}$$

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.

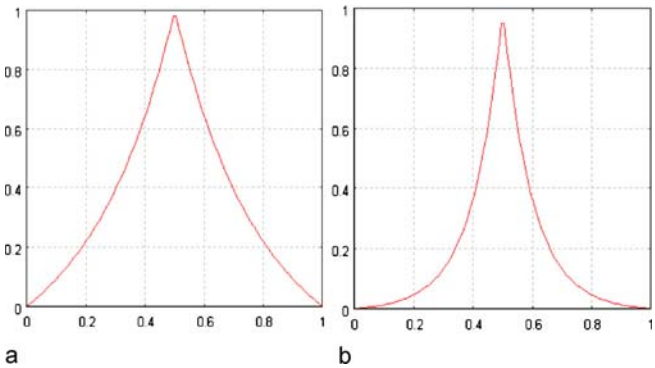

Fig. 9a,b. Example of complete coarticulation functions with different α parameters: **a** $\alpha = 2$ and **b** $\alpha = 5$

Because of the generic structure of the MIRAnim engine (see Sect. 4), it is a simple task to create the facial animation from the viseme timing information. We define a blending action for each viseme, where the dominance function acts as the weight curve. The blending schedule containing the different blending actions will then automatically perform the viseme blending. Because we use direct FAP blending, our approach also handles tongue movements (see Fig. 10), as opposed to earlier work by Kshirsagar [22], who used a principal component representation of recorded face motions that did not include tongue movements.

Next to the facial speech motion, are also any facial gestures that need to be added, defined as tags in the text. An example of an eyebrow raising facial gesture could be defined as follows:

```
Are you <begin_gesture id="g1"
anim="raise_eyebrows"/>really sure about that?
<end_gesture id="g1"/>
```

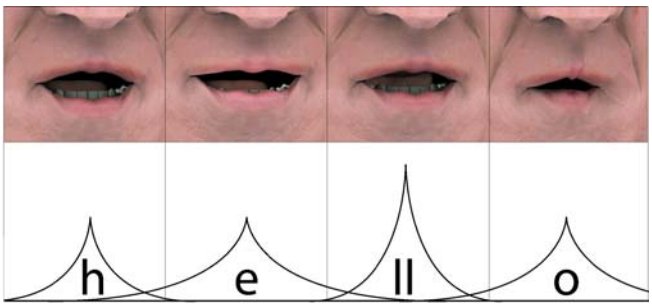

Fig. 10. Facial animation for *hello* that takes tongue movement into account

Finally, a face blinking generator is added for increased realism. Each face animation track has different weights for the FAPs. The speech animation has a higher weight on the mouth area, whereas the face expression weights are higher on the eye and eyebrow area. By blending the different facial animation tracks, the final animation is obtained.

5.2 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 Sect. 4.4 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 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. Figure 11 shows some examples of full body postures obtained using our approach.

5.3 The interaction service

As a higher level control mechanism on top of the animation service, we have developed the *interaction service*. This service mainly handles the spoken interaction between a user and any virtual human in the scene. The core of the interaction service is a dialogue manager [8], that responds to the user according to a predefined script that follows a similar approach as Alice [1]. In order to provide for a more natural interaction, we have integrated the Microsoft Speech SDK (SAPI5.1) [23] into our MR framework, so that automatic speech recognition (ASR) as well as text-to-speech (TTS) is available.

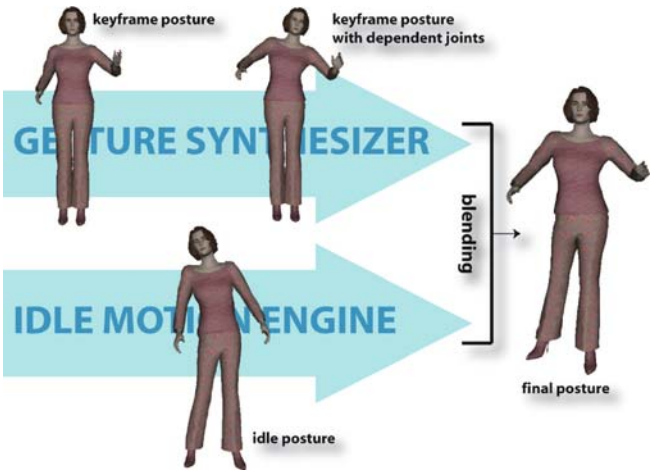

Fig. 11. Integration of gesture animations, dependent joint motion synthesis and idle motion synthesis

From the output text generated by the dialogue system, the corresponding face and body motions are generated according to the approach explained in the previous section. Because the same service is used for both scripted and interactive animation, it is possible to first play a script as shown in the previous section, and then dynamically switch to animation controlled by the interaction service. Since the animation playing itself is continuously handled by the animation service, the character animation is not disrupted when this switch is made.

6 Our dynamic precomputed radiance transfer algorithm

Our main contribution in precomputed radiance transfer (PRT) methods is in the identification of the elements of an illumination model that allows for precomputing the radiance response of a modelled human surface and dynamically transforms this transfer in the local deformed frame. Previous basic research efforts from the areas of computational physics/chemistry in the rotation of atomic orbitals has uncovered a wealth of tools and identified spherical harmonics (SH) [16] as the most appropriate basis for approximating spherical functions in the sphere. Finally in the area of AR we extend current image-based lighting techniques, which produced the most accurate and seamless, yet offline augmentations and substitute the costly global illumination radiosity part with our real-time dynamic precomputed radiance transfer (dPRT) model.

6.1 PRT for skinned virtual characters

In standard PRT, for diffuse objects, self-transfer at some point can be represented as a transfer vector, which is dotted with the light vector to reproduce the diffuse shader at that point. We aim to prove that it is possible to rotate the SH projected light to the rest frame for each vertex, using SH and a fast rotation algorithm allowing it to be applied for any order of SH. A schematic view of our extended PRT method, termed dynamic PRT (dPRT) is illustrated in Fig. 12. We illustrate the various different mesh segments that our VH mesh consists of, as a result of the Hanim [13] specification; but it could be a similar multi-segmented topology if multiple materials, textures, shaders were to be specified in different parts of the VH body. The distant natural illumination is captured via a mirrored ball and processed in the input form of a light probe (angular environment map), as shown in the background of the Fig. 12. Then for each vertex in the mesh, a new dPRT DSM transfer vector is preprocessed that encodes in SH coefficients the response of that vertex to the environment and how the incident radiance from each direction around the hemisphere of directions is transformed to exit radiance. A more thorough explanation on dPRT and the real-time response for skeleton-based deformable characters is illustrated in Sect. 6.1.1.

Since the main focus of our physically principled global illumination model is the multi-material, multi-segmented virtual characters, we have chosen the H-Anim [13]. As the topology (level of articulation) is well-defined we have categorized the various regions of the human body according to the anticipated behaviour of light scattering. Thus the categorization of the virtual character has been at two different levels, the posture dependent (PD) and posture independent (PI) segments. PI are the segments which exhibit low-frequency light scattering and when self-shadowing is calculated for these regions at the skin binding posture it can be assumed as realistic since these polygons when deformed are not affecting significantly their neighbouring vertices. Thus PI segments are identified as: *Skull*, *T1-L/R_Clavicle*, *Sacrum*, *Pelvis*, *L/R_Thigh* and *L/R_Hand*. Since such PI segments are identified, the remaining H-Anim segments are characterized as PD and a high-frequency shadowing term is calculated as shown in the following section, e.g. shadow of an arm on trunk.

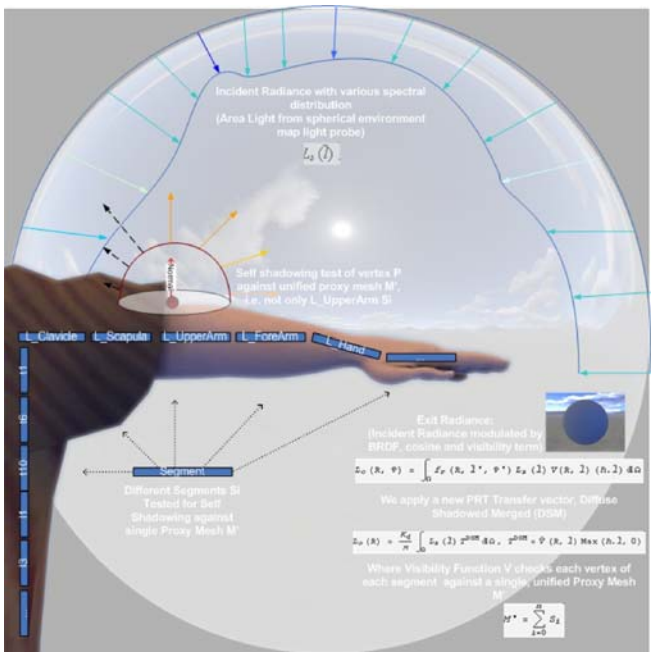

Fig. 12. Schematic view our dPRT illumination model

6.1.1 Diffuse shadowed merged (DSM) transfer

DSM transfer constitutes our new transfer function, aimed to solve the PRT shortcoming of applicability to multi-segmented mesh hierarchies like those of articulated virtual characters, when self-shadowing and correct lighting rotation is desired during deformation and animation of individual body segments. Specifically three main categories of problems arise from the application of previous diffuse shadowed PRT, which we believe is one of the reasons that it has not yet enjoyed widespread acceptance in the area of real-time character simulations, despite the fact that PRT methods provide the most realistic real-time results to date [17]. PRT methods have not yet appeared in augmented reality simulations, where the comparison with reality is more striking and mandatory due to the appearance of the real video image as the background of virtual augmentations. Thus, in order to enhance the consistency of illumination of the real scene and the virtual characters, we have chosen to extend the precomputed radiance transfer (PRT) illumination model, in order that it is applicable to the multi-segmented, deformable and animatable hierarchies that the MIRAnim system is controlling. The main issue that we resolved was to allow for the multiple segments that the virtual skeleton consists of (due to different joints, segments, cloth, hair meshes) to respond to a high dynamic range area light that can be captured from the real scene, similar to the method described in [36]. Thus, starting from the rendering equation approximation in diffuse PRT methods [17]:

$$L_0(x) = \frac{K_d}{\pi} \int_U L_s(l) T^{DSM} \mathrm{d}U \tag{4}$$

where

$$T^{DSM} = V(x, l) \cdot \max(n \cdot l, 0). \tag{5}$$

The evaluation of the above PRT equation has been shown [36] that it can be achieved by projecting both

incident radiance from the high dynamic range environment map and the product of the visibility function V with the BRDF and due to orthonormality of spherical harmonics (SH), the integral of the SH projected functions simplifies to a dot product of their coefficients. At this point we depart from standard PRT as we allow for our dynamic PRT algorithm to be able to calculate precomputed radiance transfer for multiple-segments of articulated, deformable meshes. In this case the previously shown SH light rotations to local space [36] are now more efficiently performed in global eye space. In the special case of SH rotation, previous algorithms were based on precalculated SH rotation matrix coefficients, that when above order two, would exhibit exponentially increasing computational expense. In our case we utilize an algorithm that is being used in computational chemistry [16], where SH are heavily being used for calculation on molecule orbitals, as the origin of the SH section above denotes.

Therefore, we start by accessing the multiple hierarchies of polygonal meshes that constitute our final deformable virtual character and rearranging them in the form of unique vertex, normal and color indices. Next we process each individual, unique vertex, once via its vertex, color and normal indices and calculate a transfer function for each vertex. A transfer function is a transport linear operator that includes half-cosine, visibility, self-shadowing and interreflection effects and describes the object's shaded response to the environment, mapping the incoming to outgoing radiance. That expensive transfer function is precomputed in an offline stage and stored as vectors per vertex for our Lambertian surfaces. The incident radiance however is not precomputed but sampled dynamically from a high dynamic range environment map. At run-time, this incident radiance is projected to spherical harmonics. For diffuse objects, the precomputed transfer vector at each vertex is dotted with lighting coefficients (input luminance) for self-scattered shading, which gives the approximated lighting at that point.

In our approach we implemented three different transfer functions:

- Diffuse unshadowed (DU) transfer, same as in Ramamoorthi and Hanrahan [32]
- Diffuse shadowed (DS) transfer, same as in Sloan et al. [36]
- Our new diffuse shadowed merged (DSM) transfer function for multi-segmented, articulated deformable meshes

We have investigated that previous transfers of DU and DS are not sufficient or applicable for multi-segmented, deformable VHs and thus we advocate the new extension of DSM transfer for our dPRT approach.

For the case of non-rigidly deformable objects, a significant performance issue has been identified when rotating the area light to the rest frame for every vertex, ensuring the object correctly responds to the environment lighting. Such a response to the lighting environment contains two sub-problems:

- Pure SH coefficients rotation. Most current methods have been based on the direct SH rotation matrix with its coefficients derived from explicit formulas based on the original matrix in Euler angles form. Such an approach is quite costly as it involves $(2l+1)(2l+1)$ sparse matrix multiplications and we demonstrate the working solution of employing a recursive solution based on the algorithm of Ivanic et al. [16]. It has to be noted that this algorithm has been suggested before in the bibliography but not verified as a possible solution to the problem above.
- In order to have a correct response of the object to the lighting environment, the visibility function in the rendering equation integral has to be evaluated for every vertex in real-time. This is a separate active field of research in the rendering community in terms of visibility determination and no clear and robust solution for real-time has been proposed so far.

In Fig. 13 we illustrate the process of the DSM transfer applied during deformation of a virtual human and how the segments respond correctly to the low-frequency lighting environment by applying the Ivanic [16] algorithm for spherical harmonics coefficient rotation.

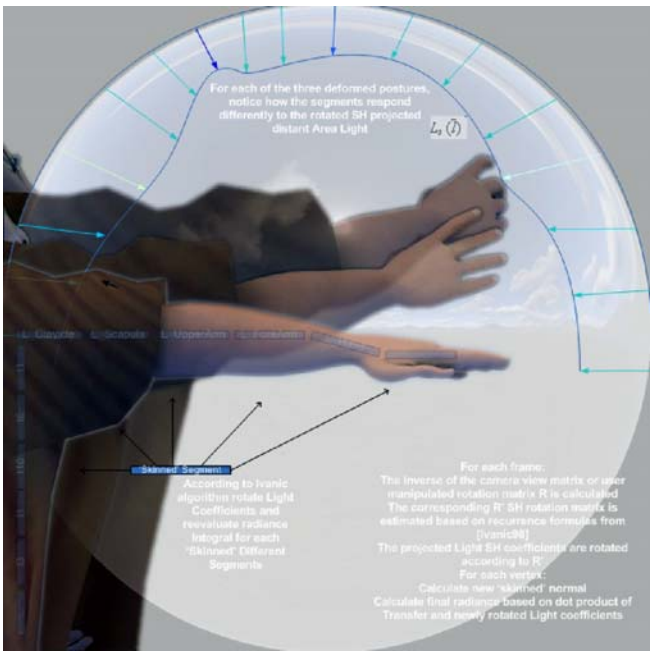

Fig. 13. Our dPRT DSM transfer for deformable VH and correct, per-frame response to environment, low-frequency area light

6.1.2 Our PRT ray tracer and grid acceleration structure

A very efficient and fast acceleration structure for our multi-segmented articulated characters is the uniform grid. We quickly and easily initialize it with the segment geometry during the preprocessing step and employ it in the ray-casting process for the estimation of hits and the diffuse shadowed transfer function via a simple computation that determines the sequence of voxels, which a given ray traverses. The uniform grid accelerator divides the axis-aligned spatial region of our articulated multi-segmented mesh hierarchy into equal-sized cells, called voxels, where each voxel stores a reference to the triangles that overlap it. Given a ray, the grid traversal steps through each of the voxels that the ray passes, checking for possible intersections with only the triangles in each voxel. A simple, fast and stable grid traversal algorithm that we employed is the Amanatides–Woo traversal [2]. However, other non-uniform grid acceleration algorithms (such as octrees) could also be employed, with the expense of additional complexity for a well-known, fixed topology as the VH.

6.1.3 Group proxy mesh for dPRT

In order to allow for correct PRT self-shadowing for multi-segmented, multi-material skinned VHs we aim to define the group [15] M' which encompasses the set of distinct group elements $S_1, S_2, \ldots, S_i$ that correspond to the individual mesh segments as defined by their geometry and global joint transformation matrix. This group proxy mesh cannot be assumed only as the original unsegmented mesh as it can encompass other VH elements that have been added later in the modelling phase. Furthermore, our methodology can be assumed to be applicable to both segmented virtual characters, but also to a wider range of complex real-time deformable models (with different materials, shader, clothes, hair, etc.), for which modelling tools create multiple surfaces (e.g. cloth surface, skin surface, etc.). However, due to all these materials and segments, artefacts often arise due to ray-casting hits between boundaries. Thus, extra relaxation criteria have to be employed. In our dPRT algorithm, we define a new visibility function $V(x, l)$, which checks visibility of the current vertex position against not only the current geometric mesh that resides but against a proxy mesh M_0:

$$M' = \sum_{i=0}^{n} S_i \tag{6}$$

where S_i are all the mesh segments of the skeleton hierarchy of the virtual character. However, not all segments are desired to be able to cast or receive self-shadowing and other GI effects. Thus, the construction of the proxy mesh M_0 is based according to the "type" of the mesh segment, tagged as *receiver* or *occluder* during the artist mesh preparation loop:

- Receiver can receive but cannot cast self-shadowing to other objects of the proxy mesh.
- Occluder can cast but cannot receive self-shadowing from other objects of the proxy mesh.

This is a disjoint set, i.e. an element cannot participate in both groups. If a *receiver* segment is found, it is excluded from the proxy mesh, thus being transparent to ray-casting hits. If an *occluder* segment is found, then this segment is excluded from further dPRT processing and calculation proceeds to next segment. However, this segment is part of the proxy mesh and candidate for ray-casting hits by other segments. Figure 4 illustrates the VH result of DSM with and without the receiver-occluder specifications. The main artefacts appear between the skull and hair segments, which is a typical case of VH arrangement, as separate meshes due to separate creation processes and materials involved.

However, with the application of our receiver-occluder set of elements, this artefact is avoided, as shown in Fig. 14.

We employ a voxel ray tracer and uniform grid acceleration structure for offline ray-tracing for PRT self-shadowing tests. This uniform grid is being initialized in a first pass with this proxy mesh while tested in a second pass, similarly as before with rays originating from every vertex of every segment in the skeleton hierarchy. The main difference is that in the simple diffuse shadowed transfer from Sloan et al. [36], the ray tracer was initialized containing only the parent mesh of the current vertex x for which rays arriving from directions l where tested. This simple and efficient scheme of our proxy mesh results in our diffuse shadowed merged transfer, which allowed us to correct for global illumination effects such as

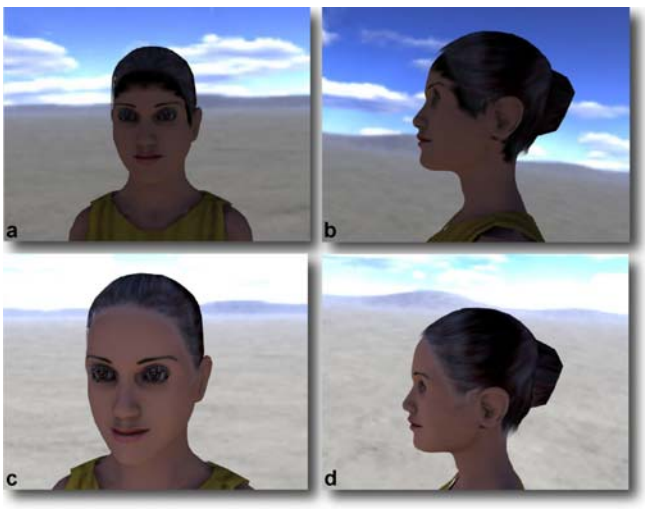

Fig. 14a–d. Image without (**a**, **b**) and with (**c**, **d**) receiver-occluder set processing for dPRT

Fig. 15a,b. Tensor distance ray-hit relaxation criteria for DSM transfer and dPRT. Images shown with (**a**) and without (**b**) tensor distance threshold applied

self-shadowing and successful application of diffuse PRT methods for multi-mesh virtual humans.

As shown in Fig. 15, in the case of multi-segmented, multi-material skeleton hierarchies, even with our diffuse shadowed merged transfer, incorrect self-shadowing is performed due to the proximity of the different mesh segments. It is usually manifested as wrong dark coloring in the vertices in the boundary edges of different skeleton segments, as these segments are modelled adjacent in the skin bind pose, but as they are separate segments, still maintain a distance between them, not visible to the eye but large enough for ray-casting. Thus one relaxation criteria that we have been applying with positive results is the checking for the tensor distance between the ray origin and the ray-triangle intersection point, against a value e corresponding to the minimum distance between the different mesh segments, e.g. for the H-Anim [13] structure this corresponds to 1.71×10^{-4} m. Hence all false hit-point candidates returned by the ray tracer below this e value correspond to hits between the segment boundaries and thus ignored.

Another issue for the shadow feeler visibility rays is that polygons adjacent to the current vertex ray origin are going to be coplanar to many rays originating from that vertex. Thus any ray tested against a polygon that included the ray origin will at best return a hit at that vertex origin. This also leads to incorrect shadowing so it is important that we exclude the polygons that contain our ray origin, from the M' proxy mesh. Thus, in the construction of M' we create full face, edge, vertex information adjacency in order to account for this special case. This second criteria has been already known to the ray-tracing community and similar approaches have been discussed.

Since the final shading information is stored per vertex in the H-Anim virtual human hierarchy, it is independent of the underlying animation approach. Therefore, the MIRAnim system is easily integrated within our MR framework to animate the skeleton of the virtual humans with shading according to the blending schedule that is defined (see Sect. 4).

7 Results

7.1 VR scenario

For the VR case study, we aimed in the reenacting of ancient theatrical plays in reconstructed ancient theatres. Specifically we were involved in the reconstruction of the Hellenistic ancient Aspendos theatre in Minor Asia, Turkey and the simulation of virtual actors of that historical period, reenacting parts of ancient plays. Key elements in this effort have been (1) the idle motions of virtual actors during different dialogues, (2) the ability of the user to interact with them after their performance as well as (3) their multi-segmented, area light-based PRT-based rendering. The hardware platform involved has been a desktop P4 3.2 Ghz, with an NVIDIA Quadro 3450 graphics card. The resulting performance was approximately 20 fps, with two humans comprising of 30 000 polygons. In Fig. 16 an example of this case study is illustrated, where at the far left a standard Phong-based model is shown compared to our PRT method with diffuse shadowed merged transfer.

7.2 AR scenario

For the AR case study we have extended the system described in [26] by allowing for interaction with a virtual human in MR. This aims to allow visitors of ancient Pompeii to be equipped with a mobile AR guide and experience real-time digital narratives of ancient virtual life

Fig. 16a–c. Standard Phong-based shadowing (**a**) in comparison with our PRT method with diffuse shadowed merged transfer (**b**, **c**)

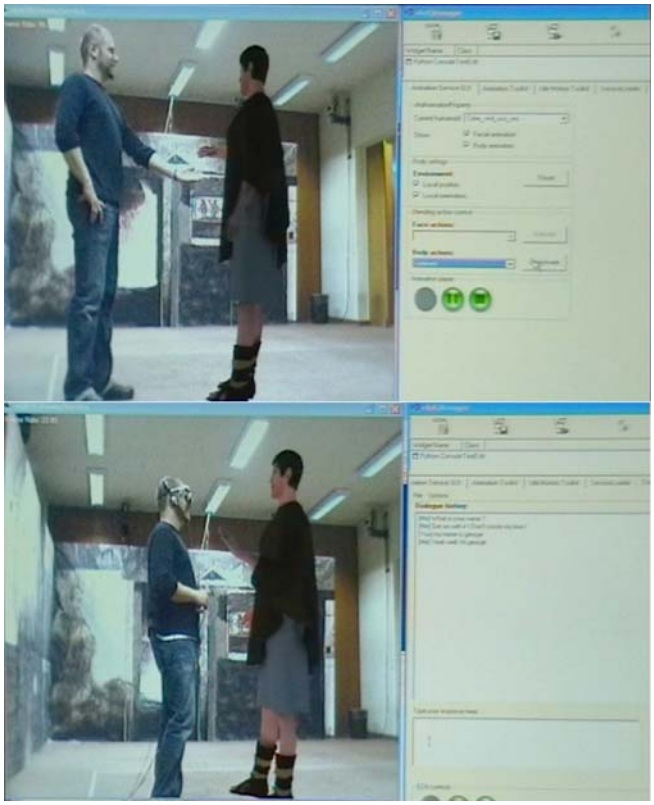

Fig. 17. Real human and interactive virtual human in a mixed reality

coming to life within their natural *real* Pompeian environment. The two crucial components for realizing these AR experiences are (1) the feature-based camera tracker and (2) the MR framework for virtual life simulation and AR registration. The work presented in this paper provides three new components in the previous framework: (1) procedural animation, (2) interaction and (3) MR rendering based on PRT methods. As our original storytelling experience has been intended to revive the life in ancient Pompeii, Italy in an AR simulation manner, a real-size paper maquette of the walls of the Thermopolion of Vetutius Placidus has been recreated in the lab as depicted in Fig. 17. The employed hardware platform was based on a P4 3.0 GHz mobile workstation, with a NVIDIA Geforce5600Go graphics card and Unibrain firewire web camera attached on an I-Glasses HMD. The resulting performance was approximately 30 fps for 12 000 polygons, depending on the performance of the markerless feature-based camera tracker.

8 Conclusion

In this paper, we have presented a mixed reality framework for interactive virtual humans. We have presented a flexible interaction and animation engine, in combination with a robust real-time rendering engine that uses a global illumination for real-time PRT extension for virtual humans. The animation engine allows to switch dynamically between interaction and scenario playing, without interrupting the animation cycle. All components have been integrated as plug-ins in a modern framework, and both VR and AR applications exhibit very suitable performance for real-time MR applications. Finally we illustrate that the amalgam of animation, rendering and interaction with virtual characters, particularly in real scenes, could pave the way for a new suite of applications in augmented reality and a wider adoption of this technology that still lacks a clear application domain.

As with any MR application, many features can still be added to increase the realism and the presence of the users in the scene and that will be the focus of our continuing work in this area. First, our virtual human animation system currently does not consider interaction with objects in the MR scene. This would include picking up virtual objects, or perhaps an integration with a haptic device in order to interact with objects from the real world. Also, a look-at behaviour would be an interesting extension, which would drastically improve the sense that the virtual human is aware of the user in the scene. Finally, we are currently investigating the inclusion of more elaborate real-time simulation and rendering of real-time virtual hair and clothing in the MR scene, while still ensuring that the system can operate at an acceptable framerate on a commodity desktop PC.

Acknowledgement This research has been developed at MIRALab–University of Geneva and has been supported through the EU funded projects ERATO (INCOMED-ICFP502A3PR03) and HUMAINE (IST-507422).

References

1. Alice chat bot: http://www.alicebot.org/. Cited November (2005)
2. Amanatides, J., Woo, A.: A fast voxel traversal algorithm for ray tracing. In: G. Marechal (ed.) Proceedings of Eurographics '87, pp. 3–10. Elsevier, Amsterdam (1987)
3. Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S., MacIntyre, B.: Recent advances in augmented reality. IEEE Comput. Graph. Appl. **21**(6), 34–47 (2001)
4. Balcisoy, S.S.: Analysis and development of interaction techniques between real and synthetic worlds. Dissertation, EPFL (2001)
5. Cassell, J., Vilhjálmsson, H., Bickmore, T.: BEAT: the Behavior Expression Animation Toolkit. In: Proceedings of SIGGRAPH '01, pp. 477–486 (2001)
6. Cavazza, M., Martin, O., Charles, F., Mead, S., Marichal, X.: Users acting in mixed reality interactive storytelling. In:

Proceedings of 2nd International Conference on Virtual Storytelling, pp. 189–197 (2003)
7. Cohen, M.M., Massaro, D.W.: Modeling coarticulation in synthetic visual speech. In: N. Magnenat-Thalmann, D. Thalmann (eds.) Models and Techniques in Computer Animation, pp. 139–156. Springer, Berlin Heidelberg New York (1993)
8. Egges, A., Kshirsagar, S., Magnenat-Thalmann, N.: Generic personality and emotion simulation for conversational agents. Comput. Anim. Virt. Worlds **15**(1), 1–13 (2004)
9. Egges, A., Magnenat-Thalmann, N.: Emotional communicative body animation for multiple characters. In: Proceedings of the 1st International Workshop on Crowd Simulation (V-Crowds), pp. 31–40 (2005)
10. Egges, A., Molet, T., Magnenat-Thalmann, N.: Personalised real-time idle motion synthesis. In: Pacific Graphics 2004, pp. 121–130 (2004)
11. Garchery, S.: Animation faciale temps-reel multi plates-formes. Dissertation, MIRALab, University of Geneva (2004)
12. Grassia, F.S.: Practical parameterization of rotations using the exponential map. J. Graph. Tools **3**(3), 29–48 (1998)
13. H-Anim Humanoid Animation Working Group: Specification for a standard humanoid. http://www.h-anim.org/. Cited May (2006)
14. Hartmann, B., Mancini, M., Pelachaud, C.: Formational parameters and adaptive prototype instantiation for MPEG-4 compliant gesture synthesis. In: Computer Animation 2002, pp. 111–119 (2002)
15. Inui, T., Tanabe, Y., Onodera, Y.: Group Theory and its Applications in Physics. Springer, Berlin Heidelberg New York (1990)
16. Ivanic, J., Ruedenberg, K.: Additions and corrections: rotation matrices for real spherical harmonics. J. Phys. Chem. **102**(45), 9099–9100 (1998)
17. Kautz, J., Lehtinen, J., Sloan, P.P.: Precomputed radiance transfer: theory and practise. In: ACM SIGGRAPH '05 Course Notes (2005)
18. Kopp, S., Wachsmuth, I.: Synthesizing multimodal utterances for conversational agents. Comput. Anim. Virt. Worlds **15**(1), 39–52 (2004)
19. Kovar, L., Gleicher, M.: Flexible automatic motion blending with registration curves. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 214–224 (2003)
20. Kovar, L., Gleicher, M., Pighin, F.: Motion graphs. In: Proceedings of SIGGRAPH '02, pp. 473–482 (2002)
21. Krenn, B., Pirker, H.: 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, pp. 107–115. University of Leeds, UK (2004)
22. Kshirsagar, S., Molet, T., Magnenat-Thalmann, N.: Principal components of expressive speech animation. In: Computer Graphics International 2001, pp. 38–44. IEEE Press, Washington, DC (2001)
23. Microsoft Speech SDK version 5.1 (SAPI5.1): http://www.microsoft.com/speech/download/sdk51/. Cited May (2006)
24. Mueller, M., Roeder, T., Clausen, M.: Efficient content-based retrieval of motion capture data. In: Proceedings SIGGRAPH '05, pp. 677–685 (2005)
25. Openscenegraph: http://www.openscenegraph.org/. Cited May (2006)
26. Papagiannakis, G., Foni, A., Magnenat-Thalmann, N.: Practical precomputed radiance transfer for mixed reality. In: Proceedings of Virtual Systems and Multimedia 2005, pp. 189–199. VSMM Society, Yanagido, Japan (2005)
27. Papagiannakis, G., Kim, H., Magnenat-Thalmann, N.: Believability and presence in mobile mixed reality environments. In: IEEE VR2005 Workshop on Virtuality Structures (2005)
28. Perlin, K.: An image synthesizer. In: Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques, pp. 287–296. ACM, Boston (1985)
29. Perlin, K.: Real time responsive animation with personality. IEEE Trans. Visual. Comput. Graph. **1**(1), 5–15 (1995)
30. Poggi, I., Pelachaud, C., Rosis, F.D., Carofiglio, V., Carolis, B.D.: Greta: a believable embodied conversational agent. In: O. Stock, M. Zancanaro (eds.) Multimodal Intelligent Information Presentation, vol. 27. Springer, Berlin Heidelberg New York (2005)
31. Ponder, M., Papagiannakis, G., Molet, T., Magnenat-Thalmann, N., Thalmann, D.: Vhd++ development framework: towards extendible, component based VR/AR simulation engine featuring advanced virtual character technologies. In: Proceedings of Computer Graphics International (CGI), pp. 96–104. IEEE Press, Washington, DC (2003)
32. Ramamoorthi, R., Hanrahan, P.: An efficient representation for irradiance environment maps. In: Proceedings of SIGGRAPH '01. ACM, Boston (2001)
33. Ren, Z., Wang, R., Snyder, J., Zhou, K., Liu, X., Sun, B., Sloan, P.P., Bao, H., Peng, Q., Guo, B.: Real-time soft shadows in dynamic scenes using spherical harmonic exponentiation. In: Prococeedings of ACM SIGGRAPH '06, pp. 977–986 (2006)
34. Rose, C., Cohen, M., Bodenheimer, B.: Verbs and adverbs: multidimensional motion interpolation. IEEE Comput. Graph. Appl. **18**(5), 32–48 (1998)
35. Rose, C., Guenter, B., Bodenheimer, B., Cohen, M.: Efficient generation of motion transitions using spacetime constraints. In: Proceedings of ACM SIGGRAPH '96, Annual Conference Series, pp. 147–154 (1996)
36. Sloan, P.P., Kautz, J., Snyder, J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In: Proceedings of ACM SIGGRAPH '02, pp. 527–536. ACM, Boston (2002)
37. Tamura, H.: Mixed reality: future dreams seen at the border between real and virtual worlds. IEEE Comput. Graph. Appl. **21**(6), 64–70 (2001)
38. Thomas, B., Close, B., Donoghue, J., Squires, J., De Bondi, P., Morris, M., Piekarski, W.: Arquake: an outdoor/indoor augmented reality first person application. In: Proceedings of Symposium on Wearable Computers, pp. 139–146 (2000)
39. Unuma, M., Anjyo, K., Tekeuchi, T.: Fourier principles for emotion-based human figure animation. In: Proceedings of ACM SIGGRAPH '95, Annual Conference Series, pp. 91–96 (1995)
40. Virtual Human Markup Language (vhml):http://www.vhml.org/. Cited November (2005)
41. Vacchetti, L., Lepetit, V., Ponder, M., Papagiannakis, G., Fua, P., Thalmann, D., Magnenat-Thalmann, N.: Stable real-time AR framework for training and planning in industrial environments. In: S.K. Ong, A.Y.C. Nee (eds.) Virtual Reality and Augmented Reality Applications in Manufacturing. Springer, Berlin Heidelberg New York (2004)
42. Vlahakis, V., Ioannidis, N., Karigiannis, J., Tsotros, M., Gounaris, M., Stricker, D., Gleue, T., Daehne, P., Almeida, L.: Archeoguide: an augmented reality guide for archaeological sites. IEEE Comput. Graph. Appl. **22**(5), 52–60 (2002)
43. Wiley, D., Hahn, J.: Interpolation synthesis of articulated figure motion. IEEE Comput. Graph. Appl. **17**(6), 39–45 (1997)

DR. ARJAN EGGES is an assistant professor at the Center for Advanced Gaming and Simulation at Utrecht University in the Netherlands. He obtained his PhD at MIRALab–University of Geneva on the topic of real-time animation of interactive virtual humans. His main interests involve the modelling of intelligent behaviour, non-verbal communication, such as gestures and facial animation, real-time face and body animation, and personality and emotion modelling. He is currently involved in various research topics related to gaming and graphics, and he teaches a course on game programming as well as a seminar on animation.

DR. GEORGIOS PAPAGIANNAKIS is a computer scientist and senior researcher at MIRALab–University of Geneva. He obtained his PhD in Computer Science at the University of Geneva, his BEng (Hons) in Computer Systems Engineering, at the University of Manchester Institute of Science and Technology (UMIST) and his MSc (Hons) in Advanced Computing, at the University of Bristol. His research interests are mostly confined in the areas of mixed reality illumination models, real-time rendering, virtual cultural heritage and programmable graphics.

PROF. NADIA MAGNENAT-THALMANN has pioneered research into virtual humans over the last 25 years. She has 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 Geneva in 1989, 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 modelling 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 Marilyn 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 OF A MANNEQUIN (2003) and THE AUGMENTED LIFE IN POMPEII (2004). She is editor-in-chief of *The Visual Computer* published by Springer.