# Simulating Virtual Humans in Networked Virtual Environments

**Igor Pandzic[+], Christian Babski[*], Tolga Capin[*], WonSook Lee[+], Nadia Magnenat-Thalmann[+], Soraia Raupp Musse[*], Laurent Moccozet[+], Heywon Seo[+], Daniel Thalmann[*]**

[+] *MIRALab, University of Geneva, Switzerland {Igor.Pandzic, WonSook.Lee, Nadia.Thalmann, Laurent.Moccozet, seo }@cui.unige.ch*

[*] *Computer Graphics Lab, EPFL, Lausanne, Switzerland {babski, capin, soraia, thalmann}@lig.di.epfl.ch*

## Abstract

In the past decade Networked Virtual Environments have been an increasingly active area of research, with first commercial systems emerging recently. Graphical and behavioral representation of users within such systems is a particularly important issue that has lagged in development behind other issues like network architectures, space structuring etc. In this paper we expose the importance of using Virtual Humans within these systems and provide a brief overview of several Virtual Humans technologies used in particular for simulation of crowds. As the main technical contribution the paper presents the integration of these technologies with the COVEN-DIVE platform, the extension of the DIVE system developed within the COVEN project. In conjunction with this, we present our contributions through the COVEN project to the MPEG-4 standard concerning the representation of Virtual Humans.

# Simulating Virtual Humans in Networked Virtual Environments

**Igor Pandzic[+], Christian Babski[*], Tolga Capin[*], WonSook Lee[+], Nadia Magnenat-Thalmann[+], Soraia Raupp Musse[*], Laurent Moccozet[+], Heywon Seo[+], Daniel Thalmann[*]**

[+] *MIRALab, University of Geneva, Switzerland {Igor.Pandzic, WonSook.Lee, Nadia.Thalmann, Laurent.Moccozet, seo }@cui.unige.ch*

[*] *Computer Graphics Lab, EPFL, Lausanne, Switzerland {babski, capin, soraia, thalmann}@lig.di.epfl.ch*

## Abstract

In the past decade Networked Virtual Environments have been an increasingly active area of research, with first commercial systems emerging recently. Graphical and behavioral representation of users within such systems is a particularly important issue that has lagged in development behind other issues like network architectures, space structuring etc. In this paper we expose the importance of using Virtual Humans within these systems and provide a brief overview of several Virtual Humans technologies used in particular for simulation of crowds. As the main technical contribution the paper presents the integration of these technologies with the COVEN-DIVE platform, the extension of the DIVE system developed within the COVEN project. In conjunction with this, we present our contributions through the COVEN project to the MPEG-4 standard concerning the representation of Virtual Humans.

## Importance of Virtual Humans for NVEs

The participant representation in a networked VE system has several functions:

- perception
- localization
- identification
- visualization of interest focus
- visualization of actions
- communication

*Perception* and *localization* are the very basic functions of participant representation in NVEs. They allow us to perceive the presence of others in the environment and see where they are. Even a crude embodiment can fulfill these tasks.

*Identification* is an important function because we usually want to know who is in front of us. Means of identification can range from simple ones, like displaying the first letter of one's name to complex body and face models resembling a particular person.

*Visualization of interest focus* can be achieved by any embodiment that somehow represents the direction of gaze - usually this means a graphical model that has eyes or symbols representing eyes, so we can see in which direction it is looking.

*Visualization of actions* requires the embodiment to have some end-effectors that perform actions. In a low-end implementation this might be a simple line reaching to the manipulated object, or it might be a virtual hand grasping the object.

*Communication* in real life is in many ways tied to our body - gestures and facial expressions (including lip movement that improves speech understanding) are natural part of our daily communication. If such communication is to be supported in NVEs, the participant

representation needs to be fairly sophisticated.

Although many of these functions can be fulfilled with very simple embodiments, it is obvious that most can be fulfilled better using more sophisticated Virtual Humans, and some functions can absolutely not be fulfilled without them

Virtual Humans can fulfill these functions in an intuitive, natural way resembling the way we achieve these tasks in real life. Even with limited sensor information, a virtual human frame can be constructed in the virtual world, reflecting the activities of the real user. Slater and Usoh (Slater and Usoh 1994.) indicate that such a body, even if crude, already increases the sense of presence that the participants feel. Therefore it is expected that a better and more realistic embodiment will have a further positive effect on the sense of presence and mutual presence.

While there is already substantial work done on integrating Virtual Humans in NVEs (Capin et al., 1997., Pandzic et al.,1997., Capin et al., 1999.), in this paper we present an overview of the aspects of this research topic concerning the simulation of crowds: the crowd representation, crowd behavior, generating populations and Level of Detail management which is essential when large number of humans is simulated. We then describe the integration of these techniques in the Coven-Dive platform and present contributions to the MPEG-4 standard resulting from this work.

## Populating NVEs with crowds

The goal of this section is to provide information about why and how populate NVEs using virtual human crowds. Several works have explored such importance of crowds in NVEs (Bouvier, 97) (Brogan and Hodgins, 97) (Reynolds, 87) among others. Imagine participants sharing a virtual museum existent in a NVE. Depending on the number of participants connected at a specific time, the virtual museum may be non-inhabited when a new participant gets connected. In this case, the degree of realism as well as the possibility of interaction with others is reduced. Assuming that NVEs can be populated using virtual crowds, the sense of presence provided during immersion in virtual environments like the virtual museum is substantially increased.

Moreover, the virtual crowd can act autonomously because its members are able to perceive the state of the virtual environment and interact with it. Anyway, the participants can be interested in interacting and/or guiding virtual crowds during the simulation. In order to deal with this variable level of autonomy (from autonomous to guided crowds) we have introduced ViCrowd model (Musse et al, 1999). The main goal of our approach is being able to built autonomous animated creatures for interactive virtual environments.

Considering the needs of controling and interacting with the crowds at the same time that programming them, we decided to use a multi-level hierarchical model where crowds and agents can present different levels of autonomy. In this way, groups of agents can be programmed (before the simulation), act in a autonomous way as well as be guided by the user (during the simulation). Moreover, the hierarchical structures presented in ViCrowd like agents, groups and crowd aim to provide the simulation of families, leadership and other aspects which are not possible if the crowd is represented by one-level of agents' structure. These decisions have been taken into account in order to comply with the needs and requirements of our applications.

The next sections describe succinctly how ViCrowd model is used to define the autonomous and guided behaviors of virtual crowds in NVEs, as well as the methods used for the generation of physical appearance of population.

## ViCrowd Model

We defined a crowd as a set of groups composed of virtual agents. Our model distributes the crowd behaviors to the groups and then to the individuals. There are two ways for setting the parameters of ViCrowd: scripted and external control. Scripted control defines *scripted behaviors* of the crowd whereas external control specifies *guided behaviors*. Yet, ViCrowd is represented through a hierarchical architecture where the minor entity to be treated consists of groups. In this case, the groups are more “intelligent” structures, while individuals follow groups’ specification. This decision is due to real time requirements existing in NVEs applications and aims to optimize the information needed to provide intelligent agents.

At a lower level, the individuals have a repertory of basic behaviors that we call *innate behaviors*. An innate behavior is defined as an "inborn" way to behave. Goal seeking behavior, the way trajectories are processed and collision avoided, the ability to follow scripted or guided events/reactions are examples of innate behaviors of individuals. While the innate behaviors are included in the model, the specification of scripted behaviors is done by means of a script language where pre-defined commands are specified in order to model different behaviors of crowds. We call the groups of virtual agents **<*programmed groups*>** who apply the scripted behaviors and do not need user intervention during simulation. Using the script language, the user can directly specify the crowd or group behaviors. In this case, the system automatically distributes the crowd behaviors among the existing groups. Events and reactions have been used to represent behavioral rules. This reactive character of the simulation can be programmed in the script language (scripted control) or directly given by an external controller. We call the groups of virtual agents **<*autonomous groups*>** who apply the behavioral rules. Yet, externally controlled groups, that we call <***guided groups***>, no longer obey their scripted behavior, but act according to the external specification.

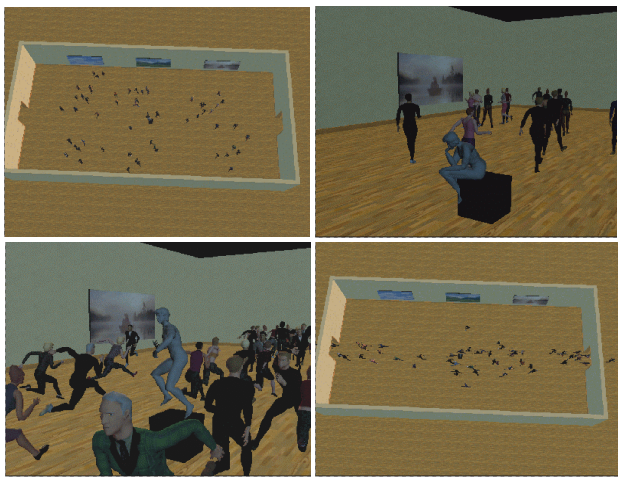

Figure 1: Scenes of simulation of evacuation due to a panic situation. Up-left and up-right: before the panic situation, the crowd walks. Down-left and down-right: crowd reacts because an event generated when the statue becomes alive

For example, if a group's intention is to visit a museum (scripted group behavior), but a panic situation occurs (event), this group can then perform the programmed reaction associated with the event. This reaction can either be externally specified (during the simulation) or pre-programmed in the script, e.g. exit the environment. Figure 1 shows a sequence of animation in a panic situation, where 100 agents react exiting the museum because a statue which becomes alive.

Figure 2: Dialogue between the leader and the crowd. On the left, the leader talks and on the right, the crowd reacts.

A script language has been used in order to program specific behaviors in ViCrowd (Musse et al, 1998).

In addition to autonomous crowds, ViCrowd also supports externally guided crowds. The information concerning the entities that can be controlled are: motion, action, events, reaction and internal status.

We integrated ViCrowd system in DIVE in order to provide a module responsible for the control of crowd behaviours. This module can provide autonomous crowds that have some knowledge about the virtual environment and are able to interact with it. Figure 3 presents some images of crowd simulation in world dive where a participant is observing autonomous crowds.

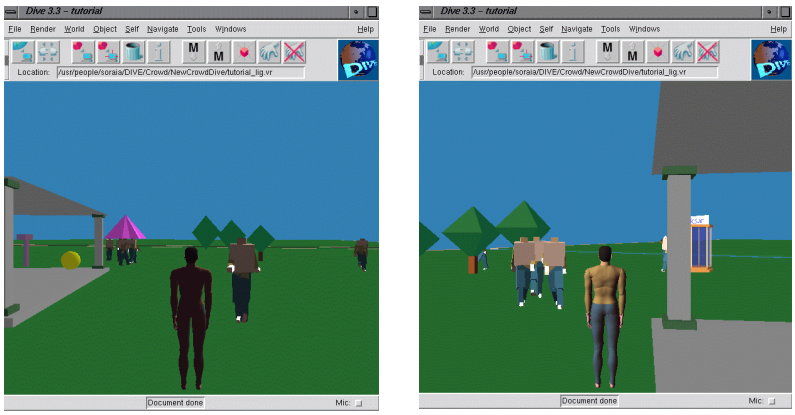


Figure 3: A participant observing the autonomous virtual crowds.

In accordance with the parameter settings, which could be recognised by the guided crowds, DIVE may provide some information from the participants (e.g., location, direction). Afterwards, the guided crowd is able to recognise the information sent by DIVE and distribute it among groups behaviours.

## Creating Populations

Given the high number of elements in a population, we need a fast and efficient way to create a physical appearance for each element. Physical appearance of an agent should be realistic but also unique in order to differentiate the agent from other agents within a population. We divide the physical appearance into the face part and the body part. We will describe some methods allowing the automatic construction of an unlimited number of faces and a fast method for the construction of bodies. These constructed faces and bodies could be stored in a database allowing a user to choose whatever body and face he wants to assemble in order to create a complete physical appearance for an agent.

### Construction of faces

Animators agree that the most difficult subjects to model and animate realistically are humans and particularly human faces. The explanation resides in the universally shared (with some cultural differences) processes and criteria not only for recognizing people in general, but also for identifying individuals, expressions of emotion and other facial communicative signals, based on the covariation of a large number of partially correlated shape parameters within narrowly constrained ranges. Though some of this may be amusingly conveyed in a schematic

manner by various 2D or 3D animation technologies, the results are easily identified as cartoons by the most naive observer, while the complex details and subtle nuances of truly realistic representation remain a daunting challenge for the field.

Approaches to the realistic reconstruction of individuals, some of them with a view to animation, include the use of a laser scanner (Lee et al, 1996), a stereoscopic camera (http://www.turing.gla.ac.uk), an active light stripper (Proesmans and Gool, 1997), or a video stream (Fua, 1998) to reconstruct heads and natural expressions. Modeling has also been done from picture data (Kurihara and Arai, 1991) (Akimoto et al, 1993) (Ip and Yin, 1996) (Lee et al, 1997) detecting features, modifying a given generic model and then mapping texture on it. Not all of these, however, combine sophisticated and reliable shape deformation methods with seamless, high-resolution texture generation and mapping.

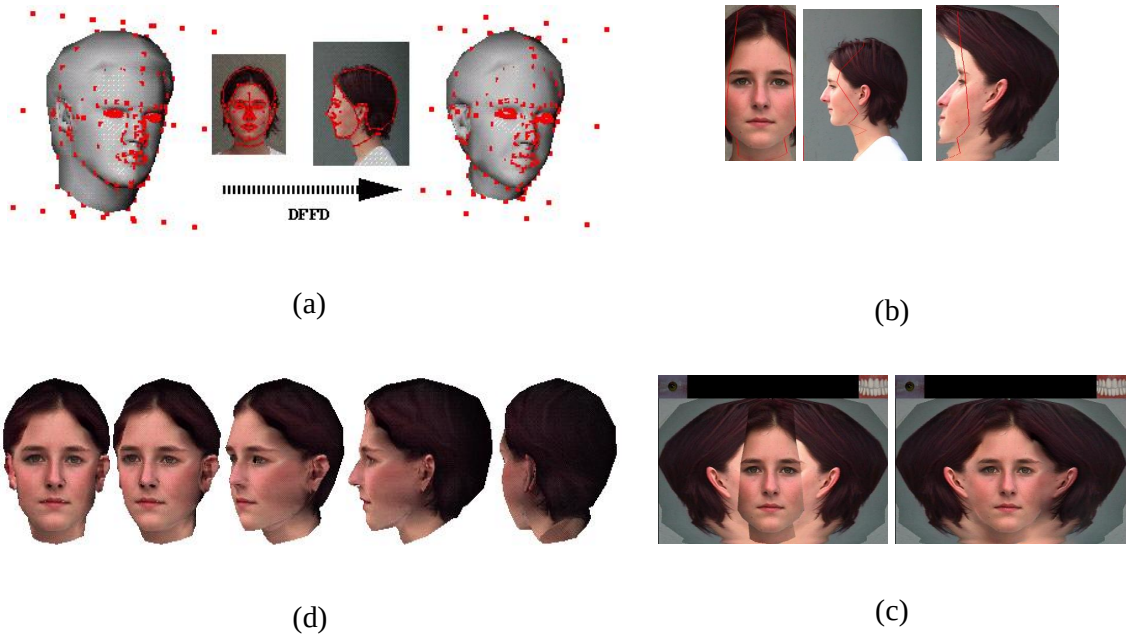


Figure 4: (a) Modification of a generic head according to feature points detected on pictures. Points on a 3D head are control points for Dirichlet Free Form Deformation (DFFD). (b) Geometrical deformation for texture merging. (c) Multiresolution technique for texture. (d) Snapshots of a reconstructed head in several views.

2D photos offer cues to the 3D shape of an object. It is not feasible, however, to consider

3D-points densely distributed on the head. In most cases, we know the location of only a few visible features such as eyes, lips and silhouettes. Figure 4(a) depicts an orthogonal pair of normalized images, showing the features detected. The two 2D sets of position coordinates, from front and side views, are combined to give a single set of 3D points. We use 3D feature points as a set of control points for the Dirichlet-based Free Form Deformation (DFFD) (Moccozet and Magnenat Thalmann, 1997). Thus the deformation of the whole surface is smoothly controlled by the control points. As shown in Figure 4 (a), the result is quite respectable considering the input data (pictures from only two views).

To improve realism, we make use of automatic texture mapping. If the texture mapping is not correct, the accurate shape is useless in practice. We use information from the set of feature points detected to generate texture fully automatically, based on the two views. The main criterion is to obtain the highest resolution possible for most detailed portions. We first connect two pictures along predefined feature lines using geometrical deformations as shown in Figure 4 (b). Then to avoid visible boundary effects, a multiresolution technique (Burt and Andelson, 1983) is used as shown in Figure 4 (c) . We then obtain appropriate texture coordinates for every point on the head using the same image transformation. Figure 4 (d) shows several views of the head reconstructed.

From the population produced by this semi-automatic method, we can automatically produce much larger populations using methods such as morphing and statistical analysis.

**Construction of bodies**

The modeling of human body is an important but difficult problem. The human body shape is rather complex, it can have a variety of sizes and has 2 main types (male/female). A realistic modelisation comprises an articulated skeleton that can support various body elements such as skin and muscles that has to be deformed during an animation. Our goal is to make realistic body with animation and deformation capabilities without the need of physical prototypes or scanning devices. On Figure 5 are described the four elements that are used in the construction of our animated bodies: the skeleton, the deformation model, the skin and the texture.

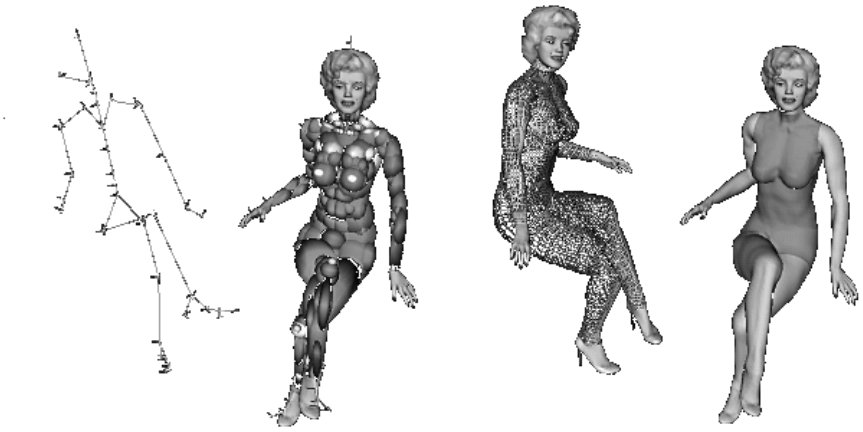

Figure 5 a, b, c, d: The 4 elements used in the construction of body.

Figure 6: bodies with metaball-based cloth

For the construction of the deformation model, we use metaballs. Metaballs are geometric and ellipsoidal shapes that can be smoothly mixed together. A designer interactively positions metaballs around the skeleton in order to approximate the overall form of the set of human muscles (see Figure 5-b). A cross-sectional method combined with B-Spline merging techniques gives the body its final geometry that includes the skin (Figure 5-c). A texture is

finally applied on the body shape. Each metaball is attached to a skeleton joint and define a part of the body. The designer's work consists mainly in varying the relative proportion, orientation and size of each metaball.

Various transformations can be easily applied to an already constructed body such as global scaling that allows the creation of child body. Lateral scaling is another transformation that allows the construction of stronger bodies. The use of agent with different body proportion gives a more realistic appearance to a crowd. A further enhancement of the body shape is possible by using the metaball technique to simulate the cloth (Figure 6). This technique is suitable for jacket or pants but not for skirts for instance. The body has much more realism with the added cloth and still can be used in real-time animation.

## Level of Detail management for Virtual Humans

In our system where participants are represented as highly realistic virtual humans, human models are the main contributors to the geometric complexity of the scene. In fact, each of our virtual human is made up of approximately 15000 textured polygons. Obviously acceleration techniques are necessary to reduce the amount of data for rendering and animating virtual humans in real time. This becomes more crucial as the number of virtual humans in the networked virtual environment increases.

The basic idea of adopting LoD(Levels of Detail) techniques for virtual humans is more or less the same as for other objects in virtual environments : a detailed description of a virtual human is necessary only when it's close to the viewpoint. A description of a virtual human includes both geometry and animation. However, virtual humans have some characteristic features. They are animated or deforming objects. LoD generation and control should incorporate both geometry and animation. Geometric importance (sharp edges, high curvatures, etc.) may not be the only criteria for generating simplified models. In a face model, for example, some parts like lips move more frequently, attracting attention, thus are more important than other parts like ears, even though ears take much more triangles in an original

model.

**LoD on animated faces**

In our real time face animation system, high level actions like ‘smile’, ‘surprise’ are mapped to mid and low level actions (Kalra et al., 1998).The low level actions are composed of 64 MPAs(Minimum Perceptible Actions), each of which corresponds to a visible feature of the related region in the face model. A region is defined as a set of selected polygons in the mesh, where a muscle action is simulated using Rational Free Form Deformations (RFFD).

The generation of different levels of detail on the geometry is done based on several observations on our face models to achieve the best visual effects possible. Ears, teeth are considered to be least important and are simplified at level1. On the other hand, lip movements are considered to be important and are preserved at level2.

Figure 7 shows the hierarchical region management structure. Level 1 model is constructed by simplifying the extra(ears, teeth, and eyeballs) regions. In level 2 model, the extra region is simplified again or removed and upper_mask region is simplified. In level 3, simplification is applied also to the lower_mask, covering the whole head (Figure 8).

By simply discarding animation parameters that correspond to simplified regions, the same hierarchy can be used for animation parameter filtering, too. We define an active tree as a set of regions currently being used by the animation system (Figure 7).

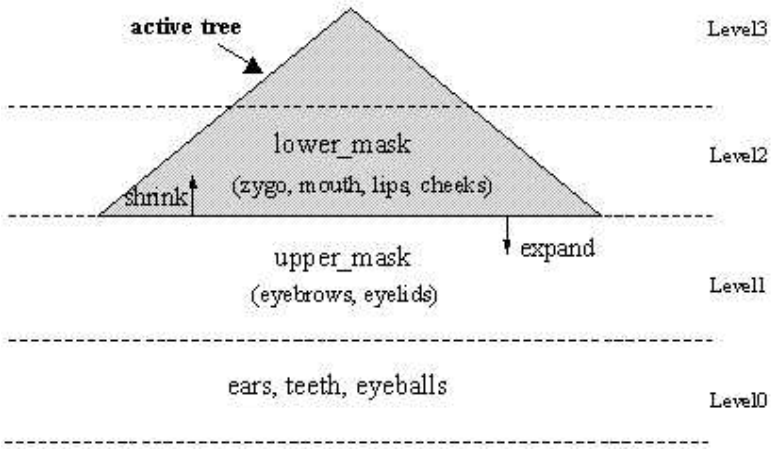


Figure 7: Hierarchical region management structure and active tree

During runtime, the active tree on this hierarchy is managed to select an appropriate level as well as to filter some of animation parameters depending on the distance from the viewpoint Those regions below the active tree are simplified as well as corresponding animation parameters are excluded. As level changes from one to the other, the active tree either shrinks or expands.

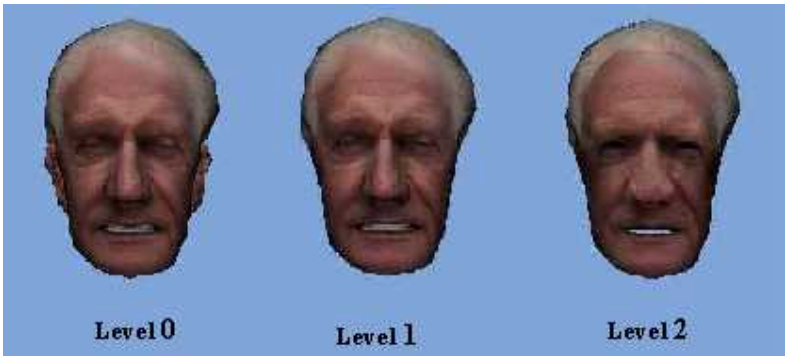


Figure 8: Different levels of a face model

**LoD on animated bodies**

Body models are more complex than face models and the most dominant objects in most of

our applications. Similar to the face models, both geometric simplification and animation parameter optimization have been used for our body models. For geometry, 4 levels are generated by applying simplification (Figure 9). For animation parameters, frame data skipping factor is used to reduce the amount of data – different frame update rates are applied to the body model depending on its distance to the viewer. Also a similar hierarchical approach to face models is begin developed to efficiently cull out animation parameters.

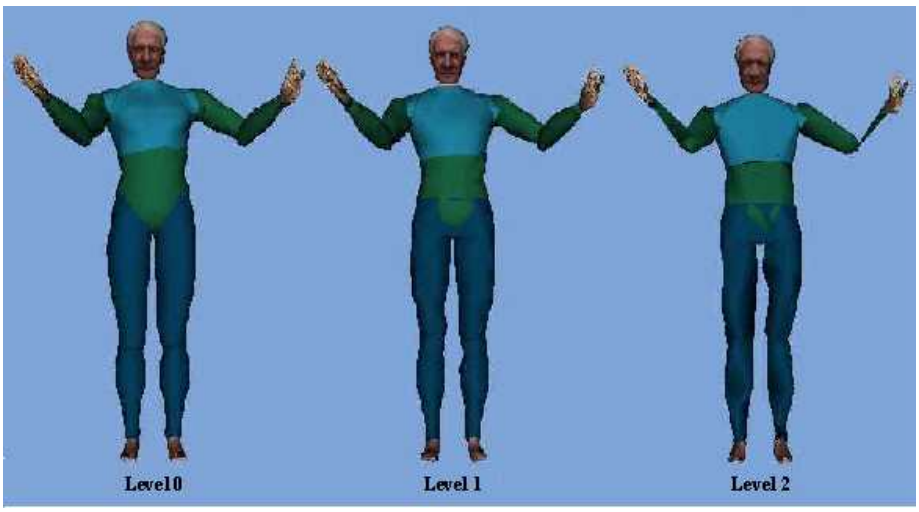


Figure 9: Different levels of a body model

## Integration of Virtual Humans in the COVEN-DIVE platform

Within the COVEN project, two already existing platforms were involved: a commercial solution represented by the work done by Division with the dVS system and an experimental one represented by SICS with the DIVE application. The goal of the COVEN project was to produce a final common platform that integrates major research and development made by each partner all linked to collaborative virtual environment technologies.

The goal of the virtual human integration inside COVEN project, was to integrate in a generic way virtual human representation capabilities. It includes the definition of a hierarchy for the skeleton representation and an entire animation system in order to make the avatar walking, grasping objects, playing animation or reacting to a given event. The result should be

a method that permits the integration of virtual humans in the two platforms present at the beginning of the project as well as inside the final COVEN platform. To be completely generic, this method should also permit the integration of virtual humans inside any kind of existing system based on collaborative virtual environment that generally needs a realistic representation of virtual humans.

We will present this generic solution in the next section and its application to the user's avatar management but also to the crowd management inside a 3D shared world.

## A Generic Solution

For embodiment, in order to avoid having to make the same work each time virtual humans are needed to perform a given set of task in a collaborative environment, it is important to take in account the actual technology at the collaborative virtual environment application level. By examining existing 3D shared world application (Dive, Capin et al., 1997., Pandzic et al., 1997.), it is obvious that all of them propose the plugin technology to developers using their systems. It permits to anybody to establish a connection between the kernel of the application and an external process. This process is then able to access internal information as well as it was a part of the application. The connection is usually established by following a given interface. Of course, from one application to another, this interface is different. This mainly means that if we want to perform a specific task within a set of different systems, we will have to write a specific plugin for each of them. So the plugin solution is a starting point but not a final one. We should be able to separate the common task to perform from the specific interface of each system. This can be done by using shared memory capability which is now available under all major operating systems. It permits to share a given memory segment by two independent processes: information can then be exchanged between the two processes through the memory segment. By combining the plugin technology and the shared memory technology, it becomes possible to reuse a process in charge of a given task from one system to another system by defining two sides:

- The plugin side: it is specific to a given system and has to be written each time. The task of this part is to access any kind of 3D shared information (3D world, events,

etc…) manage by the application. It also allocates a shared memory segment.

- The external process side: it is written once and usable with any kind of system. This process is in charge of a given set of tasks. It has to establish a connection to the shared memory segment allocated by the plugin.

The access to the shared memory segment is done through the use of a given interface. The way the shared memory segment is allocated is explained in Figure 10 in the case of an external body controller. The COVEN Platform first loads the standard body definition (1). Then it allocates a shared memory segment (2) to be used for communication with the external body controller. This operation yields a shared memory ID number (3) that is passed to the external body controller as a parameter when the body controller process is spawned (4). The controller uses the ID to connect to the shared memory segment (5) and thus the link with the COVEN platform is established. The shared memory segment is used for all further communication.

This system was tested to add realistic looking and behaving avatars as user's graphical representation and for adding crowds to these 3D shared world. These two examples are presented in the following sections.

The common part (the external process side), which is in our case the external body controller and the external crowds controller, is based on libraries developed in both labs LIG/EPFL and University of Geneva. These libraries basically permit us to perform several kinds of animation like walking motion (based on a parametrically walking motor), keyframe animation or real-time recorded animation (by using a set of magnetic sensors fixed on the body of a real user). In a more complex way, it is also possible to mix several animations and applying a notion of weight to animation (Boulic et al., 1997). A walking body can then play an animated sequence with his right arm while continuing to walk. Any COVEN platform developer can write their own body controller process, based on different libraries (to be able to make a direct interface between a specific device and the body animation for example) just by following the shared memory interface.

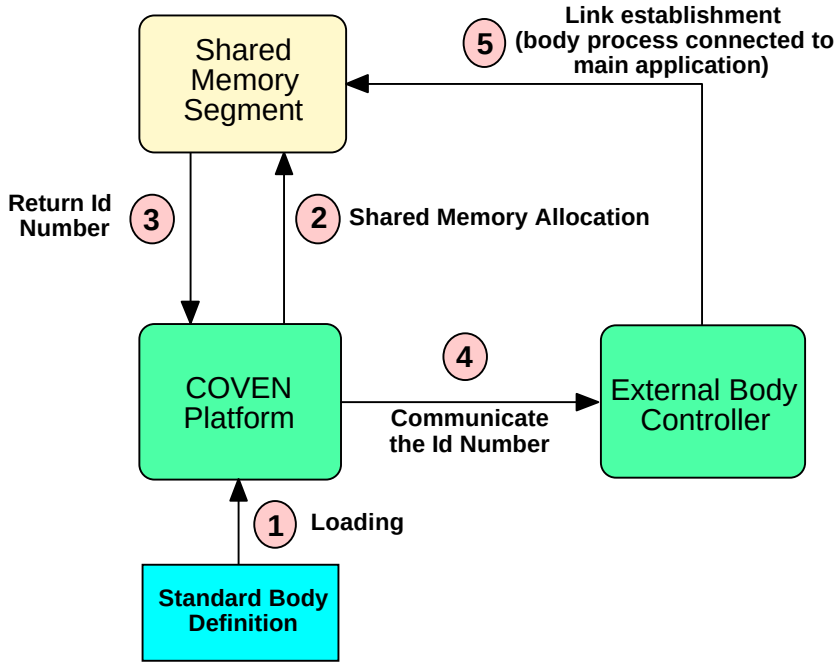


Figure 10 – The shared memory allocation system.

As long as each application has different functionality, performances from one platform to another are different (mainly in network configuration). But, in terms of body animation, capabilities are the same whatever the platform. Another point is that the interface for animating the avatar or the crowds could be the same. This also permits the developer to avoid having to learn a new way to control virtual humans each time we move from one system to another.

## User's Avatar Controller

We will present the generic solution described in the previous section applied to user's graphical representation.

In the following graph (Figure 11), we have an overview of how the system is working. The plugin side has two major functions. The first set of functions will permit the detection and retrieval the body structure in the database of the main application. The second set will use network data distribution functions of the application to perform animation and send it to connected clients.

According to the kind of avatar used by the user (the body should be represented by a valid hierarchy (Boulic et al., 1995), the application will or will not allocate the shared memory. Once the shared memory is allocated, the external body controller process can be launched. When the connection with the main program is established, user's avatar will be animated according to what he is doing in the 3D shared world.

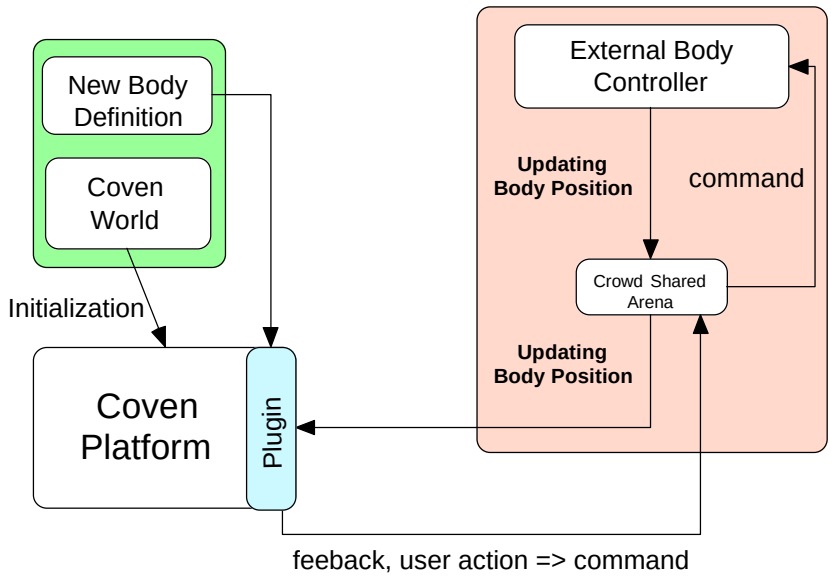


Figure 11 The generic solution applied to avatar animation.

The shared memory segment is used to transmit data in both ways from and to the COVEN application (Figure 11). Data flow coming out from the main application to the body controller (way 1) is composed of interactions performed by the user: outputs like speed change, direction change or object to pick are sent to the external process. The body animation is then adapted to the kind of action performed, displayed locally and sent to the network using the way back (way 2) to the application. The way 1 is also useful at the beginning of the connection to inform the external body controller about the structure of the 3D shared world. As long as the body controller is not integrated at the level of the main application, it is not possible for this process to access to the world database to locate objects to avoid for example. This means that a set of information has to be sent to initialise the external process concerning the surrounding 3D world. The incoming data flow, from the external process to the main

application (way 2), concerned only the body animation. These animations are computed according to interactions received by the body process from the main application: resulting animations can be a mix between a walking movement and a specific arm animation asked by the user (Boulic et al., 1997).

All clients who want to participate to an on-going session do not have to use the specific plugin. A default version of the application (which does not include the plugin) can also participate to the trial. This user will be able to see body animation from other clients but of course he won't take benefit of the same kind of capabilities.

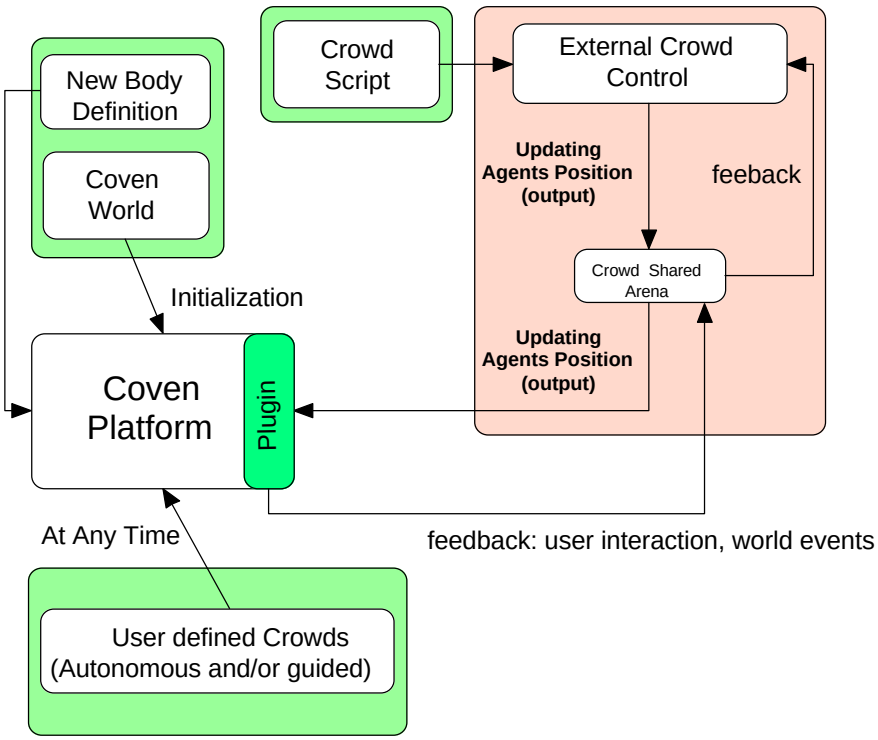


Figure 12 The generic solution applied to a crowd controller.

## Crowds Controller

The second example of application of the generic method is linked to crowd animation. Here, we still have to animate virtual humans but with an autonomous behaviour. As described in the section specific to crowds, a crowd can react to user's interaction as well as world

events. This needs a specific set of information on the world as well as dynamic information that will transit through the shared memory segment to the external crowd controller in order to adapt the crowd behaviour. An additional set of information concerning the static part of the world is given to the controller through a script file (Figure 12)

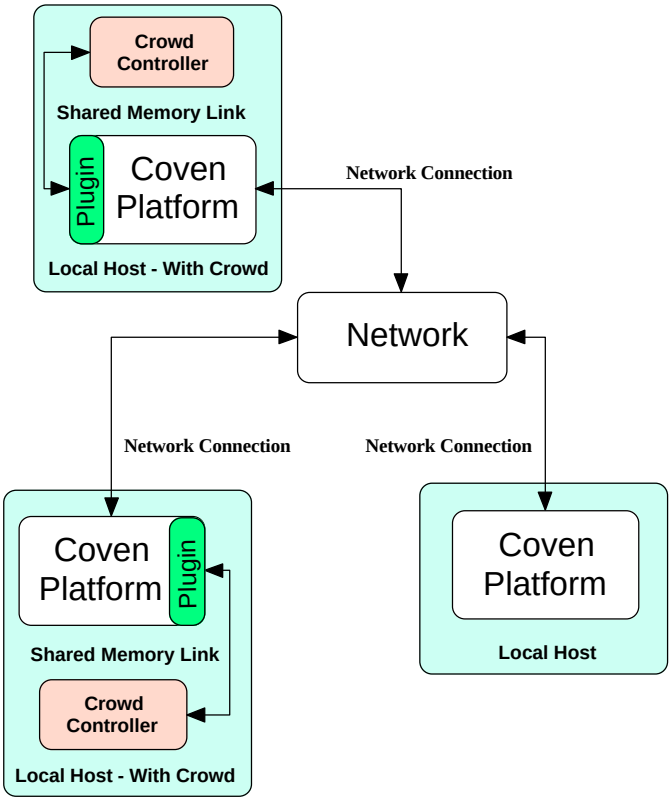


Figure 13 Overview of a network with two clients with a locally controlled crowd.

The system is exactly the same than for the user's avatar animation, except for the needed information for the external crowd controller. The shared memory segment is used to communicate in both directions between the COVEN platform and the crowd controller. In one direction, the COVEN platform sends to the crowd controller the user instructions and the world events. The crowd controller reacts to those instructions and events with respect to the crowd script which controls the crowd behaviour. As a result, the updated agents´ positions

are communicated through the shared memory to the COVEN platform.

Whatever the structure of the 3D shared application, crowds can be controlled at the central server level or at the level of each client which permits to any participant to import a crowd in the world, running locally an external crowd controller (Figure 13).

In the same way as for the avatar controller, a participant who does not have the needed plugin for crowd animation will still be able to see other crowds.

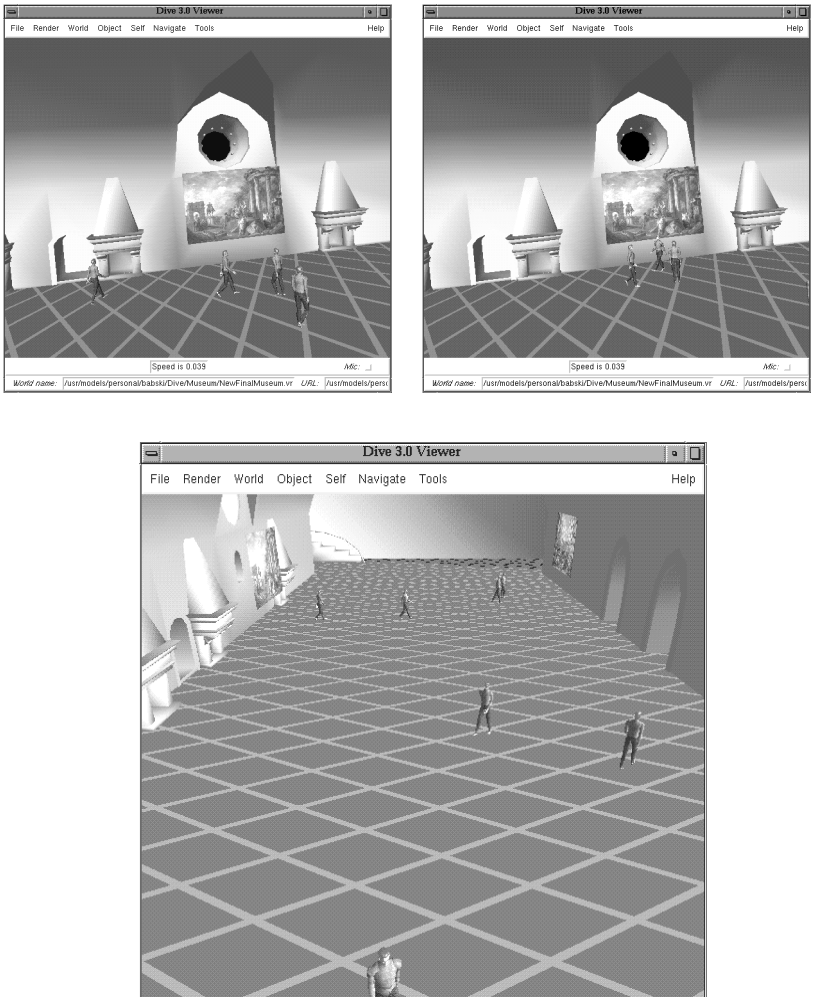


Figure 14 A museum simulation involving 3 crowds in Dive application.

# Human representation in MPEG-4: COVEN contribution

In an industry-wide effort beginning in mid-1990ies, the Moving Pictures Expert Group – MPEG (ISO/IEC JTC1/SC29/WG11) has recently produced versions 1 and 2 of the MPEG-4 International Standard (ISO/IEC 14496). In a world where audio-visual data is increasingly stored, transferred and manipulated digitally, MPEG-4 has set its objectives beyond 'plain' compression. Instead of regarding video as a sequence of frames with fixed shape and size and with attached audio information, the video scene is regarded as a set of dynamic objects. Thus the background of the scene might be one object, a moving car another, the sound of the engine the third etc. The objects are spatially and temporally independent and therefore can be stored, transferred and manipulated independently. The composition of the final scene is done at the decoder, potentially allowing great manipulation freedom to the consumer of the data.

Video and audio acquired by recording from the real world is called natural. In addition to the natural objects, synthetic, computer generated graphics and sounds are being produced and used in ever increasing quantities. MPEG-4 enables integration of synthetic objects within the scene. It provides support for 3D Graphics, synthetic sound, Text to Speech, as well as synthetic faces and bodies. In this paper we concentrate on the representation of bodies in MPEG-4, and in particular the efficient coding of body animation.

The following section provides the introduction to the representation of bodies in MPEG-4. We explain how Body Animation Parameters and Body Definition Parameters are used to define the shape and animation of bodies. Then we present our approach for efficient coding of Body Animation Parameters and Body Definition Parameters which we contributed to MPEG-4 as part of the COVEN project.

## Body Animation in MPEG-4

An FBA object in MPEG-4 is the representation of the face and body including their representation and animation. Conceptually the FBA object consists of a collection of nodes in a scene graph which are animated by the FBA object bitstream. The shape, texture and expressions of the face are generally controlled by the bitstream containing instances of Facial

Definition Parameter (FDP) sets and/or Facial Animation Parameter (FAP) sets. Upon construction, the FBA object contains a generic face with a neutral expression and a generic body with a default posture. This model can already be rendered. It is also immediately capable of receiving the FAPs and BAPs from the bitstream, which will produce animation of the face and body. If FDPs and BDPs are received, they are used to transform the generic model into a particular model determined by its shape and (optionally) texture.

Upon construction, the Body object contains a generic virtual human or human-like body with the default posture. This body can already be rendered. It is also immediately capable of receiving the BAPs from the bitstream, which will produce animation of the body. If BDPs are received, they are used to transform the decoder's generic body into a particular body determined by the parameter contents. Any component can be null. A null component is replaced by the corresponding default component when the body is rendered. Similar to the face, the BAPs can be transmitted also without first downloading BDPs, in which case the decoder animates its local model.

No assumption is made and no limitation is imposed on the range of defined mobilities for humanoid animation. In other words the human body model should be capable of supporting various applications, from realistic simulation of human motions to network games using simple human-like models.

## Structure of the FBA bitstream

A face and body object is formed by a temporal sequence of face and body object planes. An FBA object represents a node in an ISO/IEC 14496 scene graph. An ISO/IEC 14496 scene is understood as a composition of Audio-Visual objects according to some spatial and temporal relationships. The scene graph is the hierarchical representation of the ISO/IEC 14496 scene structure (see ISO/IEC 14496-1).

Alternatively, an FBA object can be formed by a temporal sequence of FBA object plane groups (called segments for simplicity), where each FBA object plane group itself is composed of a temporal sequence of 16 FBA object planes

## Body Animation Parameters

BAP parameters comprise joint angles connecting different body parts. These include: toe, ankle, knee, hip, spine (C1-C7, T1-T12, L1-L5), shoulder, clavicle, elbow, wrist, and the hand fingers. The detailed joint list, with the rotation normals, are given in the following section.

Note that the normals of rotation move with the body, and they are fixed with respect to the parent body part. That is to say, the axes of rotation are not aligned with the body or world coordinate system, but move with the body parts.

The hands are capable of performing complicated motions and are included in the body hierarchy.

The unit of rotations is defined as $10^{-5}$ radians. The unit of translation BAPs (BAPs tr_vertical, tr_lateral, tr_frontal) is defined in millimeters.

## BAP Grouping

In order to further decrease the bandwidth requirements and facilitate communication, the joints comprising the body can be partitioned into a finite set of groups with respect to their interrelationships and importance. For example, joints related to the spine can be grouped. In this way, if the motion affects only one part of the body, only the joints of that part of the body which change in the frame are coded and sent through the bitstream to the server, and then other clients. For example, if the virtual human is waving with their right arm, only joints involved in moving the right arm are sent through the network.

We divide the body degrees of freedom into groups. Complete degrees of freedoms are given in MPEG-4 Version 2 (PDAM1) specification. The groups can be sent separately by introducing a mask for each group, and inserting this mask in the beginning of the message. The mask has the following format:

<13-bit mask><4-bit mask><dofs for each group in mask>...

For example, to send only arm joints, the message has the following format:

<0000000011000><0000><5floats><7floats>

This decreases the size of the message from 408 bytes to 60 bytes. Thus, with an additional overhead of 3 bytes, we can decrease the message size significantly.

## BAP Coding

For each joint in the state vector, the quantization module stores a quantum value. The quantum value indicates what the step size is going to be for that joint angle in the compressed representation of the joint angle parameter. Thus, a quantum value of 1 indicates the angle will be encoded with the most precision, and 255 indicates a lower precision. Note that each degree of freedom has a different precision requirement. Therefore different quantization step sizes are applied to each degree of freedom. The base quantization step sizes for each joint angle are presented in the next paragraph.

The actual formula to obtain quantized state vector $S'$ from $S$ is

Quantized Value ($i$) = StateVector($i$)/(Quantum($i$)*Global_Quantization_Value)

$\rightarrow$Rounded to the nearest integer (for each joint angle $i$)

During decoding, the dequantization formula works in reverse:

StateVector' ($i$) = QuantizedValue($i$) * Quantum ($i$) (for each joint angle $i$)

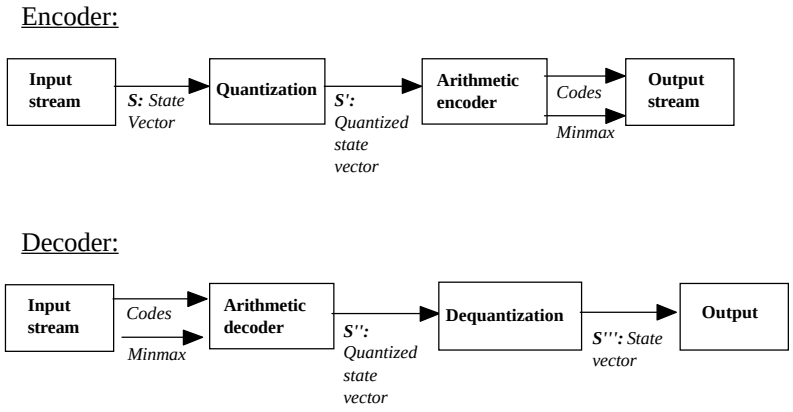


Figure 15: Dataflow of scalable compression

The bit rate is controlled by adjusting the quantization step via the use of a quantization scaling factor called *Global_Quantization_Value.* This value is applied uniformly to all DOFs. The magnitude of the quantization parameter ranges from 1 to 31. By modifying this value, we can control the bit rate requirements. For example, a global quantization value of 1 requires higher bit rates, changing it to 31 gives less accurate quantized values, letting the next step, arithmetic coding, to compress for lower bit rates. We measure the precision requirement for *Quantum*(*i*)

# Conclusions

In this paper we have presented an overview of some advanced aspects of Virtual Humans simulation for Networked Virtual Environments concerning the simulation of crowds: the crowd representation, crowd behavior, generating populations and Level of Detail management which is essential when large number of humans is simulated. We have described the integration of these techniques in the Coven-Dive platform and the contributions to the MPEG-4 standard resulting from this work.

# References

Akimoto T., Suenaga Z., and Wallace R.(1993), Automatic Creation of 3D Facial Models,

*IEEE Computer Graphics & Applications*.

Boulic, R., Capin, T., Huang, Z., Moccozet, L., Molet, T., Kalra, P., Lintermann, B., Magnenat-Thalmann, N., Pandzic, I., Saar, K., Schmitt, A., Shen, J., Thalmann, D. (1995). The HUMANOID Environment for Interactive Animation of Multiple Deformable HumanCharacters. Computer Graphics (Proceedings of Eurographics'95), 337-348

Boulic, R., Becheiraz, P., Emering, L., and Thalmann, D. (1997). Integration of Motion Control Techniques for Virtual Human and Avatar Real-Time Animation. *Computer Graphics (Proceedings of VRST '97)*, 111-118

Bouvier, E., Cohen, E., Najman, L. (1997). From crowd simulation to airbag deployment: particle systems, a new paradigm of simulation. *Journal of Electronic Imaging*, 6(1), 94-107.

Brogan, D. and Hodgins, J. (1997). Group Behaviours for Systems with Significant Dynamics. *Autonomous Robots*, 4, 137-153.

Burt P., and Andelson E. (1983), A Multiresolution Spline with Application to Image Mosaics, *ACM Transactions on Graphics*, 2(4):217-236.

Capin T.K., Pandzic I.S., Noser H., Magnenat Thalmann N., Thalmann D., (1997). Virtual Human Representation and Communication in VLNET Networked Virtual Environments, *IEEE Computer Graphics and Applications, Special Issue on Multimedia Highways*

Capin T.K., Pandzic I.S., Thalmann D., Magnenat Thalmann N. Avatars in Networked Virtual Environments, John Wiley, June 1999.

Dive, http://www.sics.se/dive

Fua P.(1998), Face Models from Uncalibrated Video Sequences, In *Proc. CAPTECH'98*, pp. 215-228.

http://www.turing.gla.ac.uk/turing/copyrigh.htm

Guenter B., Grimm C., Wood D. (1998). Making Faces. *Computer Graphics (Proceedings of SIGGRAPH'98)*, pp. 55-66

Ip H., Yin L.(1996), Constructing a 3D individual head model from two orthogonal views. *The Visual Computer*, Springer-Verlag, 12:254-266.

ISO/IEC 14496-1: MPEG-4 PDAM1, available on MPEG official web site:www.cselt.stet.it/mpeg

Kalra P, Mangili A, Magnenat-Thalmann N, Thalmann D (1992), Simulation of Facial Muscle Actions Based on Rational Free Form Deformations, *Proc. Eurographics'92*, pp. 59-69, NCC Blackwell.

Kalra, P., Magnenat-Thalmann, N., Moccozet, L., Sannier, G., Aubel, A., Thalmann, D., Real-time Animation of Realistic Virtual Humans, *IEEE Computer Graphics and Applications*, Vol.18, No5, 1998, pp.42-55.

Kendall, M.G. and Stuart, A. Advanced Theory of Statistics, vol. 3. Griffin, 1976.

Kurihara T. and Arai K. (1991), A Transformation Method for Modeling and Animation of the Human Face from Photographs, In *Proc. Computer Animation'91*, Springer-Verlag Tokyo, pp. 45-58.

Lee W. S., Kalra P., Magnenat-Thalmann N (1997), Model Based Face Reconstruction for Animation, In *Proc. Multimedia Modeling (MMM'97)*, Singapore, pp. 323-338.

Lee Z., Terzopoulos D., and Waters K. (1996), Realistic Modeling for Facial Animation, In *Computer Graphics (Proc. SIGGRAPH'96)*, pp. 55-62.

Moccozet L., Magnenat Thalmann N.(1997), Dirichlet Free-Form Deformations and their Application to Hand Simulation, In *Proc. Computer Animation'97*, IEEE Computer Society, pp.93-102.

Musse, S.R., Babski, C., Capin, T. and Thalmann, D. (1998). Crowd Modelling in Collaborative Virtual Environments. ACM VRST '98, Taiwan, (pp. 115-123).

Musse, S.R., Garat, F. and Thalmann, D. (1999). Guiding and Interacting with Virtual Crowds. Workshop on Computer Animation and Simulation of Eurographics'99 (pp. 23-33).

Pandzic I.S., Capin T.K., Lee E., Magnenat Thalmann N., Thalmann D., (1997). A flexible architecture for Virtual Humans in Networked Collaborative Virtual Environments. *Proceedings Eurographics 97*

Proesmans M., Van Gool L. (1997). Reading between the lines - a method for extracting dynamic 3D with texture. In *Proc. of VRST'97*, pp. 95-102.

Slater M., Usoh M. (1994) Body Centered Interaction in Immersive Virtual Environments, *Artificial Life and Virtual Reality*, N. Magnenat Thalmann, D. Thalmann, eds., John Wiley, pp 1-10

Thalmann D., Çapin T.K., Magnenat Thalmann N., Pandzic I.S. (1995) Participant, User-Guided and Autonomous Actors in the Virtual Life Network VLNET, *Proc. ICAT/VRST '95*, Chiba, Japan, pp. 3-11.