Realistic Avatars and Autonomous Virtual Humans in VLNET Networked Virtual Environments

Tolga K. Capin(1), Igor Sunday Pandzic(2),
Nadia Magnenat Thalmann(2), Daniel Thalmann(1)

(1) Computer Graphics Laboratory
Swiss Federal Institute of Technology (EPFL)
CH1015 Lausanne, Switzerland
{capin, noser, thalmann}@lig.di.epfl.ch
http://ligwww.epfl.ch/

(2) MIRALAB - CUI
University of Geneva
24 rue du General-Dufour
CH1211 Geneva 4, Switzerland
{Igor.Pandzic,Nadia.Thalmann}@cui.unige.ch
http://miralabwww.unige.ch/

Abstract

Networked Collaborative Virtual Environments (NCVE) have been a hot topic of research for some time now. However, most of the existing NCVE systems restrict the communication between the participants to text messages or audio communication. The natural means of human communication are richer than this. Facial expressions, lip movements, body postures and gestures all play an important role in our everyday communication. Part of our research effort in the field of Networked Collaborative Virtual Environments thrives to incorporate such natural means of communication in a Virtual Environment. This effort is mostly based on the use of realistically modeled and animated Virtual Humans. This paper discusses several ways to use virtual human bodies for facial and gestural communication within a Virtual Environment.

1. Introduction

The pace in computing, graphics and networking technologies together with the demand from real-life applications made it a requirement to develop more realistic virtual environments (Ves). Realism not only includes believable appearance and simulation of the virtual world, but also implies the natural representation of participants. This representation fulfills several functions:

- The visual embodiment of the user,
- The means of interaction with the world,
- The means of feeling various attributes of the world using the senses.

The realism in participant representation involves two elements: believable appearance and realistic movements. This becomes even more important in multiuser networked virtual environments (NVE), as participants' representation is used for communication. A NVE can be defined as a single environment which is shared by multiple participants connected from a different host. The local program of the participants typically store the whole or a subset of the scene description, and they use their own avatars to move around the scene and render from their own viewpoint. This avatar representation in NVEs has crucial functions in addition to those of single-user virtual environments:

- perception (to see if anyone is around)
- localization (to see where the other person is)
- identification (to recognize the person)
- visualization of others' interest focus (to see where the person's attention is directed)
- visualization of others' actions (to see what the other person is doing and what she means through gestures)
- social representation of self through decoration of the avatar (to know what the other participants' task or status is).

Using virtual human figures for avatar representation fulfills these functionalities with realism, as it provides the direct relationship between how we control our avatar in the virtual world and how our avatar moves related to this control. Even with limited sensor information, a virtual human frame that reflects the activities of the user, can be constructed in the virtual world; and this increases the sense of presence in this virtual world.

NVEs with virtual humans is emerging from two threads of research with a bottom-up tendency. First, over the past several years, many NVE systems have been created using various types of network topologies and computer architectures. The practice is to bring together different previously-developed monolithic applications within one standard interface; and consists of building multiple logical or actual processes that handle a separate element of the VE. Second, at the same time, virtual human research has developed to the level to provide realistic-looking virtual humans that can be animated with believable behaviors in multiple levels of control. Inserting virtual humans in the NVE is a complex task. The main issues are:

- selecting a scalable architecture to combine these two complex systems,
- modeling the virtual human with believable appearance for interactive manipulation,
- animating it with minimal number of sensors to have maximal behavioral realism,
- investigating different methods to decrease the networking requirements for exchanging complex virtual human information.

In this paper, we survey problems and solutions for these points, taking the VLNET (Virtual Life Network) system as a reference model. The VLNET system has been developed at

MIRALab at University of Geneva, and Computer Graphics Laboratory at Swiss Federal Institute of Technology, Lausanne. In VLNET, we try to integrate artificial life techniques with virtual reality techniques in order to create truly virtual environments shared by real people, and with autonomous living virtual humans with their own behavior, which can perceive the environment and interact with participants [Capin97][Noser96].

2. Realistic Avatars and Autonomous Virtual Humans: Design Issues

Realistic avatars representing participants, and autonomous virtual humans share similar design issues and techniques, while they have a number of differences. In this paper, we overview these differences and similarities.

In networked virtual environments, it is crucial to differentiate between user embodiments and other objects in the scene at first glance. The realistic human bodies are easily differentiable from the other 3D models and objects in the scene, hence using them as participant representation allows to identify them easily. Using this embodiment regularly, the participant has a bounded, authentic, and coherent representation in the virtual world. In addition, by changing decoration of the body through clothes and accessories, the representation also has an emergent identity.

Various applications have different requirements from avatar representation. For example, for a computer supported collaborative work application, the most realistic representation and animation of the body and the face might be the goal. On the other hand, a 3D chat application might have different modeling and animation requirements. For example, the current simple multiuser chat applications hides the real identity of the user, and this is likely increase the interaction as it eliminates the problem of shyness. The animation in these chat applications can be simple as playing predefined gestures of the body and the face, and can be complicated including interactions with the environment using the embodiment (e.g. grasping objects).

Including autonomous virtual humans that interact with participants increases the real-time interaction with the environment. Therefore, it is likely to increase the sense of presence of the real participants in the environment. The autonomous virtual humans are connected to the VLNET system in the same way as human participants, and also improve the usage of the environment by providing services such as replacing missing partners, helping in navigation. As these virtual humans are not guided by the users, they should have sufficient behaviors to act autonomously to accomplish their tasks. This requires building behaviors for motion, as well as appropriate mechanisms for interaction.

Our autonomous virtual humans are able to have a behavior, which means they must have a manner of conducting themselves. Behavior is not only reacting to the environment but should also include the flow of information by which the environment acts on the living creature as well as the way the creature codes and uses this information. Behavior of autonomous virtual humans is based on their perception of the environment.

These varieties in requirements for virtual human representation necessitate different approaches to modeling and animation depending on the application. We have developed various applications, as described below, for investigation of these techniques. However, this variety in techniques should not eliminate the fact that these different types of virtual humans should be present together within a single environment in a standard way. In VLNET, we try to create truly virtual worlds with autonomous living actors with their own behavior where real people should be able to enter and meet their inhabitants. The ultimate objective is to build intelligent autonomous virtual humans able to take a decision based on their perception of the environment and their interaction with the real participants.

2.1 Control of Virtual Humans

Whether avatar or autonomous, there is a need for the virtual human to interact with the environment: it should be animated using motion control techniques, and it should be provided enough information about the environment for perception.

Previously, we introduced three types of virtual humans according to the techniques to control them, direct-controlled, user-guided, and autonomous [Capin97]:

- Directly controlled virtual humans: the joint and face representation of the virtual human is modified directly (e.g. using sensors attached to the body) by providing the geometry directly.

- User-guided virtual humans: the external driver *guides* the virtual human by defining tasks to perform, and it uses its motor skills to perform this action by coordinated joint movements (e.g. walk, sit).

- Autonomous virtual humans: the virtual human is assumed to have an internal state which is built by its goals and sensor information from the environment, and the participant modifies this state by defining high level motivations, and state changes (e.g. turning on vision behavior).

All these motion control techniques can be used for avatars representing real participants, as well as autonomous virtual humans. For example, a participant can control her representation by directly updating the body with magnetic trackers; or by guiding her embodiment through commanding the actor to sit, walk; or she can control by changing the high-level state of her embodiment. Similarly, the same control techniques can be used for autonomous actors too. Then, what makes the avatars and autonomous actors different? The main difference is not in how to control the embodiment, but in what kind of information that the networked virtual environment provides to the participant or the autonomous actor program. For example, consider navigation. The real participant sees the rendered images on her display or HMD, and decides which direction to walk. In this case, the NVE provides to the participant only the rendered image. Ideally, the same image should be the only information to the autonomous actor so that it can navigate by avoiding collisions with the environment. However, this would require excessive computing power for vision techniques for processing this image; and it could be done without losing the realism of the motion, as all the information for the scene is already stored. For this purpose, Renault et al. proposed a synthetic vision technique, where the environment is rendered through the eyes of the embodiment using the z-buffer hardware [Renault90]. Instead of storing the color of each pixel in the z-buffer, a pointer to the object is stored with the depth value. This simplified information should be given to the autonomous actor instead of the normal image. Therefore, the information provided to the real and autonomous actors is not the same. Typically, perceptions to the autonomous actor should be coded in this simplified way in order to avoid additional computation. We are continuing work on how to provide general information about the scene to external applications controlling autonomous actors, particularly in the visual, auditory and tactile senses, and behavioral coding of the virtual environment. The next section describes the VLNET system, which our work is founded on.

3. The VLNET System

Typically, the virtual environment simulation systems are complex software systems, therefore a modular design imposes itself. It is appropriate to design the runtime system as a collection of cooperating processes, each of which is responsible for its particular task. This also allows easier portability and better performance of the overall system through the decoupling of different tasks and their execution over a network of workstations or different processors on a multiprocessor machine. In VLNET, we use this multiprocess approach. Figure 1 shows the architecture of the VLNET client. We separate the two types of processes: the core VLNET processes, and external driver processes.

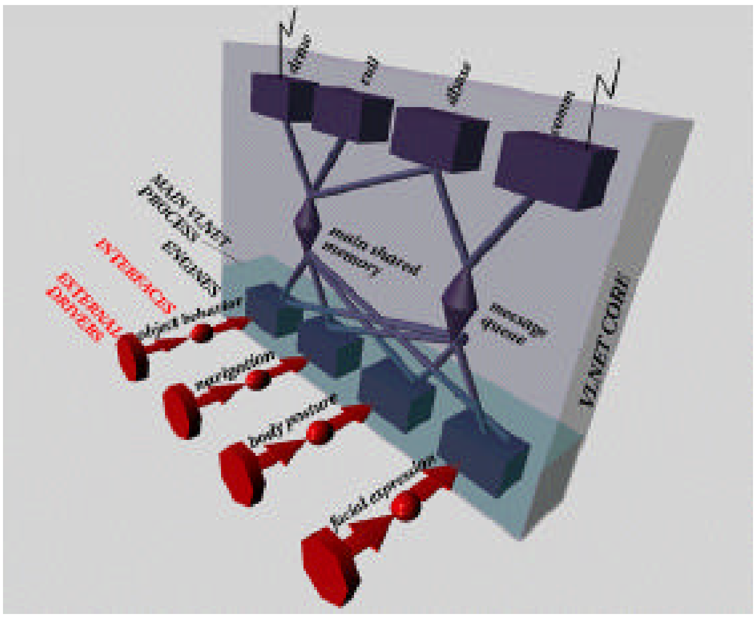

Figure 1: Client architecture of the VLNET system and its interface with external processes

3.1 VLNET Core Processes

Within the VLNET core, the main process executes the main simulation and provides services for the basic elements of VEs to the external programs, called drivers. The display, cull and database processes are standard IRIS Performer processes, and allow asynchronous loading and display of the scene with the main simulation. The main process consists of four logical units, called engines. The role of the engine is to separate one main function in the VE to an independent module, and provide an orderly and controlled allocation of VE elements. Moreover, the engine manages this resource among various programs which are competing for the same object. The communication process is responsible for receiving and sending messages through the network, and uses incoming and outgoing message queues to implement asynchronous communication.

The object behavior engine is responsible for the requests for changing or querying the object definition and behaviors in the scene, and collision detection among them. The navigation engine connects the user input to the navigation, picking and manipulation of objects. The input is in the form of relative and absolute matrices of the global position of the body, and the requests for picking or releasing an object.

Similarly, the face and body representation engines are specialized for the virtual human figure. The face engine is responsible for bridging between VLNET and external face drivers. The engine obtains the camera video images or face model parameters discussed below from the external face driver, and places in VLNET internal shared memory and outgoing message queue.

The body representation engine has an external interface for the body posture including joint angles or global positioning parameters, and high level parameters to animate the body. The role of this engine is to provide possibilities to define multiple levels of control for the human body, and to merge the output of different external body drivers to a single final posture.

3.2 External Drivers

The drivers provide the simple and flexible means to access and control all the complex functionalities of VLNET. Simple, because each driver is programmed using a very small API that basically consists of exchanging crucial data with VLNET through shared memory.

Flexible, because using various combinations of drivers it is possible to support all sorts of input devices ranging from the mouse to the camera with complex gesture recognition software, to control all the movements of the body and face using those devices, to control objects in the environment and stream video textures to them, to build any amount of artificial intelligence in order to produce autonomous or semi-autonomous virtual humans in the networked virtual environment.

The Drivers are directly tied to the Engines in the VLNET Main Process. Each engine provides a shared memory interface to which a driver can connect. Most drivers are optional and the system will provide minimal functionality (plain navigation and manipulation of objects) without any drivers. The drivers are spawned by the VLNET Main Process on the beginning of the session, based on the command line where all combinations of drivers can be specified. The drivers can be spawned on the local host or on a remote host, in which case the transparent networking interface processes are inserted on both hosts. In a simple case, as with most drivers shown in Figure 1, a driver controls only one engine. However, it is possible to control more than one engine with a single driver, insuring synchronization and cooperation.

The **Facial Expression Driver** is used to control expressions of the user's face. The expressions are defined either using the Minimal Perceptible Actions (MPAs) [Kalra 93]. The MPAs provide a complete set of basic facial actions, and using them it is possible to define any facial expression. Examples of existing facial expression drivers include a driver that uses the video signal from the camera to track facial features and map them into the MPAs describing expressions and a driver that lets the user choose from a menu of expressions or emotions to show on his face. The facial expression driver is optional.

The **Body Posture Driver** controls the motion of the user's body. The postures are defined using a set of joint angles corresponding to 75 degrees of freedom of the skeleton model used in VLNET. An obvious example of using this driver is direct motion control using magnetic trackers [Molet 96]. A more complex driver is used to control body motion in a general case when trackers are not used. This driver connects also to the Navigation Interface and uses the navigation trajectory to generate the walking motion and arm motion. It also imposes constraints on the Navigation Driver, e.g. not allowing the hand to move further then arm length or take an unnatural posture. This is the standard body posture driver which is spawned by the system unless another driver is explicitly requested.

The **Navigation Driver** is used for navigation, hand movement, head movement, basic object manipulation and basic system control. The basic manipulation includes picking objects up, carrying them and letting them go, as well as grouping and ungrouping of objects. The system control provides access to some system functions that are usually accessed by keystrokes, e.g. changing drawing modes, toggling texturing, displaying statistics. Typical examples are a spaceball driver, tracker+glove driver, extended mouse driver (with GUI console). There is also an experimental facial navigation driver letting the user navigate using his head movements and facial expressions tracked by a camera [Pandzic 94]. If no navigation driver is used, internal mouse navigation is activated within the Navigation Engine.

The **Object Behavior Driver** is used to control the behavior of objects. Currently it is limited to controlling motion and scaling. Examples include the control of a ball in a tennis game and the control of graphical representation of stock values in a virtual stock exchange.

The **Video Driver** is used to stream video texture (but possibly also static textures) onto any object in the environment. Alpha channel can be used for blending and achieving effects of mixing real and virtual objects/persons. This type of driver is also used to stream facial video on the user's face for facial communication [Capin97].

3.3 VLNET Server

A VLNET server site consists of a HTTP server and a VLNET Connection Server. They can serve several worlds, which can be either VLNET files or VRML 1.0 files. For each world, a World Server is spawned as necessary, i.e. when a client requests a connection to that particular world. The life of a World Server ends when all clients are disconnected.

Figure 2 schematically depicts a VLNET server site with several connected clients. A VLNET session is initiated by a Client connecting to a particular world designated by a URL. The Client first fetches the world database from the HTTP server using the URL. After that it extracts the host name from the URL and connects to the VLNET Connection Server on the same host. The Connection Server spawns the World Server for the requested world if one is not already running and sends to the Client the port address of the World Server. Once the connection is established, all communication between the clients in a particular world passes through the World Server.

In order to reduce the total network load, the World Server performs the filtering of messages by checking the users' viewing frusta in the virtual world and distributing messages only on as-needed basis. Clients keep the possibility of contouring this mechanism by requesting a higher delivery insurance level for a particular message, e.g. for heartbeat messages of a dead reckoning algorithm [Capin 97-1].

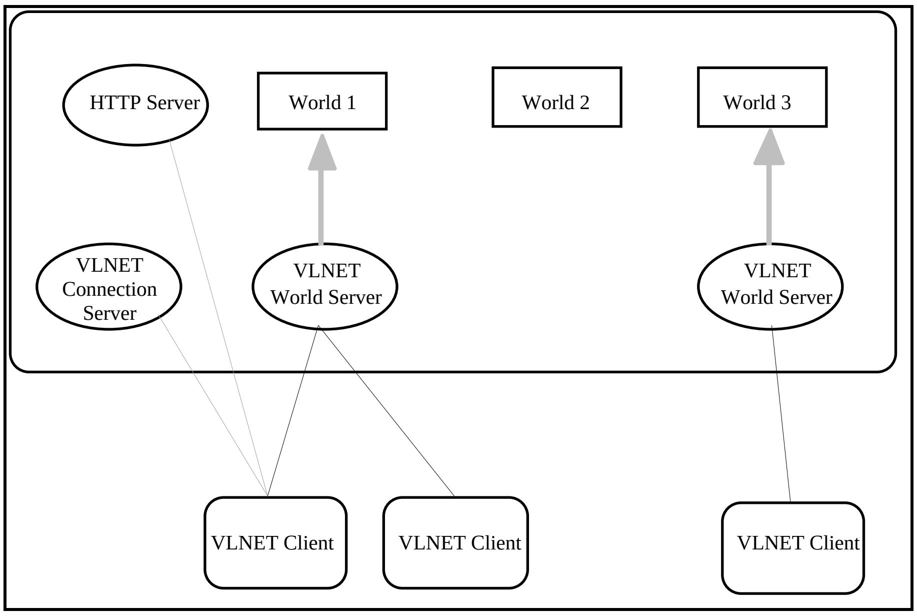

Figure 2: Connection of several clients to a VLNET server site

4. Communication in VLNET

Natural human communication is based on speech, facial expressions and gestures. Ideally, all these means of communication should also be supported within a Networked Collaborative Virtual Environment. This means that the user's speech, facial expressions and hand/body gestures should be captured, transmitted through the network and faithfully reproduced for the other participants on their sites. The capturing should be done in a non-intrusive way to increase interaction.

Obviously, the way to a complete system as described above is long and paved with problems. Capturing facial expressions or gestures non-intrusively and with enough precision is a complicated task. The synthesis of realistically looking human bodies and faces, and their animation in real time is also very demanding. Communication protocols must insure that the multi-modal data is transmitted to all the participants, and in the final synthesis the multi-modal outputs have to be synchronized.

We are trying to solve some of these problems within the Virtual Life Network system and to provide solutions leading to the complete communications as described above.

So far our work was not particularly concentrated on the audio (speech) communication. We use public-domain audio conferencing tools (VAT) to integrate this capability in the VLNET system. Therefore, audio communication is not discussed in this paper.

Next two sections will present several solutions for the facial communication, as well as some solutions for the gestural communication of the body.

Facial Communication

Facial expressions play an important role in human communication. They can express the speaker's emotions and subtly change the meaning of what was said. At the same time, lip movement is an important aid to the understanding of speech, especially if the audio conditions are not perfect or in the case of hearing-impaired listener.

We discuss four methods of integrating facial expressions in a Networked Collaborative Virtual Environment: video-texturing of the face, model-based coding of facial expressions, lip movement synthesis from speech and predefined expressions or animations.

Video-texturing of the face

In this approach the video sequence of the user's face is continuously texture mapped on the face of the virtual human. The user must be in front of the camera, in such a position that the camera captures his head and shoulders. A simple and fast image analysis algorithm is used to find the bounding box of the user's face within the image. The algorithm requires that head & shoulder view is provided and that the background is static (though not necessarily uniform). Thus the algorithm primarily consists of comparing each image with the original image of the background. Since the background is static, any change in the image is caused by the presence of the user, so it is fairly easy to detect his/her position. This allows the user a reasonably free movement in front of the camera without the facial image being lost. The video capture and analysis is performed by a special Facial Expression Driver.

Figure 3 illustrates the video texturing of the face, showing the original images of the user and the corresponding images of the Virtual Human representation.

Model-based coding of facial expressions

Instead of transmitting whole facial images as in the previous approach, in this approach the images are analyzed and a set of parameters describing the facial expression is extracted [Pandzic94]. As in the previous approach, the user has to be in front of the camera that digitizes the video images of head-and-shoulders type. Accurate recognition and analysis of facial expressions from video sequence requires detailed measurements of facial features. Currently, it is computationally expensive to perform these measurements precisely. As our primary concern has been to extract the features in real time, we have focused our attention on recognition and analysis of only a few facial features. The set of extracted parameters includes: vertical head rotation (nod), horizontal head rotation (turn), head inclination (roll), aperture of the eyes, horizontal position of the iris, eyebrow elevation, distance between the eyebrows (eyebrow squeeze), jaw rotation, mouth aperture, mouth stretch/squeeze.

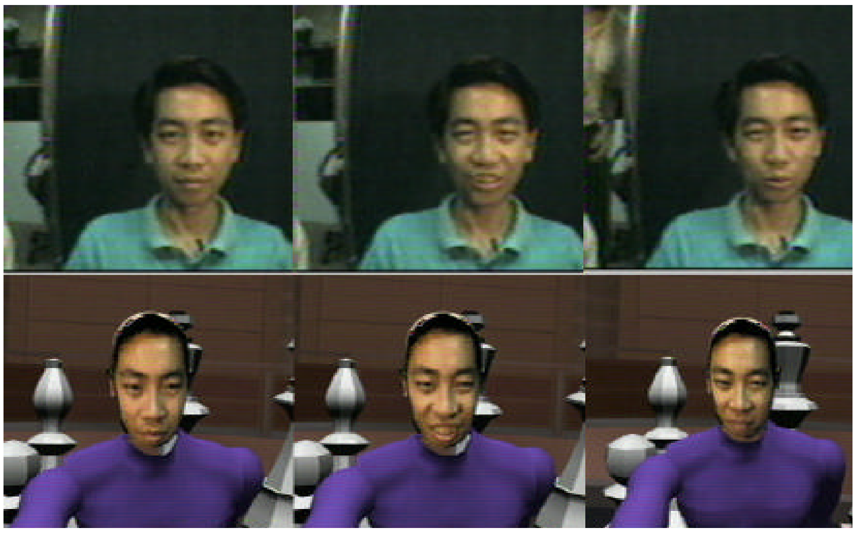

Figure 3: Video texturing of the face

The analysis is performed by a special Facial Expression Driver. The extracted parameters are easily translated into Minimal Perceptible Actions, which are passed to the Facial Representation Engine, then to the Communication process, where they are packed into a standard VLNET message packet and transmitted.

On the receiving end, the Facial Representation Engine receives messages containing facial expressions described by MPAs and performs the facial animation accordingly. Figure 4 illustrates this method with a sequence of original images of the user (with overlaid recognition indicators) and the corresponding images of the synthesized face.

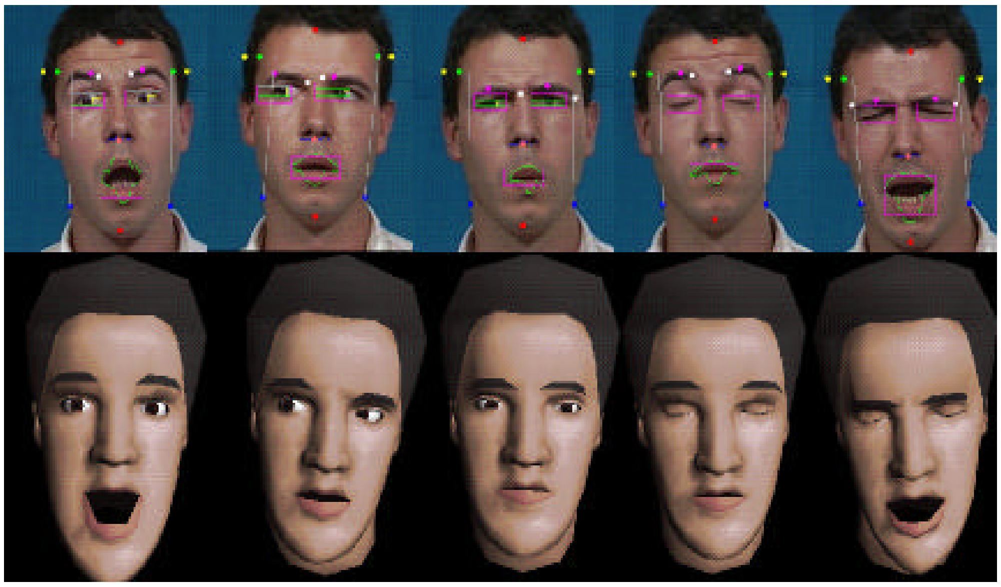

Figure 4: Model-based coding of the face - original and synthetic face

This method can be used in combination with texture mapping. The model needs an initial image of the face together with a set of parameters describing the position of the facial features within the texture image in order to fit the texture to the face. Once this is done, the texture is fixed with respect to the face and does not change, but it is deformed together with the face, in contrast with the previous approach where the face was static and the texture was changing. Some texture-mapped faces with expressions are shown in figure 5.

Lip movement synthesis from speech

It might not always be practical for the user to be in front of the camera (e.g. if he doesn't have one, or if he wants to use a HMD). Nevertheless, the facial communication does not have to be abandoned. It is possible to extract visual parameters of the lip movement by analyzing the audio signal of the speech. An application doing such recognition and generating MPAs for the control of the face can be hooked to VLNET as the Facial Expression Driver, and the Facial Representation Engine will be able to synthesize the face with the appropriate lip movement. An extremely primitive version of such system would just open and close the mouth when there is any speech, allowing the participants to know who is speaking. A more sophisticated system would be able to actually synthesize a realistic lip movement which is an important aid for speech understanding.

Predefined expressions or animations

In this approach the user can simply choose between a set of predefined facial expressions or movements (animations). The choice can be done from the keyboard through a set of "smileys" similar to the ones used in e-mail messages. The Facial Expression Driver in this case stores a set of defined expressions and animations and just feeds them to the Facial Representation Engine as the user selects them.

Figure 5 shows some examples of predefined facial expressions.

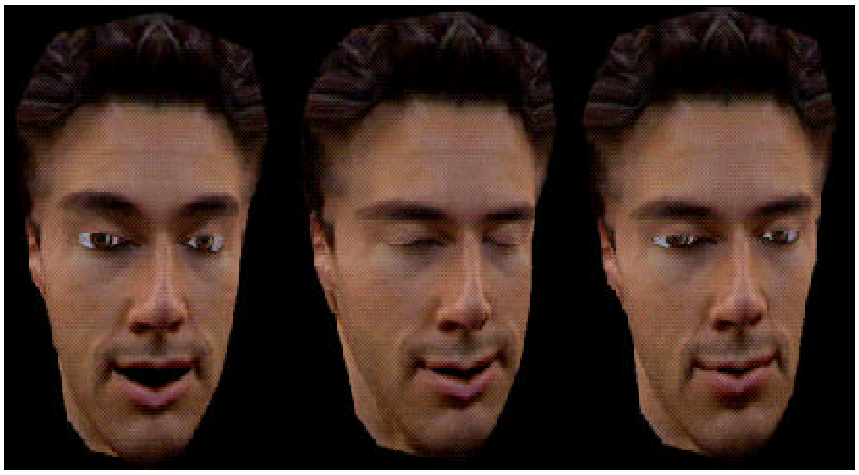

Figure 5: Predefined facial expressions - surprise, sleep, boredom

Gestural Communication

Gestures play an important role in human communication. Using the body, many messages can be communicated. The body movements can be roughly divided into three groups:

· *instantaneous gestures:* Most of the time, often even unconsciously, we accompany our speech with gestures. They stress the speech and give emphasis on particular words. They also very often have a meaning in themselves. The whole body posture also conveys information about the person's state and possibly emotions. For example, from the posture it can be determined if the person is tired, tense or relaxed.

· *gesture commands:* these are gestures that the user makes to specify some action. For example, the sign 'come here' can be speficied by raising the arm. These movements can change from one person or culture to another, therefore there is no well-defined set of rules for the meanings.

· *rule-based sign language:* these are gestures, for example used by deaf people that essentially follow well-defined rules to specify words or sounds. The signs typically work as a metaphor for defining other objects or language. The gestures can also be used by the software to define special tasks (e.g. showing forward direction to initiate a walk).

All these types of gestures can be controlled by two different methods: direct tracking and predefined postures or gestures. The type of control might be suitable for a specific type of gestures, however a combination of them can be used for different tasks.

Direct tracking

A complete representation of the participant actor's body should have the same movements as the real participant body for more immersive interaction. This can be best achieved by using a large number of sensors to track every degree of freedom in the real body. Molet et al.[Molet96] discuss that a minimum of 14 sensors are required to manage a biomechanically correct posture. However this is generally not possible due to limitations in the number and technology of the sensing devices, as it is either too expensive to have this many sensors, or it is too difficult for the participants to move with so many attached sensors. Therefore, the limited tracked information should be connected with behavioral human animation knowledge and different motion generators in order to "interpolate" the joints of the body which are not tracked [Capin96]. The main approaches to this problem are: inverse kinematics using constraints, closed form solutions, and motor functions.

The raw data coming from the trackers has to be filtered and processed to obtain a usable structure. The software developed at the Swiss Federal Institute of Technology [Molet96] permits to convert the raw tracker data into joint angle data for all the 75 joints in the standard HUMANOID skeleton used within VLNET [Boulic95][Capin97], with additional 25 joints for each hand. As shown in Figure 1, this software is viewed as Body Posture Driver by the VLNET system, and VLNET communicates with it through the Body Posture Interface. VLNET Body Representation Engine obtains this joint table from the Body Posture interface and uses such data to produce deformed bodies ready for rendering. The posture data in the form of joint angles fits into a VLNET message packet by reducing each angle to 8 bits, with a maximal error of 1.4 degrees in 360 degrees. This error rate is sufficient enough to provide body postures which is visually similar to the real body. By coupling the Flock of Birds driver with VLNET we can obtain full gestural communication in a very direct, though intrusive, way.

Predefined postures or body gestures

In a similar fashion as for the facial expressions, the body postures or gestures can also be predefined and chosen by a metaphor. For example, the smileys normally used within emails can be used to set a subset of the joints in the current body using the keyboard. The body posture driver just stores the predefined postures and gestures (i.e. animated postures) and feeds them to the Body Posture Engine as the user selects them. Figure 6 shows some examples of predefined postures.

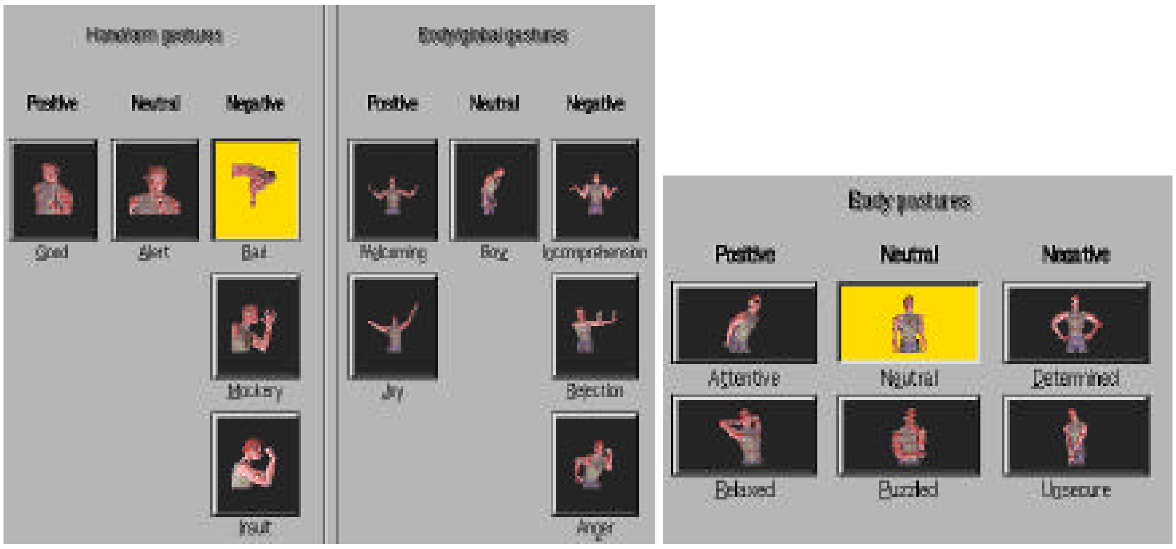

Figure 6: Basic set of gestures and postures

The main difference between direct control and predefined postures/gestures is that the direct control provides more correpondence to the real posture of the participant. Therefore, it is expected to provide more immersive feeling. However, the predefined postures can increase the communication among participants in networked environments in the absence of enough number of trackers. These two types of control can be combined to the animate the participant's body for different types of gestures. There is a need to investigate and define a set of tools that provide sufficient proprioceptive information for instantaneous gestures, while providing easy and natural control for rule-based signs and sign gesture commands.

5. Networking

The articulated structure of the human body together with the face introduces a new complexity in the usage of the network resources because the size of a message needed to convey the body posture is greater than the one needed for simple, non-articulated objects. This might create a significant overhead in communication, especially as the number of participants in the simulation increases. In order to reduce this overhead it is possible to communicate the body postures in more compact forms, accepting some loss of accuracy in the posture definition. This is not the only trade-off to be considered when choosing the optimal approach. Conversions between different forms of posture definition require potentially expensive computations which might induce more overhead in computation than was reduced in communication. The choice will also depend on the quality and quantity of raw data available from the input devices, the posture accuracy required by the application and the projected number of participants in the simulation.

For the networking analysis, we separate the discussion into body and faces, as they use different control methods and as they use different channels for communication. In any case, we can decompose the communication into three phases: coding, transmission, and decoding of the data. The transmission lag for a message will be the sum of the lag of all these phases.

In addition, each message type contains an accuracy loss of data which is a trade-off to decrease the lag. In this section, we analyze different message types with respect to the following aspects:

- *coding computation at the sender site:* we evaluate the amount of computation needed in order to convert the input data into the message to be sent, at the sending site;

- *bitrate requirements:* we evaluate the bandwidth requirements for different parameters to describe the motion. We assume a minimum limit for real-time computation as 10 frames/second.

- *decoding computation at the receiver site:* we evaluate the amount of computation needed to interpret the message and obtain the body posture(s) for display at the receiving site. The weight of this computation on the simulation is typically more than the one at the sender site because the messages from a potentially large number of participants have to be processed contrary to coding which is done only for the locally controlled figure.

- *accuracy loss:* We evaluate the loss of accuracy of the body posture with respect to the original input data. This is typically the trade-off to be considered against decreasing coding/decoding computations and transmission overhead.

We compare these issues in Figure 7 for various types of human body motion control. We consider four types of message packets that can be used to convey the body posture information:

- *global positioning parameters*: The global positioning of 17 body parts can be sent as 3 rotation and 3 translation values. This data can be used directly to display the body by-passing the conversion to joint angles.

- *joint angles.* These values are the degrees of freedom comprising the body, each represented by a floating point value. We evaluate three possibilities for the message type: the actual floating point representation of the angle, and 2-byte integer and 1-byte angle information, discretized between 0 and 360. This data has to be transformed into global positioning for display.

- *end-effector matrices.* A 4x4 floating point matrix is used to determine the position of the end effectors, in this example the head and the right hand. An inverse kinematics with two end-effectors (head and hand) has to be applied to the received message to obtain the final posture.

- *state information.* Only the high level state information is conveyed which makes the messages small. Moreover, the messages are sent only when state changes. The computation complexity involved to produce the posture(s) from the state information can range from quite simple (in the case of predefined static postures like sitting, standing) through medium (in the case of predefined dynamic states like walking or running) to very complex (in the case of more complex dynamic states like searching an object). In this evaluation, we take the medium-level walking action as an example.

We analyze three typical situations with respect to different real-time control data: i) complete body posture data is available, ii) only head and hand end-effector data is available, iii) walking motion guiding data is available from an external driver.

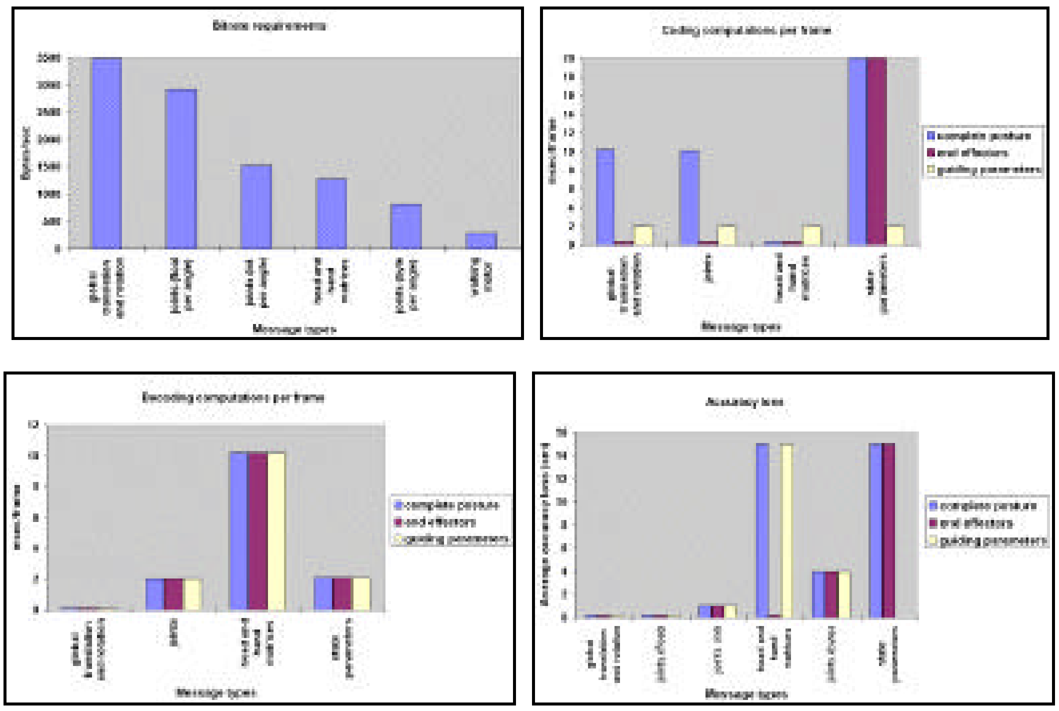

Figure 7: Networking body data a. Bitrate requirements for each message type b. Coding computations for each body posture c. Decoding computations for each body posture d. Accuracy loss with respect to original input data

Figure 7 shows the bitrate requirements, coding and decoding computations per frame, and accuracy loss for different message types. Figure 7a is calculated with the assumption of a minimum real-time speed of 10 frames/second. We see that the bitrate requirement varies between 4 Kbytes/sec and 240 bytes/second for one human body. Figures 7b and 7c show the coding and decoding results on an SGI Indigo2 Impact workstation with 250 MHz processor. The results show that there are a wide range of possibilities to define and transmit human figure information with respect to computation and bandwidth requirements. The choice of the control and message type will depend on the particular application requirements. Where high accuracy is needed (e.g. medical training applications) the transfer of body part matrices or at least end effector matrices will be required; in the large-scale simulation with numerous users it might be efficient to convey small messages containing the state information and use filtering and level of detail techniques to reduce the computational overhead. We chose the joint angles transfer as the optimal solution covering a wide range of cases since it offers fair or good results on all criteria, balancing the network and compression/decompression computational overhead (the traversal of the human hierarchy to convert joint values to transformation matrices of body parts, to be rendered on display), and accuracy loss. Using the state parameters for high level motions decreases the network bandwidth, however it requires a fast decoding process at the receiving site. The example walking motor showed the possibility of decreasing bandwidth requirements for sending motion data. Figure 7d shows the accuracy loss of posture data with varying message types and input methods. The results were computed by averaging the Euclidean distance between the corresponding body parts in the initial body posture at the sender's site and the decoded posture at the receiver's site.

Similarly, Figure 8 shows the bandwidth requirements and experimental results for coding and decoding of the face. Figure 8a shows that using model-based coding drastically decreases the network overhead, while Figures 8b and 8c demonstrate that the overheads of coding and decoding are insignificant.

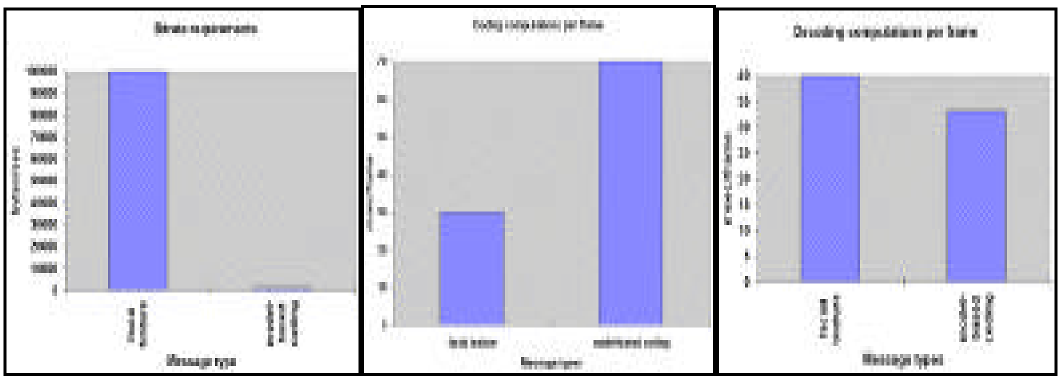

Figure 8: Networking face data a. Bitrate requirements for each message type b. Coding computations for face data c. Decoding computations for face data

As the number of participants increases, the transmission and decoding overheads will be excessive, and the speed might decrease significantly. Therefore, methods should be investigated to decrease this overhead. An approach is not to send the information to a site at all if there is no or little interaction, using filtering techniques [Pandzic97]. In addition, the *dead-reckoning* techniques may be applied to extrapolate the human information from the last received information of body and face, and the last speed. The initial results on human body dead reckoning have shown that up to 50% of the traffic can be decreased by applying simple predictive filtering on the joint angles [Capin97b].

6. Conclusions and Future Work

The human figure representation in networked virtual environments is not an easy task. First, we presented an easy architecture how they can be included in a complex NVE system. Next, we showed the different levels of controlling the human body, and compared them. Then, we presented different possibilities to send human information, and the load they put on the sender, network and receiver; as well as accuracy loss. The human figure information can put a load on the computational and networking resources, and the best control, representation and transmission form should be selected depending on the application and the resources.

Virtual human representation and communication in networked virtual environments is in an early stage. We will further our research on compression of virtual human models, and networking techniques to decrease communication requirements. The initial results are promising, and we hope to achieve very low bitrate virtual human communication.

Moreover, we will work on human motion control for direct, guided and autonomous virtual humans control. We are currently investigating techniques for providing simplified perceptual information about the virtual environment to the autonomous actors, and interaction between real participants and autonomous actors.

Acknowledgments

This research is financed by "Le Programme Prioritaire en Telecommunications de Fonds National Suisse de la Recherche Scientifique" and the TEN-IBC project VISINET.

Numerous colleagues at LIG and MIRALab have directly or indirectly helped this research by providing libraries, body and environment models, scenarios for applications, in particular Elwin Lee, Eric Chauvineau, Hansrudi Noser, Marlene Poizat, Laurence Suhner and Jean-Claude Moussaly. We would like to thank Mireille Clavien for designing body gestures and postures, and Anthony Guye Vuilleme for implementing this user interface.

References

[Boulic 95] Boulic R., Capin T., Huang Z., Kalra P., Lintermann B., Magnenat-Thalmann N., Moccozet L., Molet T., Pandzic I., Saar K., Schmitt A., Shen J., Thalmann D., "The Humanoid Environment for Interactive Animation of Multiple Deformable Human Characters", *Proceedings of Eurographics '95*, 1995.

[Capin97] T. K. Capin, H. Noser, D. Thalmann, I. S. Pandzic, N. Magnenat Thalmann, "Virtual Human Representation and Communication in VLNET Networked Virtual Environment", *IEEE Computer Graphics and Applications*, March 1997.

[Capin97b] T. K. Capin, I. S. Pandzic, N. Magnenat Thalmann, D. Thalmann, "A Dead-Reckoning Algorithm for Virtual Human Figures", *Proc. IEEE VRAIS'97*, IEEE Computer Society Press, 1997.

[Kalra 93] P. Kalra, "An Interactive Multimodal Facial Animation System", PhD Thesis nr. 1183, EPFL, 1993

[Molet 96] T. Molet, R. Boulic, D. Thalmann, "A Real-Time Anatomical Converter for Human Motion Capture", *Proc. Eurographics Workshop on Computer Animation and Simulation*, R. Boulic ed., Springer, Wien, 1996, pp.79-94.

[Noser96] H. Noser, T. K. Capin, I. S. Pandzic, N. Magnenat Thalmann, D. Thalmann, "Playing Games through the Virtual Life Network", *Proc. Artificial Life'96*, Chiba, Japan, 1996, pp.114-121.

[Pandzic97] I. S. Pandzic, T. K. Capin, E. Lee, N. Magnenat Thalmann, D. Thalmann, "A flexible architecture for Virtual Humans in Networked Collaborative Virtual Environments", *Proc. Eurographics'97*, Budapest, 1997.

[Pandzic 94] Pandzic I.S., Kalra P., Magnenat-Thalmann N., Thalmann D., "Real-Time Facial Interaction", *Displays*, Vol. 15, No 3, 1994.

[Renault90]] Renault O., Magnenat-Thalmann N., Thalmann D., "A Vision-based Approach to Behavioral Animation", The Journal of Visualization and Computer Animation, Vol.1, No.1, 1990.

[Thalmann96] D. Thalmann, J. Shen, E. Chauvineau, "Fast Realistic Human Body Deformations for Animation and VR Applications", Proc. Computer Graphics International '96, Pohang, Korea,1996.