Interaction with Virtual Human Crowds Using Artificial Neural Networks To Recognize Hands Postures

Abstract

Interaction between real and virtual humans spans several research topics from creation and animation of virtual actors to user-interfaces and virtual agents' control. In this paper we present a system to provide the user interaction with virtual human crowds using hands postures capture. An artificial neural network process recognizes the hand postures and a Client/Server system handles the communication between the several modules. In addition, ViCrowd, a system to manage crowds is responsible for dealing with crowd behaviors.

Key-Words: neural network, user interface, Client/Server system, hands posture recognition, human crowds' model.

1. Introduction

In the last few years, research on virtual humans concentrated its efforts on simulating virtual human behaviours and movements. Recently, the interaction between real and virtual humans has attracted many researchers due to the wide range of possible applications. In this paper we discuss an application of Virtual Reality (VR) technology associated to others sophisticated methods, e.g. neural networks, in order to provide a user-friendly interaction with virtual actors. The main advantage of this approach is the easy way of control it, direct it, and interact it with virtual actors (in spite of traditional techniques such as scripts and complicated graphical interfaces).

This paper discusses an application to interact with virtual human crowds *(Musse and Thalmann, 1997)* using hands posture capture. The hands postures are recognised using NeuSim *(Osorio, 1999)* which is a simulator of an Artificial Neural Network (ANN) based on the Cascade-Correlation learning algorithm *(Fahlman, 1990)*. A Client/Server system *(Schweiss et al, 1999; Musse et al, 1999)* is presented in order to integrate the different processes.

In addition, a case study to direct and interact with virtual human crowds in a theatre is discussed.

2. Related Works

Artificial Neural Networks have been used successfully in several applications such as medical diagnosis, robotics, image processing/classification, and speech and signal recognition. Recently, some applications integrating ANN and Computer Graphics have been developed in order to improve realistic effects *(Grzeszczuk et al, 1998)*. ANN are perfectly adapted to be applied in classification or regression tasks, where we have some examples available from the target problem. We use these examples within the ANN

learning algorithm to create a more powerful and generic model of the problem (generalisation). We have used in this project the system NeuSim, which is a simulator of an Artificial Neural Network that implements a Multi-Layer Perceptron (MLP) *(Rumelhart et al, 1986; Widrow et al, 1990)* architecture based on the Cascade-Correlation learning algorithm *(Fahlman, 1990)*. This ANN simulator is part of a more powerful machine learning system named INSS *(Osorio, 1998; Osorio and Amy, 1999).*

With respect to virtual human crowds, the emphasis has been on presenting different methods and techniques to model crowds motion and behaviours such as flocking systems *(Reynolds, 1987; Tu and Terzopoulos, 1994)*, physically based *(Bouvier et al, 1997; Brogan et al, 1998)* procedural-based, *(Thompson, 1995)* and hybrid systems (used in *AntZ, 1999*). Musse et al *(1998)* presented the term "guided crowds" which represent the group of virtual human agents that can be guided and directed during the simulation.

Considering the interaction using VR, many manipulations metaphors have been proposed. A recent overview of techniques for object manipulation, navigation and application control in Virtual Environments is presented by Hand *(1997)*. Enrico Gobbetti described Virtuality Builder (1994), an object-oriented architecture that allows the integration of various interaction techniques such as gestural input and constrained simulation. Poupyrev et al *(1997)* present a manipulation metaphor based on three main steps: Selection, Positioning and Orientation. Kallmann and Thalmann describe 3D interactions with Smart Objects (1999) which have the ability to describe its possible interactions.

3. Overview of the Architecture

In this paper, we are interested to propose an architecture to interact with virtual human crowds using VR technology. Our prototype uses a Cyber touch data glove from Virtual Technologies Inc.

Initially, the user programs hand postures and associates them to events in order to interact with virtual crowds. For instance, the event "clapping" can be associated to a specific posture such as presented in Fig. 1.

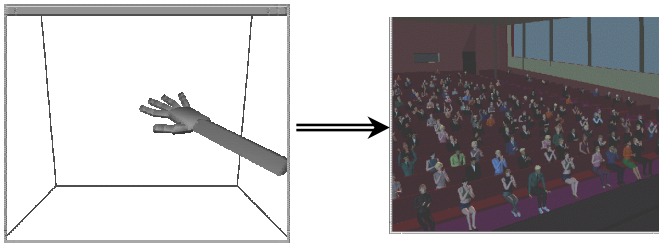

Fig. 1: Hand posture used to apply the clapping action.

During the simulation, the artificial neural networks module is responsible for the recognition of hand postures applied by the user. The recognised posture activates the associated event, which is sent to the ViCrowd module (*Musse et all, 1999*). The communication between the modules is made using the Client/Server system.

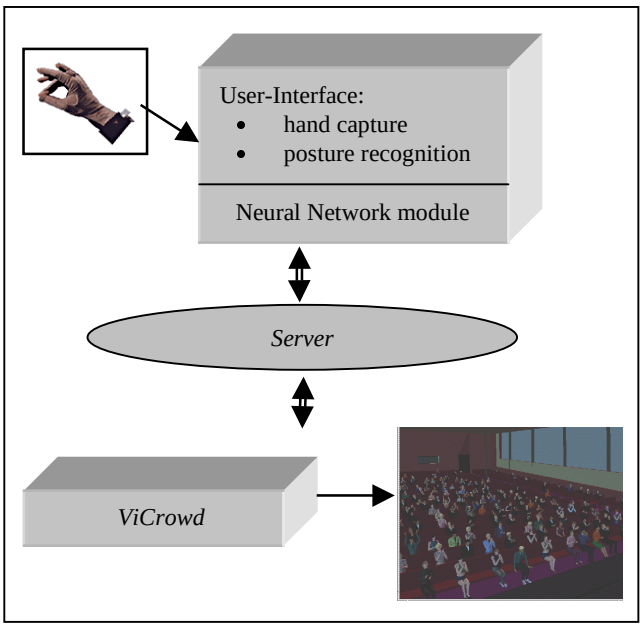

Fig. 2: Architecture of the system

The Client/Server system allows the interconnection of various kinds of clients such as: virtual environment database, Behavioural LISP system responsible for management of high-level behaviours, as discussed in previous works *(Farenc et al, 1999 and Schweiss et al, 1999).*

The ViCrowd system is responsible for the virtual crowd management and can treat messages received by other clients connected in the Client/Server system. When a posture is recognised, ViCrowd receives the event triggered as a function of posture and apply the reactions defined in the script language. In addition, ViCrowd deals with the display of the simulation. Fig. 2 shows the architecture of the system.

3.1. Virtual Crowd Management

This section aims at presenting some concepts of our crowd model and more explicitly about the guided crowd. The simulation of human crowds for populating virtual worlds provides a more realistic sense of virtual group presence. In some virtual environments, it would be useful to simulate populations in an autonomous way, thus the agents have a kind of environment knowledge and are able to move and interact within this environment. However, depending on the application, more ways of interaction can be required in order to provide a real time communication between participants and virtual agents. We have worked with three levels of autonomy: guided, programmed and autonomous in order to establish the required control of crowds, depending on the application.

Using ViCrowd script language, action, motion and behavioural rules are defined in order to specify the crowd behaviours. While action and motion describe explicit behaviours of crowd, called ***programmed crowd***, the behavioural rules are used to define ***autonomous crowd***. All these information can also be sent by an external process in order to guide crowds explicitly, during the simulation. We called this type of crowd as ***guided crowd***. The small window on the bottom right of Fig. 3 represents the Textual User Interface (TUI) client where textual commands can be specified. Fig. 3 shows a guided agent reacting according to a textual command: “GOTO Station”.

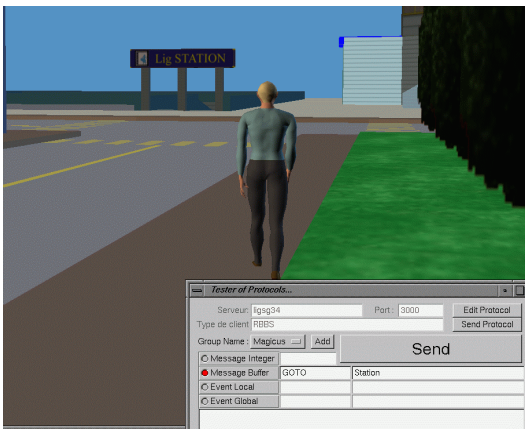

Fig. 3: A guided agent going to the train station as specified in the Textual User Interface Client (TUI).

The guided crowd represents groups of virtual agents, which can be externally guided. As the crowd in our approach is goal-based (the groups of agents always try to seek goals), the guided crowd receives dynamic goals in order to reach during the simulation. We have defined some interaction functions that set different ways to interact or guide crowds. The exchanged information is basically classified into 8 types:

1. Selection: Selection of a group/agent to interact.
2. Motion: Defines a new motion paradigm to the selected entity.
3. Action: Selection of an action to be applied.
4. State: Changes the internal status of group or agents.
5. Density: Increases or decreases the number of agents in the output area (camera's view).
6. Events/reactions: activates pre-programmed events and change event/reaction parameters.
7. Request: Requires about the selected entity.
8. Knowledge: Defines environmental data.

Each one of these functions presents different information that can be dealt in order to interact with a group or agent of the crowd. In this work, we have used mainly two of them: selection (aims at selecting the agents in the virtual environment which have to react as a function of triggered events) and events/reactions functions (which goal is to activate events).

When an event is activated, ViCrowd search in the behavioural rules for the appropriated reactions to be applied as well as the agents affected by the event. In this work, two hand postures are required in order to select the group of agents that have to react to the triggered event and the proper event. Listing 1 shows an example of the ViCrowd script language to define events and reactions externally specified in order to affect the virtual crowds. <EXTERNAL_INFO> describes information, which is sent by an external module, during the simulation.

```
EVENTS
        Event Clapping
                WHO EXTERNAL_INFO
                WHEN EXTERNAL_INFO
        end_event
        Event Do_nothing
                WHO ALL
                WHEN ALL
        end_event
REACTIONS
        Reaction_Clapping event Clapping
                KEYFRAME CLAPPING
        End_reaction
        Reaction_DoNothing event Do_nothing
                KEYFRAME NOTHING RANDOM
End_reaction
```

Listing 1: Example of events and reactions information.

While the external controller, through the simulation, activates event called Clapping, the Do_nothing event occurs during all the simulation (WHEN ALL) and is applied to all agents (WHO ALL). The reaction applied for both events describe key-frames postures animations (KEYFRAME CLAPPING).

3.2 Neural Networks Concepts

We have chosen to use a simulator named NeuSim *(Osorio, 1998)* to classify and recognize the hand postures captured from a data glove. NeuSim is a simulator of an Artificial Neural Network (ANN) that implements a Multi-Layer Perceptron architecture (MLP) *(Rumelhart et al, 1986, Widrow et al, 1990)* based on a supervised learning algorithm, Cascade-Correlation *(Fahlman and Lebiere, 1990).* This ANN simulator is part of a more powerful machine learning system named INSS *(Osorio, 1999).*

The use of Cascade-Correlation learning algorithm in NeuSim instead of Back-Propagation (a widely used model) *(Rumelhart, 1986),* allows quicker learning, with higher performance results *(Fahlman and Lebiere, 1990; Schiffmann et al, 1993).* This algorithm allows especially constructive learning. The Cascade-Correlation (CasCor) algorithm developed by Fahlman and Lebiere, in contrast to static neural learning algorithms such as Back-Propagation, is a generative technique to network construction and learning. Instead of merely adjusting weights in a network of fixed topology, Cascade-Correlation starts with a minimal network of input and output units. During learning, it may add hidden units one at a time, installing each one on a separate layer. This is done in the following way: if the learning process is not reducing output classification error fast enough with its current topology, it will select and install a new hidden unit (between input and output layers) whose output activation correlate best (over all training cases) with the existing network error. Once a new unit has been installed in the network its weights are frozen, and this unit keeps unchanged its learned weights. So, Cascade-Correlation will reduce step by step the network output error by cyclic process of output units learning and hidden unit addition/learning. In essence, the Cascade-Correlation algorithm searches not only in weight space but also in the space of network topologies.

NeuSim is used to classify hand postures. The Neural Simulator learns how to classify postures from some labeled examples (supervised learning). The learning is done in order to achieve the best generalization score, so the network will be able to classify new unseen postures more correctly. An example of NeuSim usage is discussed in Section 4.

3.3 The Client/Server System

This section discusses the Client/Server system as a way to provide communication between different processes.

Indeed, there are several ways to communicate information between different applications: shared memory, peer-to-peer connection, or Client/Server type architecture. We have chosen the last solution for its flexibility and its facility to add new features in adding new clients.

Shared memory would be faster than any connection via sockets, but it would imply that all applications are running on the same computer. In such case, it's difficult to have 2 distant users interacting in the same environment

Peer-to-peer connection assumes that each application is connected to other programs which will share information with it during simulations. This approach requires from user a good knowledge of all applications, and it suffers a lack of flexibility as changing one application means changing all other applications.

Architecture Client/server consists in a central application (server) to which other applications are connected (clients). Each client communicates to others via the server, without knowing any physical locations of them, nor the way to communicate. The server can be considered as a black box that externally offer services, and internally distributes information using a protocol, which is unique for each client.

The Server

The server is the central application that distributes the information sent from one client to another connected clients. Each client ignores the internal way of process of other clients, but knows which services they can offer via a protocol with the server. In this protocol, a request command specifies how to ask the server the list of clients and their services.

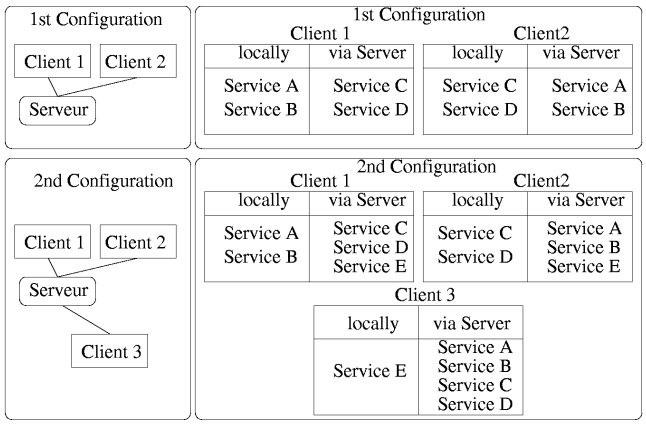

Fig. 4: Example of connections in the Client/Server system

The server does not know which clients will be connected, nor the services they can offer. Indeed, it allows clients to use some commands in order to send message or to manage priority of received messages. During the connection, each client sends to the server each type of message it can send (stimulus), and the sequence of server commands (reactions) that have to be performed for each received stimulus. All these information constitutes the protocol that allows client and server to communicate. In Fig. 4, we can see in the <1st configuration> that Client 1 can offer Service A and Service B and can use Service C and D from the Client 2 via the server. On the other hand, Client 2 can access to Service A and B from Client 1. When a third Client connects to the server <2nd configuration>, Client 1 and 2 can ask to Server which services Client 3 is able to offer. Once this request is done, Server sends them a message and updates the list of accessible services for all clients. Both clients 1 and 2 can access service E, and Client3 can then access services A,B,C, and D.

The Protocol

At the connection to the server, a client sends its protocol, in order to inform to the server which events and reactions can be treated. The protocol consist of stimulus, which are the messages sent from clients to server, and reactions, which are the sequence of commands that the server performs when a specific stimulus is received. Reactions can be executed sequentially or in a parallel way, and each stimulus can be set to a specific priority. Reactions are commands known by the server: to send messages, to update data field, to set variables, etc. Stimuli are always messages of type buffer (containing string of char) or integer ones.

3.4 Interface to Define and Recognize Hands Postures

The user interface allows the user to define new hand postures, which will be used to send orders to the agents of the virtual crowd, in order to make them react.

The cyberglove and the VirtualHand Inc. library were designed to give a huge amount of information regarding the current hand posture. For this application, we concentrated on a set of 27 relevant parameters. We were interested in keeping the Distal Interphalangeal Joint (DIJ), the Proximal Interphalangeal Joint (PIJ), the Metacarpal Phalangeal Joint (MPJ) and the abduction angle of each finger, as well as the wrist pitch, the wrist yaw, and the palm arch. The four last parameters are the relative angles between fingers (thumb-index angle, index-middle angle, middle-ring angle and ring-pinkie angle). Fig. 5 shows the hand model.

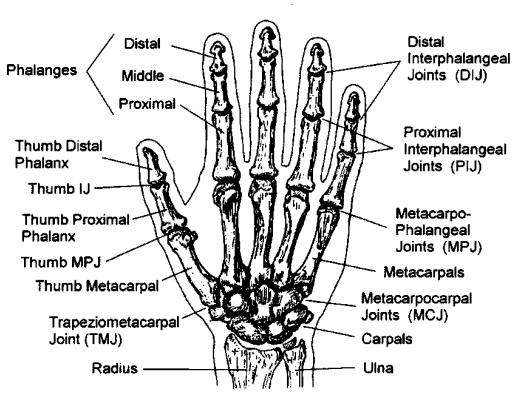

Fig. 5: Posterior dorsal view of right hand

Because of the accuracy of the values returned by the cyberglove, each hand posture is a unique instantiation of this set of parameters. Thus, the size and the shape of the hand change from one to another posture. VirtualHand Inc. provided a cyberglove calibration technique so that the glove is more adapted to the current user.

In order to use the interface, the user has to define new hands postures. An OpenGL window displays a virtual hand as a feedback for the user. Each time the user is satisfied with the defined posture, he/she snaps all the parameters in a file. Then, the recorded postures are stored in a database, which is used to train a neural network. A name is given to each posture, so that association with events can be done using significant names.

The neural network has been adapted to our particular representation of a hand. Its input-layer consists of 27 neurons, one for each parameter. The output-layer could have had many different configurations. We concentrated on representing a posture by a neuron (there are as many output neurons as the number of postures). This configuration simplifies the interpretation of the network output: only one neuron can be active at one time, i.e. the one whose posture has been recognized. The CasCor algorithm (discussed in Section 3.2) optimizes the network topology, until 100% of the patterns are correctly classified.

Once the postures are defined and saved in the database, they have to be associated with events in order to be applied by the crowd. The crowd events (Section 3.1) are related to the scene, which will be simulated. For instance, the events and reactions required to simulate a crowd in a theatre are not the same needed in a museum. The events defined in the simulation are described in ViCrowd script (Listing 1) and are sent to the module responsible for managing the interface when the simulation starts. Fig. 6 shows the user interface to associate hand postures and crowd events.

Each hand posture made by the user has a different meaning that can be classified in *selection* or *event*. The selection postures describe the group of virtual people that has to be affected by the events, and the event posture describes the events activated by the user and sent to ViCrowd. A complete discussion about a case study using this architecture is presented in the next section.

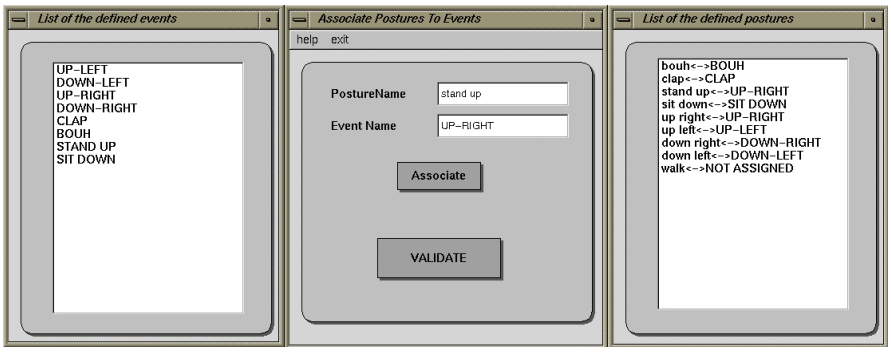

Fig. 6: Interface to associate postures and event: On the left, the window shows the names of defined events and on the right the names of postures. The window on the center provides functions to associate the information.

4. Case-Study: Interacting with a Crowd in a Theatre

The theatre is a virtual environment where the crowd can evolve and react as a function of triggered events. We have subdivided the theatre surface in order to define sub-groups that have to be affected by the events. There are four subgroups as illustrated in Fig. 7.

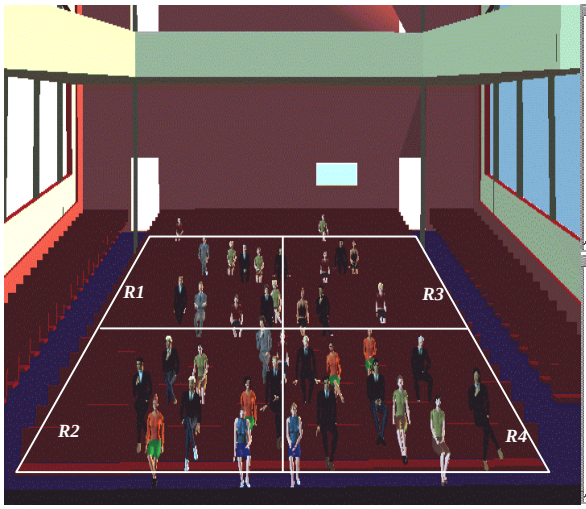

Fig. 7: Regions defined in the theatre

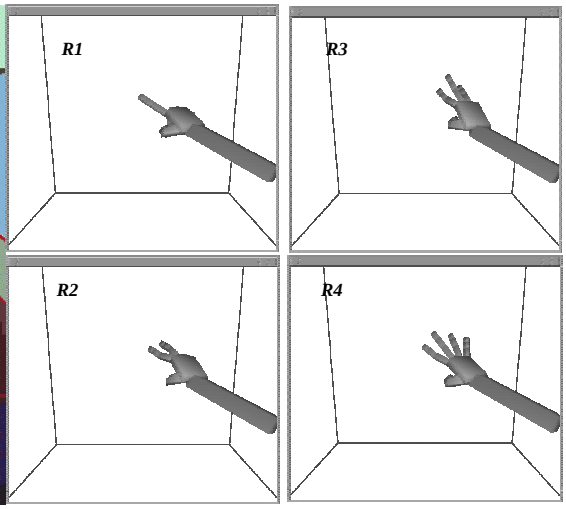

Fig. 8: Postures associated to regions in the theatre

Using the hand postures defined by the user, groups can be selected as a function of the pre-defined regions. We defined a different posture for each region as shown in Fig. 8.

There are 4 different actions that can be applied by the virtual groups: to stand up, to sit down, to clap and to hoot. In addition, 4 postures were associated to each action as shown in Fig. 9.

Concerning the neural network recognition, as we mentioned before (Section 3.4) we have defined 27 neurons in the input layer and 8 neurons in the output layer in order to recognise 8 different hand postures as illustrated in Fig. 10.

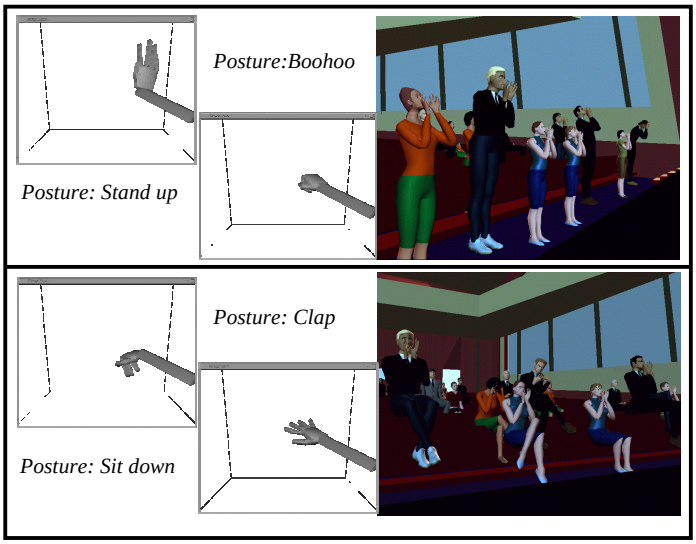

Fig. 9: Actions and postures programmed to the crowd

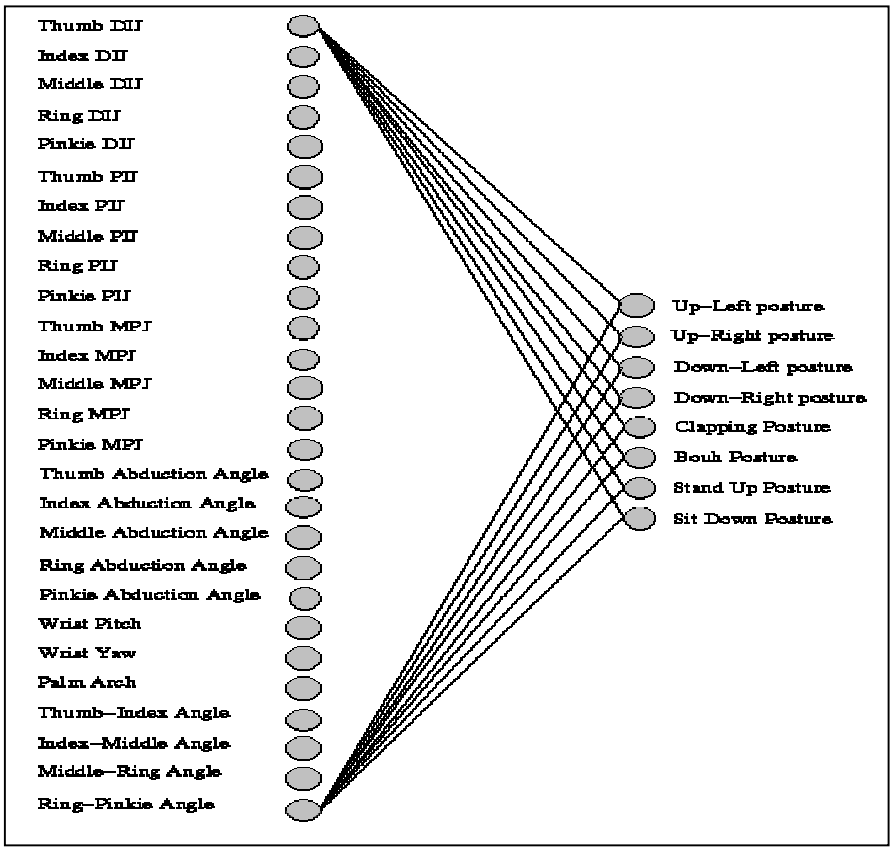

Fig. 10: Input and output layers used in neural network recognition

The frame rate (fps = frames/sec) obtained to simulate virtual agents in an Onyx 2, 2048 Mbytes of RAM, 128 TRAM (texture memory) and 6 processors varied according to the table and chart in Fig. 11.

Nb Agents	5	10	20	40	100
FPS	22	20,1	19,2	18,5	15

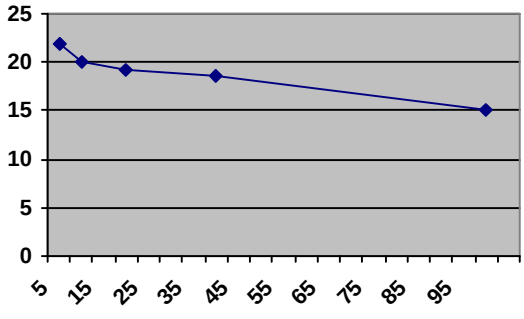

Fig. 11: Variation of frame rate obtained simulating from 5 to 100 virtual agents

Fig. 12 shows an user interacting with a virtual crowd in the theatre.

Fig. 12: User interacts with a virtual crowd in a Theatre

5. Final Remarks

We presented in this paper a framework dedicated to interact with virtual human crowds using VR devices.

Moreover, we presented the Client/Server system responsible for the integration between different applications, which are independent from each other. In addition, the neural network module (INSS) was concerned with the recognition of hand postures.

We believe that such approach of recognizing hand postures and interacting with virtual human crowds provides a flexible framework that can be used in different applications.

Yet, the Client/Server system provides the integration of different metaphors since it allows different and independent applications to communicate messages to each other. For instance, we are investigating the usage of speech recognition in order to guide crowds as well as the hand postures presented in this work. Our goal is to provide a multi-modal interface to guide and interact with virtual agents.

Acknowledgements

The authors are grateful to Olivier Renault for appearing in the photo of Fig. 12. The research was sponsored by the Swiss National Research Foundation, the Federal Office for Education and Science (ESPRIT eRENA project), FUNDEPE and CAPES - Fundação Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Brazilian office of Education and Science).

References

- AntZ (1999) http://www.antz.com.
- Bouvier E.; Cohen E. and L. Najman. (1997) "From crowd simulation to airbag deployment: particle systems, a new paradigm of simulation". Journal of Electronic Imaging 6(1), 94-107, January.
- Brogan, D.; Hodgins, J. (1997) “Group Behaviors for Systems with Significant Dynamics”. Autonomous Robots, 4, 137-153.

- Fahlman, S. E.; Lebiere, C. “The Cascade-Correlation Learning Architecture”. Carnegie Mellon University - CMU, Computer Science Technical Report - CMU-CS-90-100. February 1990. http://www.cs.cmu.edu/Reports/index.html .
- Gobbetti, E. “Virtuality Builder II, Vers une Architecture pour l’interaction avec des mondes synthétiques”. Thése de doctorat. Lausanne, EPFL, 1994.
- Grzeszczuk, R.; Terzopoulos, D. and Hinton, G. “NeuroAnimator: Fast Neural Network Emulation and Control of Physics-Based Models”. Proc. ACM SIGGRAPH’98 Conference, Orlando, FL, July, 1998, in Computer Graphics Proceedings, Annual Conference Series, 1998, pp. 9-20..
- Hand, C. « A Survey of « D Interaction Techniques », Computer Graphics Forum, 16(5), 269-281, 1997.
- M. Kallmann, D. Thalmann, Direct 3D Interaction with Smart Objects, Proc. ACM VRST 99, London, England.
- Musse, S.R.; Thalmann, D. (1997) “A Model of Human Crowd Behavior: Group Inter-Relationship and Collision Detection Analysis”. Proc. Workshop of Computer Animation and Simulation of Eurographics’97, Sept. Budapest, Hungary.
- Musse, S.R.; Babski, C.; Capin, T.; Thalmann, D. (1998) “Crowd Modelling in Collaborative Virtual Environments”. Proc. of ACM–VRST’98. Nov. Taipei, Taiwan.
- Musse, S. R., Garat, F. and Thalmann, D (1999). "Guiding and Interacting with Virtual Crowds". Proceedings of Workshop Eurographics Computer Animation and Simulation. Milan, Italy.
- Osorio, Fernando S. "INSS: Un systeme Hybride Neuro-Symbolique pour L'Apprentissage Automatique Constructif". These de Doctorat, INPG/IMAG - Laboratoire Leibniz. Grenoble, France. 1998. http://www-leibniz.imag.fr/RESEAUX
- Osorio, F. S. & Amy, B. “INSS: A Hybrid System for Constructive Machine Learning”. NeuroComputing 28(1999) 191-205. Elsevier Science, Amsterdam.
- Poupyrev, I.; Weghorst, S.; Billinghurst, M. and Ichikawa, T. “A Framework and Testbed for Studying Manipulation Techniques for Immersive VR”, Proceedings of the ACM Symposium on Virtual Reality Software and Technology, VRST, Lausanne, Switzerland, 21-28, 1997.
- Reynolds, C. (1987) “Flocks, Herds and Schools: A Distributed Behavioral Model”. Proc. SIGGRAPH’87, Computer Graphics, v.21, n.4, July.
- Rumelhart, D.; Hinton, G. & Willians, R. “Learning Internal Representations by Error Propagation”. In : Rumelhart & McClelland: Parallel Distributed Processing - Explorations in Microstructure of Cognition - Vol.1. Cambridge: MIT Press, 1986.
- Schiffmann, W.; Joost, M. & Werner, R. “Comparison of Optimized Backpropagation Algorithms”. Proceedings of ESANN'93, Brussels, p.97-104, 1993. http://www.uni-koblenz.de/~schiff/ publications.html
- Schweiss, E., S. R. Musse, and F. Garat. 1999. An Architecture to Guide Crowds based on rule-based systems. Autonomous Agents'99, Seattle, Washington, USA.
- Thompson, P.A., Marchant, E.W. (1995) “A Computer Model for the Evacuation of Large Building Populations”. Fire Safety Journal, n. 24, pp-131-148.
- Tu, X.; Terzopoulos, D. (1994) “Artificial Fishes: Physics, Locomotion, Perception, Behavior”. Proc. SIGGRAPH’94, Computer Graphics, July.
- Widrow,B.; Lehr,M.“30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Back-Propagation”. Proc. of the IEEE, NY, V.78(9), pp.1415-1441. Sept. 1990.