DEXTROUS HAPTIC INTERFACE FOR JACK^{Tm}

Viorel G. Popescu and Grigore C. Burdea
Department of Electrical and Computer Engineering,
Rutgers-The State University of New Jersey,
Piscataway, N.J. 08854, USA.
http://www.caip.rutgers.edu/vrlab/

ABSTRACT

This paper presents a simulation system which integrates a real-time hand sensing and force-feedback interface (the Rutgers Master II) with a full body modeling software (JACK^{Tm}). The RM–II system and the Polhemus Fastrak^{Tm} 3D magnetic tracker were initially integrated with "JACK" full-body simulation software. The system performance (graphics update rates, sensor reading updates) is evaluated on two graphic workstations (SGI High Impact and SGI Infinite Reality). Next we propose the concept of a Virtual Human Agent enhanced with haptic feedback. This concept is illustrated by a simulation developed using the RM–II system, JACK software library and a speech recognition engine.

INTRODUCTION

Many haptic systems are currently limited to providing feedback to only one location of user's body (mainly the hand) [1]. Due to advances in VR technology [2] and increased workstation computation power the idea of a full body haptic suit comes closer to reality. Virtual humans can be used as graphic interfaces for such complex VR simulations. They have been used mostly for human factors design, ergonomics visualization and virtual prototyping and less in real-time simulation systems. Therefore some limitations are inherent to the currently available Virtual Humans when used in real-time applications.

Designing complex systems with full-body tracking and force feedback require very powerful simulation tools. However, tracking and feeding back forces for those parts of the body with the highest haptic activity for a given application can be currently realized with a single workstation and additional VR I/O devices. This paper presents an implementation of a Virtual Human enhanced with force feedback for the hand.

The following section presents the integration of the JACK software with the RM–II haptic interface. The system performance is evaluated for two platforms, medium and high-end graphic workstations. A proof-of-concept demonstration of a Virtual Human Agent (VHA) performing tasks in a machine shop is described next. Multimodal input needed to communicate with the VHA is outlined briefly. Graphics and sensors update rates are evaluated subsequently. Concluding remarks are given in the last section.

THE HAPTIC INTERFACE SYSTEM

System Components

JACK^{Tm} [4] is a full-body simulation environment which provides 3D modeling capabilities as well as extensive human factors and analysis tools. The software provides a 3D human model currently used for human factors design, ergonomics visualization and virtual prototyping. The articulated human figure consists of 39 segments, 38 joints, and 88 degrees of freedom, including a 17-segment flexible torso; the fully-articulated hands add 30 segments, 30 joints, and 33 degrees of freedom. The JACK software includes modules for torque and strength computation for the human model, anthropometric scaling, walking behaviors, balance control, reach and grasp behaviors, real-time animation previewing, real-time collision detection, free-form deformations of surfaces, import and export interfaces to CAD, and support for several input devices (Ascension FOB, Cyberglove). The real-time application developer is provided with an API which includes a set of "C" functions which allow accessing components of the virtual environment and issuing simulation commands. Command files (in "jcl" format) and geometry files (in "psurf" format) complement the development toolkit. Objects created with CAD modeling packages (such as Autocad) and converted to "psurf" format can be loaded in the simulated environment.

The Rutgers Master II (RM–II) system is a portable haptic interface designed for interaction with virtual environments [3]. Its two main subsystems, shown in Fig. 1, are the *hand–master* and the *Smart Controller Interface* (SCI).

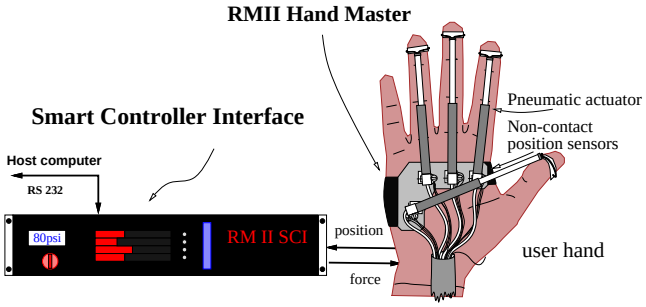


Figure 1: The Rutgers Master II system [3]

The RM–II hand master can read hand gestures (fingertip positions relative to the palm) and apply forces on the fingertip. The hand–master unifies hand motion sensing and force display in a small and compact structure which weighs approximately 100 grams. The main structure consists of a small "L" shaped palm base on which are mounted four custom-designed linear-displacement pneumatic actuators and 8 Hall effect sensors. The actuators are equipped with a special non-contact sensor to measure their linear displacement. A 3D magnetic tracker (Fastrack^{Tm}) [11] attached to the back of the hand provides wrist position/orientation.

RM–II and Fastrak^{Tm} devices were initially connected with the JACK simulation environment through the C API. The C API interface included in JACK v1.1, has a severe limitation for real-time application development: in this version the simulation loop executes one command (issued by the interface) per frame. Therefore hand motion tracking requires 16 commands: one for palm position and orientation and 5(fingers)x3(joints) joint adjustment commands. The next version of JACK (2.0) will allow for direct linking of C/C++ modules into the JACK run-time. Transom (Ann Arbor, MI), which commercializes JACK^{Tm}, provided us with a sample code for loading a module into JACK 1.1 while still programming against the C API. We were thus able to load the interface module at run-time; the module contains a simulation function (including all hand motion tracking commands) executed once per frame. The experimental setup is shown in Fig. 2 [13].

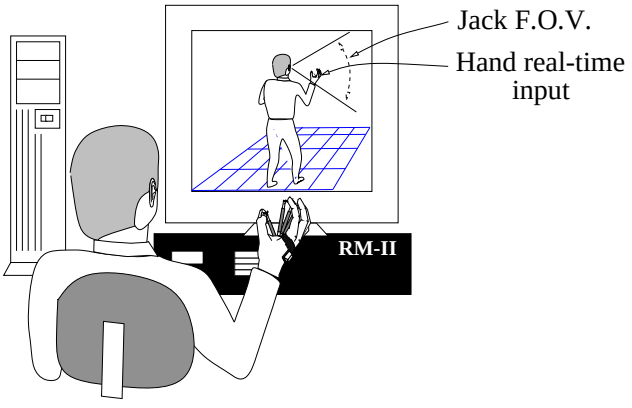


Figure 2: Hand input interface setup [13]

System Design

While JACK provides different types of automatic behaviors (reaching, grasping, walking), we explored the enhancement of JACK simulation system with real-time input and force feedback for the hand. The task to be implemented with the RM–II system is object grasping. In this case hand motion can be decomposed in two distinct phases: "reaching" and "grasping". "Reaching" is implemented using JACK inverse kinematics and Polhemus sensor readings. Arm motion uses the shoulder as the starting point and provides data only for palm position and orientation. The arm motion is solved using a real-time inverse kinematics mechanism implemented in JACK software [5]. RM–II sensor readings are used to control JACK fingers. Only a few "grasping" configurations [7] are achievable through our interface. The RM–II hand master limits by design user finger motions to nearly half of the maximum range. As a consequence, "circular" is the most suitable type of grasping for our system.

JACK is able to execute 15 out of 16 types of grasping configurations using an automatic behavioral control mechanism (PaT-Nets - Parallel Transition Networks) [8]. To implement grasping a contact detection queue has to be maintained for the hand. Finger positions are checked against the object to be grasped. For automatic behavior, this information is enough to control fingers' motion according to the constraints implemented in PaT-Nets. In our system, force feedback can be used to help control fingers' motion during grasping. When the palm has the appropriate position and orientation, force feedback guides the user's fingers in a stable grasping configuration according to the shape of the object. Unfortunately, in the absence of wrist and elbow force feedback, palm position and orientation cannot be controlled through this mechanism. Therefore hand position relative to the object to be grasped can be quite arbitrary, resulting in unsatisfactory results. Constraints have to be imposed to simulate a realistic grasping task. Two types of constraints can be implemented: fingers' positioning or the grasped object automatic positioning. The former solution gives priority to the graphic simulation realism and suspends user input over a short period of time. Once a grasping configuration is encountered, the hand real-time input is suppressed and automatic behavior activated, until the grasp action is completed and the object is attached to the hand. The latter solution maintains the direct input from the user's hand at the expense of unrealistic virtual object grasping. The position and orientation of the grasped object are set such as to optimize the grasping configuration (see Fig. 3 [13]).

Real-time constraints force us to choose the second approach, since automatic behavior used in the first solution is computationally intensive, leading to unacceptable time delays in user input. A contact detection queue is maintained during the simulation. The virtual human right hand is checked for intersections with objects in the simulated environment. If an intersection is found, the hand and intersecting objects are highlighted (in red), the grasped object is set in an optimal grasping position while information about object stiffness is sent to RM–II. The RM–II system provides force feedback to the user hand according to the specified object stiffness. Once a grasping configuration is detected, the object is attached to the hand until a release configuration is signaled. Gravity simulation adds realism to the virtual environment, as objects fall on the

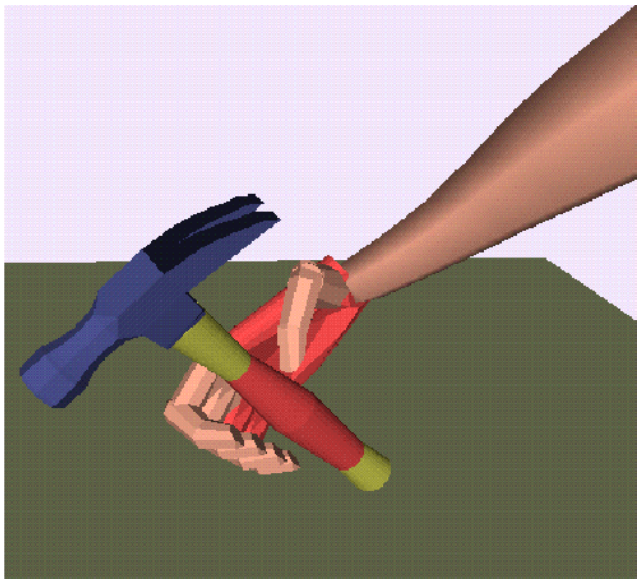


Figure 3: Grasping a hammer [13]

floor. A block diagram of the integrated system is shown in Fig. 4 [13].

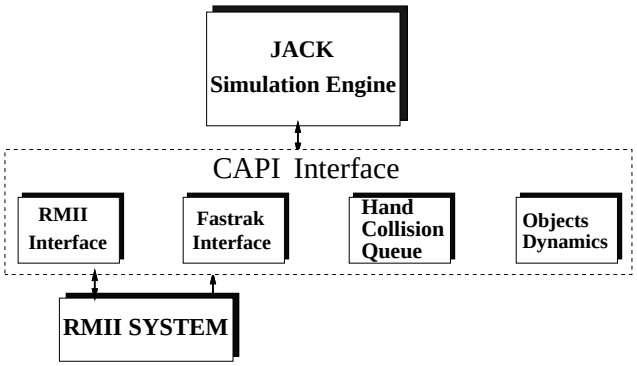


Figure 4: Block diagram of the integrated system [13]

We tried the interface for grasping of simple objects (sphere, cylinder) on a simple environments with 2460 shaded polygons. When only hand position tracking was activated, the hand movement was rendered at 10 fps on the SGI High Impact workstation. When finger joints updates were added, hand movement was rendered very slowly with considerable delays (1-2 sec) between user action and simulation response. Simplifications of the interface were needed in order to achieve a reasonable time response for simulated actions. The interface was designed according to the grasping task characteristics, and the configuration of the RM–II hand master. During the “reaching" phase, the user’s fingers move very little; therefore no finger motion is issued and the global hand movement is rendered at the maximum speed. During the “grasping" phase, the hand position changes little while the fingers dynamics are important. Therefore finger movements are quantized (not continuous at sensor update rate); the quantization threshold used is 1/9 of the full dynamic range of RM–II Hand Master. A joint adjustment command is issued only if the change between two successive readings of the RM–II is larger than the specified threshold. In addition, “circular power grasping" with the RM–II was simulated with only one joint update per finger. The experiments with the new interface show smaller delays and a satisfactory graphics display of hand movements. Hand interface with JACK was redesigned subsequently using a module loading technique. No simplifications were necessary for the hand interface since the main limitation of simulation speed is the graphic rendering engine and inverse kinematics computation.

System Evaluation

The interface was tested on two graphics platforms: a High Impact and an Infinite Reality SGI workstations. Previous experiments with hand positioning using magnetic sensors (Ascension Flock of Birds) implemented in [6], achieved a frame rate of about 8-10 frames per second with a shaded environment of 2000 polygons on a SGI 310/VGX. The limitation was due to the computation overload of the inverse kinematics routine. The same limitation applies to our simulation. In addition to hand positioning, the interface we implemented uses RM–II sensors readings to control finger positions. After implementing the interface, the performance of the Virtual Human with real-time hand input system was tested. The test program consists of a virtual human grasping objects in an environment with 5315 shaded polygons. The results of the simulation tests are summarized in table 1.

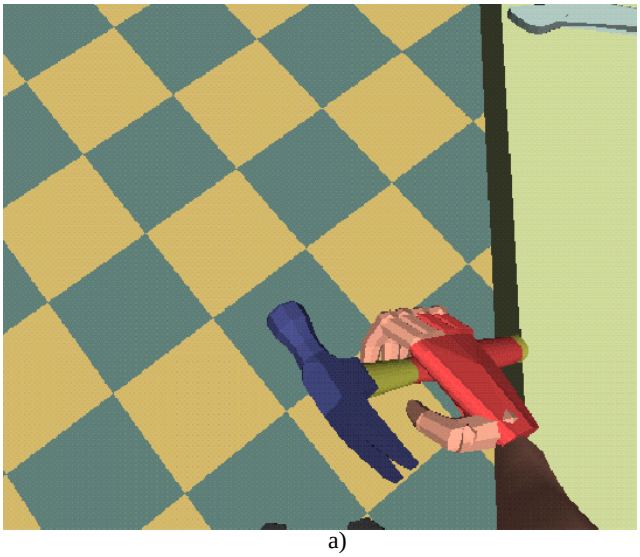
Table 1: **Comparison between two implementation of RM–II – JACK interface**

Workstation	*High Impact*		*Infinite Reality*			
Method	C API		C API		Dynamic linking	
Views	1	2	1	2	1	2
Frames/sec	10	5	22	12	17	9.5
Sensors updates/sec	2	1	3.5	2.1	17	9.5

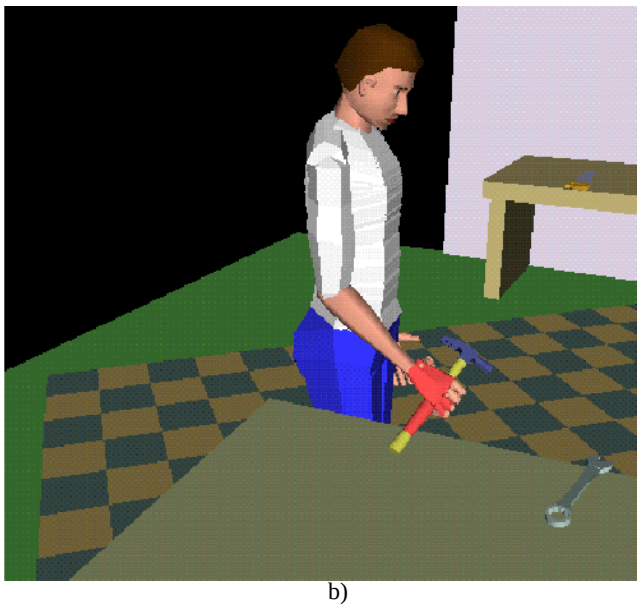
With the simplifications described in the previous section, an average frame rate of 10 fps (one view) was achieved on High Impact SGI. The same simulation runs at about 22 fps on the Infinite Reality machine. When using the C API the sensor update rate is very small (3.5 updates/sec), even though the simulation runs at a reasonable frame rate. No simplifications were necessary for the hand interface when implementing run-time module technique since the main limitation of simulation speed is the graphic rendering engine and inverse kinematics computation. That creates however a little bit of overload, reducing the frame rate from 22 fps to 17 fps for one window on SGI Infinite Reality machine. The sensor update rate is about five times larger than in the previous case, allowing real-time grasping. When loading interface module in JACK, real-time motion tracking delays depend only on the frame rate, and are not affected by the overhead of communication between interface and JACK simulation. The force feedback loop is executed locally on the RM–II SCI, and is independent of the frame rate.

APPLICATION: VIRTUAL HUMAN AGENT WITH REAL-TIME INPUT

A large scale simulation involving real-time full-body motion tracking and force feedback requires very powerful tools. The first step towards such a complex simulation is to provide real-time user input and force feedback for some body segments used in VR simulations (like the hand). A hybrid simulation system which combines real-time input with automatic behavior of the virtual human can be implemented using JACK - the Virtual Human - integrated with a hand haptic device (as described in the previous section).



a)



b)

Figure 5: Simulation scene: a) virtual hand grasping a hammer (between eyes view); b) JACK grasping a hammer (bird's eye view) [13]

Automatic behaviors like walking, bending, turning and arm movement add realism to the simulated environment. Additionally the haptic device allows the user to feel forces in the hand while grasping and manipulating objects. We define the concept of a Virtual Human Agent (VHA) as a simulation system embedding a Virtual Human with real-time haptic input and with user control over the simulated actions. Such an agent could be useful in different VR simulations like CAD prototyping, medical simulations, user training etc.

We illustrate the concept of a VHA in a machine shop simulation in which the user grasps objects and controls the movement of the virtual agent to different places in the simulated environment. The virtual environment contains tables, shelves, a chair and some tools (hammer, saw, etc.). Some objects were provided by the JACK software package, while others were created in Autocad [9] and translated to “psurf" formated files. The virtual environment contains 5315 polygons rendered with flat shading. Two camera views were implemented: virtual human view and “bird’s eye" view, as shown in Figure 5 [13].

The simulation starts with JACK walking toward a table. Once he reaches the table, JACK “looks" down to his right hand. Next the contact detection mechanism for his right hand is activated and real-time input enabled. VHA is now ready to grasp objects from the table. The user can grasp tools in real-time and feel forces in the hand. The grasping mechanism was implemented as described in the previous section. The user can control the virtual agent using the following verbal commands: “go to table", “go to shelf", “lower head", “raise head", “turn around". JACK can carry tools between work benches, change the viewpoint and execute tasks involving grasping. Automatic walking behavior is used to move

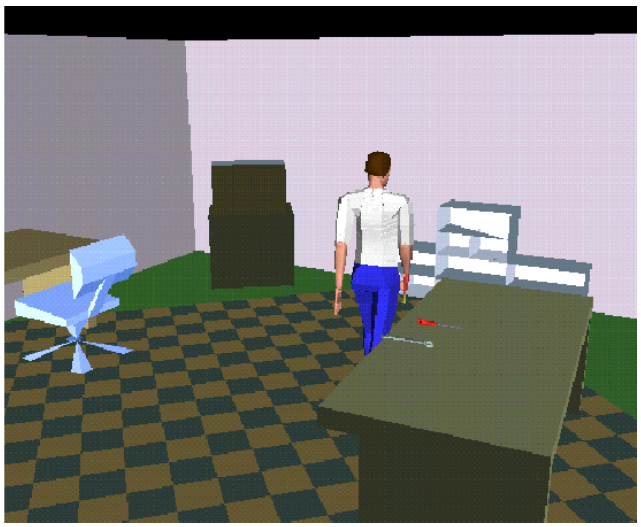


Figure 6: Simulation scene: JACK walking between two workbenches [13]

between work sites. To execute the “walk" command, the actual position of the virtual human is first detected and the path trajectory is interpolated to predefined locations corresponding to work sites. To have a smooth path, the difference in orientation between the

actual position and the end position is calculated. If this difference is above a threshold the virtual human is rotated before the walk command is issued, otherwise the motion becomes unstable and VH looses its balance. Figure 6 shows JACK carrying a tool (“hammer") between two workbenches [13].

Real time hand motion tracking is suspended during walking; when figure motion is detected the real-time input flag is invalidated. The force feedback mechanism is always on, allowing the user to feel the grasped object during walking. To facilitate the interaction between the user and the Virtual Human Agent, a speech input interface was added to the simulation as shown in Figure 7 [13].

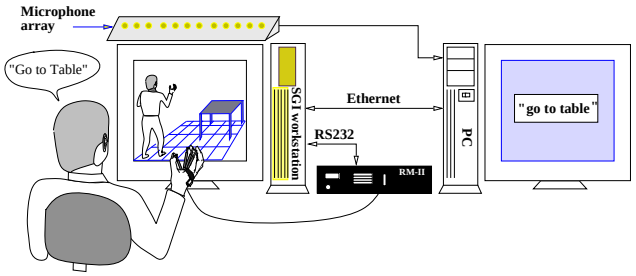


Figure 7: Multimodal interface to “JACK" using the RM–II and the Microsoft SDK [13]

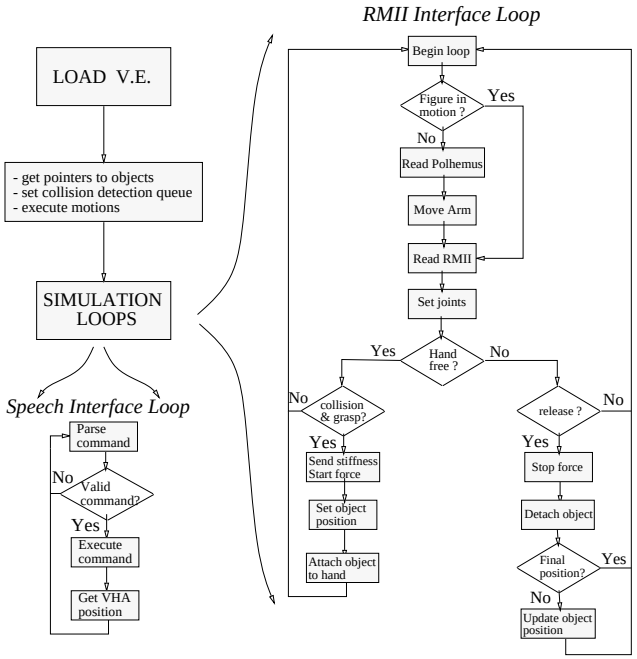


Figure 8: Simulation block diagram [13]

The speech interface uses a Microsoft speech recognition engine [10] running on a PC Pentium Pro. A microphone array provides a better input (with spatial localization) for hands free communication with the simulated agent [12]. A small grammar was implemented for the speech recognition engine: it includes commands for walking, head movement and turning. For this small vocabulary, the recognition rate of the Microsoft engine is high (95 %). A socket connection transmits string commands between the PC and the graphics workstation. The command interface program uses C API functions to control the simulated human figure, as illustrated in Figure 8 [13].

CONCLUSIONS

This paper outlined the design and functional evaluation of a haptic interface for JACK^{Tm}. Design solutions were optimized according to the characteristics of the simulated task and simulation tool limitations. Evolving from this interface a Virtual Human Agent with a multimodal interface was subsequently designed. The concept of Virtual Human Agent (VHA) with an “active" hand can be applied to several VR simulations involving hand dexterity: luggage inspection, aircraft inspection, surgical operation, palpation training. With the increasing 3D graphics capabilities of computing systems, this will substitute current simulations which render only flying hands navigating in VE. The concept can be extended to include real-time tracking and haptic feedback to other body segments.

The haptic interface for JACK can be improved using more complex stiffness models. RM–II design specifications include a protocol for transmitting stiffness model parameters (polynomial model, exponential model, etc.). Stiffness parameters have to be added therefore in the objects representation, allowing for a better quality haptic rendering of virtual objects. Future developments of the VHA will include more simulated tasks and an enhanced communication between user and the virtual agent. The virtual human will be “instructed" to perform different types of automatic behavior like bending, siting, crawling, using speech commands. Speech synthesis will allow the VHA to “answer" the user in response to his commands.

ACKNOWLEDGMENTS

Research reported here was supported by grants National Science Foundation (grant BES-9708020), from Rutgers University (SROA) and the CAIP Center with funds from the New Jersey Commission on Science and Technology and the CAIP’s industrial members.

REFERENCES

[1] Burdea G., 1996, *Force and Touch Feedback for Virtual Reality*, John Wiley & Sons, New York.

[2] Burdea G. and Coiffet P., 1994, *Virtual Reality Technology*, John Wiley & Sons, New York.

[3] D. Gomez, G. Burdea, N. Langrana, 1995, “Integration of the Rutgers Master II in a Virtual Reality Simulation”, *Proc. of IEEE VRAIS’95 Conference*, Research Triangle Park, NC, March, pp. 198–202.

[4] Transom, Inc., 1996, *JACK User’s Guide, version 5*, Ann Arbor, Michigan.

[5] Tolani D. and N. Badler, 1996, “Real-Time Inverse Kinematics of the Human Arm", *Presence*, Vol. 5, No. 4, pp. 393–401.

[6] N. Badler, M. Hollick, J. Granieri, 1993, “Real-Time Control of a Virtual Human Using Minimal Sensors", *Presence*, Vol. 2, No. 1, pp. 82–86.

[7] Cutkosky M. R. and Howe R. D., 1990, “Human grasp choice and robotic grasp analysis", in *Dextrous robot hands*, T. Iberall & S. T. Venkataraman (Eds.), Spriger-Verlag, New York, pp. 5–31.

[8] B. Douville, L. Levison, N. Badler, 1996, “Task-Level Object Grasping for Simulated Agents", *Presence*, Vol. 5, No. 4, pp. 416–430.

[9] Autodesk Inc., 1994, *AutoCAD User's Manual*, Sausalito, CA.

[10] Whisper Speech Recognizer by Microsoft Corp., http://research.microsoft.com/ msrinfo/demodwnf.htm

[11] Polhemus, 1993, *Fastrak User's Manual*, Colchester, VT.

[12] Q. Lin, C. Che, E. Jan and J. Flanagan, 1994, “Speaker / Speech Recognition Using Microphone Arrays and Neural Networks." Proc. SPIE, Vol. 2277, San Diego, CA, pp.121–132.

[13] Popescu V. G. and Burdea G., 1998, *Research on full body modeling and force feedback*, CAIP Report CAIP-TR-223, Rutgers University, Piscatawy, NJ, April, pp. 1–33.