CoPuppet: Collaborative Interaction in Virtual Puppetry.

Paolo Bottoni
University of Rome "La Sapienza"
Via Salaria 113
00198 Roma, Italy
+ 39 06 4991 8426
bottoni@di.uniroma1.it

Stefano Faralli
University of Rome "La Sapienza"
Via Salaria 113
00198 Roma, Italy
faralli@di.uniroma1.it

Anna Labella
University of Rome "La Sapienza"
Via Salaria 113
00198 Roma, Italy
+ 39 06 4991 8512
labella@di.uniroma1.it

Alessio Malizia
Universidad Carlos III de Madrid.
Avda. de la Universidad, 30.
28911-Leganés, Madrid, Spain.
phone: +34-91-624-5935
alessio.malizia@gmail.com

Mario Pierro
University of Rome "La Sapienza"
Via Salaria 113
00198 Roma, Italy
pierro@di.uniroma1.it

Semi Ryu
Virginia Commonwealth University
1000 W Broad st
Richmond, VA, USA 23284
+1 804 543 1971
sryu2@vcu.edu

ABSTRACT
CoPuppet is a framework for the development of performances of virtual puppetry. In particular, it defines a class of interactive systems in which to realize collaborative virtual puppet performances involving several puppeteers. Users are able to control puppet's body parts and interact with the puppets by producing gestures which are captured by video devices and translated into control parameters for the movements of the puppet. Moreover, a storyteller realizes another form of control, as the sounds captured by a microphone are used to steer in real time mouth movements and facial expressions of the virtual puppet on the screen. The result of such interactions will see the emergence of a cooperative management of the puppets' movements.

Categories and Subject Descriptors
J.5 [ARTS AND HUMANITIES: *Performing arts*].

Keywords
Virtual puppetry, collaborative performance, mixed reality environments.

1. INTRODUCTION
Puppetry is one of the most ancient forms of representation, diffused all over the world in different shapes, degrees of freedom in movements, and forms of manipulation. As an example, Neapolitan puppets (known as Punch and Judy in the Anglo-Saxon world) are operated by single hands fitting inside the puppets, while Sicilian ones may be significantly sized, with different components steered via threads moved from above. Shadowplays, such as the Javanese Wayang Kulit, or the Turkish Karagöz, happen behind screens with the puppets' limbs controlled via horizontally held sticks.

The Prague Black Theatre incorporates puppets and real actors whose body parts are made invisible by wearing black clothes. Korean puppetry works with rods, and sometimes hands inside, with limited control, showing very primitive expressions. One of the puppeteers sits in front of the stage with the audience and talks constantly with the puppet during the play, breaking the boundary between the puppet and the real world. In the Japanese Bunraku puppet theatre, 3 puppeteers in black costumes work together to establish the puppet's emotional expressions and gestures as a whole.

Puppets are a great support for storytelling, allowing representation and replacement of settings with little effort and relying on conventional representations of ellipses. Stories are typically stereotyped and set in the historical and popular traditions of the different cultures. Very often, they were parts of rituals, where particular stories would be told in specific occasions. For example, puppets were often worshipped as images of gods by ancient populations in Korea, and shamans could evoke the puppet's spirit through specific rituals.

Sometimes the puppet drama has a fixed narrative to be told exactly as trained. For example, before a Bunraku performance the chanter holds up the text and bows before it, promising to follow it faithfully. On the other hand, particularly in the Mediterranean area, stories could also be improvised or developed under the influence of recent chronicles. In these cases, they could convey transgressive contents which would have not made it to the theatrical stage. Korean puppet drama shows another interesting case. It is a folk art form preserved only by oral tradition, as in many cases puppeteers were not literate, and the narrative in the puppet theatre was orally transmitted over a long period of time [1]

As many other media, puppets have gone digital. Even if many artists and scientists are interested in translating puppetry in digital form, focus has been mostly placed on development of digital controls, rather than on providing interesting and interactive playground. This way, digital puppetry has not retained the playful energy of original puppetry.

CoPuppet aims at replicating this sort of experience, resulting into free improvisation and collaboration between people, using interactive tools such as virtual sensors and voice activation. Virtual sensors are here intended as maps of physical phenomena onto a virtual support.

In this paper, we explore the possibilities offered by multimodal and cooperative interaction with puppets, in constructing a communicative experience among performers, or even audience members, called to affect different parts of a puppet through gestures and voice. The proposal is based on the use of the CHAMBRE architecture for the creation of multimodal interfaces [2,3], in which users can interact with multimedia sources through WIMP- as well as multimodal-based widgets. CHAMBRE can also integrate virtual reality environments, and users may affect them through modifications of their parameters by gesture or voice commands. As a result, CoPuppet allows the creation of "live improvised" storytelling and actions between puppeteers.

The rest of the paper proceeds as follows. A brief review of related work is given in Section 2, while the CHAMBRE architecture exploited in CoPuppet is presented in Section 3. Section 4 presents the CoPuppet application, while Section 5 and 6 discuss its configuration and illustrate the performance, respectively. Finally, Section 7 draws conclusions.

2. RELATED WORK

Digital puppetry is often categorized in three families [4]:

Waldo puppetry: the digital puppet is controlled onscreen by a puppeteer who uses a telemetric input device connected to the computer.

Motion capture puppetry: an object (puppet) or human body is used as a physical representation of a digital puppet and manipulated by a puppeteer.

Machinima: a production technique that can be used to perform digital puppets. Machinima involves creating movies employing computer-generated imagery rendered using low-end 3D engines in video games.

One of the authors has already developed a virtual interactive puppet performance, "YONG-SHIN-GUD" [5], as a digital translation of ancient puppetry. This involves a live artist performance with music and storytelling, together with the use of 3D motion graphics to represent a virtual puppet. The puppet movements and facial expressions are steered by the sounds captured by a microphone, either produced by instruments or by the storyteller voice. These reactions occur in real-time, while the virtual puppet constantly speaks and sings back to the puppeteer, as a sort of real-time echo and mirror reflection. The audience itself might participate in the interaction by producing sounds in reaction to the artist's or puppet's actions. The goal of the "YONG-SHIN-GUD" performance was to let the puppet eventually acquire the ultimate trans-state of shaman, by spiraling interactive dialogues with the real puppeteer. Drawing on the oriental philosophy of yin and yang, the real-time lip synchronization process produces a continual change of roles between the virtual puppet and the human puppeteer.

An interesting combination of storytelling and multimodal interaction is presented in [6]. Here, humans interact with puppets in a virtual environment: human gestures and speech are recognized and used to steer dialogues and interaction with the puppets.

Camille Utterback's "text rain" [7] is an interactive installation in which participants use their own bodies, to do what seems magical—to lift virtual falling letters and play with them. In a *Text Rain* installation, participants stand or move in front of a large projection screen, where they observe a mirrored video projection of themselves in black and white, combined with a color animation of falling letters. Like rain or snow, the letters appear to land on participants' heads and arms. The letters respond to the participants' motions and can be caught, lifted, and then let fall again. The falling text can 'land' on anything darker than a certain threshold, and 'fall' whenever that obstacle is removed.

Pamela Jenning's "Constructed Narratives" [8] shows interesting collaborative aspects between people in a public space. Constructed Narratives is a block-based construction game designed for adults and older teenagers. The goal of the Constructed Narratives project is to develop a framework for the design of tangible social interfaces. It has been designed for use in public spaces where there is the opportunity for individuals and groups of people, who are not acquainted with each other, to encounter the game and subsequently each other. CoPuppet is also designed for challenging social relationship between people in a public space.

The construction of virtual sensors can take advantage of the availability of tools for pattern recognition and motion detection. Currently, original and simple algorithms have been incorporated into CHAMBRE, typically based on finger position identification. Differently from [9], we are not restricted to 2D positioning, but can also exploit (partial) 3D information. The open structure of CHAMBRE and the simplicity of its component model, however, make it easy to embody more sophisticated ones.
In any case, video capturing and pattern recognition through computer vision techniques is still a challenging problem largely faced in the last years [10, 11, 12]. Objects detection can burst multimodal applications, in fact by recognizing detected objects, the system can automatically analyze their behaviors. The main feature utilizable for maintaining the identity of a moving object is its visual appearance. To this aim color and shape ratios are widely adopted [13].

3. CHAMBRE

CHAMBRE is an open framework able to accommodate different protocols, sensors, generation and interaction techniques [2]. The generative process can be steered, both explicitly and implicitly, by human users whose inputs are acquired through external multisensor devices. For example, a webcam can capture user movements, while image analysis tools can interpret them to detect presence in specific zones or evaluate variations with respect to previous or fixed reference images. As a result, parameters are generated to steer the system response. Specific inputs can also trigger, in real-time, modifications of the interpretation process. This ability makes CHAMBRE a flexible and open tool, easily adaptable to different situations and applications.

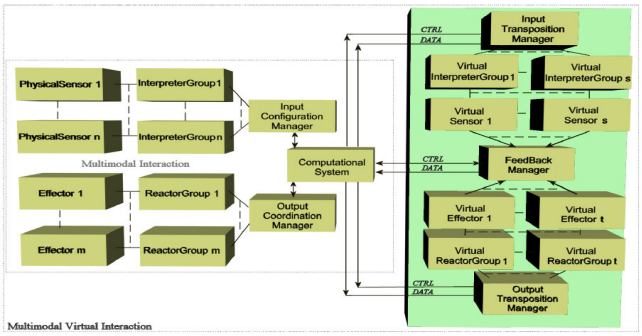

Figure 1: The architectural model of Virtual Multimodal Interaction.

The CHAMBRE architecture allows a component-based style of programming, where components are endowed with communication interfaces and a system results from their connection. A designer can interactively define a CHAMBRE network in the form of a graph, where nodes are components and edges are communication channels among them.

The CHAMBRE framework was started as a distributed component-based architecture for the production of multimedia objects and incorporates several plug-ins developed to cope with several multimedia data formats and streams [2]. Currently available plug-ins offer: 1) sensor stimulus interpretation from real sensors (physical measures) such as: webcams, mouse and keyboard signals, etc.; 2) signal generation; 3) signal mapping; 4) mapping-driven multimedia streaming generation (audio and video synthesis). These plug-ins favor rapid prototyping of *Multimodal Interfaces* (*MI*) [14] which allow users to interact with the system through several input devices like: keyboard, mouse, voice, face and gesture recognition, and so on.

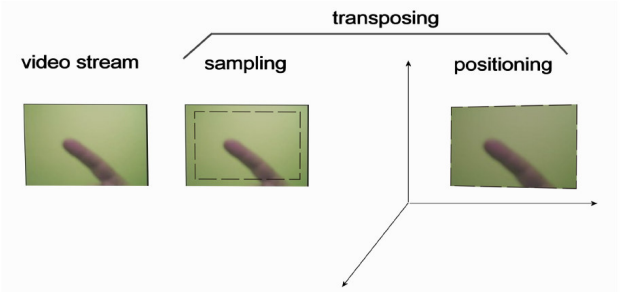

Figure 2: Construction of a support for virtual sensor.

Symmetrically to the real case, virtual actuators, or groups of actuators can interact with and change virtual environments through *Virtual Multimodal Interfaces* (VMI). In order to manage the state of virtual actuators, the *FeedBack Manager* module sends streams of control commands and data to the *Computational System*.

A VMI is characterised by the presence of particular CHAMBRE nodes in charge of mapping real stimuli to virtual ones through an interpretation process. A virtual stimulus is a configuration of a data collection produced by some computational process, which can be interpreted as the result of a measurement process. In particular, a *Virtual Component* defines an *Appearance*, a *Behavior* and a measurement method.

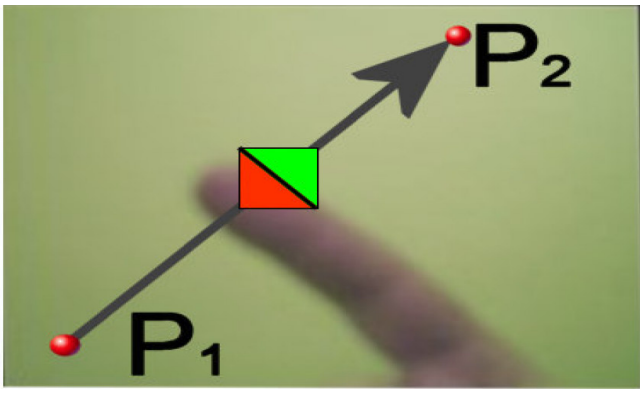

Figure 3. An example of Virtual Slider.

An innovative feature in the CHAMBRE implementation of VMIs is the possibility of positioning virtual sensors onto a virtual support. As an example, a video virtual support can transpose the frames taken from a video stream by sampling (clipping) and positioning them into the virtual space (see Figure 2). As another example, a virtual slider acting onto a video virtual support can perform measurements by detecting the point closest to an extreme P_2, in a segment P_1P_2, which intersects a moving shape, as shown in Figure 3.

Figure 4 shows instances of *Virtual Button* and *Virtual Slider* widgets performing their measurements on video supports.

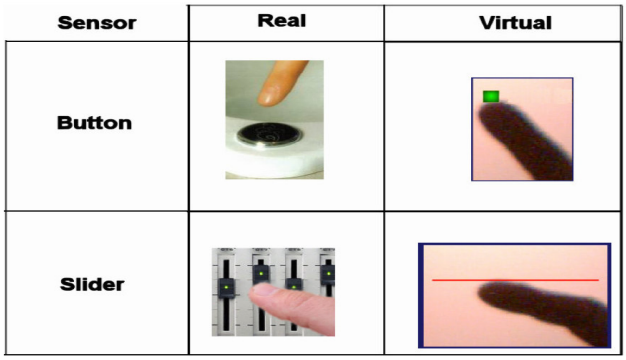

Figure 4: Examples of Virtual Sensors.

4. CoPuppet

CoPuppet is a class of CHAMBRE applications realizing different paradigms for Virtual Puppetry. A CoPuppet application is a CHAMBRE network featuring instances of *MVI_VirtualPuppet* (MVI_VP for short), a CHAMBRE software component which, differently from virtual sensors, does not define a measurement method but only an *Appearance* and a *Behavior*. It relies on other CHAMBRE components, possibly running on remote machines, to perform the measurements needed to generate the data determining the puppet's posture. An MVI_VP appearance is given by a 3D digital puppet, produced as a tree of labeled Java 3D nodes (see Figure 5).

The Java 3D graph-based scene model [15] provides a simple and flexible mechanism for representing and rendering scenes. CHAMBRE embedded graphical engine is based on Java 3D technology and is capable of rendering skinned bones. A scene graph contains a complete description of the entire scene. This includes the geometric data, the surface attributes and the

visualization position information needed to render the scene from a particular point of view. Hence, the *Appearance* of an MVI_VP is a labeled extension of a Java 3D scene graph, with nodes of two kinds: BranchGroup and TransformGroup. A BranchGroup identifies body parts, and its elements have labels in the set $BG = \{B, H, M, R, R', L, L'\}$, for body, head, mouth, right and left arm and forearm, respectively. A TransformGroup specifies control elements and a *Transformation Matrix* for the different body parts. The labels in the set *TG* are obtained by concatenating a label from *BG* with one from $Par = \{Tx, Ty, Tz, x, y, z, s\}$. Here, "*T*" labels indicate translation parameters along the three Cartesian axes, the next three indicate rotation angles around these axes, and *s* denotes the scale factor. As an example, the label *MTy* identifies the TransformGroup node used to translate the mouth BranchGroup along the *y* axis. This produces a vertical shift of the puppet's mouth.

The *Behavior* of an MVI_VP is determined by a simple protocol: the object accepts messages of the form **label value**, with **label**$\in$*TG*, and **value** is a parameter used to define the new transformation matrix associated with the *Appearance*'s node identified by the label. Values generated by Virtual Sensors are normalized in the interval [0.0, 1.0] and mapped to labels in *TG*. A message can also be mapped to modify attributes defining the material used to render the corresponding nodes of the puppet's 3D representation. For example the message "*MTy* value", besides defining the vertical position of the mouth, can also be used to change the texture of the puppet's head, thus modifying its facial expressions.

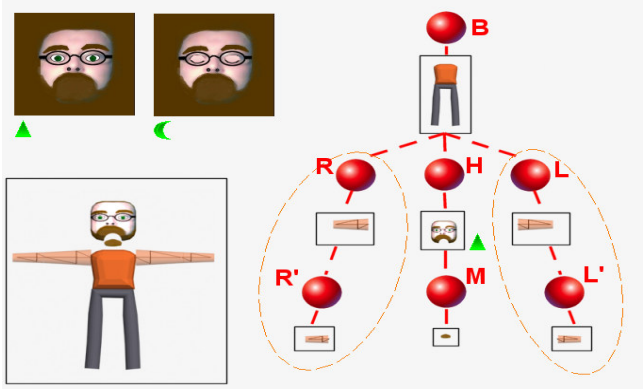

Figure 5: The appearance of a Virtual Puppet.

The definition of Virtual Puppet as a CHAMBRE component favors the use (and reuse) of puppets in different contexts of execution. Actually, every software component (even non CHAMBRE ones) which respects the *Behavior* protocol can produce messages for the puppet controls.

We focus now on two specific puppetry paradigms which can be realized in CoPuppet.

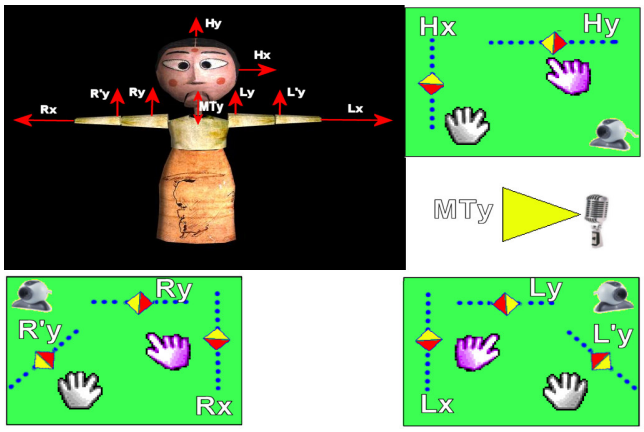

Figure 6: Puppet's controllers.

4.1 Single puppet, collaborative controls

Single puppets can be operated on by one or more performers, through interactions in which separate channels control different aspects of the puppet, such as limb movement or facial expressions. As new controllable aspects are introduced, so can new multimodal input channels be added, to be managed by a single performer, or distributed among many. In particular, we are interested in the evolution of patterns of real time collaboration among performers during free improvisation of the puppet performance.

In the proposed scenario, the puppet is set in a virtual environment of arbitrary complexity, and its movements, utterances and facial expressions are determined by the actions of independent users. The environment itself can present objects which can be animated by user interaction. Hence, users log into a performance either as puppet body parts such as mouth, arms, lower body, or as objects in the scene, such as trees, umbrellas and hand mirrors.

In its current stage, their are no constraints on relative motions of body parts, except that they be kept connected, so that physically impossible situations can arise. This is not in contrast with the aims of the project, and is possibly a desired effect of it, as it would be interesting to see how the performers draw themselves out of such situations, or engage in power struggles to make others conform to their choices. However, constraints could be added by exploiting the tree structure of the objects managed with Java 3D.

4.2 Multiple puppet, collaborative controls

A Multiple Puppet scenario can be produced by replicating a number of instances of the Single Scenario. CHAMBRE capability of distribution helps the configurations of scalable stages, where teams of puppeteers can perform real-time shows and or produce databases of recorded session for further post productions needs.

An asynchronous form of multiple puppet can be realized by recording actions of single puppets and merging them afterwards projected on a new environment.

In a multiple puppet scenario based on replication of the single one, users enrol not only on body parts, but have to define which puppet they are managing. This may lead to the design of puppet chatting stages, in a way similar to how avatars may be used. Hence, users can engage in virtual dialogues, observing the

reactions of the different participants through the modifications of their representative puppets. Sessions can also be recorded and replayed by the participants or by external observers.

5. Configuring CoPuppet

As mentioned before, there are no constraints as to which software components can be used to steer puppet's behavior, and even third party suites like Virtools [16] can be used as a front-end for a CoPuppet application.

The use of well-known tools can favor the artistic and technical development process of script creation and relieve the composer from advanced programming aspects as the development of particular graphical routines.

A Single or Multiple scenario of CoPuppet usually consists of a network of *n* computer with *m* instances of CHAMBRE running on them. Each CHAMBRE instance contains a replication of the *k* Virtual Puppets used for the performance, but they may differ in the graph of connected software components. Puppeteers can thus have a global view of the virtual puppets and of the current configuration of CHAMBRE software components needed to drive the body parts for which they are responsible.

Each puppeteer is focused on at least one of the *m* CHAMBRE instances where the global behavior of the *k* puppets is represented. The owner of a set of CHAMBRE instances steers some behavioral aspects of a puppet which are transmitted to each CHAMBRE instance in the network. This way, computation is distributed over the network and synchronization is achieved by message passing.

As mentioned before, external tools can be used for graphical and audio improvements. For example in Infinite Cemetery [17], spatial information about Virtools entities is transmitted via network to an audio spatialization and synthesis algorithm running on a separate machine.

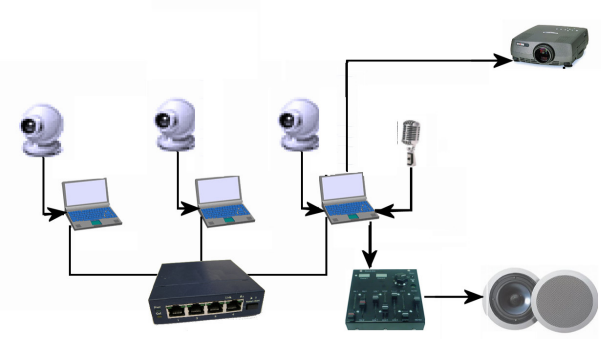

Figure 7: Setup of the peripherals for CoPuppet.

6. CoPuppet Performance

In a CoPuppet collaborative performance, the puppeteers are placed at different locations in front of a screen on which the appearance of the virtual puppet is projected, as shown in Figure 8. Users login as a puppet body part for which controls have been defined, and produce collaborative movements of the virtual puppet. A voice puppeteer tells a story into a microphone, to which the system responds in real time by producing mouth movements and facial expression of the virtual puppet on screen. Body puppeteers use their fingers in front of a webcam, thus activating virtual sensors connected with the corresponding body part on screen. The movements are reminiscent of those performed by real puppeteers steering movements through threads. While each puppeteer can create only simple movements, the whole gesture of the puppet results from their combination, producing powerful and interesting effects.

The performance can also allow for audience intervention. In particular, while retaining a single storyteller, CoPuppet can partially open several body controls to the public, to foster forms of collaboration between the performers and the audience. A completely open instantiation of the framework can also be envisaged, in which computers, microphone and webcams would be setup for users to participate in the performance space.

6.1 Collaboration aspects

The main intended use of CoPuppet involves a storyteller, a crew of body parts performers, and possible intervention from the audience. These users do not have to define in advance the kind of story they want to tell, but can come up at any time with ideas about the situation and the story, based on the materials in the scene. In a sense, all objects in the scene can become alive as "performing objects". It is expected that the real time observation of the consequences of one's own actions, as well as those of the other puppeteers, will foster interaction between puppeteers, who will engage into collaborative or competitive behavior patterns. Typical patterns might be mimicking or counteracting the actions of one another, introducing delays in replicating some movement, achieving unison through the iteration of the same movements. Interestingly, these patterns may occur directly among performers as well.

We envisage that CoPuppet, through its simple and intuitive controls, can become a playground for children, while adults might find it interesting as a form of stage for free speech, as well as imaginative storytelling.

The following paragraph describes the technical setup for the demonstration of the current version of CoPuppet.

6.2 Technical Description

Each puppeteer uses one or more *VirtualSliders* to translate his/her hand movements into puppet actions. The slider position is determined from the hand position, acquired via webcam. Puppeteers can see their hands projected into the virtual space, and interact with the virtual controllers (via their laptop computer screen) and the virtual puppet (via the projection screen).

Figure 6 shows how the different transform nodes can be operated on through different multimodal channels. Labels are taken from the set *TG* and indicate both a puppet body part and the corresponding *VirtualSlider* setting the value of the control parameter. In particular, R_X, R_y, R'_y, L_X, L_y, L'_y are the labels for angle rotation around the *x* and *y* axes for the articulations of the puppet's arms, while H_x, H_y are labels for angle rotation around the *x* and *y* axis for the puppet's head. Finally, MT_y is the parameter that determines the puppet's mouth opening: it is determined by calculating the average amplitude value *m* of the signal coming from a microphone connected to the puppeteer's laptop. The value of *m* is calculated on a buffer of 1024 audio samples acquired every 0,023 seconds, so that the system response to audio storytelling is fast and accurate.

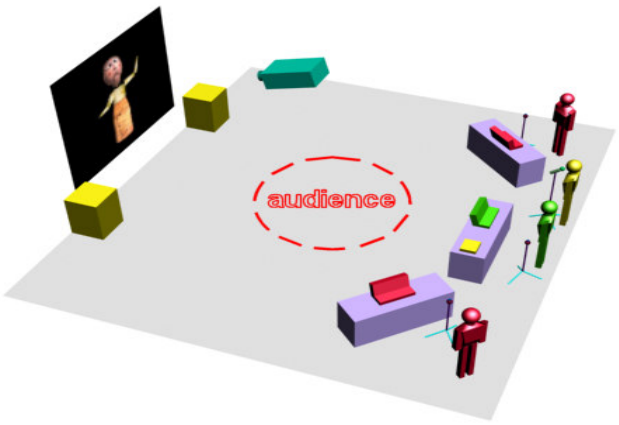

Figure 8: Performance system architecture.

The global setup (see Figure 7) consists of three laptops connected via a Local Area Network (LAN), using a hub. Each laptop is connected with a USB webcam : the first laptop used for the R_x, R_y, R'_y, controls, the second for the L_x, L_y, L'_y and the third for the H_x and H_y controls. For MT_y, a microphone is connected to the audio *in* port of the second laptop. Finally, the second laptop is connected to the video projector through a VGA cable, and, using an audio cable, is also connected to the mixer which drives the audio speakers for the audience. Effective interaction among puppeteers relies on the high speed of the 100mps LAN connection between the laptops, which transfers raw data generated by the virtual sensors. Collaborative performance by puppeteers in remote locations would require more sophisticated handling of messages exchanged between system components, including time-stamping of messages and reconstruction of missing data in case of network delays and/or failures.

7. CONCLUSION

In this paper we have presented *CoPuppet*, a distributed system for digital puppetry which exploits the collaborative performance of multiple puppeteers.

Currently, CoPuppet allows interactions between puppeteers in near distance. However, it aims for remote collaborative puppet control over the network in the future.

We are also investigating the possibility of employing software agents in puppet controls, to enable more sophisticated forms of correspondence between data incoming from the virtual sensors and puppet movements.

Our future works on multiple puppet scenario aim to set both puppets and humans in a mixed reality environment [18]. This last scenario requires the introduction of new software components able to solve the problems concerning occlusion, contact, avoidance between human and virtual actors [19].

The open nature of the CoPuppet framework allows the incorporation of different sensors and the definition of different articulations and forms of rendering built on the Java 3D tree. On the other hand, its incorporation within the CHAMBRE environment, with its component-based structure, makes it possible to reuse definitions of behavior, appearance and measures in different contexts, such as collaboration scenarios, remote conferencing, or to associate puppets' controls to different sources of measures, whether interactive or automatic ones.

Acknowledgments. The authors with the University of Rome were partially supported by the PRIN 2006 Project: *Ambient Intelligence: event analysis, sensor reconfiguration and multimodal interfaces.*

8. REFERENCES

[1] Oh Kon Cho, Korean Puppet Theatre: KKOKTU KAKSI, Asian studies center East Asia series occasional paper no.6, Michigan State University, pp. 20 (spring 1979).

[2] P. Bottoni, S. Faralli, A. Labella, C. Scozzafava, "CHAMBRE: A distributed environment for the production of multimedia events", *Proc. DMS 2004*, pp.51-56, KSI, 2004.

[3] Bottoni, P., Faralli, S., Labella, A., Malizia, A., and Scozzafava, C. "CHAMBRE: integrating multimedia and virtual tools", *Proc. AVI '06*. ACM Press, 285-292, 2006.

[4] Walters, Graham. The story of Waldo C. Graphic. Course Notes: 3D Character Animation by Computer, ACM SIGGRAPH '89, Boston, July 1989, pp. 65-79.

[5] S. Ryu, "Ritualizing interactive media: from motivation to activation", *Technoetic Arts*, 3(2):105-124, 2005.

[6] Marc Cavazza, Fred Charles, Steven J. Mead, Olivier Martin, Xavier Marichal, Alok Nandi, "Multimodal Acting in Mixed Reality Interactive Storytelling" *IEEE MultiMedia*, 11(3):30-39, 2004.

[7] Camille Utterback, *Text Rain,* http:// www.camilleutterback.com/textrain.html.

[8] P..Jennings, "Constructed narratives a tangible social interface", *Proc. 5th Conference on Creativity & Cognition*. C&C '05, 263-266 ACM Press, 2005.

[9] B. A. Myers, R. G. McDaniel, R. C. Miller, A. S. Ferrency, A. Faulring, B. D. Kyle, A. Mickish, A. Klimovitski, , and P. Doane. "The Amulet environment: New models for effective user interface software development", IEEE Trans. Softw. Eng., 23(6):347-365, 1997.

[10] G. Foresti, C. Micheloni, L. Snidaro, P. Remagnino, and T. Ellis. "Active video-based surveillance system", IEEE Signal Processing Magazine, pages 25-37, March 2005.

[11] C. Jaynes. "Multi-view calibration from motion planartrajectory", Image Vis. Comput., 22(7), July 2004.

[12] M. S. S. Khan. "Consistent labeling of tracked objects in multiple cameras with overlapping fields of view", IEEE Trans. on PAMI, 25(10):1355-1360, October 2003.

[13] F. Porikli and A. Divakaran. "Multi-camera calibration, object tracking and query generation" Proc. of IEEE Intl Conference on Multimedia and Expo, 1(1):653-656, July 2003.

[14] N. Bianchi-Berthouze, P. Bottoni, "Articulating actions in multimodal interaction", *3D Forum*, 16(4): 220-225, 2002.

[15] Java 3D API Specification http://java.sun.com/products /java-media/3D/forDevelopers/j3dguide/Intro.doc.html #4739

[16] Virtools. www.virtools.com.

[17] Infinite Cemetery, Original Concept and 3D Virtual space: by Semi Ryu, Generative music composition: by Claudio Scozzafava, Software Plug-In: by Stefano Faralli, Execution: December 16 2005, *GenerativeArt 2005*. Politecnico, Milano, 2005.

[18] D. Thalmann, R. Boulic, Z. Huang, H. Noser, “Virtual and Real Humans Interacting in the Virtual World”, Proc. International Conference on Virtual Systems and Multimedia `95, Gifu, Japan, pp.48-57.

[19] M.J.Schuemie, P.van der Straaten, M.Krijn. C.A.P.G.van der Mast, "Research on Presence in VR: a Survey", *Cyberpsychology and Behavior*, 4(2): 183-202, 2001