

VIRTUAL THEATER for Industrial Training: A Collaborative Virtual Environment

J.C.OLIVEIRA¹, S. SHIRMOHAMMADI¹, M.HOSSEINI¹, M.CORDEA¹,
N.D.GEORGANAS¹, E.PETRIU¹ AND D.C. PETRIU²

¹School of Information Technology and Engineering

²Dept. of Systems and Computer Engineering

¹University of Ottawa ²Carleton University

161 Louis Pasteur Priv., Ottawa, ON K1S 5J6

CANADA

jauvane@mcrlab.uottawa.ca <http://www.mcrlab.uottawa.ca>

Abstract: Collaborative Virtual Environment concepts have been used in many systems in the past few years. Applications of such technology range from military combat simulations to various civilian commercial applications. In this paper we present a CVE prototype developed for industrial tele-training. We are showing that users can successfully control the environment through wireless input based on video processing and speech recognition.

Key-Words: Collaborative Virtual Environments, Virtual Reality, Collaboration, Java3D, Virtual Theater, Multimedia, Video Processing, Industrial Training.

1 Introduction

Over the past few years, a number of interactive virtual reality (VR) systems have been developed. A Collaborative Virtual Environments (CVE) is a special case of a VR system where the emphasis is more on “collaboration between users” rather than on simulation. CVEs are used for applications such as collaborative design, training, telepresence, and tele-robotics.

From an industry perspective, CVEs can be an attractive solution for reducing training expenses [1]. Instead of working with physical objects, these are represented by virtual objects that can then be placed in a virtual environment accessible to many users. The users, represented by avatars, can then manipulate and interact with the objects as in the real world, gaining valuable experience and training before using the real equipment

Motivated by these advantages, we have designed and implemented an industrial teletraining prototype. At the request of our industrial sponsors, the prototype has been created for training operators of ATM switching equipment; however it can be modified for other types of training. Such a prototype can be fully controlled in a wireless fashion where

speech commands and user gestures are used as input, allowing for a more comfortable interaction between the human participants. This paper is organized as follows. Section 2 describes the prototype and its components, section 3 outlines the underlying architecture, while section 4 focuses on the video processing technique employed to track a user’s behaviour.

2. Prototype

Our prototype is a multiuser teletraining application, which allows users, represented by avatars, to learn how to operate on a faulty ATM switch. The avatars repair the switch in steps which precisely reflect those necessary to perform the same actions in the real world. The prototype consists of two general modules: user interface, and network communication. The user interface itself consists of a graphical interface (GUI), a 3D interface (VR), and media interfaces (speech recognition, voice streaming, head tracking). Figure 1 shows the user interface of the prototype.

The upper right area of the interface, which takes the largest part, is the *3D environment*. On the left

and below the 3D environment are the *controls* used by the trainees to interact with objects and navigate in the environment. At the top left is the *head-tracking facility* which will be discussed in more detail later in section 3.6 and 4. Below the head-tracking window is a *utility panel* that is used for different purposes as discussed later. There is also a *chat space* where users can exchange textual messages. In order to avoid navigation problems with inexperienced users, it is possible to view the world from a set of predefined camera views as shown in Figure 2a.



Figure 1. Training Application's Interface

A user is able to approach and verify the operation of the switch and its cards, remove a faulty card and put it on the repair table, and replace it by installing a new card into the switch. Other parties will be able to watch that user's avatar taking such actions. All of the above actions can be performed by directly navigating in the scene and manipulating objects with the mouse or by selecting the action in a menu as shown in Figure 2b. Figure 3 shows a sequence of screenshots illustrating the execution of the previously mentioned steps performed by a trainer and watched by a trainee.

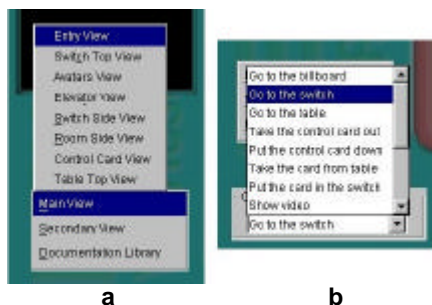


Figure 2. Camera View/Action Choice Menu

As can be seen from figure 3, a trainee can adjust his/her viewpoint to whichever angle/position that is more suitable to watch the actions. These adjustments can be done quickly by choosing the appropriate views from the camera view menu. When the trainer selects a new view of the World the trainees get the same view as well. That was implemented based on feedback from users and trainers, which is aimed at enhancing trainee's experience by viewing the actions through the best angle suggested by the trainer. A trainee may change the view if he/she so desires.

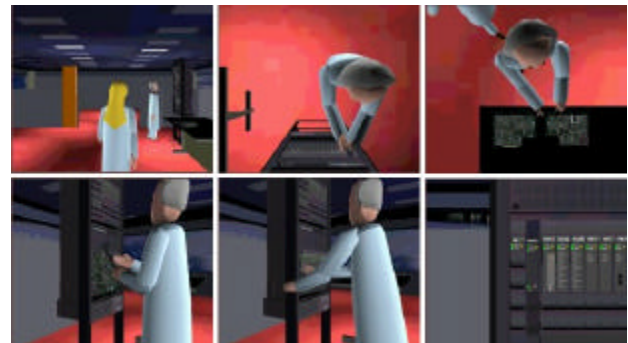


Figure 3. Replacing a Faulty Card.

In addition to receiving help from the trainer, a user can also view video segments showing the correct procedure for performing certain actions. The utility panel is used to display the video clips as shown in Figure 4b. If chosen by the trainer, the video will be displayed in every participant's screen.

Another use for the utility panel is the *secondary-view* feature, where a user can simultaneously watch the current actions from an alternative point of view as shown in Figure 4a.



**Figure 4. Utility Panel functionality:
a) secondary view (top) b) video clips (bottom)**

In addition to the above explained interfaces, the prototype offers voice recognition technology

whereby the user simply enters commands by talking into the computer's microphone. In this case, the user may simply say pre-defined commands such as "Go to the table" for the avatar to perform, or may change views by saying "Table Top View", and so on.

3. Architecture

Rather than developing the system from scratch, we decided to make use of a wide spectrum of available technologies, reuse software packages, and concentrate our efforts in integrating them. Hence, the system brings together a large number of components and technologies, namely for 2D graphics and interface design, 3D rendering, multi-user communications, voice recognition, audio and video streaming, and head tracking. In this section we will briefly describe the prototype architecture and the role of each component.

Figure 5 presents the system architecture including all of the components.

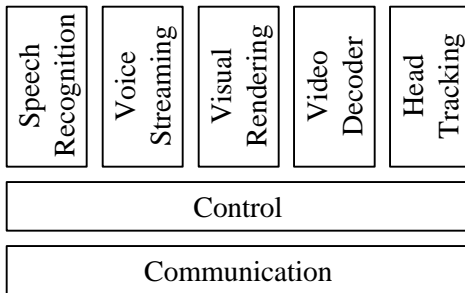


Figure 5. System Architecture.

3.1 Communication

The communication layer is responsible for all exchange of information amongst users. There are three components in this layer: directory server, control communication, and media communication layers.

The *directory server* is an entity which holds information about the participants in a session; the directory server is also responsible for checking on the status of every participant through alive bits.

Command	Int Parameter 1	Int Parameter 2	Str Parameter 1	Str Parameter 2	Str Parameter 3
---------	-----------------	-----------------	-----------------	-----------------	-----------------

Figure 6. Control Packet Format.

The *control communication layer* is responsible for all communication with exception of audio data. Packet have the format shown in Figure 6, where the

field *command* indicates what kind of information lies within the packet.

The directory server has been written in ANSI C and the client side of the control communication in C++ resulting in a Dynamically-Linked Library (DLL) to be loaded at run time. There are IPv4 and IPv6 implementations available so that depending on their capabilities, a group of users can choose whichever version they find appropriate by selecting the corresponding DLLs and directory server. The IPv6 implementation will eventually make use of quality of service parameters, multicast groups and flow labels when the users would benefit from better behaviour of the system.

The communication layer decouples all networking issues from the upper layers. The native-platform communication DLL is loaded into the Java environment using Java Native Interface (JNI). Incoming messages are sent to the Java core via callback of Java methods from the native code.

3.2 Rendering & Interface

For the GUI part, we used JavaSwing, which is a set of high-level graphics APIs for more advanced and sophisticated graphical components. Java technology was chosen due to its easy utilization of high-level APIs. When designing the 3D interface, we examined both Java3D and various VRML browsers. Other works have shown that Java3D is faster, easier to control and manipulate, and more interactive than the interface offered by VRML browsers [2]. Java3D also eliminates certain complexities that are inherent to using VRML along with its Java External Authoring Interface, Web browser and VRML browser. On the other hand Java3D has no specific file format and a developer must convert 3D descriptions of other famous formats to Java3D code.

Our prototype makes use of VRML objects designed in PowerAnimator. In order to make such objects treatable under Java3D a comprehensive optimization is performed in the original VRML objects aiming at simplifying the geometry (reducing redundant polygons). The VRML behavior of the objects is converted to Java3D code using VRML to Java3D conversion tools. A view of a world created in this manner is then presented in a Canvas3D. Hot spots are defined in the world in order to allow the user to interact with them in a point-and-click fashion. User actions are sent to the communication

layer, which in turn informs the other parties involved in or affected by such actions, creating thus a collaborative environment.

3.3 Speech Recognition

We use Microsoft Speech API (SAPI) to provide a speaker independent recognition of pre-defined commands described by a SAPI grammar. This way commands such as "go to the table", "pick up the card", are recognized by the SAPI module. An example of SAPI code is shown below:

```
[<Animation>]
<Animation>=go to the "go to the" <Select3>
<Select3>=billboard "billboard"
<Select3>=switch "switch"
<Select3>=table "table"
```

In the above example, all the "go to the X" commands are programmed into the SAPI engine, where X is one of "billboard", "switch", or "table". These commands will invoke the appropriated actions.

The speech recognition module is implemented using ActiveX components and its integration to the prototype is made via another native DLL loaded by JNI. The speech recognition module works independently from the other modules, by sending recognized commands into the prototype via a pre-defined local socket.

3.4 Audio Conferencing

The audio capturing module allows participants to enter into an audio-conferencing session. The module is based on the Microsoft NetMeeting SDK. Due to Soundcard limitations, this module will compete with the voice recognition module described above. Text chat facilities are also optionally provided and can be used in cases where audio is not supported.

3.5 Video Player

The Video Decoder module is an H.263 Video Decoder developed entirely in Java which is able to receive and decode streamed video from a video server. The video is presented in the utility panel.

3.6 Head Tracking

The head-tracking module captures the user's head

motion wirelessly by video processing techniques using a simple camera installed on the user's computer. Head movements are sent to the prototype and are used to control the corresponding avatar's head. The head-tracking module is implemented using ActiveX components which generate a series of rotation parameters which are sent to the prototype via a JNI connection through yet another DLL. More details are disclosed in section 4.

3.7 Control

The control layer is implicitly included in the other layers and is composed of a set of rules which ensures that all components work together. The coordination of the DLLs, which enable the exchange of data between the Java core of the prototype and the native components, comprises the Control Layer.

4. Video Processing

Model-based video coding (MBVC) has recently emerged as a very low bit rate video compression method suitable for Collaborative Virtual Environment (CVE) applications [9]. The MBVC increases coding efficiency by using knowledge about the scene content and describing the real world geometry of 3D model objects. The principle of this compression is to generate a parametric model of the image seen at the emission end and to transmit only the characteristic parameters that show how the model changes in time. These differential parameters are then used to animate the model of the image recovered at the reception end.

The first step in a full automatic MBVC system is the *face detection* allowing the identification and location of the face in first image frames. The next step is *motion estimation* encompassing global 3D-motion recovery, local motion estimation, expression and emotion analysis, etc. The problem is technologically difficult, as 3D motion parameters have to be extracted from a sequence of 2D images of the performer's head-and-shoulders.

In our system, we use a *3D tracking method* for the real-time measurement of six head motion parameters, namely 3D position and orientation, and the focal length of the camera. This method uses a 3D wireframe head model, a 2D feature-based matching algorithm, and an Extended Kalman Filter (EKF) estimator. Our global motion tracking system is

meant to work in a realistic CVE without makeup on speaker's face, with uncalibrated camera, unknown lighting conditions and background.

The EKF converts the 2-D feature position measurements, using a perspective camera model into 3-D estimates of the position and orientation of the head [3, 4, 5]. The EKF recursive approach captures both the cause-effect and the dynamic nature of the tracking, offering also a probabilistic framework for uncertainty representation.

The EKF procedure is applied to nonlinear systems and consists of two stages: time updates (or prediction) and measurement updates (or correction). At each iteration, the filter provides an optimal estimate of the current state using the current input measurement, and produces an estimate of the future state using the underlying state model. The values, which we want to smoothen and predict independently, are the tracker state parameters.

The EKF state and measurement equations can be expressed as:

$$s(k+1) = As(k) + x(k) \quad (1)$$

$$m(k) = Hs(k) + h(k) \quad (2)$$

where s is the state vector, m is the measurement vector, A is the state transition matrix, H is the Jacobian that relates state to measurement, and $x(k)$ and $h(k)$ are error terms modeled as Gaussian white noise.

The observations are the 2D feature coordinates (u, v) , which are concatenated into a measurement vector $m(k)$ at each time step. The observation vector is the back-projection of the s state vector containing the relative 3D camera-scene motion, and the camera internal geometry, namely the focal length. In our case the state vector is $s(\text{translation}, \text{rotation}, \text{velocity}, \text{focal_length})$ that contains the relative 3D camera-object translation, rotation and their velocities, and camera focal length.

The EKF requires a physical dynamic model of the motion and a measurement model relating image feature locations to motion parameters. Additionally, a representation of the object (user's head) is required.

The dynamic model is a discrete-time Newtonian physical model of a rigid body motion, moving with constant velocity. The measurement model relates the state vector s to the 2D-image location (u_k, v_k) of

each image feature point, using a perspective projection model.

We employ a three-parameter incremental rotation (w_x, w_y, w_z) , similar to that used in [4] and [6] to estimate inter-frame rotation. The incremental rotation computed at each frame step is combined into a global quaternion vector (q_0, q_1, q_2, q_3) used in the EKF linearization process and rotation of the 3D-model [7].

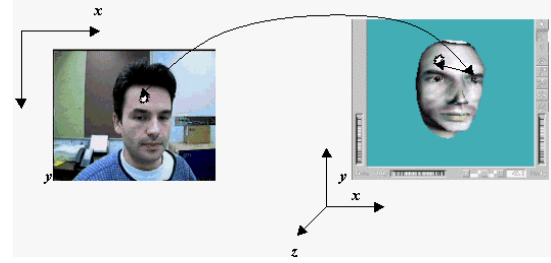


Figure 7. Hotspot in Video and Correspondent 3D Head

4.1 EKF Initialization

The 3D-model provides the initial structure parameters of the Kalman filter. Each 2D-feature point (u_i, v_i) corresponds to a structure point $p_i(X_i, Y_i, Z_i)$. As shown in Fig. 7, these (u_i, v_i) points are obtained by intersecting the 2D image plane with a ray rooted in the camera's center of projection COP and aiming to the 3D structure point on the head model.

- Step 1. The user positions his/her face** at the center of the screen, and adjust the matching of the live image and the projected mesh, so that the projected mesh covers the entire facial region.

Step 2. Left mouse click on every rigid feature point of interest on the live image. An automatic program function takes care to properly align the selected live feature point to a vertex of the projected mesh.

Step 3. Right mouse click anywhere on the active Windows “live” image triggers the tracking process (booting EKF module).

Figure 8. The EKF “Boot” Algorithm

The typical point identification problem of the 3D pose recovery from 2D images is solved in our case by identifying corresponding points in both the 2D

live image of the subject and the 3D model of the subject's head. In order to aid the point identification process, we are using an augmented reality technique by projecting in the 2D live image the 3D mesh used to model the head. A multiple "point identification" procedure using this augmented reality technique is summarized in Fig. 8. At this development stage it is still up to the user to arrange the scale matching between the live face image and the projected mesh.

4.2 EKF Update

At each iteration, the EKF computes an estimate of the rigid 3D motion that must probably correspond to the motion of the 2D live image. We employ the Kanade-Lucas-Tomasi (KLT) [8] 2D-gradient feature tracking method, which robustly performs the tracking reinforced by the EKF estimation output. An estimate of motion and camera focal length is found at each step. After the 3D-motion and focal length are recovered, a perspective transformation will project feature points back onto the image to determine an estimated position of the 2D feature trackers. At the next frame in the sequence a 2D tracking is performed starting at this 2D estimated position. The current matching coordinates of tracked features are fed back into the Kalman filter as the observation vector, and the loop continues. The feedback from EKF is used to update the 3D-model pose parameters, i.e. provides the 3D head tracking information.

The recovered 3D position and orientation are propagated to the *Head Modeling* block of the CVE system, which renders a new posture of the 3D-model as illustrated in Fig. 9.



Figure 9. Tracking the head motion

5. Conclusion

We described a prototype developed for industrial teletraining which deployed a comprehensive set of components. Among the features in the prototype, speech recognition and head-help to interface the

human users with the virtual environment in a way which is more natural to users.

Acknowledgements

The authors acknowledge the research and development contributions of François Malric, Ramsey Hage and Pierre Desmarais, as well as the financial assistance of Newbridge Networks and CITO. We also acknowledge the financial assistance of the Brazilian Ministry of Education Agency's CAPES scholarship, and the Natural Sciences and Engineering Research Council of Canada (NSERC) Scholarship Program.

References:

- [1] J. Leigh, "A Review of Tele-Immersive Applications in the CAVE Research Network", Proceedings of the IEEE International Conference on Virtual Reality, Texas, March 1999.
- [2] J. C. de Oliveira; S. Shirmohammadi and N. D. Georganas, "Collaborative Virtual Environments Standards: A Performance Evaluation", IEEE DiS-RT'99, Greenbelt, MD, October 1999.
- [3] A. Azarbayejani, T. Starner, B. Horowitz, and A. Pentland "Visually controlled graphics", IEEE Trans. Pattern Analysis and Machine Intelligence, 15(6): pages 602-605, June 1993.
- [4] T.J. Broida and R. Chellappa "Estimation of object motion parameters from noisy images", IEEE Trans. Pattern Analysis and Machine Intelligence, 8(1): pages 90-99, January 1986.
- [5] D.B. Gennery. "Visual tracking of known 3-dimensional object", Int. J. of Computer Vision, 7(3), pages 243-270, 1992.
- [6] A. Azarbayejani and A. Pentland "Recursive estimation of motion, structure, and focal length", IEEE Transactions on Pattern Analysis and Machine Intelligence, 17(6), 1995.
- [7] K. Shoemaker. "Quaternions", Department of Computer and Information Science University of Pennsylvania Philadelphia, PA 19104.
- [8] J. Shi, and C. Tomasi, "Good Features to Track", IEEE Conf. on Computer Vision and Pattern Recognition (CVPR94) Seattle, June 1994.
- [9] K. Aizawa, T.S. Huang, "Model Based Image Coding: Advanced Video Coding Techniques for very Low Bit-Rate Applications", Proc. IEEE, vol. 3, No. 2, Feb. 1995.