# A Realistic Video Avatar System for Networked Virtual Environments

Vivek Rajan, Satheesh Subramanian, Damin Keenan
Andrew Johnson, Daniel Sandin, Thomas DeFanti

Electronic Visualization Laboratory
University of Illinois at Chicago, Chicago, IL, USA
vrajan@evl.uic.edu

## Abstract

*With the advancements in collaborative virtual reality applications there is a need for representing users with a higher degree of realism for better immersion. Representing users with facial animation in an interactive collaborative virtual environment is a daunting task. This paper proposes an avatar system for a realistic representation of users. In working towards this goal, this paper will present a technique for head model reconstruction in tracked environments, which is rendered by view dependent texture mapping of video. The key feature of the proposed system is that it takes advantage of the tracking information available in a VR system for the entire process.*

## 1. Introduction

The word avatar, inspired from hindu mythology, stands for an incarnation or embodiment of human form. In the context of the virtual reality environment, an avatar is a graphical representation of the human form. The human face is endowed with a myriad of gestures which need to be represented in a avatar in order to impart it with realism in the context of a networked virtual environment. With continuing research and new technologies, combined with growing expectations, there is a need for avatars that are more human-like.

In this paper, we describe how view dependent texture mapping can be used to produce a realistic head of an avatar, and eliminating in the process the constraints posed by background and lighting requirements. The process essentially involves projecting real-time video images on a 3D model of the user's head to render the head of the avatar. This paper also presents an automated image based model generation technique for reconstructing the head model of a user, making the system cost effective and usable.

## 2. Previous Work

A large amount of research has gone into representing users with facial animation. In general, the methods for representing facial expression fall into two categories. One method is to extract various facial parameters from the video and use the parameters to animate a model. The second method is to use the video directly by texture mapping the video onto some model. It is observed that video directly used for rendering avatars achieves higher degree of realism in comparison to a generic model that may have been modified using extracted parameters from the video data. This section will review some of the significant methods.

One of the previous methods in this area was video texturing of the face [1]. This technique uses a video image of the user's face as a texture map onto a simple model. The background image captured by the camera is processed to extract the subset of the image containing the user's face using a simple background subtraction algorithm. The texture mapping is done using a simple frontal projection. This technique is a compromise between mapping on a simple shape (e.g. box, ellipsoid) which would give unnatural results and mapping on a full-featured human head model where more precise image-feature alignment would be necessary.

In model-based coding of facial expressions, as in the previous technique, the user's head and shoulder video image is captured by a camera, but instead of transmitting whole facial images the images are analyzed and a set of parameters describing the facial expression are extracted [3] [9]. This method can be used in combination with texture mapping. The model needs an initial image of the face together with a set of parameters describing the position of the facial features within the texture image in order to fit the texture to the face. Once this is done, the texture is fixed with respect to the face and does not change, but it is deformed together with the face. This differs from the approach dis-

cussed earlier where the face was static and the texture was changing. The main drawback of the facial module is that it is not always possible for users to be in front of the camera.

A 2D video avatar [12] is a flat polygon onto which real-time video of the user is texture mapped. The background of the video is removed either by chroma keying or background subtraction. The main disadvantage with this method is that it is two dimensional and the user has to always face the camera for a good representation. Secondly, chroma keying and background subtraction require a fixed color background with proper lighting for obtaining a good background-cutout of the user. These constraints make it not usable in virtual environments like the CAVE.

Static photo-realistic 3D representation of users was demonstrated by Insley [7]. The process involved acquisition of views from 360 degrees of the user and selecting the appropriate view depending on the remote user's view point to render the avatar. As 2D images are used to represent the avatar, they are not truly 3D and look good only from the same distance and height of the camera used in recording. These avatars are mainly significant for recognition/identification of the user.

We experimented with generating 3D avatars using range information from stereo cameras [5]. Range information obtained from a stereo camera is used to segment the background, and the foreground points are meshed to obtain an approximate model of the user. As the model of the avatar is constructed in real-time for every frame, the 3D shape or size of the user need not be known. However, the quality of the model generated depended upon the background as well as the lighting. Moreover, the method could not be used in real-time applications since the frame rate achieved by this technique was about 3-4 fps.

Most of the constraints regarding background and lighting requirements are overcome by using the view dependent texture mapping technique, though we require one or two spot lights to light the user's face. Also, since video texturing of the face uses a simple model and model-based technique of facial expressions uses a generic model, the results are not very realistic. Our technique utilises an approximate model of the user's head which is then rendered by projecting real-time video of the user onto the model, resulting in a realistic video avatar.

## 3. Overview

A video camera on a tripod is placed in one of the front corners of the CAVE. More cameras can be placed in the CAVE to increase the field of view. The first step is to acquire the data required for calibrating the camera. An LED is attached to the wand using velcro for automatically locating the 2D location of the wand (tracker) in the video image. Images of the LED are captured from the video camera while the wand is moved around in the CAVE. The 3D location of the wand and it's corresponding 2D image points are obtained automatically, which is then given as input to the camera calibration program. The camera calibration program uses this data for computing the intrinsic and extrinsic parameters of the camera (i.e. the position, orientation and focal length of the camera).

Having calibrated the camera, the next step is to reconstruct the head model of the user. Different views of the user are captured by the calibrated camera. These images are segmented using background subtraction, which are then used to reconstruct the model of the user. The model, stored in the performer binary format (pfb), can be used as head model of the avatar in collaborative virtual reality applications. The generated head model of the user and the camera parameters of the calibrated camera(s) are sent to all the remote sites. The process of head model reconstruction is explained in detail in section 5.

The outline of the operation of the system is shown in figure 1. The calibrated camera placed in the CAVE captures the video of the user in real-time and the tracker data is also acquired simultaneously. To keep the video in sync with the tracking information, the video and tracker data are packed together and then sent over the network. As low latency is important in collaborative applications, the UDP protocol is used to send the information across the network.

At the remote site, a UDP receiver receives and unpacks the data to separate the video and tracker information. The transformations of the avatar's head are updated based on the tracking information received. Using projective texture mapping, the video is projected onto the head model to obtain a realistic head representation of the user.

A single camera provides only a limited field of view and may not be able to capture the user's face all the time. Multiple cameras may be used to increase the field of view. Based on the remote user's position, the video from the appropriate camera is projected onto the avatar's head model.

## 4 Camera Calibration

Camera calibration is the process of determining the internal camera's geometric characteristics (intrinsic parameters) and the 3D position and orientation of the camera frame relative to a certain world coordinate system (extrinsic parameters). For the proposed video avatar system, the camera needs to be calibrated in the CAVE tracker space for two reasons - head model reconstruction of the user and view dependent texture mapping of video. This section describes a method for calibrating a camera in the CAVE tracker space using Tsai's camera calibration technique [11].

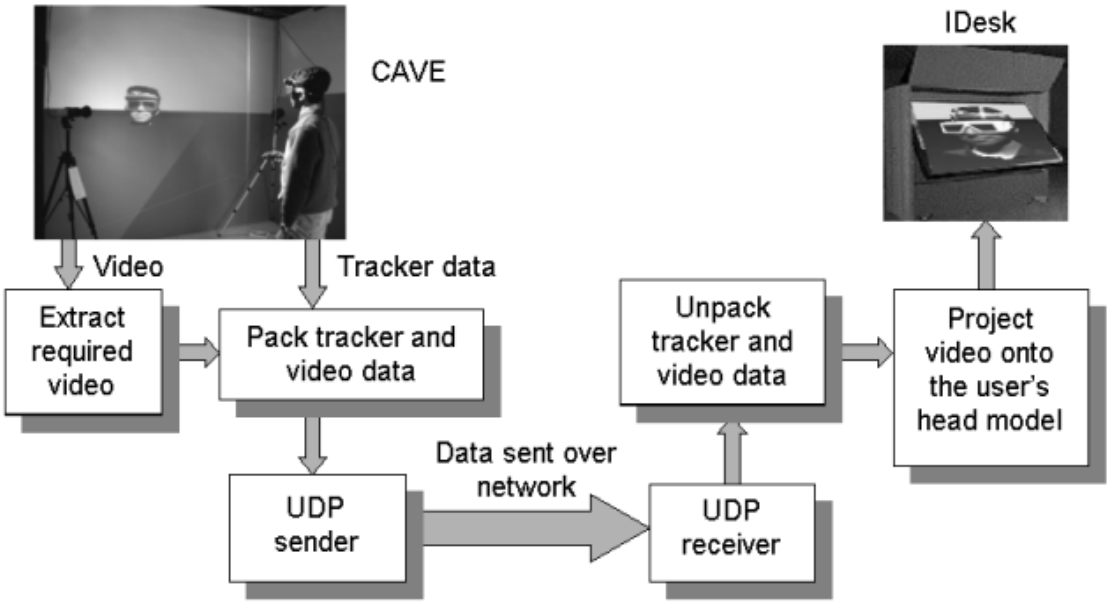



**Figure 1. Operation of the Video Avatar System. Camera(s) placed in the CAVE captures the video of the user. The portion of the video containing the user's head is extracted and transmitted along with the tracking information. On the remote site, the information is received and the video is projected onto the head model of the user, to obtain a realistic head representation.**

### 4.1 Acquisition of Calibration Data

Calibration of the camera for the pin-hole model consists of the 3D world coordinates of a feature point and the corresponding 2D coordinates of the feature point in the image.

For acquisition of the 3D world coordinates, the tracking system in the CAVE is used. Obtaining the corresponding 2D image point, requires an LED to be attached to the tracker, thereby making it possible to detect the position of the tracker in the image. LEDs have a high saturation value and can be easily detected in the HSV color space. The image of the LED captured by the camera is segmented by thresholding the image with value almost equal to the value of the LED. Segmentation can be improved further by applying a sequence of morphological operators, thereby segmenting the LED portion from the rest. The centroid of the segmented pixels is the 2D coordinates of the LED. Thus, the 2D image points of the tracker are obtained.

### 4.2 Obtaining Camera Parameters

Once a set of 3D and 2D points are acquired, the equations relating the 3D world coordinates to the 2D image coordinates are solved to obtain the camera parameters. From the camera parameters obtained, the projection matrix of the camera can be computed. A set of at least 11 points is required for calibration but a set of about 30 points are necessary for a good calibration. Even with a large number of input points the registration may not be accurate. To correct the registration an interactive tool described in section 7 is used.

It is important that the camera calibration error is less than 20 mm for a good head model reconstruction and a good registration of video with the head model. According to Tsai's camera calibration paper [11], it is essential to keep the accuracy of the 3D points (world coordinate points obtained for camera calibration) atleast one order of magnitude tighter than the final goal of 3D measurement using the calibrated camera; for example, if the the final accuracy is desired to be 10 mm, then the 3D points have to be 1 mm accurate. Hence, to achieve a calibration error less than 20 mm we need to use a tracking system that has a resolution less than 2 mm. The Intersense IS900 tracking system, which was used for this work, is very accurate and has a resolution of 1.5 mm.

The camera used for this work is a Sony DXC950 3CCD model (with Fujinon VCL 714BXEA zoom lens), which has a separate CCD for each primary color (red, green, and blue). The focal length of the camera varies with the zoom, and hence, it is important to maintain the zoom of the cam-

era constant throughout the operation.

## 5. Head Model Reconstruction

The head model of the user is reconstructed using a volumetric intersection of the silhouettes of the user's head. The volumetric intersection technique suggested by Gibson et al. [4] uses a turntable on which the object is placed and the images of the object are captured by a calibrated camera while it is turned through $360$ degree. This method would work well for static objects that doesn't move but may not work well for human subjects as they may tend to move while the camera captures the different views. Moezzi et al. proposed a technique [8] that simultaneously captures different views of the object from multiple calibrated cameras. This technique does solve the problem of reconstructing moving objects but would require a large number of cameras. Unlike these techniques, our technique takes advantage of the tracking information available in VR systems to determine the position and orientation of the object for each view. Thereby, we eliminate the need for multiple calibrated cameras.

Image based modeling in the CAVE requires a camera to be placed in the CAVE and calibrated in the tracker space as described in section 4. Different views of the tracked user are captured by the calibrated camera. These views are segmented and then using a volumetric intersection technique [4] the model of the head is reconstructed. The following sections describes each step in detail.

### 5.1 Image Acquisition

The image acquisition process requires the user to stand approximately in the center of the CAVE and turn through approximate steps of 45 degrees. A video camera, calibrated in the CAVE coordinate system, captures the image of the user at each step of rotation. The user has to wear the standard stereo glasses during this process as the head model needs to be generated with the glasses. The transformation matrix of the head tracker, required for the model reconstruction, is also recorded for each of the views. The quality of the model improves as the number of views of the user increases.

### 5.2 Image Segmentation

The user's head needs to be segmented from each of the acquired images. A background subtraction method proposed by Horprasert et al. [6] is used for getting an initial segmentation. If the background in the CAVE is complex, the subtraction may not produce a good silhouette of the user. To improve the segmentation, a well lit white background may be used while acquiring the user's images. Moreover, since only the head model of the user is being reconstructed, the remaining portions of the image can be segmented by simple bounding box technique. The segmentation is further improved by applying a sequence of erosion and dilation operations.

### 5.3 Image Based Modeling

Having segmented the different views of the user, these images have to be put together to form the 3D model of the user's head. A volumetric intersection of the silhouettes [4] is used to achieve this. The head of the user is assumed to fit inside a fixed size cube. The cube is transformed based on the transformation of the head tracker, while a calibrated camera captures different views of the user in the CAVE. The cube is sampled into $64^3$ voxels for the volume representation. For each view of the user, a distorted cone is created by the user's silhouette and the camera viewpoint. The intersection of all these cones provides a volumetric representation of the user's head. Let, $(x_{min}, y_{min}, z_{min})$ and $(x_{max}, y_{max}, z_{max})$ be the corners of the fixed size cube that is sampled into $64^3$ volume of voxels $((0,0,0) - (64,64,64))$. Let, $(v_x, v_y, v_z)$ represent the coordinate of a voxel in the volume, $(c_x, c_y, c_z)$ and $(w_x, w_y, w_z)$ be it's corresponding coordinates in the fixed sized cube and world (CAVE) coordinates, respectively. Let, $(i_x, i_y)$ be the point on the image corresponding to the world coordinate $(w_x, w_y, w_z)$.

Each voxel in the volume, $(v_x, v_y, v_z)$, is mapped to it's corresponding point $(c_x, c_y, c_z)$ in the fixed size cube using equations 1 - 3. $(c_x, c_y, c_z)$ is transformed to the world coordinates $(w_x, w_y, w_z)$ by the transformation matrix of the head tracker. The image coordinate $(i_x, i_y)$ of the point $(w_x, w_y, w_z)$ can be obtained by transforming $(w_x, w_y, w_z)$ with the camera's projection matrix.

$$c_x = \frac{v_x \times (x_{max} - x_{min})}{volume\_size} + x_{min} \tag{1}$$

$$c_y = \frac{v_y \times (y_{max} - y_{min})}{volume\_size} + y_{min} \tag{2}$$

$$c_z = \frac{v_z \times (z_{max} - z_{min})}{volume\_size} + z_{min} \tag{3}$$

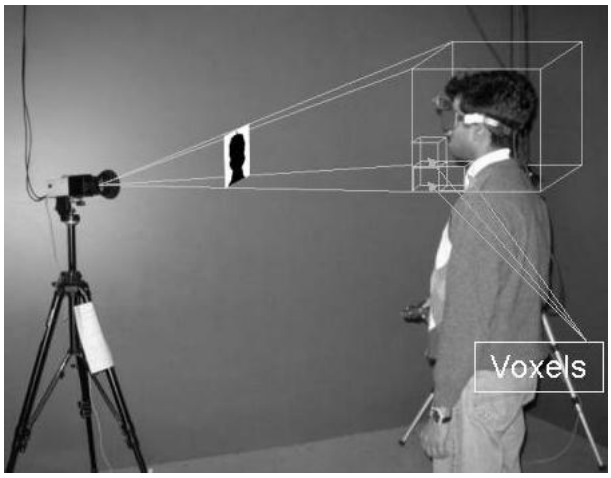
where $volume\_size = 64$.

$$\begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} = \begin{bmatrix} Head \\ Transform \\ Matrix \end{bmatrix} \begin{bmatrix} c_x \\ c_y \\ c_z \end{bmatrix} \tag{4}$$

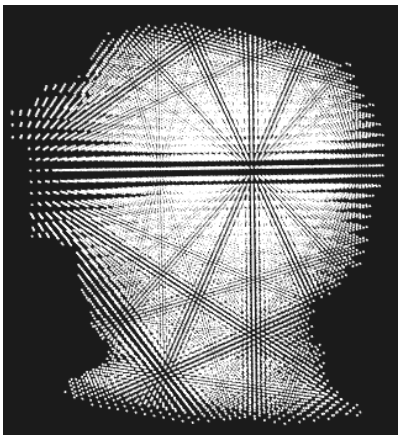
$$\begin{bmatrix} i_x \\ i_y \\ X \end{bmatrix} = \begin{bmatrix} Camera \\ Projection \\ Matrix \end{bmatrix} \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} \tag{5}$$

where $X =$ don't care.

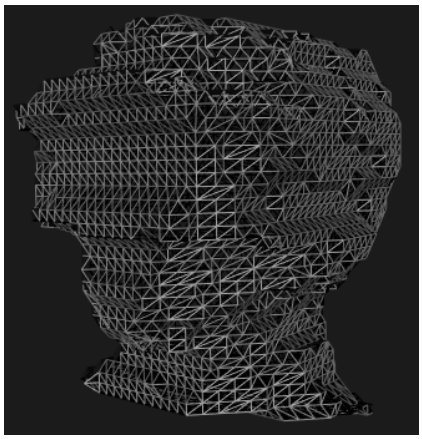
(a) The segmented views of an user standing at the center of the CAVE and turning through approximate steps of $45$ degrees.



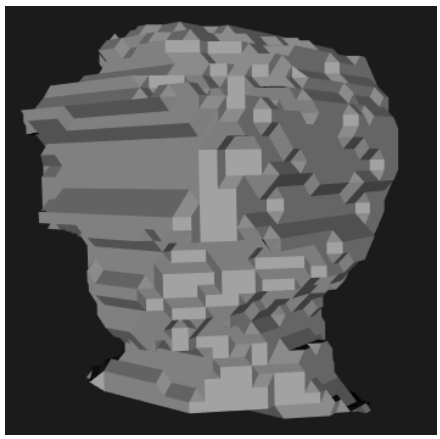

(b) The fixed size cube confines the user's head and is transformed based on the transformation of the head tracker. The cube is sampled into $64^3$ voxels. Few voxels are shown for the purpose of illustration. The voxels are projected onto the segmented images to check if they are part of the user's head.



(c) Point cloud model of the user generated by volumetric intersection of the segmented views. The image shows the view of the model with the user's face facing left.



(d) Polygonal wireframe model generated by meshing the point cloud model.



(e) Flat shaded model

**Figure 2. Volumetric intersection technique to reconstruct the head model**

The values of all the voxels in the volume are initialized to $0$. Each voxel in the volume, is transformed to the image coordinates using the above equations. If the image coordinates corresponding to the voxel is a part of the user's head (i.e. the segmented portion), then the value of the voxel is incremented. This is repeated for every view. A voxel that is a part of $n$ views would have a value $n$ as each view would have incremented the value of that voxel. Hence, all the voxels with the value $n$, where $n$ is the number of views, are a part of the bounding volume of the user's head. All these voxels form a point cloud of the user's head, which can be polygonized using the marching cube algorithm to obtained a 3D model. Figure 2 shows the model generated by the volumetric intersection of $8$ views.

The generated model is a fair approximation of the user's head. Fine details, like the lips, are not generated. The time taken for generating the model depends on the number of images used for generating it. On a SGI Onyx, it takes over a minute to perform the volumetric intersection on a single image. Usually, about 8 images are required to generate a good quality head model. Thus, the time taken for generating a good model is about 8 - 10 minutes. The time taken is high because there were no optimizations performed for the intersection calculation. The polygon count of the head model is approximately 6000. Hence, the model can be used directly for real-time interactive applications.

## 6. Rendering the head model

View dependent texture mapping (VDTM) [2] is a technique for generating novel views of a scene with approximately known geometry making use of one or more original views. This technique can be applied directly to 3D video avatars. VDTM requires to have a knowledge of the geometry of the scene being rendered and also have a set of calibrated images that can be mapped onto the model. In the case of video avatars, the geometry of the scene (i.e. the head model of the tracked user) is available and the calibrated images are also available from the calibrated camera. Hence, the technique of VDTM is directly applicable for video avatars. Moreover, to implement this efficiently in real-time, the projective texture mapping [10] feature of OpenGL is used.

In order to perform projective texture mapping for the video avatars, the user specifies the position and the orientation of the virtual camera (which is obtained by calibrating the camera), and a virtual image plane with the video as the texture. The video texture is then cast onto the head model using the camera position as the center of projection to obtain a realistic avatar head as shown in figure 3.

**Figure 3. Projecting real-time video of the user onto the head model of the user to obtain a realistic avatar head.**

### 6.1 Avoiding Projection of Video on Occluded Polygons

The video should be mapped onto the portions of the head model that are visible from the camera viewpoint. The OpenGL implementation of projective textures does not perform any visibility checks and the video is mapped throughout the geometry of the head model including the occluded polygons as shown in figure 4. In this context, occluded polygons refer to the polygons of the head model that are occluded from the viewpoint of the video camera and not the polygons occluded for the viewer. This causes the avatar to look very odd from viewpoints other than the viewpoint of the camera.

**Figure 4. The OpenGL implementation of projective textures does not perform any visibility checks and hence the video is also mapped onto the occluded polygons. This causes the head to look distorted and thus it is necessary to avoid this.**

The projection of the video texture on the backfacing polygons can be avoided using user defined clipping planes.

The basic idea is to draw the portion of the head model facing the camera with the projected texture state enabled and draw the portion of the model facing away with a default texture. To implement this, two clipping planes with opposite normals are required. The model is effectively drawn in two passes. In the first pass, one of the clipping planes clips the backfacing polygons and the frontfacing polygons are drawn with the projected texture enabled. In the second pass, the second clipping plane is enabled, which clips the frontfacing polygons and draws the backfacing polygons with a default texture. This is illustrated in figure 5.

**Figure 5. Projection of video onto occluded polygons is avoided by rendering the head in two passes. In the first pass, the occluded polygons are clipped and the frontfacing polygons are drawn with the projective textures enabled. In the second pass, the frontfacing poygons are clipped and the occluded polygons are drawn with a default texture.**

## 6.2 Projection from Multiple Views

In a CAVE, the user could be facing in any direction, and does not necessarily have to face the video camera. As the user turns away from the camera, the portions of the user's face occluded by the camera will not be visible for the remote user. This results in poor communications because of the loss of facial expressions. Moreover, identification of the user becomes difficult. This problem can be overcome by the use of multiple cameras placed in different locations in the CAVE and projecting the video of each view based on the projection matrix of the respective camera.

If the video of all the cameras are going to be streamed over the network, the bandwidth required would be quite large and would depend on the number of cameras. To minimize the network bandwidth, only the appropriate video stream depending on the viewpoint of the remote user, is sent over the network. The proposed method is described in the next section.

### 6.2.1 Projection of Video based on Remote User's Viewpoint

Rendering the video avatar using the video that looks best from the remote user's viewpoint, helps in reducing the network bandwidth by sending only one video stream. This is illustrated with an example consisting of two cameras as shown in figure 6. The camera that is used to project video can be determined from the position of the remote user. The world (or the scene) can be divided into two regions (1 and 2), based on the world position of the cameras as shown in figure 6. The line perpendicular to the line joining the two cameras and passing through it's center divides the world in two regions. Depending on the region of the remote user, a request is sent to the server to send the appropriate video. If the remote user is in region 1 then camera 1 is used to project the video, and similarly for region 2, camera 2 is used to project the video. This can be easily extended for any number of cameras.

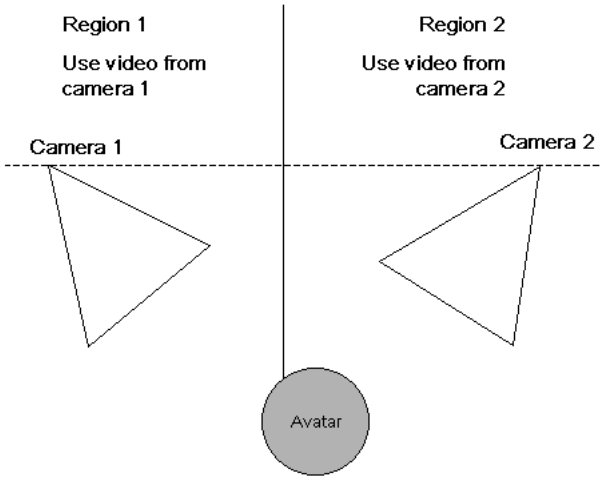



**Figure 6. Determining the correct camera for projection. This example illustrates which video image to project depending on the remote user's viewpoint for a two camera situation. The world is divided into two regions based on the camera position and video from camera 1 or 2 is used if the remote user is in region 1 or 2 respectively.**

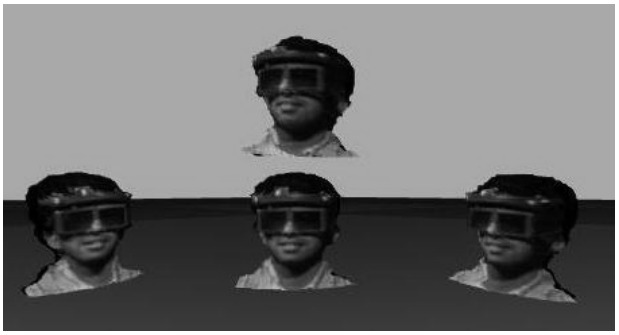
# 7 Registration Issues

The video image will not be registered correctly with the head model if the camera parameters are not accurate. The

camera calibration errors are caused because of errors in acquisition of 2D points and errors in tracker information. These errors are inherent in the system and are hard to correct. To correct the registration, an interactive tool was developed.

The tool displays the head model in three different orientations with the video texture mapped on them. It also displays a head model that spins continuously at a slow pace. The camera parameters can be manipulated using the tool with it's effect seen in real-time. Thereby, the registration of the video with the head model can be corrected with ease in 2 to 3 minutes to finish the camera calibration. Figure 7 shows snapshots of the interactive registration tool.

(a) Initial registration, using the camera parameters obtained by calibrating the camera, is not very accurate.



(b) The registration can be corrected by using the interactive tool. The tool lets the user adjust the registration in the X, Y and Z axis with it's effect seen in real-time. Thereby, the registration can be corrected with ease. This process needs to be done only once after calibrating the camera.

**Figure 7. Interactive tool for correcting registration**

Since a static model of the head is used, there is some error in registration when there is a movement of the user's mouth and jaw. In most cases, especially from the front view, this error is not very noticeable. It is the most visible from the side view of the avatar. Figure 8 clearly illustrates this error in registration. The error is quite obvious when the user's jaw is wide open and is almost not visible while the user is speaking. As static models are used, this error is inherent in the system. The use of dynamic models may be investigated to overcome this problem.

**Figure 8. The image on the left shows the side view of the avatar where the error in registration is very noticeable when the jaw of the user is open. This occurs because the head model of the user is static. The image on the right shows that the error is not very noticeable from the front view of the avatar.**

## 8 Video Streaming

Video avatars requires video to be transmitted over the network in real-time. Streaming video usually requires a great deal of bandwidth. The video frame used in this work had a dimension of $720 \times 486$, with each pixel consisting of four channels (Red, Green, Blue and Alpha), each channel being a byte long. So the size of a single video frame is $720 \times 486 \times 4 = 1399680$ bytes. To maintain a smooth video frame rate, the video needs to transmitted atleast at 15 fps. The bandwidth required to send this video at 15 fps would be $720 \times 486 \times 4 \times 8 \times 15 = 167961600$ bps (160 mbps). This amount of bandwidth is unacceptable and needs to be reduced.

### 8.1 Reducing bandwidth using a bounding box

The video avatar has the head model mapped by the video. The rest of the avatar is composed of dummy models without any video texture. Hence, the portion of video containing the user's head is only used for texture mapping. Therefore, the remaining portion of the video, which is redundant, need not be broadcasted over the network. By

broadcasting only the bounding box of the user's head in the video, the bandwidth requirements can be reduced.

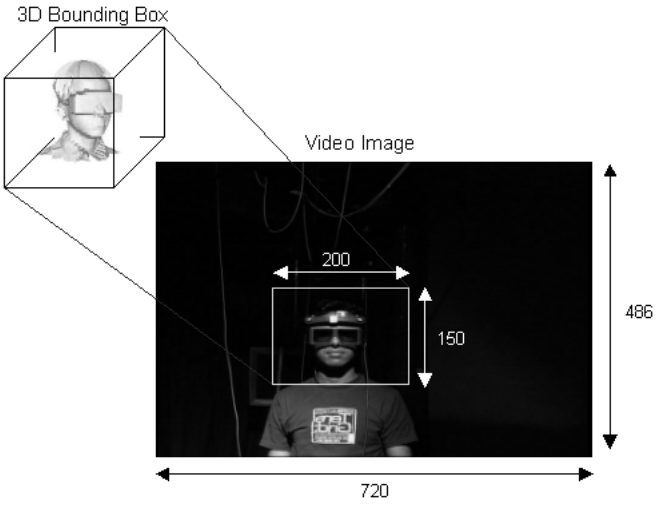



**Figure 9. The 3D bounding box of the user's head is computed by computing the bounding box of the user's head model. The corners of this 3D bounding box is projected onto the video image to obtain the 2D cutout of the user's head.**

The 3D bounding box of the user's head can be computed by calculating the bounding box of the user's head model. The vertices of the 3D bounding box can be projected onto the video image, using the camera's projection matrix, to obtain the 2D cutout of the user's head from the video as shown in figure 9. The figure shows a user standing approximately in the center of the CAVE. The 2D bounding box of the user's head is about $200 \times 150$, which is about 11 times smaller than the whole frame ($720 \times 486$). The bandwidth required to send frames of this size at 15 fps would be $200 \times 150 \times 4 \times 8 \times 15 = 14400000$ bps (13mbps). The size of the bounding box depends on the proximity of the user to the camera, but in most cases, users stand approximately in the center of the CAVE, and hence a considerable amount of bandwidth can be reduced.

OpenGL supports textures with a 5551 pixel format of RGBA with 5 bits for R, G and B channels, and 1 bit for the alpha channel. Using this format, the size of each pixel is 2 bytes long, which is half the size of the usual RGBA texture with 1 byte/channel. The network bandwidth can be further reduced by a factor of 2 by broadcasting the video in this format. This format would lead to a degradation of the quality of image compared to the 4 byte RGBA format. However, the benefit gained by using this format is significantly beneficial for networks with low bandwidths.

### 8.2 Temporal Errors

Some collaborative virtual reality applications transmit tracking information and video (if any) using separate channels. However, this is not suitable for the proposed system as it may lead to some temporal errors. Temporal errors, produced by out of sync tracker and video data, results in poor registration of the video with the head model. For the video to be registered correctly, the head tracking data and the video need to be synchronized. Therefore, the tracking information (head position and orientation) corresponding to a video frame is packed along with each video packet. Thereby, the temporal errors in the registration of the video can be eliminated.

## 9 Discussion

The presented system uses an image based modeling technique to generate the head model. However, the model can be generated using other techniques too. For example, the model could be generated using a 3D laser scanner. But this would require the user to manually align the model with the tracker coordinate system. The image based modeling technique is completely automated and requires no user intervention to align the model with the tracker coordinate system.

In many cases, there are more than one person in the CAVE. In such cases, the system would work well as long as the other people in the CAVE don't get between the camera and the tracked user. If people came between the camera and the user, the camera would not capture the video of the user, thereby resulting in projecting incorrect video onto the head model. Hence, if there are more than one person in the CAVE, care should be taken to not block the camera's view of the tracked user.

Lastly, the proposed system was mainly experimented in the CAVE but it is applicable to the ImmersaDesk and other related VR devices. The main requirement for this system to work is that the VR devices should have an accurate tracking system. Inaccurate tracking systems would lead to poor registration of the video image.

## 10 Conclusion

The concept of an avatar in a virtual environment is not merely to denote the presence of a user but to impart it with realism that would make the whole experience of interaction with an avatar life-like. This paper is a step in that direction.

This paper explores ways of imparting realism to an avatar in a convenient way, while keeping in mind the goals of minimizing the human intervention factor, ease of use and cost. The key feature of the proposed system is that

it takes advantage of the tracking information available in a VR system for camera calibration, head model reconstruction and projective texture mapping. The technique of head model reconstruction enables creation of models in tracked virtual environments in a convenient way using images, thereby, eliminating the need for expensive laser scanners and the need for any manual intervention. View dependent texture mapping helps in removing the need for most background and lighting constraints. The use of multiple cameras allow a greater degree of freedom for users without any restriction of motion. This eases the behavior of the user, an improvement in usability. The "bounding box" technique for reducing bandwidth makes the system usable in real world networks. Moreover, the interactive registration tool enables less precise tracking systems, such as electromagnetic trackers, to be usable with a small margin of error.

The rendered head model is a good representation of the user. Projecting real-time video results in a good transmission of the user's facial expressions and also helps in easy identification of the user. Imperfections are present in the system, especially in the quality of model generated, and do require enhancements, but nevertheless, it is a step forward. Static head models cause some errors in registration due to the jaw motion. Another issue is that the switching of video from one camera to another is quite visible. These issues will have to be addressed for achieving more realism.

Some of the enhancements that can be done to improve the system are listed as follows:

- Reducing the time taken for model reconstruction. Optimization algorithms, for volumetric intersection, could possibly be investigated.
- Simultaneous projection of multiple video streams to avoid switching between video and to produce more realistic avatars. This can be implemented using networks with higher bandwidths.
- Implementation of dynamic head models that can synchronize jaw movements in real-time. Video based techniques, for capturing jaw motions, could be investigated.
- Small lipstick cameras could be used to minimize the cameras from blocking the CAVE screens.

This paper was an attempt to address some issues in order to impart some realism to an avatar. The future work may be thought of as a road towards realism whose foundation was attempted by this work.

## Acknowledgements

The virtual reality research, collaborations, and outreach programs at the Electronic Visualization Laboratory (EVL) at the University of Illinois at Chicago are made possible by major funding from the National Science Foundation (NSF), awards EIA-9802090, EIA-9871058, EIA-0115809, ANI-9980480, ANI-9730202, and ANI-0123399, as well as the NSF Partnerships for Advanced Computational Infrastructure (PACI) cooperative agreement ACI-9619019 to the National Computational Science Alliance. EVL also receives funding from the US Department of Energy (DOE) Science Grid program and the DOE ASCI VIEWS program. In addition, EVL receives funding from Pacific Interface on behalf of NTT Optical Network Systems Laboratory in Japan.

The CAVE and ImmersaDesk are registered trademarks of the Board of Trustees of the University of Illinois.

## References

[1] T. K. Capin, I. S. Pandzic, N. Magnenat-Thalmann, and D. Thalmann. *Avatars in Networked Virtual Environments*. John Wiley & Sons, 1999.

[2] P. E. Debevec, Y. Yu, and G. D. Borshukov. Efficient view-dependent image-based rendering with projective texture-mapping. In *Eurographics Rendering Workshop*, pages 105–116, June 1998.

[3] I. Essa, S. Basu, T. Darrell, and A. Pentland. Tracking and interactive animation of faces and heads using input from video. In *IEEE Proceedings of Computer Animation*, 1996.

[4] D. P. Gibson, N. W. Campbell, and B. T. Thomas. Iterative generation of 3-d models from a set of 2-d images. In *Eurographics*, pages 135–145, March 1998.

[5] X. Gong. Video avatars using depth from stereo. Master's thesis, University of Illinois at Chicago, 2000.

[6] T. Horprasert, D. Harwood, and L. Davis. A statistical approach for real-time robust background subtraction and shadow detection. In *The Proceedings of IEEE ICCV'99*, 1999.

[7] J. Insley. Using video to create avatars in virtual reality. In *The Visual Proceedings of the 1997 SIGGRAPH Conference*, Aug 1997.

[8] S. Moezzi, L.-C. Tai, and P. Gerard. Virtual view generation for 3d digital video. *IEEE Multimedia*, 4(1):18–26, 1997.

[9] I. Pandzic, P. Kalra, and N. Magnenat-Thalmann. Real time facial interaction. *Displays*, 15(3):157–163, 1994.

[10] M. Segal, C. Korobkin, R. van Widenfelt, J. Foran, and P. Haeberli. Fast shadows and lighting effects using texture mapping. In *Proceedings of the 19th Annual Conference on Computer Graphics*, pages 249–252, 1992.

[11] R. Y. Tsai. An efficient and accurate camera calibration technique for 3d machine vision. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 364–374, 1986.

[12] S. Yura, T. Usaka, and K. Sakamura. Video avatar: Embedded video for collaborative virtual environment. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, volume 2, Jun 1999.