The Development of a Generic Framework for the Implementation of a Cheap, Component-Based Virtual Video-Conferencing System

Soteri Panagou, Shaun Bangay

{cssp|cssb}@cs.ru.ac.za

Multimedia Centre of Excellence, Computer Science Department, Rhodes University, Grahamstown, 6140, South Africa

Abstract

We address the problem of virtual-videoconferencing. The proposed solution is effected in terms of a generic framework based on an in-house Virtual Reality system. The framework is composed of a number of distinct components: model acquisition, head tracking, expression analysis, network transmission and avatar reconstruction. The framework promises to provide a unique, cheap, and fast system for avatar construction, transmission and animation. This approach affords a conversion from the traditional video stream approach to the management of an avatar remotely and consequently makes minimal demands on network resources.

Categories: I.3.7 [Three Dimensional Graphics and Realism], I.4.5 [Reconstruction], I.4.8 [Scene Analysis: Tracking, Shape, Time-Varying Imagery], H.1.1 [Videoconferencing], H.4.3 [Coding and Information Theory].

1 Introduction and Motivation

To be able to transmit video streams successfully over a network depends on one of two things: either high-bandwidth network connectivity or compression of the video stream to achieve acceptable frame-rates. The promise of being able to somehow "encode" a video stream in some very compact form is an exciting one. One suggestion that has attracted attention recently is based on the idea of "virtual videoconfererencing". Rather than transmitting and displaying the scenes in their original form, the images seen from the viewer are constructed from virtual participants (avatars[1]) whose facial characteristics are enhanced and manipulated so as to closely to resemble the real persons who are involved in the conference. Only pose information for the avatars need be transmitted.

This paper discusses a framework for such a system and sample implementation of the required components.

We adopt an encoding/decoding approach to solving the problem of virtual videoconferencing. Our encoder is responsible for the generation of 3D avatar representations, tracking the subject's head, classifying his/her current expression, packaging up this data and then transmitting the packaged data to the decoder via the network.

Our decoder accepts network packets and extracts the rotation/translation and expression information from the incoming data stream. It handles generating the appropriate expression, and rotating/translating the avatar representation. We implement expression classification/reconstruction via an expression/emotion database that contains a listing of all expressions known to the system as well as deformation information required to generate each of the listed expressions.

Our implementation is overlaid on top of a Virtual Reality system that has been under development for some time at Rhodes University. The system (called CoRgi) is a second generation, object-oriented component-based distributed Virtual Reality system. The choice of this system has not only enabled quick prototyping of the

[1] The term "avatar" refers to a 3D representation of a subject in a virtual environment.

framework discussed in this paper, but its component nature has also had an impact on the development of that framework.

2 Framework

Our framework can be decomposed into 6 major categories, as illustrated in **Figure 1**. It involves an encoding/decoding process.

The encoding part is responsible for **model acquisition**, audio capture, **head tracking** and **expression analysis**. The model acquisition component is not linked to any other component on the encoding side because it represents a process that occurs only once, when a new user is introduced to the system.

The decoding part of the system is responsible for the **avatar management** and is comprised of three more specific tasks: controlling the movement of the avatar (such as rotation and translation of the head), **expression generation** and expression management. Expression generation is considered core, but falls within the realm of the avatar management component.

The last major component that is responsible for communication between the encoding and decoding parts of our system is the **networking** component.

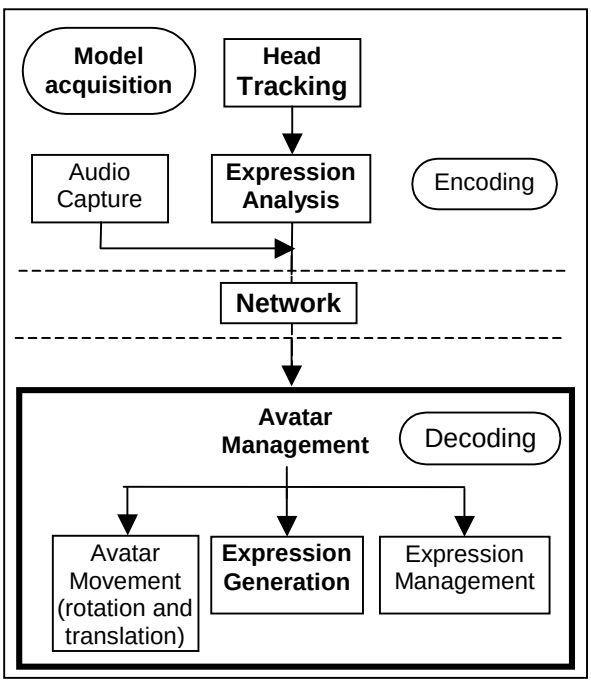

Figure 1: A conceptual design of our framework. The components in bold indicate those that are core to our design.

Although audio capture is important, it is considered beyond the scope of this discussion.

2.1 Model acquisition

We must have some method of generating 3D avatars. The aim here is to provide users with the ability to represent them in a virtual videoconference. This translates to performing some kind of "reconstruction" of the user with the result being the development a 3D model that "looks" like the user. In addition to this though, the system must be general enough to allow for the use of any predefined 3D representation.

2.2 Head Tracking

There must be some way of determining the pose and orientation of the subject. The common ways include image-processing approaches such as the one presented below in our implementation, or the use of electro-magnetic trackers similar to the ones available from Polhemus Inc[2]. These systems have the advantage of being invariant to the occlusions and shadowing that plague image-processing based approaches, and provide better than real-time feedback; the Polhemus trackers typically return 3D coordinate positions at a rate of sixty updates a second. They are ideal for any real-time tracking requirements, including pose estimation. The downside of these electromagnetic tracking systems is that they may be prohibitively expensive.

2.3 Expression Analysis

This section deals with the process of facial expression classification. There must exist of some mapping function (pre-defined table) that defines the way in which the expression identified will be overlaid over the avatar.

2.4 Expression Generation

This component refers to the method used to generate the expression with the avatar. This typically translates to some deformation approach. Deformation algorithms can be classified into two broad categories, namely:

- **Free Form Deformation** algorithms allow an object's topology to be altered by moving certain control points surrounding the object to affect its shape. The maintenance of C^1 and C^2 surface continuity is important. To this end vertices are added to and removed from the deforming object to guarantee a realistic deformation. This process tends

[2] More information about this company is available at http://www.polhemus.com/

to be computationally expensive and has been avoided for our purposes. For a more complete discussion on traditional FFD issues, refer to [6];

- **Vertex Interpolation** - these algorithms simulate the complexities of traditional FFD algorithms by moving vertices without actually introducing new vertices to the geometry. Vertex interpolation enables us to "deform" an object in real-time.

3 Background and Related Work

Research into the development of a complete virtual-videoconferencing system has been the focus of a number of projects. None of the systems discussed below provide any mention of integration of their systems with any sort of VR system, but are rather discussed as independent applications, which is a unique aspect of our work.

Escher *et al.*[2] propose a complete virtual-videoconferencing system that uses a generic mesh of a face, which is then deformed and textured to suit the real face being modelled. Although the system Escher *et al*.[2] describe provides real-time response and modelling, their parallel implementation uses 4 SGI workstations, which leads to a very expensive system.

3.1 Model Acquisition

General reconstruction techniques employed for the generation of 3D models include the use of Cyberware-type laser scanners, and pulsed lasers [15] and the deformation of generic head templates to suit the profile of the subject (Thalmann et al. [11]). Head template deformation is suited to solving the problem of human head modelling only. While this suffices for a virtual-videoconferencing system, the reconstruction system we envisage makes no assumptions regarding the structure of the final reconstructed object.

3.2 Head Tracking

In their discussion on determining the epipolar geometry of a stereoscopic scene, Zhang *et al.* [17] make use of Normalised Cross Correlation (NCC) to determine pixel matches between stereoscopic image pairs. Their motivation for this is that NCC is very robust, a fact that we have confirmed with our own experiments (see **Figure 5**) Given a pixel in a source image, NCC in its traditional form performs a 2D search around that pixel position in the target image, and returns a match for the original pixel only if a specified matching threshold has been exceeded. NCC returns -1 for a complete mismatch and 1 for a complete match. We have extended the notion of NCC matching on stereoscopic image pairs to video sequences. This approach is similar to the optical flow approach. (Reddi, [14])

3.3 Expression analysis

The analysis of facial expressions is done either by image tracking using skin colour identification and edge detection, or through the evaluation of sound generated through speech. The latter determines the list of Oxford phonemes (and associated lengths) that constitutes the input sound stream. These phonemes are then used to determine what the shape of the mouth should be for the current expression. For details of this implementation see Thalmann *et al.*[11]. Also, the effectiveness of the sound-processing component of the system with non-English speaking users is questionable.

3.4 Expression Generation

Much of the work in this field is based on research by Eckman *et al.* [1]. Their major contribution to the field of expression modelling is that of FACS (Facial Action Coding System). This system provides an enumeration of all the possible facial movements required to generate any possible expression. They call these mappings "action units". Essa *et al.* [3] criticize this system by stating that it provides only an 'approximate' mapping. This is because "some muscles give rise to more than one action unit". They go on to develop a model that enhances the basic FACS system.

The MPA (minimal perceptible action) as well as FACS systems are based on the idea that any expression that a "typical" person can generate can be decomposed into a combination of basic facial movements. The MPEG-4 face specification is also based on this approach. The standard makes use of a Face Animation Table (FAT) to determine a mapping between incoming Face Animation Parameters (FAP), the vertices affected by the current FAP and the way in which these vertices are affected. See the MPEG-4 specification for details on this approach [8].

The Motion Pictures Expert Group has ratified the final MPEG-4 specification, part of which involves the encoding of video sequences using VRML-like (Virtual Reality Markup Language) scene-graph specifications. Lee *et al*. [10] pioneered the use of this standard. The Face and Body animation Ad Hoc Group (FBA) that is a subset of the MPEG-4 group, has defined a specification for both the description and animation of human bodies and faces. The specification for facial animation is broken down into 2 categories, namely Face Animation Parameters (FAP) and Face Definition Parameters (FDP). Lee *et al.*[10] affords a brief discussion of these categories and their

implementation for a system conforming to this part of the MPEG-4 specification.

4 Design

4.1 Model Acquisition

We have developed and completely implemented the core modules listed below for our own model acquisition. This work builds on an earlier discussion in [12].

The reconstruction algorithm can be classified as a "visual hull" reconstruction method. A number of images of the object to be reconstructed are used as input to the reconstruction process. The idea is that each image contains what is called a "visual cone" enclosing the object of interest. The final reconstruction is the intersection of all the visual cones in all the images. Refer to Kutulakos *et al.* [9] and Seitz *et al.* [16] for a discussion on reconstruction approaches similar to the one discussed here. One shortcoming of these systems is that they do not discuss the efficient generation of polyhedral meshes from the reconstruction process.

4.1.1 Shape Carving

Our object is assumed to reside within a bounding volume of evenly spaced voxels[3] i.e. a regular grid. Several pictures of the object are taken from different angles and the backgrounds are stripped from these images using a chroma-keying approach. Each of these images is then projected into the voxel space orthogonally. A voxel is disabled if a background pixel projects on to. Doing this for all the images results in a set of active voxels representing the object. For the purposes of later isosurface extraction and texture generation, we keep track of the image (as well as the position in that image) that each voxel is closest to.

4.1.2 Isosurface Extraction

We have implemented an algorithm for removal of those voxels that are not associated with the surface of our object. A voxel is classified as **internal** if it is completely surrounded by active voxels. If any voxel fits this criterion, we remove it.

Applying this algorithm to the voxels remaining after shape carving results in a hollow shell, and a large decrease in the number of voxels representing our object.

[3] The term "voxel" (as used in this paper) refers to a 1x1x1 cube in 3D space.

4.1.3 Mesh Connectivity

The mesh connectivity algorithm takes as input the hollow shell of active voxels generated by the isosurface extraction algorithm. We assume that since our voxel space is regular 3D, the active voxels comprising the hollow shell will occur at fixed positions. The following algorithm generates an edge connected mesh of all the active voxels, and thus a mapping from our voxel representation to a vertex-based polyhedral representation of the object.

```
FOR all active voxels DO
  - Let position of voxel be (X,Y,Z)
  - Generate a list of active
    neighbours from position
    (X+1,Y,Z) to (X+1,Y+1,Z+1)
  - Generate all possible triangles
    by joining voxel and neighbour
    pairs.
END FOR
```

The triangle generation is achieved with a lookup table that lists all the possible triplet combinations that are allowed.

4.1.4 Mesh Decimation

We have implemented a mesh decimation algorithm to simplify the polyhedral mesh generated by the mesh connectivity algorithm. We have developed this algorithm because of in the following observation: the likelihood of our mesh containing large rectangular patches is quite high. This is a direct result of the regularity of our voxel space. Each rectangle is composed of a number of evenly spaced vertices. A simplification that forms the basis of our decimation implementation is that all vertices that are part of a rectangular patch but not part of the perimeter can be disregarded from the final polyhedral mesh.

Our algorithm can be described as a maximum growth, removal algorithm and is illustrated in **Figure 2**. Starting

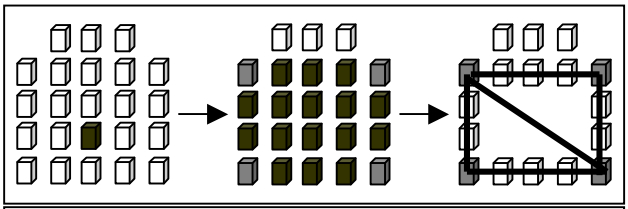

Figure 2: The decimation process in one dimension. Once the largest rectangular patch has been found, the voxels internal to the patch are disabled. The two triangles formed by the gray voxels are generated as output for the polyhedral mesh.

with each active voxel, we grow outwards in each of the 3 planes (X-Y, X-Z, Y-Z) and find the rectangle covering the largest number of active voxels for each plane. All voxels lying within each rectangle's perimeter are disabled.

The decimation algorithm differs slightly from the traditional academic mesh decimation algorithms in the way reduction is performed. Traditional algorithms will remove a vertex if the distance from the plane formed by its neighbours is less than some threshold value. We avoid this calculation because of the guarantee that our voxel space is regular. The final implementation of our decimation approach is thus significantly quicker that traditional algorithms.

4.1.5 Texture Generation

Texture generation with our algorithm is based on a relatively novel approach. It is equivalent to the approach of Seitz *et al.'s*[16] in that the voxels that remain after the space carving process has been performed are coloured to represent the object's surface. They refer to this approach as "voxel colouring". We take the notion of voxel colouring further by generating a vertex colour map of the voxels representing the surface of the reconstructed object.

We simulate traditional texture mapping with vertex colouring and Gouraud shading. The object reconstruction, isosurface extraction and finally the connectivity implementations described earlier result in a closed, connected and hollow polyhedral mesh. In addition to the vertices representing the final mesh, we also have associated RGB values with each of the vertices. We are able to "texture map" an object without actually resorting to the computational expense of doing this. Experiments by Fourie [5] reveals that texturing causes a drop of 63% in the rendering performance of an SGI Octane Graphics Workstation in experiments conducted or as much as 80% on standard PC workstations. The reason for this performance degradation arises from the fact that the traditional graphics pipeline forces the recalculation of texture information with every frame that must be rendered, discarding any previously computed results. This arises out of the necessity to maintain visual quality, in addition to other issues mentioned below.

The advantages offered by our approach are numerous:

- the removal of inverse-perspective calculations;
- the texturing of curved surfaces; this is trivial with our approach;
- objects scale very well without artefacts appearing when close to the surface.

A number of problems with our method still to be addressed include:

- large memory/bandwidth requirements with complex objects; this is as a direct result of the vertex density (how far the vertices are from one another). A higher vertex density means a superior textured result at the expense of more vertices. The situation imposes processing overhead in terms of the vertex processing.
- difficulty when performing optimisations, such as the decimation approach mentioned above; changing the structure of the object in any way will affect the texture effect.
- "albedo". The albedo of a particle is given by the "amount of light scattered by this particle in all directions in relation to the amount of incoming light" [13]. The fact that our object is textured from photographs implies that any part of a surface reflecting light in the original images will be disseminated to the textured object. To minimise this, we ensure that there is enough ambient lighting in the scene and that no light is directly reflected off the surface of our object.

4.2 Head tracking/Pose Estimation

We address the problem of head tracking by tracking a number of features on a person's face. We are able to predict which are the good matches and which matches are bad because of the video stream's temporal nature. NCC tends to be invariant to rotations of the subject's face, including yaw, and suits our needs perfectly. We are able to obtain rotational and translation information from the subject, which we can then apply to our avatar. Tracking 4 points on a person's face provides a basis for calculating the following information:

a) Rotation (rotation in X-Y)- via triangulation of the points;

b) Translation (movement in X-Y);

c) Scaling (movement in Y-Z) - The distances between the tracked pixels are used as a measure of how far the subject is from the camera.

4.3 Expression Analysis

Our experiments in emotion analysis have been restricted to two emotions/facial expressions, namely smiling and frowning.

Our implementation assumes that 4 points around the subject's mouth are tracked. These are: nose tip, area half way between chin tip and middle of the lower lip, left edge of mouth, and right edge of mouth. We monitor these four pixels, and use thresholds to decide whether, for

example, the subject is smiling or has his/her mouth open. The thresholds are determined from the initial calibration stage, when the user chooses the points to be tracked on the subject's face. The intersection between the two lines formed by the four pixels is used to decide upon the expression to be identified. As the points are tracked, so the intersection moves around. If at any point the intersection lies above the initially calculated intersection, the expression is classified as a smile, while a frown occurs only when the intersection lies below the initial intersection.

Changing the subject to be tracked causes thresholds to be automatically recalculated. This occurs during the initial calibration stage when the points around the user's face are specified.

4.4 Expression Generation/Replication

Once our Emotion/Expression analyser has managed to identify an expression (smile, open mouth or closed mouth), the expression has to be replicated. We proceed to deform our 3D model to have the same expression as the real-world subject.

Our implementation is based on **Equation 1.1**, an algorithm proposed by Hung *et al.*[7], that has complexity *O(n)*. The different components of this equation are as follows:

- P = current vertex;
- K = direction of deformation;
- P_o = point of source force; this is the position at which the deformation force originates;
- P_d = point of destination force; this denotes the place when the effect of the deformation force ends;
- R = radius of influence;
- t = the current time (time 0=no deformation; time 1 = complete deformation).

The radius of influence is a very important variable in the equation. This component dictates the extent of the deformation's impact on the object.

This algorithm provides three-tier control of the deformation process. The first method of control arises from the source/target force combination. The control of the deformation using time provides accurate control over the animation generated by the deformation. Finally, as mentioned above, the radius provides control over how much of the object gets deformed. Additionally we also support multiple deformations being performed on an object at the same time.

$$P' = P + K \times \left[\frac{1 + \cos\left(\frac{P_o - P}{R} \right) \Pi}{2} \right] \times (P - P_d) \times t$$

Equation 1.1: Vertex Interpolation Equation allowing us to perform deformations

Our implementation further introduces the concept of an **Emotion Database**. An emotion database is likened to the MPA or FACS system. It provides a mapping between high level expressions and the facial movements required to generate these expressions. Each avatar has the following entries in the database:

- the emotion/facial expression classification (e.g. a smile);
- the forces required to achieve the required expression (e.g. deform left cheek and deform right cheek);
- the deformation information (the forces to be applied and the length of time for each deformation).

This deformation algorithm inherently depends on the object's vertex density, with a higher density generating smoother deformations. Our object reconstruction implementation generates objects that are very conducive to this deformation approach.

4.5 Networking

The information that is transmitted across the network is:

- The pose the avatar is to assume;
- The expression the avatar must have.

When our system engages in a conferencing session, a network connection is established between the encoder and decoder. The decoder obtains the name of the avatar to be used during the virtual conference. If it does not have a copy of the avatar and its associated emotion database, the encoder sends it a copy of all this information.

The result of this process is the conversion of the video into a string and a number of integers. It affords an efficient compression of the video source.

4.6 Avatar management

The management of the avatar is the responsibility of the client. This core component is responsible for 3 actions, namely:

- translating and rotating the avatar to simulate the pose of the user;
- generation of expressions (expression replication);
- the management of expressions;

The first of these functions involves moving the avatar around in the virtual environment according to the data generated by the encoder's head tracking module.

The replication of facial expressions can be achieved using approaches such as Free Form Deformation. Escher *et al.* [2] make use of Dirichlet Free Form Deformation to generate expressions with their models. The process of expression management dictates that there must be methods in place to allow avatar expressions to be managed at the decoding end. If the decoder is instructed to deform the avatar to a smile and then to a frown, a protocol must exist to specify the process that is to be followed to achieve this transition. Incoming expressions can either be queued up and processed one at a time or if a more "real-time" effect is desired, the current expression being generated is interrupted (neutral expression is immediately regenerated) and the new expression generated.

4.7 Implementation in CoRgi

We mentioned in the introduction that we have made use of CoRgi to perform our sample implementation. We now outline how the implementation of the various components discussed in the previous section have been integrated into an encoding and a decoding system. This is illustrated in **Figure 3**.

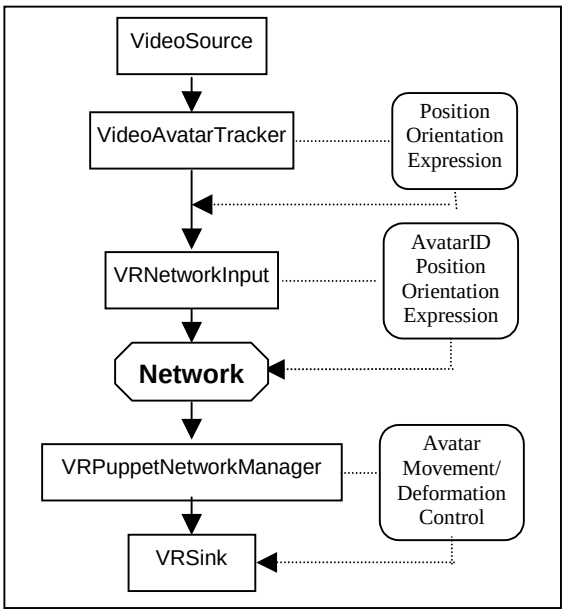

Figure 3: Component-based implementation overlaid on top of the CoRgi VR system. The beveled squares represent the data that is constructed as each components performs its task.

The encoder is responsible for collecting images from our CCD camera (VideoSource), performing expression analysis and head tracking (VideoAvatarTracker), and then preparing the data for network transmission (VRNetworkInput).

On the decoding end, the incoming network packet is analysed, and the appropriate expression deformations and Avatar translation/rotation is performed (VRPuppetNetworkManager). Finally the virtual scene is rendered (VRSink).

The implementation of the final system is almost complete. The only section still to be implemented is the Expression Analysis section of the VideoAvatarTracker component. The rest of the components have been implemented, with the basic functionality of these components in place. System integration is complete, with the interfaces for each of the components defined.

Issues relating to the usability of the system as a whole are not discussed. This will be the focus of later research.

5 Results

5.1 Model Acquisition

Figure 4 illustrates a reconstruction generated by the Model Acquisition algorithm. The reconstruction took place using (initially) a 100^3 voxel bounding volume. The resulting 3D model took 12 seconds to reconstruct from initialisation to generation of the final polyhedral representation and associated vertex colour representation. The subject was placed on a chair that could be rotated, and pictures were then taken of him at 30^o rotation intervals. The images were captured with a standard CCD camera and were then used as input for the reconstruction algorithm. The final reconstruction generated by the model acquisition implementation consists of 16386 vertices and 52456 triangular polygons.

The figure also reveals the limitations of our texture-mapping approach. The side of the head has been reconstructed quite well, while the front of the face is actually not very good. We suspect this is due to the concave nature of the front of the face. The implementation of concave support with this algorithm is imperative to ensure the removal of these problems.

One other observation to be made from the image is the shape of the resulting object; the reconstructed shape is quite good and actually looks like the person on the left. Additionally, perspective does not appear to have caused too many problems with this reconstruction. The result of our model acquisition implementation is a reconstruction method that is able to generate an approximation of a real-

Figure 4: Model Acquisition results. A vertex-coloured polyhedral avatar representation of the subject in above.

world object using a plain CCD camera. This compares favourably with traditional model acquisition systems such as Cyberware laser scanners, which tend to be prohibitively expensive.

Our reconstruction algorithm still requires a number of improvements. These problems include:

- no compensation for perspective in the source images; solutions include the use of Normalised Cross Correlation and Epipolar Geometry to perform 3D reconstruction. (Zhang *et al*,[17]);
- concave surface reconstruction; this applies when we attempt to reconstruct an object that is not equivalent to its visual hull. We are currently only able to able to provide a good reconstruction if that object being reconstructed is equivalent to its visual hull.

The inability of the present algorithm to handle non-equivalence cases implies that:

- the reconstruction has a "shrink-wrapped" look;
- our texture generation suffers; because of the "shrink-wrapped" problem, voxels that are not actually part of the real object cause the generated texture to include a noise factor.

Better reconstruction results can be achieved by looking at spline-based reconstruction methods. Here the voxels remaining after isosurface extraction are used as control points to generate a smooth B-Spline surface. An example of this approach is discussed by Forsey *et al*.[4]

5.2 Head Tracking

The images in **Figure 5** illustrate the robustness of the Normalised Cross Correlation approach to tracking and expression analysis. Our efforts are concentrated mainly around the mouth. As is illustrated in the images, the NCC algorithm is invariant to rotations, translations, and changing facial expressions (open mouths etc.) It must be mentioned though that although these images indicate promising results, the application that generated the images had to be recalibrated a number of times before the results presented here were achieved. This is due (as explained previously) to the sluggishness of our NCC implementation. In all situations where the algorithm failed, the person being tracked moved too quickly or changed expressions too rapidly, causing large inter-frame changes.

Tracking time for the 4 areas in each of the images was approximately 3 seconds per frame. A 4x4 search window was used in this case.

The current implementation is a very naïve one. It does not maintain a historical view of the tracked points. A fast implementation of the NCC algorithm (possibly in hardware) would imply that inter-frame changes were kept to a minimum. This would be a source of improved tracking results.

The sluggishness of the NCC algorithm also results in false matches being generated when a person's expression changes too much. This is simply because the inter-frame changes are too great.

While NCC offers a robust way of tracking a person's face, it is very slow, sometimes taking 1 second to return a single match. Despite this lack of performance, we feel that the benefits to be had from NCC vastly outweigh its performance penalty. The primary benefit of NCC is its ratio-based approach. Because it performs comparisons using intensity ratios, it can track points of interest on curved surfaces without the results being affected by the

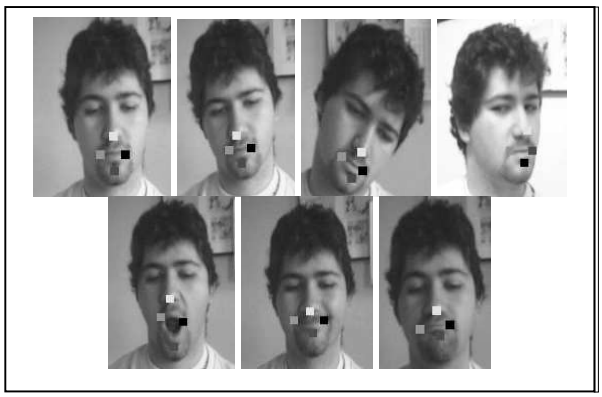

Figure 5: Head tracking performed using the NCC tracking algorithm. From top left to bottom right: neutral pose, roll left, roll right, yaw right, open mouth, smile, frown.

illumination changes due to surface curvature.

5.3 Facial Modelling

Figure 6 illustrates the resulting expressions generated using our deformation implementation. The resulting smile and frown are the most basic of expressions, but indicate the versatility of this deformation algorithm. The smile deformation was generated using four global deformations. The expressions generated were correctly identified by a number of individuals.

The implementation with CoRgi has shown that this algorithm has a negligible impact on the execution/rendering speeds of our graphics sub-systems, even with objects containing as many as 5000 vertices. This has been confirmed with the rendering performance measured for the expressions in **Figure 6**. The performance decreased from an average of 15.4 fps (frames per second) to approximately 14.6 fps during the generation of the expressions. This result was achieved on an SGI Octane with a dual MIPS R10000 processor configuration. The performance of the deformation implementation is a factor of the processor speed and not the graphics hardware.

5.4 Overall System Performance

We have implemented the major and most computationally intensive components of the system, namely that of head tracking and expression generation. The final expression analysis implementation will not add too great an overhead to performance, since it is designed to make use of the points tracked by the head-tracking module. The implemented components provide us with a relatively accurate indication of the performance of the complete implementation.

The time taken to transmit an identified expression and head tracking information to the decoding end for expression generation and avatar movement does not require real-time constraints. This is because the expression generation always trails the expression identification. The fact that deformation implementation (for expression generation) is also dependent on a time factor (t) means that an expression is generated independent of the time taken for the user to exhibit the relevant facial expression.

We have mentioned previously that the major performance penalty with our system is the NCC tracker. Its slowness also affects its robustness, with large inter-frame changes (due primarily to changing facial expressions) causing the system to fail during the tracking process. A faster NCC algorithm (possibly a hardware implementation) offers two benefits:

- reduced inter-frame changes, leading to more robust tracking;
- more points can be tracked, allowing for more expressions to be classified and generated.

With the current implementation, our performance sits at about 4 seconds to classify an expression and (as mentioned above) a rendering performance of approximately 15 fps for the expression generation.

An idea of the networking performance is illustrated by the following example: a 100x100 uncompressed 8 bit video frame, represents about 78Kb of data. Our "compression" method transforms this into about 25 bytes of data (5 bytes for the avatar id, about 16 bytes for position and orientation, and 4 bytes for the expression id).

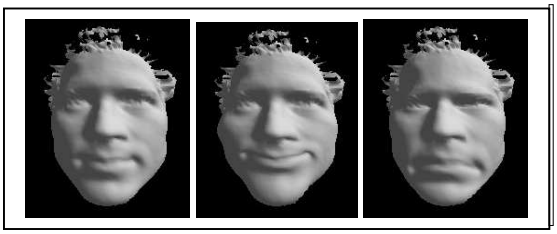

Figure 6: A deforming head. The first image indicates a neutral expression while the second shows a smile, The last image depicts the avatar frowning. These expressions have been generated using our Vertex Interpolation Deformation implementation. The head model depicted here contains 7013 vertices and 13338 polygons, all triangulated.

6 Conclusion

We have discussed the implementation of a framework for the development of a virtual-videoconferencing system using a low bandwidth network model and overlaid on top of a VR system. We have identified six main components required for the development of such a system, namely model acquisition, head tracking, expression analysis, avatar management and expression modelling using a low-bandwidth network model.

We have discussed our model acquisition implementation and mentioned the current shortcomings of this system, namely the lack of appropriate depth recovery in terms of concave surface reconstruction as well as the lack of perspective correction. Results indicate that these shortcomings do not really affect the generation of avatar heads for a virtual-videoconferencing system, bar texture generation. The major advantage our

acquisition implementation has over implementations using laser-range scanners is that it is cheap, only requiring the use of a general CCD camera.

We have described our implementation of an efficient, yet flexible deformation system that enables real time expression generation. The framework developed achieves our primary goal with this project, namely the development of a virtual videoconferencing framework for low-bandwidth conditions that is also acceptably cheap.

Many improvements are still to be made to the current implementation/framework development. We feel that the most pertinent developments are required in the following areas:

- Normalised Cross Correlation – because this algorithm forms an integral part of our system as a whole, the fastest possible implementation is required. We are investigating the use of Field Programmable Gate Arrays or DSPs for a possible hardware implementation;
- Expression Database – we want to expand the number of expressions;
- Expression analysis – complete the implementation.

References

[1] Eckman, P., Friesen, W. "Facial Action Coding System". Consulting Psychologists Press Inc., 577 College Avenue, Palo Alto, California, 1978.

[2] Escher, M., Magnenat-Thalmann, N. "Automatic Cloning and Real-Time Animation of a Human Face". MiraLab, University of Geneva, 1997. Available at www: http://miralabwww.unige.ch/ARTICLES/FACA97.htm

[3] Essa, I., Pentland A. "A Vision System for Observing and Extracting Facial Action Parameters". *IEEE CVPR 1994 Conference*, Seattle, Washington, 1994.

[4] Forsey, D., Wong, D. "Multiresolution Surface Reconstruction for Hierarchical B-splines", University of British Colombia. Available at www: http://www.cs.ubc.ca/spider/forsey/Approx/App_1.html

[5] Fourie, F. "Developing Efficient Graphics Rendering Components". Honours Thesis, Computer Science Department, Rhodes University, 1998.

[6] Gain, J. "Virtual Sculpting: An Investigation of Directly Manipulated Free-Form Deformation in a Virtual Environment". MSc Thesis, Department of Computer Science, Rhodes University, 1996.

[7] Hung, D., Huang, S. H. "Project Deidre (I) – Modeling Human Facial Expressions". Cornell Theory Center, December 1995. Available at www: http://www.tc.cornell.edu/Visualization/contrib/cs490-95to96/hjkim/deidre.html

[8] Koenen, R. "Coding of Moving Pictures and Audio: Overview of the MPEG-4 Standard" International Organisation For Standardisation, March 1999. Available at www: http://drogo.cselt.it/mpeg/standards/mpeg-4/mpeg4.htm

[9] Kutulakos, K. N., Seitz, S.M. "What Do N Photographs Tell Us About 3D Shape?" *Technical Report TR680*. Rochester University, January 1998. Available at www: http://www.cs.cmu.edu/~seitz/papers/tr680.pdf.

[10] Lee, W., Escher, M., Sennier,G., Thallman, N. "MPEG-4 Compatible Faces From Orthogonal Photographs". MiraLab, CUI, University of Geneva, 1998.

[11] Magnenat-Thalmann, N., Cazedevals, A., Thalmann, D. "Modelling Facial Communication Between an Animator and a Synthetic Actor in Real Time". *Proc. Modeling in Computer Graphics*, pp. 387-396.Geneva

[12] Panagou, S., Bangay, S. "An Investigation into the feasibility of Human Facial Modelling". *Proceedings of 1st South African Telecommunications*, Networks and Applications Conference, September 1998.

[13] POV-Team. "POV-Ray(tm) User's Documentation 3.0",1997. Available at www: http://mirror.det.mun.ca/doc/povray-manual/povray.htm.

[14] Reddi, S. "Using optical flow for the recovery of motion parameters and for gaze control". PhD Thesis, Birkbeck College, University of London, 1993.

[15] Roach, L., Jacobs, G. "3D Laser Scanning Speeds 3D Visualization of Existing Plants and Extends 3D Visualization into New CPI Applications". *Chemical Engineering*, June 1999.

[16] Seitz, S. M, Dyer, C. R. "Photorealistic Scene Reconstruction by Voxel Coloring", *Proc. Computer Vision and Pattern Recognition Conf.*, 1997, pp 1067-1073. Available at www: http://www.cs.cmu.edu/vcolor.html.

[17] Zhang, Z., Deriche, R., Faugeras, O., Luong, Q. "A Robust Technique for Matching Two Uncalibrated Images Through the Recovery of the Unknown Epipolar Geometry". *Rapport de recherche no 2273*, Inria Sophia Antipolis, Projet Robotvis, May 1994. Available at www: http://www.inria.fr/