Video-Compression With Virtual Reality
An implementation of a Cheap, Component-Based Virtual Videoconferencing System

Soteri Panagou, Dr. Shaun Bangay
{cssp|cssb}@cs.ru.ac.za
Multimedia Centre of Excellence, Computer Science Department, Rhodes University, Grahamstown, 6140, South Africa
http://www.cs.ru.ac.za/vrsig/

Abstract
We address the problem of virtual videoconferencing. The proposed solution is effected in terms of an implementation based on a Virtual Reality system. Our implementation is divided into a number of distinct components, an approach that illustrates the requirements from each of the components and at the same time constraining them in terms of clearly defined specifications. The different components that have been identified and implemented are: avatar or model acquisition, head tracking, expression analysis and expression replication/reconstruction. Part of the focus of this paper is also on how our implementation has led to a number of requirements from the VR system used during development. To this end, we discuss our implementation of a deformation component to assist our expression generation. The head-tracking component is responsible for controlling the pose of our avatar, while the expression analysis component is responsible for evaluating the user's expression. These 2 components are developed as data sources that generate avatar control information.

Categories: Compression, virtual-videoconferencing, object reconstruction, image processing, virtual reality, expression analysis, free-form deformation, low bandwidth networking.

1 INTRODUCTION AND MOTIVATION

To be able to transmit video streams successfully over a network demands 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 compact form is a very exciting one, and is a constant source of research. We essentially want to perform a mapping between video frames captured from a source such as a CCD camera, into data that can be used to control an avatar[1] in a virtual environment. Control here implies not only head movement and rotation, but also expression replication.

This paper presents a framework of such a system and sample implementation thereof. This implementation is overlaid on top of a Virtual Reality (VR) system that has been under development for some time at Rhodes University.

During the process of this discussion we will firstly look at the different components required for the development of a virtual-videoconferencing system in isolation. We then discuss the implementation of this system with the VR system (called CoRgi). The issues here have been to conform as much as possible to the CoRgi approach and where this has not been possible, extend the existing system to support our needs.

2 BACKGROUND AND RELATED WORK

Escher *et al.*[2] propose a framework for a complete virtual-videoconferencing system that uses a generic mesh of a face, which it then deforms and textures to suit the real face being modelled. The modelling of facial expressions is done in one of two ways. The first method involves image tracking using skin colour identification and edge detection. The second approach to expression generation is based on the evaluation of sound that is generated through speech. This process involves the evaluation of the input sound stream to determine the list of Oxford phonemes (and associated lengths) that constitutes the input sound stream. These phonemes are then used to essentially determine what the shape of the mouth should be for the current expression for the current expression. For details on this implementation see Thalmann *et al.*[7].

Although the system Escher *et al.*[2] describe provides real-time response and modelling, their parallel implementation using 4 SGI workstations, which leads to a very expensive system. Also, the effectiveness of the sound-processing component of the system with non-English speaking users is questionable. Additionally their model acquisition system is constrained to solve the problem of human head modelling. While this suffices for a virtual-videoconferencing system, the reconstruction system we envisage makes no assumptions regarding the structure of the final reconstructed object. The advantage their model acquisition has over our approach is that their use of a generic head means the final reconstruction leads to a relatively efficient polygonal representation.

The Motion Pictures Expert Group has ratified the final MPEG-4 specification, part of which involves the encoding of video sequences using VRML-type (Virtual Reality Markup Language) scene-graph specifications. Lee *et al.*[6] 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

[1] A 3D representation of a person in a virtual environment.

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.*[6] affords a brief discussion of these categories and their implementation for a system conforming to this part of the MPEG-4 specification.

As with MPEG-4's FAP/FDP approach to face animation, there exists much research in the way of facial expression generation. Much of this work is based on work done 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 as providing only an 'approximate' mapping because "some muscles give rise to more than one action unit" and go on to develop a model to enhance the FACS system.

3 CoRgi

This centre of excellence has been actively involved in research with respect to Virtual Reality. This research has spawned off the development of 2 VR systems, namely RhoVer and its current incarnation, CoRgi (suitably called the child of RhoVer). It is a second-generation VR system that has been designed to incorporate all the best current technologies and programming paradigms. The aim of CoRgi is to be a VR operating system, since its primary goal is to act as a platform for quick prototyping and component construction, with the aim of investigating different paradigms relating to Virtual Reality.

CoRgi flexibility arises out of its component-based nature, an approach that has been adopted by many researchers with respect to visualisation and image processing. Examples of this approach are the Visualisation Toolkit (VTK) (see Schroeder *et al.* [9] for a discussion on this system) that is essentially a visualisation package, as well as Khoros, a high-end image processing and modelling package. These systems adopt a data flow approach, whereby data is generated by one or more source components and modified by a number of components connected to these sources. The connections between the different components can be described as a **component chain**. The component chain identifies which components are connected to one another, as well as their connection order. For example, a video source could be connected to a video filtering component that is in turn connected to a video sink. The video source generates a stream of video that is then passed to the video filter that modifies each frame in some way. This data is then passed on to the sink that is responsible for eventually displaying the modified stream to the screen. The data is thus passed to each of these components in the order specified by this chain. This component chain essentially forms an "application".

Every component that is part of the CoRgi system is responsible for performing some specific operation, whether it is the generation of data as in the case of a source, the processing of this data, as in the case of a standard component, or the generation of some output by a sink.

4 FRAMEWORK and IMPLEMENTATION

Our framework can be decomposed into 5 major categories, as illustrated in **Figure 1**.

The server 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 server side because it represents a process that occurs only once when a new user is introduced to the system.

The client is responsible for the management of the avatar (called Avatar Management in the figure). This core component can be broken down into 3 more specific areas, namely: controlling the movement characteristics of the avatar (such as rotation and translation of the head), expression generation and finally, expression management.

These 5 categories comprise our basic system and are discussed in the next section. In addition to a description

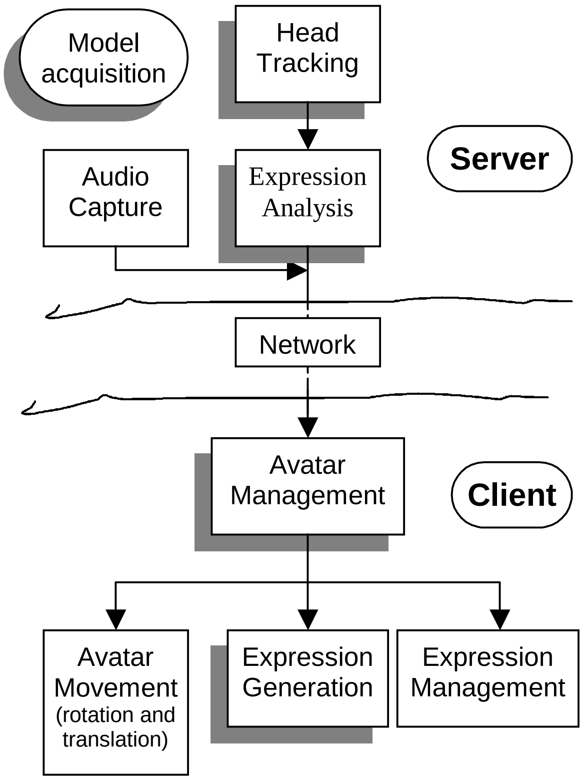

Figure 1: A conceptual design of our framework, based on the client-server networking model. The shaded components are the core in the design.

of each components, we also include a basic description of our networking component. Audio capture is important, but at the moment not regarded as an integral component of the framework, and so will not be discussed any further.

4.1 MODEL ACQUISITION

A general approach to model acquisition must exist. This includes the use of a Cyberware-type laser scanner and structured light. In addition to this though, the system must be general enough to allow for the use of any predefined 3D representation A system should thus allow for the use of a "Mickey Mouse" virtual presence if it is required.

We have developed and completely implemented the core modules listed below for our own model acquisition. This work builds on an earlier discussion in [8], and is based on a brute-force method similar to the marching cubes reconstruction method (Schroeder *et al.*[9]).

The first acquisition module is that of **shape reconstruction**. Our object is assumed to reside within a bounding volume of voxels[2]. Several pictures are taken from different angles and the backgrounds stripped (we classify background pixels as being pure white). We then proceed to project each of the images into the voxel space, disabling each voxel that a background pixel projects on to. Doing this for all the images leaves us with the voxels representing the object we are attempting to reconstruct. During the extraction process we keep track of the image (as well as the position in that image) each voxel is closest to. This information is later used to perform texture mapping.

The second module we have developed is an **isosurface extraction** component. We have implemented a basic algorithm for removal of those voxels that are not associated with the surface of our object. Applying this algorithm to the voxels remaining after the shape extraction algorithm results in a hollow shell, and a large decrease in object size.

The third component in our implementation is the **mesh connectivity** module. Once the isosurface extraction has been performed, we have to connect the remaining algorithm takes as input the voxels remaining from our Isosurface Extraction algorithm. It is based on the assumption that our voxel space is regular 3D and permits the generation of completely connected 3D meshes

Finally, to simulate the real-world object we are reconstructing, we have implemented a **texture generation** component. Texture generation with our algorithm is based on a relatively novel approach. Instead of attempting to generate a traditional texture map that we then apply to our reconstructed object via texture coordinates, we simulate this texture mapping with Gouraud shading. The result of the previous 3 modules is a 3D connected triangular mesh with RGB triplets at each of the vertices in the mesh. Using Gouraud shading we are thus able to colour each of the triangles and achieve a "textured" 3D model.

The generated 3D model along with its texture information conforms to the CoRgi specification and is thus compatible with the VR system.

Before continuing, it is necessary to discuss composition of a typical virtual scene. A basic virtual scene, called a **VREnvironment**, can contain a number of virtual "things". Each "thing" can have a **VisualRepresentation** associated with it. A VisualRepresentation can be thought of as CoRgi's internal polyhedral representation for virtual objects. Once a VisualRepresentation has been defined, we use the **SetPhysicalRepresentation** method to make the conceptual linkage between the "thing" and its "VisualRepresentation". Our virtual "thing" now has a virtual shape, which we can see and interact with, associated with it.

The result of our model acquisition implementation is a reconstruction method that is able to generate an approximation of a real-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. The reconstructed object is also compatible with the CoRgi system and can be used in virtual scenes, as is illustrated in **Figure 3**.

Our reconstruction algorithm requires a number of improvements before it can compete with the results generated by Cyberware scanners. These problems include that of perspective correction, as well as concave surface reconstruction. As the implementation currently stands, no attempt has been made to compensate for perspective, although potential solutions include camera calibration and the use of Normalised Cross Correlation and Epipolar Geometry to perform 3D reconstruction. (Zhang *et al*. in [10])

A more critical problem, a solution to which we are currently investigating, is to provide support for concave surfaces. The inability of the present algorithm to handle concave surfaces means that:

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

4.2 HEAD TRACKING

There must be some way of determining the pose and orientation of the user of the system. The most common approach to this is through the use electromagnetic trackers similar to the Polhemus Inc[3]. Trackers are unfortunately expensive and also not very common.

In their discussion on determining the epipolar geometry of a stereoscopic scene, Zhang *et al.* [10] introduce the notion of using Normalised Cross Correlation (NCC) as a robust method for determining pixel matches between stereoscopic image pairs. Given a pixel in a source image, NCC in its traditional form will perform a 2D search around that pixel position in the target image, and will return 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

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

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

NCC matching on stereoscopic image pairs to video sequences.

We address the problem of head tracking by simply tracking a number of features on a person' s face. Because of the video stream's temporal nature, we are also able to predict which are the good matches and which matches are bad, by evaluating the velocity of previously tracked points and searching the areas where a new match could be based on these evaluations first. NCC tends to be invariant to rotations of the subject' s face (provided that the rotation is not too great), and thus 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 allows us to calculate the following information:

a) **Rotation, Pitch and Yaw** (rotation in X-Y)- via triangulation of the points;
b) **Translation** (movement in X-Y); and lastly
c) **Scaling** (movement in Y-Z) - The distances between pixels are used as a measure of how far the subject is from the camera.

CoRgi, being a VR system, has standard components that are able to generate data that is then passed on to the rest of the system for evaluation and control. Our video-based head tracker thus fits the criteria of being a component that is able to take a video stream as input and generate positional information from a number of tracked points in the video stream. Our component is appropriately called **VideoAvatarTracker**.

4.2.1 PROBLEMS

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 thus be a source of improved tracking results.

The sluggishness of the NCC algorithm 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.

Lighting also plays quite an important role in the robustness of the NCC approach. It tends to generate better results under natural lighting than under fluorescent lighting conditions.

4.2.2 SUMMARY

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 illumination changes due to surface curvature.

With respect to our VR implementation, we now have a seamless way of converting video into a representation specifying avatar parameters such as rotation, scaling and translation. Our component thus acts as a VR data source for the rest of the pipeline.

4.3 EMOTION ANALYSIS / FACIAL MODELING

4.3.1 Expression analysis

Having evaluated the user's facial expression, there must be some way to classify it. The process of expression classification implies that there must exist some mapping from an expression to the simulation of that expression overlaid on our avatar.

The notion of table mappings was mentioned in section 2, with regards to FACS. The general approach followed is to make use of a lookup table to perform this mapping. 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.

We have mentioned above that we use NCC to track features on a person' s face. Our implementation assumes that 4 points are tracked. These are: nose tip, area half way between chin tip and lower lip middle, left edge of mouth, and right edge of mouth.

Our experiments in emotion analysis have been limited to two basic emotions/facial expressions, namely smiling and frowning. These expressions involve evaluation of the pixels tracked around the mouth of the subject. We monitor the 4 pixels around the mouth, and use thresholds to decide whether, for example, the subject is smiling or frowning.

4.3.2 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;
- the management of expressions;

The first of these functions is a relatively trivial process that will not be discussed further. It involves moving the avatar around in the virtual environment according to the data generated by the server's head tracking module.

The latter two areas are of greater interest to us. 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 on the client side. If the client 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. The client, for example, could receive the request for a smile, deform until maximal deformation has been achieved or another expression is requested, and then process the incoming requests.

Once we have correctly identified an expression (smile or frown) we then proceed to deform our 3D model to have the same expression as the real-world subject.

We have implemented a basic deformation algorithm that is based very closely on work done by Hung *et al.* [5]. We have incorporated work done by Gain [4] in our design as well.

Unlike traditional Free Form Deformation algorithms, that allow an object's topology to be altered by moving control points around, deformations with our implementation occur by moving vertices that are part of the polyhedral structure.

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

Equation 1.1: Vertex Interpolation Equation allowing us to perform deformations

Our implementation is based on **Equation 1.1**, an algorithm proposed by Hung *et al.* [5] 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;
- P_d = point of destination force;
- R = radius of influence; and finally
- 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 thus allows the deformation of a system to be controlled on three levels. The first method of control arises from the source/destination force combination. The control of the deformation using time provides us accurate control over the animation generated by the deformation. Finally, as mentioned above, the radius allows us control over how much of the object does actually get deformed.

Our implementation further introduces the notion of an **Emotion Database**. An emotion database is associated with each avatar we wish to use in our system. It contains the following information for each of the relevant avatars, namely:

- The emotion/facial expression classification (e.g. a smile);
- The deformation forces required to achieve the desired expression (e.g. deform left cheek filter and deform right cheek filter); multiple deformations are supported;
- The deformation information (such as combination of the forces to be applied and the length of each deformation *(t)*).

The emotion database can essentially be likened to the MPA or FACS system. It provides a mapping between high level expressions and the facial movements required to generate these expressions. A smile for example consists of 4 deformations, one just below each eye of our avatar, and one for each edge of the mouth outwards.

4.3.3 CoRgi-SPECIFIC EXTENSIONS

CoRgi as such has no way to allow for deformation of objects. This shortcoming has caused us to define a new class we call **DeformableVisualRepresentation** (DVR), which is derived from the VisualRepresentation base class. This new type of representation adds methods to allow objects of this class type to be deformed using the equation mentioned in the previous section. In addition to allowing for single deformations only, this class allows multiple deformations to be applied to a object at the same time. In order to facilitate multiple deformations occurring at the same time, we derive **AnimatedDefVisRep** (ADVR) from DVR. This new class offers us the ability to manage multiple deformations in terms of the emotion database. We are thus able to control the deformation of the avatar by simply specifying the expression we want.

Our 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.

4.4 NETWORKING

There should be an efficient method of remotely controlling an avatar. The minimum information that must be transmitted across the network is:

- The pose the avatar is to assume; this is the orientation of the head i.e. the rotation;
- The expression the avatar must assume.

The client-server architecture is the desired network model. An example of the duties that are required from the networking implementation is discussed below.

A network connection is established between the client and the server. The client obtains the name of the avatar to be used during the virtual conference. If the client does not have a copy of the avatar and its associated emotion database, the server will send it a copy of all this information.

The server is thus responsible for performing the head tracking and expression analysis. Once all the relevant information has been collected, a network packet is composed that has the following components:

- **AvatarID** - this is a unique avatar identifier;
- **Position** - <X,Y,Z> coordinate that allows for translations of the avatar on the client side;
- **Orientation** – specifies the rotation, pitch and yaw alterations to the avatar;
- **Expression** – An integer identifying the expression the avatar must have.

The client possesses an expression FIFO queue. The current expression is at the front of the queue. The avatar is deformed according to the current expression until it reaches a maximum, or another expression is received from the network. The client will then restore the avatar to the neutral expression and then remove that expression from the front of the queue (marking it as having been processed). It will then begin processing the next expression.

In order to automate expression management with networking support, we have developed a component called **VRPuppetNetworkManager** (VPNM) that is derived from ADVR. This component adds networking functionality to the standard ADVR and thereby allowing remote expression management to be achieved. The VPNM receives a packet from the network destined for itself, decodes it and depending on the expression it has to make to avatar assume, controls the deformation.

The result of our network implementation is that we are able to now convert each frame captured from a CCD camera into 6 integers (position, orientation and expression) and 5 characters for the AvatarID. The bandwidth requirements for this kind of network traffic are thus minimal.

5 RESULTS

5.1 MODEL ACQUISITION

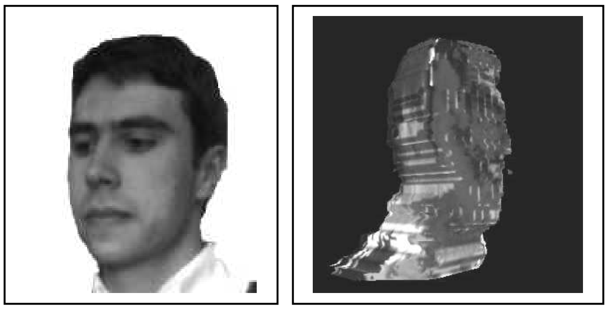

Figure 3: Reconstruction of the person on the left resulted in the reconstructed avatar (which we fondly refer to as Avatar Austin) on the right. The output on the right was captured from CoRgi.

Figure 3 illustrates a reconstruction generated by the Model Acquisition algorithm. The reconstruction took place using (initially) a 100^3 bounding volume. The resulting 3D model took 12 seconds to reconstruct, from initialization to generation of the final .OFF polyhedral representation and associated .COL texture/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 they 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 thus 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.

5.2 HEAD TRACKING AND EXPRESSION ANALYSIS

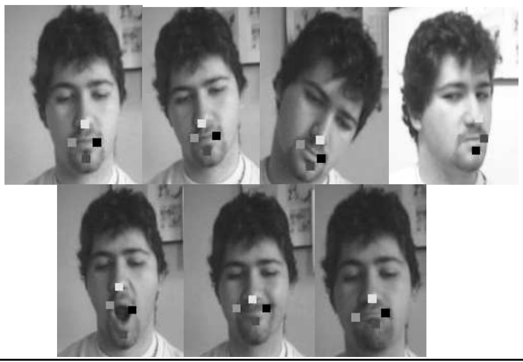

Figure 4: Head tracking performed using the NCC tracking algorithm. From top left to bottom right: neutral pose, pitch left, pitch right, rotate right, open mouth, smile, frown.

Images (a) – (g) in **Figure 4** illustrates the robustness of the Normalised Cross Correlation approach to tracking and expression analysis. Note that 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 primarily 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, thus causing large inter-frame changes.

5.3 FACIAL MODELING

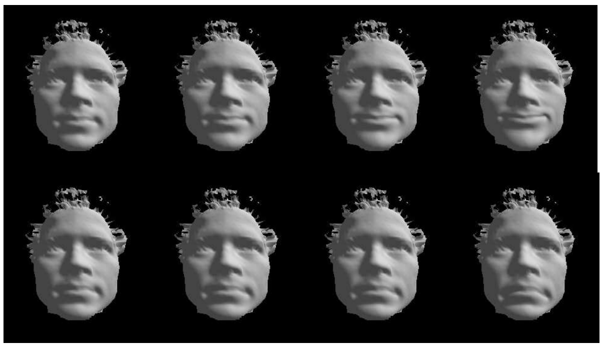

Figure 5: A deforming head. The top row of images illustrates a 3d model of a head smiling, while the bottom 4 images depict the generation of a frown. These expressions have been generated using our Vertex Interpolation Deformation implementation. The head model depicted here contains 7013 vertices and 13338 polygons, all triangulated.

Figure 5 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

5.4 CoRgi COMPONENT CHAIN

The final component chain for our virtual videoconferencing application is illustrated in **Figure 6**.

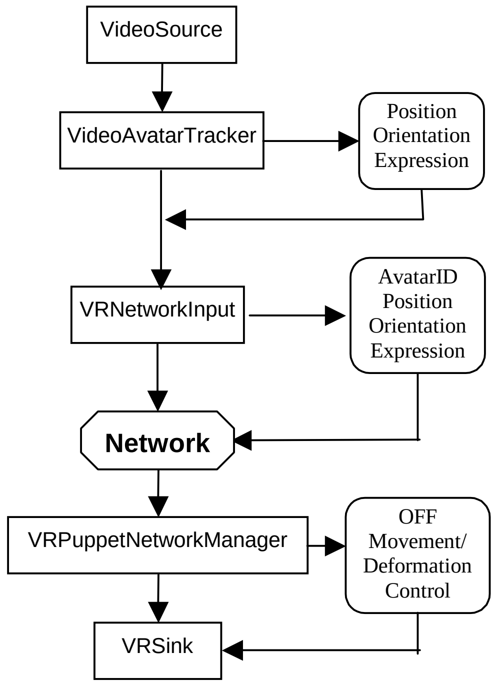

Figure 6: Conceptual system design for the implementation of a virtual videoconferencing system under CoRgi. The beveled squares are representative of the data as it is constructed by the component chain.

6 CONCLUSION AND FUTURE WORK

We have discussed the implementation of a basic framework for the development of a virtual-videoconferencing system, overlayed on top of a VR system using a low-bandwidth networking model. We identified 5 main components required for the development of such a system, namely model acquisition, head tracking, expression analysis and expression modelling.

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 allows for 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 using inexpensive equipment.

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 to an acceptable level.

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, June 1994.

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

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

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

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

[8] Panagou, S., Bangay, S. "An Investigation into the feasibility of Human Facial Modeling". Proceedings of 1st South African

Telecommunications, Networks and Applications Conference, September 1998.

[9] Schroeder, M., Martin, K., Lorensen, B. The Visualisation Toolkit: An Object Oriented Approach to 3D Graphics. pp 146-152, Prentice Hall, 1996.

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