Low Bandwidth Video Streaming using FACS, Facial Expression and Animation Techniques

Dinesh Kumar and Jito Vanualailai
School of Computing, Information and Mathematical Sciences, The University of the South Pacific, Suva, Fiji

Keywords: Facial Expression, Facial Animation, Low Bandwidth, Video Streaming, FACS.

Abstract: In this paper we describe an easy to use real-time 3D facial expression and animation system that takes the creation of individual facial expressions to the atomic level. That is instead of generating and recording known facial expressions we propose a mechanism that will allow us to create and store each *atomic facial distortion*. We can then combine some of these singular distortions to create meaningful expressions. FACS Action Units (AUs) is one such technique that describes the simplest visible movement, which cannot be decomposed into more basic ones. We use this as the basis for creating these atomic facial distortions. The Waters muscle based facial model has been used and extended to allow the user to calibrate and record each facial deformation as described in FACS AUs. The user can then create any facial expression by simply stating the series of AUs and its degree of activation in a controlled fashion. These features all form part of the Facial Animation System (FAS). Our FAS is implemented in such a way that enables it to be used as a low bandwidth video streaming player - a real time facial animation player driven only by FACS AUs transmitted as plain text over TCP sockets.

1 INTRODUCTION

In this paper we study the Facial Action Coding System (FACS) as described in (Ekman and Frieson, 1977) and its application to facial expression animation. We also investigate the area of facial animation by examining some common facial modeling and animation techniques and implementing a Facial Animation System (FAS). In FAS we demonstrate the application of the FACS standard on a 3D face model based on Waters muscle model (a popular technique in facial expression creation described in (Waters, 1987)). The animation of the face therefore is controlled by incoming stream of FACS Action Unit (AU) parameters.

This ties our system closely with the already available information about human facial behaviour given in FACS. The computational cost of facial animation has been solved with the advent of high processing power and advanced graphics rendering capabilities of modern computers. What is left therefore is the creation of a human face model and a novel way to animate the model from one state to another. At this point we realize that the first step in the animation process is creation of a synthetic face model. There are many commercial 3D modeling software that allows development of 3D face models for animation. However this process is rather tedious and very time consuming.

Next step is the creation of individual facial expressions. Each expression therefore describes a human emotion such as smile, anger and surprise etcetera. In many facial animation systems developed the user simply creates an expression by tweaking parameters that directly deforms parts of the synthetic face model. In other words basic human expressions are pre-programmed into the system. The resultant expression is normally validated through visual inspection. These expressions are then sent to animation engine. Therefore most facial animations systems lack a systematic way for creation of these facial expressions.

This situation motivated our research. We propose a 3D facial expression and animation system that takes the creation of individual facial expressions to the atomic level. That is instead of generating and recording known expressions we propose a mechanism that will allow us to create and store each atomic facial distortion. We can then combine some of these singular distortions to create

Kumar, D. and Vanualailai, J.
Low Bandwidth Video Streaming using FACS, Facial Expression and Animation Techniques.
DOI: 10.5220/0005718202240233
In *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2016) - Volume 1: GRAPP*, pages 226-235
ISBN: 978-989-758-175-5

meaningful expressions. We use FACS AUs as the basis for creating atomic facial distortions. In summary the key contributions made to the field in this paper are:

- the ability of FAS to create atomic facial distortions based on information about human facial behaviour described in FACS with ease.
- a technique to execute a set of AUs to generate an expression.
- a simple technique proposed to apply the degree of activation of AUs in expression creation.
- development of a simple animation engine for simulation.
- implementation of a low bandwidth video streaming system; a real-time facial animation player driven only by FACS AUs transmitted as plain text over TCP sockets.

It is therefore envisaged that our system can prove useful to researchers studying the relationship between the generating of facial expression using FACS AUs as the basis. We wish to demonstrate the feasibility of using the FACS paradigm for the animation of facial expressions hence we selected a subset of AUs and implemented their behaviour. In this research we also attempt to understand some of the principles that are used to model face expressions based on FACS and implement a basic animation system on these. This paper therefore deals with FACS and animation but does not cover FACS AU recognition.

The rest of the paper is organised as follows: In section 2 we review related work in the field of FACS driven facial expression and animation. Section 3 describes the architecture of our system and in section 4 we describe the components of our FAS and algorithms developed. In section 5, we present the experimental results followed by conclusion in section 6.

2 RELATED WORK

The human face is a complex architecture which is capable of creating a vast range of facial expressions on the fly. Not only are the same named expressions different from person to person, mimicking the same on a computer has remained a challenge to computer scientists for over several decades now. The primary focus had been to create models capable of rendering high quality and realistic facial expressions. Initial efforts in 3D facial modeling began with (Parke, 1972) who developed the first parameterized facial model. (Waters, 1987) and (Magnenat-Thalmann et al., 1988) follow soon by developing pseudo muscle based models. The technique of free form deformations were also used to create facial expressions for example in (Kalra et al., 1992). (Kahler et al., 2001) utilized geometry based muscle modeling for facial animation where as (Ostermann, 1998) used the MPEG-4 standard to animate synthetic faces.

It is to be noted that the state of the art in facial animation has advanced tremendously over the years. For instance there is increasing effort by scientists to perfect the facial model to produce quality and realistic expressions. Example (Tena et al., 2011) developed an interactive region-based linear 3D face model that effectively divided the facial model into regions and used user-given constraints (markers placed on the face model). When activated these markers affected only the region in which they were placed and produced better effects than using the same markers on a region less model. The practice and theory of blendshape models has been well discussed by (Lewis et al., 2014). (Weise et al., 2011) discusses real time performance based facial animation that enables the user to control the facial expression of a digital avatar in real-time. Similar facial performance based models were proposed by (Pauly, 2013) and (Bermano et al., 2013). On the other hand there is equal effort by scientists to produce facial animation from video or motion capture data. Techniques have been developed to detect facial expressions using motion detectors or sensors that are susceptible to movements and then animated on a computer facial model. One such implementation is described by (Sifakis et al., 2005) who were able to automatically determine muscle activations from motion capture marker data.

It is therefore evident that most of these researches were focused on perfecting models to generate expressions and the results validated by FACS. Instead in our research we propose a method, which is to use FACS to generate expressions rather than to just use it to validate expressions. Some research is evident in this area such as in (Alkawaz et al., 2015), (Wojdel and Rothkranz, 2005) and (Wojdel and Rothkranz, 2001). (Wojdel and Rothkranz, 2005) in their research developed a parametric performance based model where each parameter corresponded to one of the AUs as defined in FACS.

They also developed co-occurrence rules that described how different AUs influenced each other. We employ the co-occurrence rules developed by (Wojdel and Rothkranz, 2005) in our paper, but

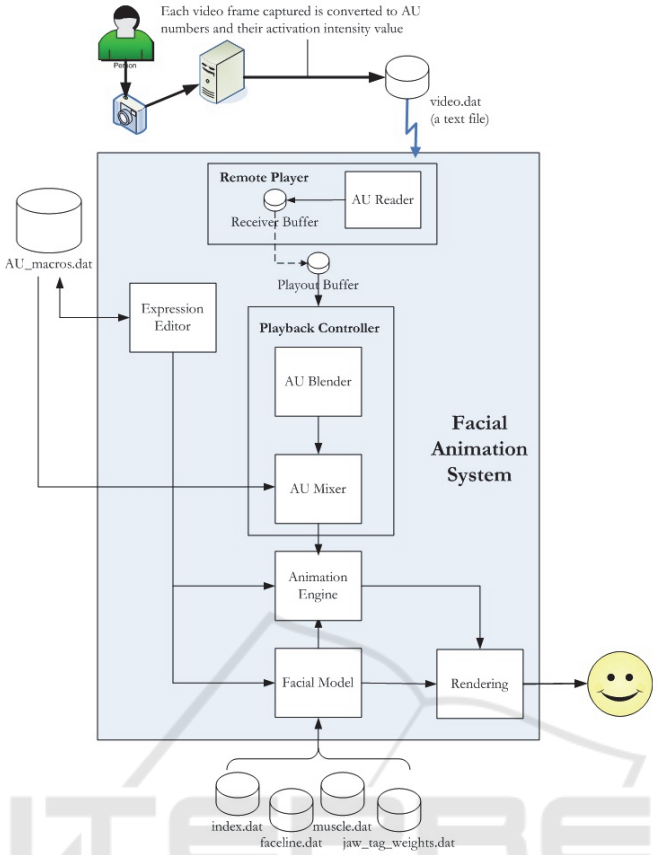

Figure 1: Architecture of our Facial Animation System (FAS).

instead we use these rules to generate expressions on Waters muscle model rather than on a parametric face model.

Furthermore, research in using computer generated face models to support low bandwidth virtual videoconferencing or video streaming is sparse. However, quite interestingly (Liu and Wang, 2010) proposed a low bandwidth teleconferencing system by combining techniques of facial muscle model, face detection and facial feature extraction to create life-like face animation. Using the symmetry property of the face, feature points were detected. Transformation values were then computed based on the position of the feature points which were then sent via a network to the receiver frame by frame. We note here that our proposed system sends AU numbers and their activation values rather than transformation values. Finally (Prakash and Balasubramanian, 2010) provide a good review of techniques developed in the area of facial modeling and animation while (Alkawaz et al., 2014) presents the future directions and challenges in the facial animation field.

3 SYSTEM OVERVIEW

Our Facial Animation System (FAS) was developed in three parts. The first part included work done on the facial model itself. Waters initial model was used and enhanced by giving it an easy to use graphical user interface (GUI) to control the functionalities of the program. Also greater control over rendering and animation is provided. The second part involved work done on the expression editor. Waters original linear muscle set used in the program is extended to include more muscles (in particular sphincter and sheet muscles) in order to create more interesting facial expression. The expression editor gives user the control over all muscles used on the face model including jaw rotation. Using this control the user can contract and relax muscles and can record expressions for each FACS AUs quite easily. The recordings for individual AUs are used by the *Remote Player* which is the third component developed. The remote player enables live video only streaming. It opens up a TCP socket and

receives streaming data from a remote source. This streaming data is made up of series of AUs defining individual expressions. A small animation engine is also designed within this player. The information about expressions (that is AUs) is processed according to the co-occurrence rules (if chosen) developed by (Wojdel and Rothkranz, 2005) and then played over time creating animation on our 3D face model. Figure 1 illustrates the architecture of FAS.

4 FACIAL ANIMATION SYSTEM (FAS)

This section describes the work done on our Facial Animation System (FAS) driven by FACS in three distinct categories:

1. Enhancements and improvements made to the original Waters program to provide a better user interface (UI) and greater control over rendering options and animation. This includes inclusion of sphincter and sheet muscles to control the facial geometry for expression creation.
2. Work done on our expression editor that allows the user control over all the muscles on the face to create an expression. It gives the user the flexibility to score expressions for individual FACS AUs and even record them. These expression recordings for individual AUs are used by the Remote Player.
3. The Remote Player itself. Once a TCP connection is established with the server transferring in sequence batches of AU numbers for expressions, the player runs these batches over time creating animation.

4.1 The Reference Face Model

The original Waters face geometry is retained and used in this project. The geometry comprises of 236 interconnected vertices forming 876 triangular polygons. In addition to the 18 existing linear muscles, our model introduces 2 sheet muscles and 3 sphincter muscles. The mathematics behind these pseudo-muscles are drawn from (Tanguy, 2001) and (Edge and Maddock, 2001) which is based on the original Waters model. We note that the original sphincter muscle used around the mouth area converges vertices towards the centre of the muscle. This makes it inappropriate to use the same implementation of the sphincter muscle around the eyes. To correct the problem we modified the sphincter muscle equation so that the vertices converge towards the semi-major axis of the ellipse defining the influence area around the eyes. This gives the model the ability to "squint" eyes. Figure 2 shows all the muscles implemented in FAS.

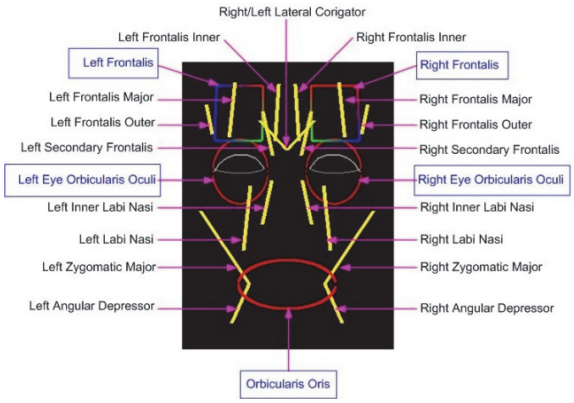

Figure 2: Muscles implemented in FAS.

4.2 Expression Editor

The Expression editor forms an integral part of our FAS. It gives the user control over all the muscles (including the jaw) used in our face model hence allowing a wide range of expressions to be generated with ease. Since our aim is to animate facial expression based on FACS AUs, it gives the flexibility to generate expressions that can approximate the various AUs defined in FACS. Using the expression editor the user can manipulate one or more muscles to match a particular AU expression. During this activity the calibration is rendered in the graphics window for the user to note the deformations on the face model. This allows the user to fine tune the expression. Once the desired expression is generated to a satisfactory level it can then be recorded with the AU number as an identifier. There is also the possibility to modify an existing AU expression if desired by the user. A total of 64 AU expressions can be defined.

By the definition of FACS, complex facial expressions are generated by applying one or more individual AUs. When these AU expressions are programmed and stored in FAS, it gives us the possibility to generate many other compound expressions simply by using combinations of AUs. The UI for our expression editor is fairly easy to use (Figure 3).

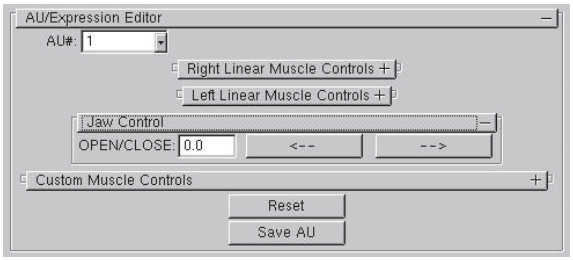

Figure 3: Expression Editor UI.

4.3 The Remote Player

Improving the Waters facial model by including the sphincter and sheet muscles and fixing the jaw movement has been the first major development task. This allowed creating more interesting expressions. Further the expression editor gives us a novel way to blend in FACS results with our modified facial model. This has been the second major task. However a system that only allows creation and recording of AU expressions will seem of not much use. It will be interesting to see hence how our system will behave if we present to it a series of random AUs to animate. Surely the end result is worth noting and whether it corresponds to a valid human expression.

This led to the implementation of a small low-bandwidth video streaming system, a real time facial animation player driven by FACS AUs transmitted as plain text over TCP sockets. The reasons for developing this are in twofold:

1. It will allow the testing for facial expression creation and animation by streaming AU numbers as plaintext over the internet instead of high-end media files.
2. Such a system would prove useful for example in live streaming or streaming recorded lecture presentations from across different university campuses. Instead of clogging bandwidth with high resolution video data to be transmitted, if AU numbers representing facial deformation of the person presenting the lecture can be indentified and sent, this would significantly reduce the usage of the bandwidth.

In general, multimedia content has a large volume, so media storage and transmission costs are still significant; even in today's computing world where bandwidths have increased significantly. To offset this somewhat, media are generally compressed for both storage and streaming. This however reduces the quality. Our system attempts to solve this problem by eliminating the need to send the actual video of the person delivering the lecture. Instead the user will be presented with a facial model on the client side which will deform according to AUs sent from the server side.

Developing media players capable of streaming live video over the internet are not trivial. There are three fundamental problem areas that have to be addressed.

- *Bandwidth* - If the sender transmits faster than the available bandwidth then congestion occurs, packets are lost, and there is a severe drop in video quality. If the sender transmits slower than the available bandwidth then the receiver produces sub-optimal video quality.
- *Delay Jitter* – This is a problem because the receiver must receive/decode/display frames at a constant rate, and any late frames resulting from the delay jitter can produce problems in the reconstructed video, e.g. jerks in the video.
- *Loss Rate* - Losses such as lost packets, bit errors or burst errors can have a very destructive effect on the reconstruction of the video on the receiver side.

It is also desirable that playback can begin while data is still being delivered. This requires the usage of an appropriate buffering algorithm and the use of an appropriate size buffer which is basically shared by two processes – one that fills up the buffer with video data to play and the other that actually plays it. However unlike conventional media players, our remote player is very much different in the sense that it is driven only by streaming AU numbers. This required custom algorithms to be designed for buffering and playback sequences.

The remote player system consists of an AU Reader, an AU Blender, an AU Mixer and a Playback Controller. All these components are part of FAS except the AU Reader that runs as a separate thread from the program. The first step is to provide all the parameters necessary to run the remote player efficiently as illustrated in Figure 4.

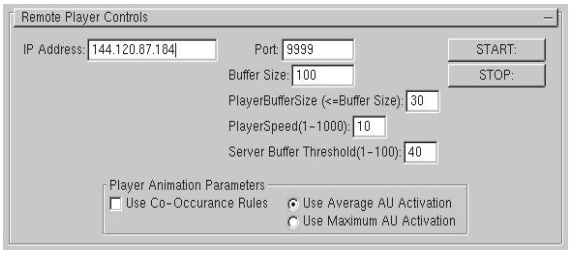

Figure 4: Remote Player user interface.

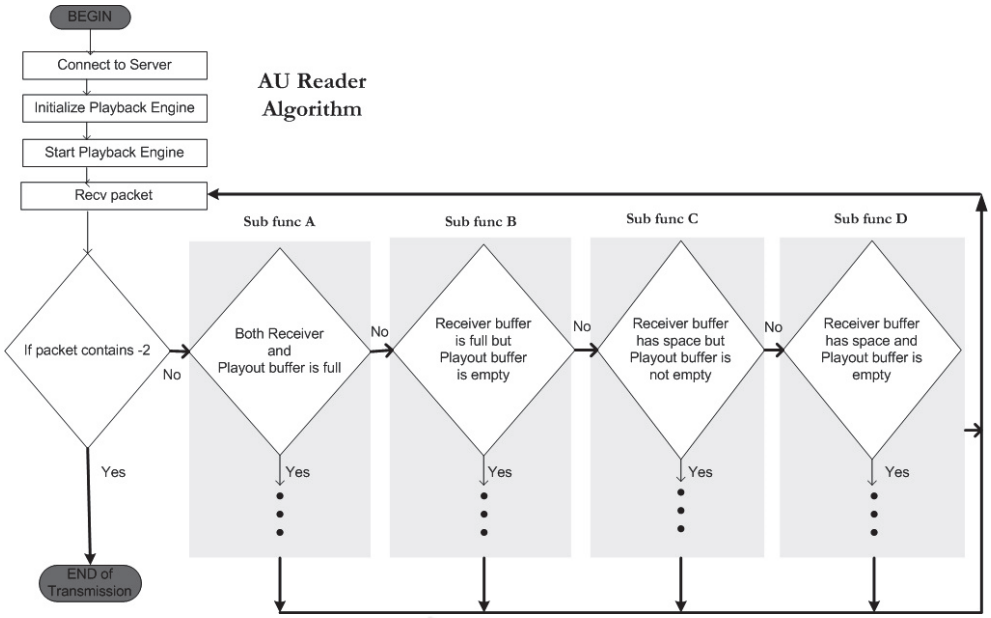

Figure 5: AU Reader Algorithm.

4.3.1 AU Reader

The AU Reader is a key component in the remote player. It is responsible for receiving the transmitted data from the server. In fact the functionality of the AU Reader is handled by a *thread* that is created as soon as the user initiates a connection. The use of a thread is desirable because our OpenGL based FAS process is already running in an endless loop drawing frames every few milliseconds. What we needed is a background process that will simply work alongside our FAS, receiving and recording AUs and feeding them to the playback controller. This leads to the obvious advantage that the parent FAS process is not overloaded to do the job of receiving AUs from the server which if it did would obviously create disruptions in rendering. Figure 5 describes the algorithm used in AU Reader.

AU Reader also accesses and manipulates the two buffers – *Receiver* and *Playout*. In order for the receiver and server to communicate with each other efficiently the following control constants are used:

1 – successful receipt of packet. (Clients signals server that packet received in good state, no loss). A 'packet' in our case is simply a series of AU numbers (a set) that corresponds to an expression. For example, for the expression - happiness, the 'packet' would simply contain the string (plaintext) {1, 6, 12, 14}, for anger {2, 4, 7, 9, 10, 20, 26} and {-2} for end of transmission etc.

- 1 – client signals server to pause transmission. (The likely cause of this would be the *Receiver* buffer has become full and the playback controller is slow in freeing up *Playout* buffer)
- 2 – server signals client for end of transmission.
- 3 – client signals server to resume transmission. (*Receiver* buffer now has space to accommodate more AUs)
- 4 – client signals server to stop transmission. (This is when the user decides to stop the streaming prematurely)

Since the *Playout* buffer is also accessed by the playback controller (a shared resource), to avoid any deadlock situation a condition variable is defined and used called `player_buffer_state`. This variable is set to zero (0) when the *Playout* buffer is or becomes empty and set to one (1) when the Playout buffer is not 100% empty. A value of one (1) also indicates to the AU Reader that the playback controller is processing the current set of AUs. The AU reader fills the *Playout* buffer when it finds the state of this condition variable is zero.

4.3.2 The Playback Controller

The Playback controller works on the concept of key frames. That is two key frames (*start/end*) representing two different expressions are determined. The in-between frames are then interpolated resulting in animation. The end frame becomes the start frame in the next sequence and so on. A control variable (`current_frame_state`) is used to keep track whether interpolating between

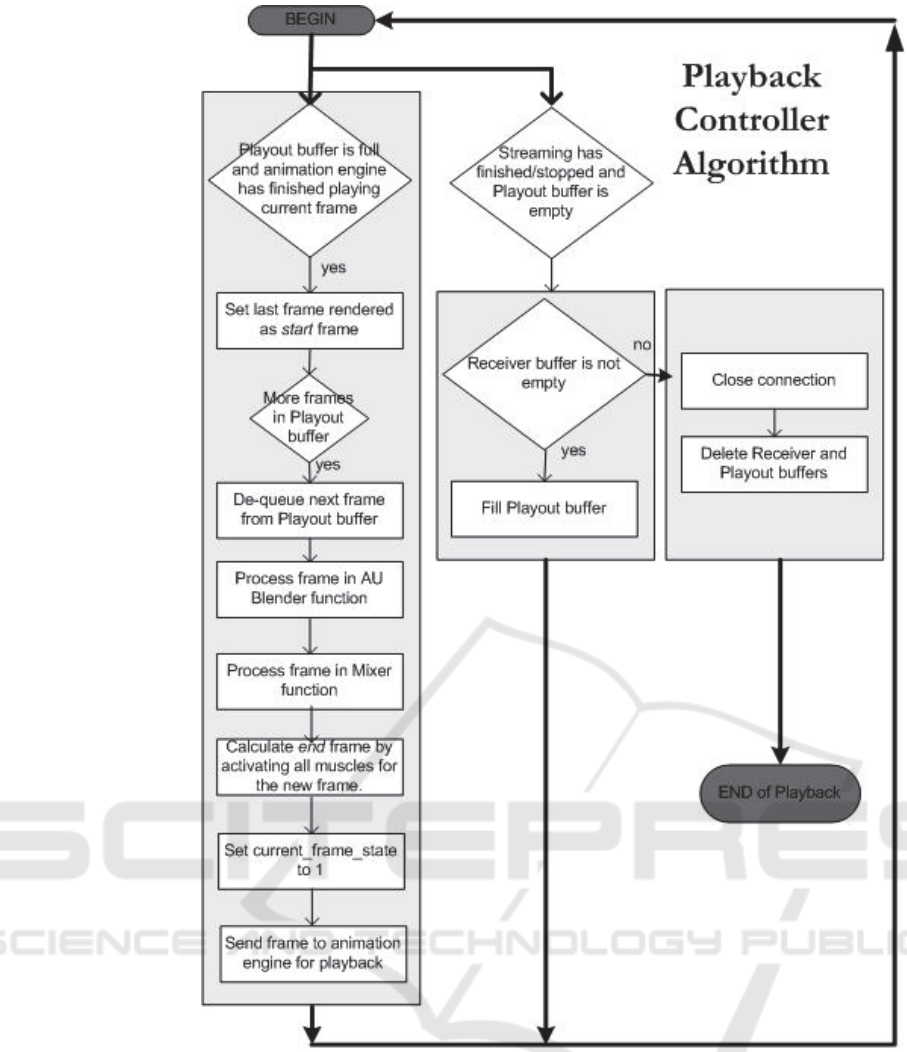

Figure 6: Playback Controller Algorithm.

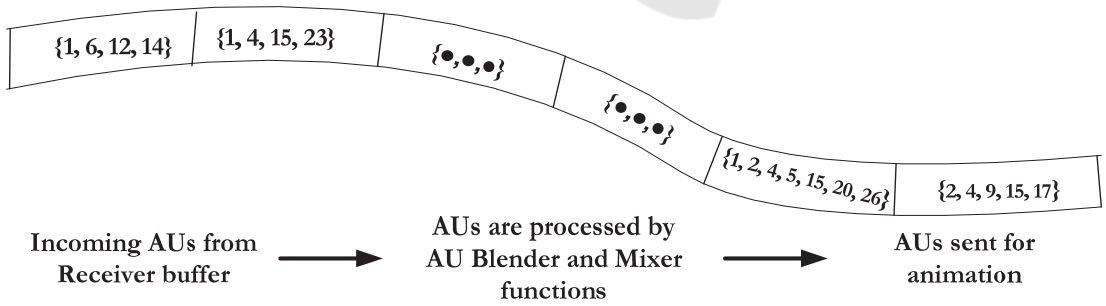

Figure 7: FIFO queue implementation of *Playout* buffer.

start and end frame has been completed.

It is set to zero (0) if the player has finished playing the end frame, one (1) otherwise. Each entry in the *Playout* buffer represents one key frame in the form of a set of AUs. Each set therefore describes one expression.

Once initiated by the AU Reader the playback controller function executes at a constant rate. Figure 6 illustrates the complete playback controller algorithm and the *Playout* buffer is implemented as first in first out (FIFO) queue structure as illustrated in Figure 7.

4.3.3 AU Blender

The function of the blender is to receive a set of AUs and apply the co-occurrence rules discussed in (Wojdel and Rothkranz, 2005) thus resulting in a refined set of AU which can then be animated. A total of 16 AU expressions are implemented in our system (Table 1). The reason for these choices of AUs is that they are used to produce the 6 basic expressions (happy, sad, anger, fear, surprise and disgust). These basic expressions are implemented in the original Water model. However they have been produced by manipulating the muscles of the face so that the desired expression can be achieved. Our system tries to arrive at the same expression but by applying individual expressions governed by recorded AUs in the program. The AU Blender therefore implements the same dependencies as provided by (Wojdel and Rothkranz, 2005) but only for the 16 AU given in Table 1.

Table 1: Action Units (AUs) implemented.

AU	FACS Name	Muscle Reference
1	Inner Brow Raiser	Frontalis
2	Outer Brow Raiser	Frontalis
4	Brow Lowerer	Corrugator supercilii, Depressor supercilii
5	Upper Lid Raiser	Levator palpebrae superioris
6	Cheek Raiser	Orbicularis oculi
7	Lid Tightener	Orbicularis oculi
9	Nose Wrinkler	Levator labii superioris alaeque nasi
10	Upper Lip Raiser	Levator labii superioris
12	Lip Corner Puller	Zygomaticus major
13	Cheek Puffer	Levator anguli oris
14	Dimpler	Buccinator
15	Lip Corner Depressor	Depressor anguli oris
16	Lower Lip Depressor	Depressor labii inferioris
17	Chin Raiser	Mentalis
20	Lip stretcher	Risorius
23	Lip Tightener	Orbicularis oris
26	Jaw Drop	Masseter

4.3.4 AU Mixer

Facial expressions in a real life rarely contain only one single AU activation (Wojdel and Rothkranz, 2005). This means meaningful facial expressions consists of activations of more than one AU. Each AU activation in our system is recorded as activation values for the 18 linear muscles, 3 sphincter, 2 sheet muscles and jaw rotation – a total of 24 values. Some of the values are none-zero if the corresponding muscle is activated and zero otherwise. In order to accumulate changes resulting from activation of multiple AUs to create an expression, we use the following mechanism.

Average – In this mixer the muscle activation values for each AU in the set of AUs to generate an expression is investigated in parallel. If the corresponding muscles are activated in the AUs, their values are added and averaged according to the number of AUs which have that muscle activated. Table 2 illustrates the average mixer functionality (only a subset of muscles is shown). It shows from this technique that muscles that are commonly used in many AUs, their activation values are averaged across the number of AUs.

Maximum – Here, activation values of common muscles across AUs are analyzed and the maximum activation value selected. The advantage of this mixer is that subtle changes would be overshadowed by higher muscle activation values giving maximum deformation. Table 3 shows the implementation of the maximum mixer.

Table 2: Average mixer technique used in FAS.

Muscle	Muscle Activation			Average Activation
	AUa	AUb	AUc	
Left Zygomatic Major	1.2	0.0	0.6	0.9
Left Angular Depressor	0.8	0.2	1.4	0.8
Left Frontalis Inner	0.0	0.0	0.9	0.9
…	…	…	…	…

Table 3: Maximum mixer technique used in FAS.

Muscle	Muscle Activation			Maximum Activation
	AUa	AUb	AUc	
Left Zygomatic Major	1.2	0.0	0.6	1.2
Left Angular Depressor	0.8	0.2	1.4	1.4
Left Frontalis Inner	0.0	0.0	0.9	0.9
…	…	…	…	…

5 TEST RESULTS

We test the Remote Player's ability to animate expressions based on a given set of AUs. We use the AU set defined for the 6 basic expressions in (Noh and Neumann, 1998). We also test the system

combining some of these basic expressions and evaluate their results. Table 4 describes the application of the co-occurrence rules as the transmitted AUs pass through the AU Blender and AU Mixer functions. Table 5 gives results of a simple test performed to measure the bandwidth usage of transferring single expressions, the duration of playback and the amount of time the remote player spent on receiving each expression. The data for this table was gathered by performing 5 executions cycles for each single expression.

Table 4: Application of co-occurrence rules for the six basic human expressions.

Expression	AUs Transmitted	Resultant list of AUs after applying co-occurrence rules
Surprise	1, 2, 5, 15, 16, 20, 26	2, 5, 11, 15, 26
Fear	1, 2, 4, 5, 15, 20, 26	2, 4, 5, 11,15, 26
Disgust	2, 4, 9, 15, 17	2, 4, 9, 11, 15, 17
Anger	2, 4, 7, 9, 10, 20, 26	2, 4, 9, 11, 15, 26
Happiness	1, 6, 12, 14	6, 11, 14, 15
Sadness	1, 4, 15, 23	4, 11, 15, 23

As an example from this analysis, it can be calculated that approximately 33 frames of "surprise" like expressions can be transmitted by using only 5% of the total bandwidth with the remaining 95% to be used for other purposes (including audio transfer). This shows video transferred only as a series of AU activations is clearly an effective way to minimize bandwidth usage and at the same time achieve animation of the desired expression

6 CONCLUSION

This research also led to study of other areas of research such as psychology (study of FACS) and anatomy (study of the human face muscles aspect – due to adoption of the Waters muscle model). Our main research objective was to show the feasibility of using the Facial Action Coding System (FACS) paradigm for the generation of facial expressions. The results obtained by our research are an affirmation that indeed FACS can be used as a basis for facial expression and animation.

We note that real muscles when contracted, contract with the skin rather than just displacing skin; have different and unequal volumes across the muscle shape that determines how much it can be contracted; and overlap each other where in many cases activation of one muscle triggers the activation of adjacent muscles. Hence future direction of this research could be development of real 3D muscle models governed by laws of fluid dynamics.

Table 5: Bandwidth usage for single expressions.

Bandwidth size: **100Mbps LAN** **Client Platform: System 1**
***Playout* Buffer Size:** **30** **CPU:** INTEL Core 2 Duo 2194 Mhz
***Receiver* Buffer Size:** **100** **RAM:** 1GB
Player Speed: **250ms** **GFX:** INTEL Q35 Express 384MB
Threshold: **40%** **AVERAGE FPS:** (1024x768) 469.6

Expression	Server Side		Receiver Side			Bandwidth used
	Bytes Transferred	Transmission send Time (average seconds)	Bytes Received	Transmission Recv time (average-seconds)	Playback duration (average-seconds)	
Surprise	19	0.0189	19	0.0149	1.0212	0.148%
Fear	18	0.0228	18	0.0150	1.0307	0.141%
Disgust	13	0.0155	13	0.0148	1.0213	0.102%
Anger	18	0.0149	18	0.0155	1.0305	0.014%
Happiness	11	0.0154	11	0.0154	1.0185	0.086%
Sadness	11	0.0214	11	0.0153	1.0152	0.086%

REFERENCES

Alkawaz, M. H., Mohamad, D., Basori, A. H. & Saba, T., 2015. Blend Shape Interpolation and FACS for Realistic Avatar. *3D Research (Springer Link),* 6(6).

Alkawaz, M. H., Mohamad, D., Rehman, A. & Basori, A. H., 2014. Facial Animations: Future Research Directions & Challenges. *3D Research (Springer Link),* 5(12).

Bermano, A. et al., 2013. Facial Performance Enhancement Using Dynamic Shape Space Analysis. *ACM Transactions On Graphics (ACM TOG).*

Edge, J. D. & Maddock, S., 2001. Expressive Visual Speech using Geometric Muscle Functions. *Proc. Eurographics UK,* pp. 11-18.

Ekman, P. & Frieson, W., 1977. Facial action coding system. *Consulting Psychologists Press.*

Kahler, K., Haber, J. & Seidel, H. P., 2001. Geometry-based muscle modeling for facial animation.. *In Proc. of Graphics Interface,,* p. 37–46.

Kalra, P., Mangili, A., Magnetat-Thalmann, N. & Thalmann, D., 1992. Simulation of facial muscle actions based on rational free form deformations.. *In Proc. of Eurographics,* pp. 59-69.

Lewis, J. P. et al., 2014. Practice and Theory of Blendshape Facial Models. *EUROGRAPHICS State of the Art Reports 2014.*

Liu, Y. & Wang, S., 2010. A virtual teleconferencing system based on face detection and 3D animation in a low-bandwidth environment. *International Journal of Imaging Systems and Technology,* 20(4), pp. 323-332.

Magnenat-Thalmann, N., Primeau, E. & Thalmann, D., 1988. Abstract muscle action procedures for human face animation. *Visual Computer,* 3(5), pp. 290-297.

Noh, J. & Neumann, U., 1998. A Survey of facial Modeling and Animation Techniques. *Technical Report, University of Southern California*.

Ostermann, J., 1998. Animation of synthetic faces in MPEG-4. *Computer Animation,* pp. 49-51.

Parke, F., 1972. Computer generated animation of faces. *ACM Annual Conference.*

Pauly, M., 2013. Realtime Performance-Based Facial Avatars for Immersive Gameplay. *Proceedings of the ACM SIGGRAPH Conference on Motion in Games 2013.*

Prakash, K. G. & Balasubramanian, 2010. Literature Review of Facial Modeling and Animation Techniques. *International Journal of Graphics and Multimedia,* 1(1), pp. 1-14.

Sifakis, E., Neverov, I. & Fedkiw, R., 2005. Automatic determination of facial muscle activations from sparse motion capture marker data. *ACM. Trans. on Graphics (SIGGRAPH).*

Tanguy, E., 2001. An Abstract Muscle Model for Three Dimensional Facial Animation. *Technical Report, University of Sheffield, UK*.

Tena, J. R., Torre, F. D. L. & Mathews, I., 2011. Interactive Region-Based Linear 3D Face Models. *SIGGRAPH.*

Waters, K., 1987. A Muscle Model for animating 3D facial expressions. *Computer Graphics (SIGGRAPH'87),* 21(4), pp. 17-24.

Weise, T., Bouaziz, S., Li, H. & Pauly, M., 2011. Realtime Performance-Based Facial Animation. *Transactions on Graphics (Proceedings SIGGRAPH 2011),* 30(4).

Wojdel, A. & Rothkranz, L. J. M., 2001. FACS Based Generating of Facial Expressions. *Proceedings of 7th annual conference of the Advanced School for Computing and Imaging, ASCI'01.*

Wojdel, A. & Rothkranz, L. J. M., 2005. Parametric generation of facial expressions based on FACS. *Computer Graphics Forum,* Volume 24, pp. 743-757.