Animating Human Faces Using Modified Waters Muscle Model.

Peter Drahoš*
Peter Kapec†

Faculty of Informatics and Information Technologies
Slovak University of Technolgy
Bratislava / Slovakia

Abstract

Facial animation plays an important role in face-to-face communication. With the power of current personal computers and their modern GPUs we are now capable of rendering high quality photo-realistic graphics. In this paper, we present an implementation of a muscle-based 3D animation method that is suitable for almost any face mesh. We also extend Waters' muscle model to support overlapping and muscle force normalization. In addition, we present techniques that significantly accelerate the animation while keeping memory requirements minimal.

Keywords: real-time animation, facial animation, muscle models, warping, optimization,

1 Introduction

Computer animation of living objects is still a challenge, mostly when the animation and visualization should be created and displayed in real-time. Modeling and animating human bodies is one of the most difficult tasks.

There are many applications of animated humans, ranging from avatars in 3D chat environments trough medicine and similar scientific fields, film and entertainment industry to education etc. Simulating virtual humans requires application of several techniques such as: the skeletal model, simulating virtual muscles and skin, eye motion, gestures and facial expression etc.

Animation of facial expressions has a crucial role in face-to-face communication[1]. Today's personal computers are already capable of rendering photo-realistic quality faces but the algorithms we use for the animation are far from perfect. To produce believable animated faces we not only need excellent rendering, the quality of the animation plays the same role here.

In this paper we describe methods for creation of such real-time animations in 2D and 3D. We describe a pseudo muscles method based on Waters'[2] models and extend these models to support interaction. In addition, a optimization method to increase animation performance is presented followed by a method to normalize forces used in these models.

We start by briefly describing the background of facial animation in Section 2. Section 3 briefly describes our effort to create facial animation using accelerated grid warping. In Section 4, we describe the present state of the muscle driven methods which seem to be more promising. In Sections 4.1 to 4.5 we focus on two most common problems that occur when muscle interaction is not used. These are followed by techniques which we subsequently developed to overcome them. Finally, in Section 5, we examine the influence of these techniques on the animation speed and quality.

2 Background

Facial expression animation can be done in 2D and 3D. Each is optimal for different applications and each has its specific drawbacks. 2D techniques are suitable for educational purposes and for simple telepresence[1]. 3D methods are more flexible, especially in 3D environments.

There are many approaches ranging from simple real-time animations to some sophisticated non real-time anatomical simulations. The current animation techniques can be divided into three categories: image manipulation, geometry manipulation and combined geometry with image manipulation.

First we introduce several 2D methods and compare them. Then we focus on the second 3D category which comprises several techniques including: muscle models, parameterization and interpolation.

One of the basic methods in 2D is image morphing. This method uses several different input photographs for different emotions. The animation is created by morphing these images between each other. The limitation of this method is the restriction to the

*drahos@webcom.sk
†pk99316@decef.elf.stuba.sk

number of input images from which the animation can be created. A similar method uses short video sequences in connection with phonemes[3] or short words. Phonemes are minimalistic sound and mimic units which when sequenced can synthesize speech. Joining of such video sequences[1] can create facial animation. The difficulty is to ensure smooth transitions between these video sequences. This can be done using other methods e.g. image morphing.

Other method for creating facial expressions animations uses image warping. The input for this method is a photograph. This image is mapped on 2D polygonal mesh. Transforming the position of several vertices of this mesh deforms the image as shown in Figure 1. The polygonal mesh can be uniform or of different polygon types. The main drawback of the uniform mesh is the inability to "open lips" without deforming the face unnaturally. Non-uniform mesh is suitable when the face is described by feature points.

An international standard MPEG-4[4] was created to standardize multimedia communication. Part of this standard describes modeling and animating virtual humans. MPEG-4 defines more then 68 feature points located on a face in order to provide reference for defining facial animation parameters. The underlying polygonal mesh for image warping can be defined using these standardized feature points.

To create facial expressions animation in 3D, several different methods can be used. The simplest technique is to map a dynamic texture onto a static 3D geometry model of a head[2]. This animation does not look very realistic, especially when the jawbone is not moving while speaking. More precise methods modify the geometry of the 3D model of the head. Probably the simplest real 3D approach is the interpolation technique. This technique stores models for each expression. Animation is achieved by linearly interpolating between these models. Morphing can be used to morph between several different 3D models with different face expressions. Methods which require several models for expressions often have high memory requirements. Having many models also limits the reusability of these methods. On the other hand parameterization techniques require only one model. This model has to be manually prepared and adjusted to be suitable for animation but is reusable. These models are prepared by hand which is very time consuming and difficult.

Facial expressions animation can be done by simulating face muscles. This approach was introduced by Waters[2] and is probably the most optimal choice for both fast, simple methods and for anatomical simulations.

Of course anatomical simulations require massive computations and they are generally not suitable for real-time animation. The simple muscle models, often called pseudo muscle models are generally very fast and their results are mostly satisfactory. This is because they directly deform the facial mesh and do not simulate the underlying anatomy as physically based muscle models do. Pseudo muscle models are often combined with image manipulation to produce realistic outputs but this characterizes them as combined animation techniques[5].

The muscle is basically a vector with features that describe the parameters of the muscle. All muscles have an area of influence and a formula to deform vertices in this area. Modifying these parameters contracts and deforms vertices on the face model.

The most common real-time animation method giving excellent outputs is a pseudo muscle model combined with an image manipulated bump-map applied upon the model[6].

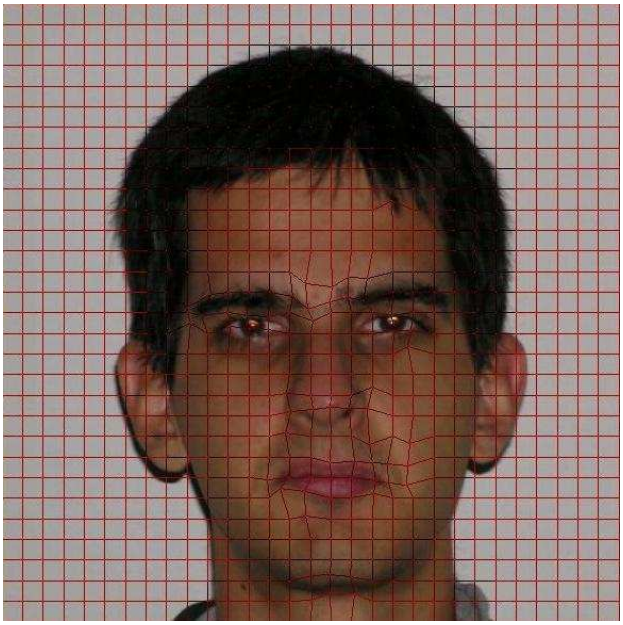

Figure 1: 2D warping with uniform grid mesh.

3 Accelerated warping

Our work is an attempt to improve visual capabilities of previous projects[7, 3] created at our faculty dealing with face animation.

We started with an application prototype which used OpenGL to accelerate 2D warping. Warping was done using a texture-mapped uniform polygon grid. We were interested in the quality of the images created using hardware accelerated grid transformations. The animation used key frame based system with linear interpolation of grid vertices. At first the quality of the output images and animation seemed suitable

[1] A nice feature of this method is the simultaneous voice output stored in these video sequences.

[2] Very common method used in games.

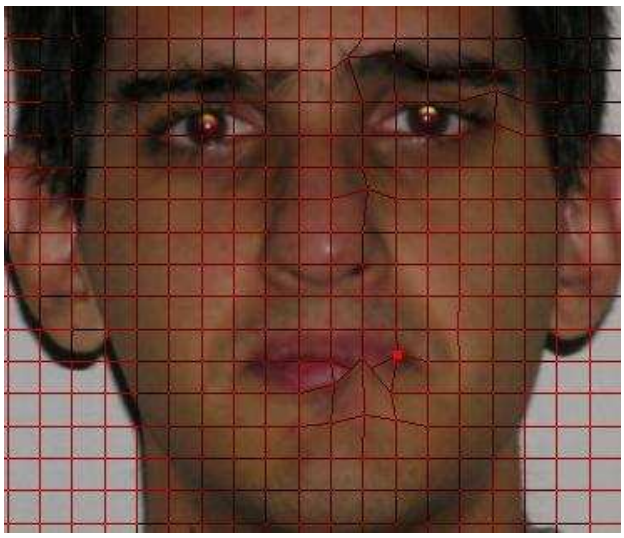

Figure 2: Effect of the cell overlaping problem described in Section 3.

for further work. The first problem we were unable to solve occurred when grid cells overlapped. This can be seen on Figure 2. The following experiments we did with this model proved the "open lips" problem with uniform grid warping which was mentioned in Section 2. Another drawback is that the grid has to be tailored specifically for every input image manually as faces often have different sizes and features. The screen of the prototype can be seen in Figures 1 and 2.

This method was abandoned because preparing animation by manually adjusting vertices of the underlying grid for every emotion was very time consuming and not reusable. We focused our efforts on 3D pseudo muscle models which are model independent.

4 The muscle model

Muscle driven animations use a set of virtual muscles. Models of these muscles are based on Waters'[2] muscle models. All these muscles define an area of influence and a deformation formula for all influenced vertices. Waters has defined three types of muscles: linear vector muscle, which is used for almost all face muscles, the sphincter[3] muscle and the sheet[4] muscle.

We focused on the linear vector model shown in Figure 3. Our goal is to extend this model to support overlapping. Overlapping and muscle combination was not described by Waters in his work[2] which focused on the mathematical models.

To understand why overlapping is important we have to first outline some effects that overlapping muscles cause. Particularly for muscles without support for overlapping. Two most common problems that occur with sequential animation of muscles will be described in the next Sections 4.1 and 4.2. Extensions to overcome these problems and to improve the model are presented in Sections 4.3 to 4.5.

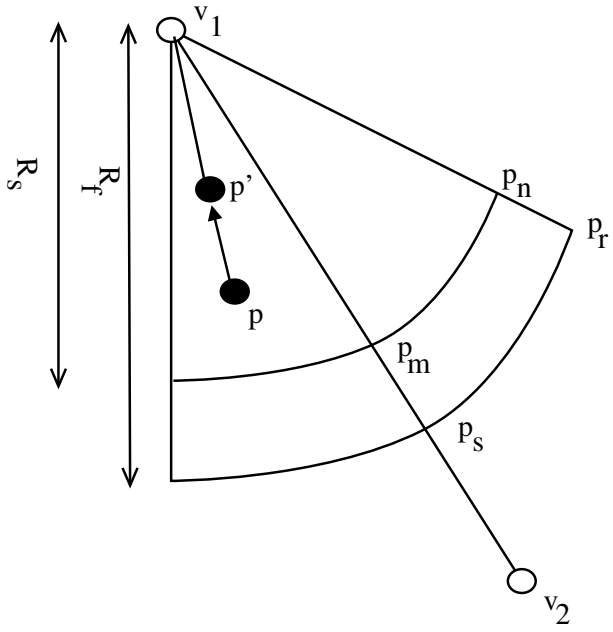

Figure 3: The linear muscle model

4.1 Ordering the sequences

This is the main problem with the Waters model when using sequential animation for each muscle independently.

Consider the situation in Figure 4. Here we have two overlapping muscles M_1,M_2 shown by their areas of influence and a vertex p. The two muscles create displacements d_1 and d_2 which affect the position of vertex p in sequence, creating p' and p''. The animation frame is then achieved by sequentially adding the influences of each muscle to the position of the vertex p[5] this process is described by the Algorithm 1. So after animating muscle M_1 the resulting vertex position of vertex p will be p' then after animating muscle M_2 the resulting position will be p''. This behavior is mostly correct, but what if the displacement d_1 is greater than shown in the figure or what if the point p is closer to the origin of muscle M_1? Then the point p' would be out of the influence of the muscle M_2! This behavior is unacceptable and produces jumpy tearing in the animation.

Most implementations solve this by arranging muscles so that this effect is minimal but they do not solve this problem. The result does not only depend

[3]Used for the Orbicularis Oris.

[4]Used for the Frontalis

[5]This is done for every vertex in the muscle influence area.

on the arrangement of the muscles but the order of animating the muscles plays a great factor.

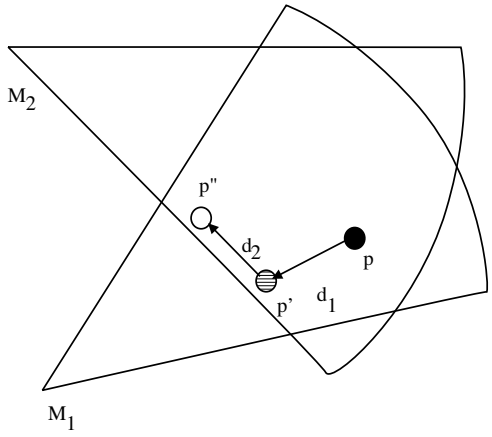

Figure 4: Overlapping with sequential animation.

4.2 Area of influence problem

As shown in the muscle model in Figure 3, only the vertexes inside the influence area are modified so there is no need to consider other vertexes. This is correct while no muscles overlap. If muscles overlap than it is possible for vertexes outside the influence area to be transformed into the area, so all vertexes must be considered for animation. This can be a huge performance drawback because while animating a muscle we have to test every vertexes position. If it is in the area of influence we transform it, else we ignore it as shown in Algorithm 1. That means when animating M muscles on a model with V vertexes we have to do MxV area tests.

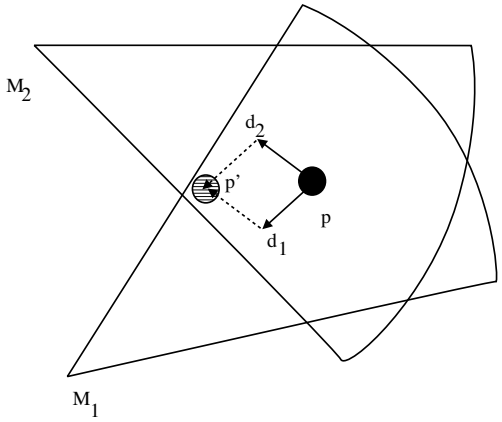

Figure 5: Overlapping using summed displacements.

4.3 Extended Waters' model

The problems described in Sections 4.1 and 4.2 can be simply solved by always transforming only the undeformed base mesh and by combining all displacements from all muscles into a single displacement for each vertex from the face. Then the order of muscle animation would not play any role on the resulting deformation as in Figure 5. This can be done due to the fact that all muscles deform only the base mesh and their displacements are summed[6]. This of course requires to modify the original formula of the Waters model[7].

The muscle is basically a vector from $v_1$to v_2. $R_s$and $R_f$represent fall-off radius start and finish respectively. The new vertex $p'$of an arbitrary vertex p located on the mesh within the segment $v_1p_rp_s$, along the vector (p, v_1), is computed as follows:

$$p' = p + \cos(\alpha)rk\frac{pv_1}{\|pv_1\|}$$

where α is the angle between the vector (v_1, v_2) and (v_1, p),D is$\|v_1 - p\|$,k is a fixed constant representing the elasticity of the skin, and r is the radial displacement parameter:

$$r = \begin{cases} \cos(1 - \frac{D}{R_s}) \\ \cos(\frac{D-R_s}{R_f-R_s}) \end{cases}$$

for p inside sector $(v_1p_np_m)$ and for p inside sector $(p_np_rp_sp_m)$.

To accumulate displacements we use the following computation instead:

$$d' = d + p + \cos(\alpha)rk\frac{pv_1}{\|pv_1\|}$$

where d is the displacement for the vertex p and $d'$is the new combined displacement. No other modifications are needed in the computation but a different approach has to be used to compute animation frames as shown in Algorithm 2. This approach produces outputs without visible tearing of unexpected behavior even for three or more overlapping muscles.

When adding up displacements some vertices may end up with extreme displacements which were added up from partial displacements of each muscle. This problem can be solved by simulating parallelism[5]. We did not notice any unnatural effects caused by such displacements in our experiments so we ignore[8] it.

[6]Sum is always the same for all permutations of its elements.

[7]We will only modify the linear vector muscle but other models are possible.

[8]Our goal is real-time animation after all.

Algorithm 1 Original sequential algorithm

```
// frame loop
while(! end ) {
        // reset all vertices to their default position
        ResetAll();
        // for all muscles
        for(int m=0;m<MAX_MUSCLES;m++)
                //for all vertices
                for(int v=0;v<MAX_VERTICES;v++) {
                        // v is in influence of m
                        if(influence(m,v))
                                // deform vertex v using muscle m
                                v=deform(m,v)
                }
        Render();
}
```

Algorithm 2 Overlapping muscles

```
// frame loop
while(! end ) {
        // reset displacement and position for all vertices
        ResetAll();
        // for all muscles
        for(int m=0;m<MAX_MUSCLES;m++)
                // for all vertices
                for(int v=0;v<MAX_VERTICES;v++) {
                        // v is in influence of m
                        if(influence(m,v))
                                // add displacement to v using muscle m
                                v.displacement+=deform(m,v);
                }
        // Apply displacement to position for all vertices
        ApplyAllDisplacements();
        Render();
}
```

Algorithm 3 Optimized animation with overlapping

```
// stores vertices in influence for each muscle
SetupMuscles();
// frame loop
while(! end ) {
        // reset displacement and position for all vertices
        ResetAll();
        // for all muscles
        for(int m=0;m<MAX_MUSCLES;m++)
                // for all associated vertices
                while (v=GetNextMuscleVertex(m))
                        // add displacement to v using muscle m
                        v.displacement+=deform(m,v)
        // Apply displacement to position for all vertices
        ApplyAllDisplacements();
        Render();
}
```

4.4 Optimizing speed

Now using the extended model and algorithm form the subsection 4.3 we have a set of muscles that always operate on the default mesh. This means no vertices can be transformed into the area of influence of any muscle as was described in Section 4.2. We are now able to pre-calculate the test of influence for all muscles and vertices before the actual animation. We will store a reference to every vertex inside the area of influence for each muscle, then while animating a muscle we only consider these vertexes for animation. The result is Algorithm 3 which if significantly faster than Algorithm 2 or 1, because no testing is done while animating a frame. The overall speedup is dependent on the alignment of the muscles to the face mesh.

4.5 Normalizing forces

All muscles are animated using the r argument in the muscle formula from Section 4.3. Setting this argument to different values generates different deformations of the face mesh.

In order to have expressions that can be applied to various face models we decided to normalize values of the r variable to the range $< 0,1 >$. We call the normalized r variable r_{norm}. We will also need a variable r_{max}which defines the upper limit of the variable r and is stored for every muscle in the face. Then before animating any muscle we can simply obtain r as $r = r_{norm}r_{max}$. Thus when $r_{norm} = 0$ the muscle generates no displacement and no visible mesh deformation[9]. When $r_{norm} = 1$ the generated displacement than produces deformation r_{max}. r_{max} must be set up by hand when creating muscle structure. Than we can store emotions with values ranging from 0 to 1 for each muscle. We also have to store r_{max} with the muscle setup which is bound to the given mesh.

The only drawback is that normalized expressions can only be applied to faces which have the same muscle structure modified for the given mesh. This is not a problem when these structures are created from templates that are only adjusted for the mesh geometry.

5 Results

We have significantly improved the animation speed by pre-calculating influences for all muscles. This requires more memory to store the pre-calculated references but removes all influence testing from the frame animation loop.

[9]These muscles can be ignored in the animation to speed it up a bit.

Our method is suitable for 3D meshes with small and moderate number of polygons. Increasing numbers of polygons may produce more detailed[10] results but the number of polygons plays a great factor in the animation speed and increasing polygon counts often prevents real-time animation.

Algorithm	FPS
1. Default	58.6
2. With overlapping	57.3
3.With overlapping and optimization	187.2

Table 1: Speed results using various algorithms.

By comparing all the algorithms in Table 1 we have noticed the significant speed improvement in the algorithm 3. The model used in the tests has about 15000 polygons and was rendered with nVidia FX 5600 video card on a Pentium 4 ,1.8 GHz.

The resulting application gave satisfactory results shown in Figures 6 and 7. Animation is done in real-time using interpolation of muscle forces between normalized emotions.

Normalization of emotions enabled us to share animation sequences between completely different models with the same muscle structure. It is even possible to combine different animations or to animate different areas independently. Fact is that normalized emotions are easy to create and their reusability is very high.

To emulate jaw movement we have simply created a linear muscle that pulls the lower part of the face down creating a similar effect. In the sample animation we use 18 muscles[11] which overlap mainly in the mouth area. More muscles can be added and removed even while the application is running. The model used in the application was not preprocessed or modified for the animation purpose.

6 Conclusions

In this paper, we have described a simple method for animating real-time facial expressions. We have developed a application to test the speed and quality of the results achieved with this method. Our method completely removes the problem of overlapping muscle interaction by using simple and fast displacement accumulation and enables us to pre-calculate muscle influences before animating. The modified method often takes only a fraction of the previous calculation time depending on the muscle arrangement.

By maintaining the simplicity of Waters' muscle model we have successfully created an usable expres-

[10]Especially when using muscles that create wrinkles or other skin effects.

[11]All are linear vector muscles.

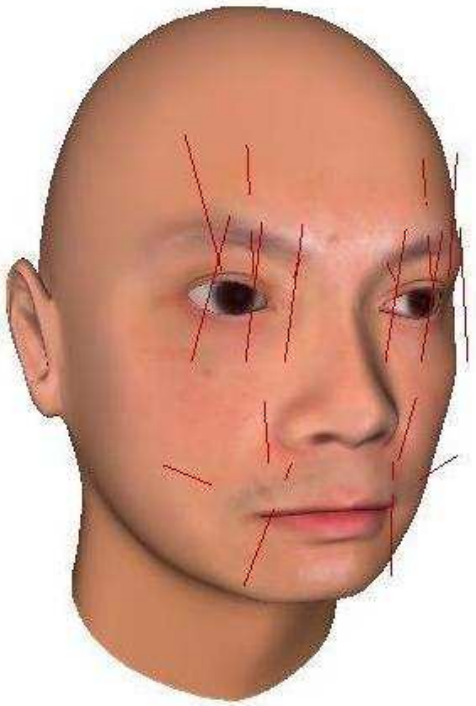

Figure 6: Default sample mesh with muscles.

sion animation method suitable for real-time face animations on a regular personal computer. Our approach is easy to apply to any other face with no need to modify its mesh.

There are still issues we would like to improve on our model in the future. First of all, simulating skin effects using bump-maps or muscles generating wrinkles would greatly increase realism. Secondly, we would like to try to implement the algorithm using OpenGL vertex shaders as this would give the method additional performance boost. Finally, we want to test the possibilities of using this method to simulate muscles not only on faces but in skeletal animation and other areas.

Figure 7: Sample expression

7 Acknowledgements

We would like to thank Martin Šperka for his guidance during the time we developed our application and for valuable help with this paper. Also many thanks to our friend Martin Kiselkov who is on the Figures 1 and 2.

References

[1] M. Šperka, "Telepresence and human body modelling," *Symposium Proceedings ISTEP*, pp. 281–285, 2000.

[2] K. Waters, "A muscle model for animating three-dimensional facial expressions," *Computer Graphics (SIGGRAPH'87)*, vol. 21, pp. 17–24, 1987.

[3] B. Pariša, "The talking head," *Master Thesis Project 2, Department of Computer Science and and Engeneering, Slovak University of Technology*, 2003.

[4] J. Ostermann, "Animation of synthetic faces in mpeg-4," *Computer Animation,*, pp. 49–51, 1998.

[5] J. Y. PeiHsuan Tu, IChen Lin, "Expression detail mapping for realistic facial animation," 2003.

[6] I. L. PeiHsuan Tu, "Expression detail mapping for realistic facial animation,"

[7] M. Horniak, "Program for the visual simulation of mimics when speaking," *Master Thesis. Department of Computer Science and and Engeneering, Slovak University of Technology*, 1990.