FEATURE POINT BASED MESH DEFORMATION APPLIED TO MPEG-4 FACIAL ANIMATION

Sumedha Kshirsagar, Stephane Garchery, Nadia Magnenat-Thalmann
MIRALab, CUI, University of Geneva

Key words: mesh deformation, real-time facial animation, performance driven animation, optical tracking

Abstract: Robustness and speed are primary considerations when developing deformation methodologies for animatable mesh objects. The goal of this paper is to present such a robust and fast geometric mesh deformation algorithm. The algorithm is feature points based *i.e.* it can be applied to enable the animation of various mesh objects defined by the placement of their feature points. As a specific application, we describe the use of the algorithm for MPEG-4 facial mesh deformation and animation. The MPEG-4 face object is characterized by the Face Definition Parameters (FDP), which are defined by the locations of the key feature points on the face. The MPEG-4 compatible facial animation system developed using this algorithm can be effectively used for real time applications. We extract MPEG-4 Facial Animation Parameters (FAP) using an optical tracking system and apply the results to several synthetic facial mesh objects to assess the results of the deformation algorithm.

1. INTRODUCTION

In this paper, we present a robust, fast, and simple geometric mesh deformation algorithm. A geometric mesh can be characterized by the locations of key feature points. Further, the animation of the mesh can be defined by the displacements of these feature points. The algorithm described here can be applied for animation of such meshes. As a specific application, we describe the use of the algorithm for MPEG-4 facial mesh, which is characterized by the Face Definition Parameters (FDP). We examine the results of the mesh deformation applied to facial animation by using the Facial Animation Parameters (FAP) obtained from an optical tracking system used for facial feature capture.

There are a variety of ways possible to represent animatable objects geometrically. The choice depends on the considerations such as precise shape, effective animation and efficient rendering. Barr introduced geometric modeling deformations using abstract data manipulation operators creating a useful sculpting metaphor [1]. Bearle applied surface patch descriptions to model smooth character form [2]. Free Form Deformation (FFD) and its variants have been used extensively for a variety of modeling and animation applications [3][4][9][13]. They involve the definition and deformation of a lattice of control points. An object embedded within the lattice is then deformed by defining a mapping from the lattice to the object.

FFDs allow volume deformation using control points while keeping the surface continuity. They provide the sculptural flexibility of deformations. FFDs have been successfully used for synthetic objects like face [6] and hand deformation [11]. FFDs have some limitations though. The locations of the control points are not very well controllable with respect to the actual mesh object. Also, the discontinuities or holes in the mesh are difficult to handle as a general case. Recently, Singh *et. al.*[15], proposed a new approach of using *wire* curves to define an object and for shaping its deformation. They illustrated the applications of animating figures with flexible articulations, modeling wrinkled surfaces and stitching geometry together.

In order to define shape and animation of a geometric mesh object, we concentrate on the use of feature points. We assume that the shape of the object is defined by the locations of the predefined feature points on the surface of the mesh. Further, the deformation of the mesh can be completely defined by the movements of these feature points (alternatively referred as control points) from their neutral positions either in absolute or in normalized units. This method of definition and animation provides a concise and efficient way of representing an object. Since the control points lie on the geometric surface, their locations are predictable, unlike in FFD.

2. GEOMETRIC MESH DEFINITION AND DEFORMATION

In this section, we describe in detail the feature point based mesh deformation algorithm. The algorithm is usable on any generic surface mesh. To begin with, the feature points or the control points with movement constraint are defined for a given mesh. A constraint in a direction indicates the behaviour of the control point in that direction. For example, if a control point is constrained along the *x* axis, but not along the *y* and *z* axes, means that it still acts as an ordinary vertex of the mesh along the *y* and *z* axes. Its movement along these axes will be controlled by the other control points in the vicinity.

Given a geometric mesh with control point locations, we need to compute the regions influenced by each of the control points. In order to get realistic looking deformation and animation, it is necessary that the mesh has a good definition of the feature points; *i.e.* the control point locations should be defined considering the animation properties and real-life topology of the object under consideration. Each vertex of the mesh should be controlled by not only the nearest feature point, but other feature points in the vicinity, in order to avoid patchy animation. The number of feature points influencing a vertex and the factor by which each feature point influences the movement of this vertex is decided by the following:

- The distances between the feature points *i.e.* if the feature points are spread densely or sparsely on the mesh
- The distances between the ordinary (non-feature point) vertices of the mesh and the nearest feature point
- The relative spread of the feature points around a given vertex

The algorithm is divided into two steps. In the *Initialization* step, the above mentioned information is extracted and the coefficients or *weights* for each of the vertices corresponding to the nearest feature points are calculated. The distance between two points is computed as the sum of the edge lengths encountered while traversing from one point to the other. We call this *surface distance*. This *surface distance* measure is useful to handle holes and discontinuities in the mesh, *e.g.* mouth and eye openings in the facial mesh models. The *Deformation* step actually takes place during the real-time animation for each frame.

2.1 Initialization

The initialization can further be divided into two substeps.

2.1.1 Computing Feature Point Distribution

In this step, the information about all the neighbouring feature points for each of the feature point is extracted. The mesh is traversed starting from each feature point, advancing only one step in all the possible directions at a time, thus growing a mesh region for each feature point, called *feature point region*. Neighbouring feature points are those feature points that have a common *feature point region* boundary. As a result, for each feature point defined on the mesh surface, we get a list of the neighbouring feature points with *surface distances* between them. This information is further used in the next step.

2.1.2 Computing Weights

The goal of this step is to extract possible overlapping influence regions for each feature point and to compute the corresponding weight for deformation for all the vertices in this influence region. Consider a general surface mesh as shown in Figure 1. During the process of mesh traversal starting from the feature points, assume that the vertex *P* is approached from a feature point FP_1. FP_1 is added to the list of the influencing feature points of *P*.

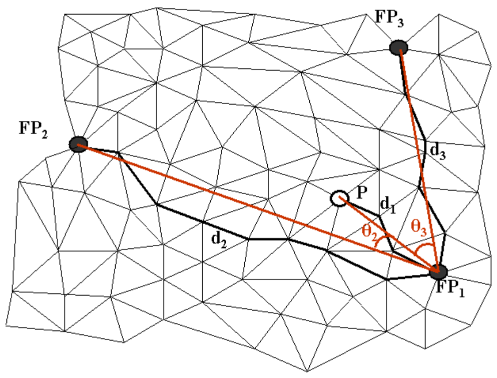

Figure 1. Computing weights for animation

From the information extracted in the previous step of mesh traversal, FP_2, and FP_3 are the neighbouring feature points of FP_1. FP_2 and FP_3 are chosen such that the angles q_2 and q_3 are the smallest of all the angles q_i for neighbouring feature points FP_i of FP_1. Also,

$$q_2 < \frac{p}{2}, q_3 < \frac{p}{2} \qquad (1)$$

The surface distances of the vertex from these feature points are respectively d_{1P}, d_{12} and d_{13} as shown in the figure. While computing the weight of FP_1 at *P*, we consider the effect of the presence of the other neighbouring feature points namely FP_2 and FP_3 at *P*. For this, we compute the following weighted sum *d*:

$$d = \frac{d_{12}\cos q_2 + d_{13}\cos q_3}{\cos q_2 + \cos q_3} \qquad (2)$$

Thus, *d* is the weighted sum of the distances d_{12} and d_{13}. The feature point in a smaller angular distance from the FP_1 is assigned a higher value of weight. If there is only one neighbouring feature point of FP_1 such that $q_2<\pi/2$, then *d* is simply computed as $d_{12}/\cos q_2$.

We compute the weight assigned to the point *P* for the deformation due to movement of FP_1 as:

$$W_{1,P} = \sin\left(\frac{p}{2}\left(1-\frac{d_{1P}}{d}\right)\right) \qquad (3)$$

or more generally

$$W_{i,P} = \sin\left(\frac{p}{2}\left(1-\frac{d_{iP}}{d}\right)\right) \qquad (4)$$

Thus, point *P* has a weight for displacement that is inversely proportional to its distance from the nearest feature point FP_1. This determines the local influence of the feature point on the vertices of the mesh. At the same time, nearer the other feature points (FP_2 and FP_3 in this case) to FP_1, less is this weight according to the equation 2 and 3. This determines the global influence of a feature point on the surrounding region, in the presence of other feature points in the vicinity.

It is possible that a vertex is approached by more than one feature point, during the process of mesh traversal. We compute the weight for this feature point following the same procedure, as long as the angular distance criterion (1) is satisfied, and the *surface distance* $d_{iP}<d$, *d* as defined in equation 2. This second criterion ensures that the feature points FP_j whose nearest neighbours are nearer to the vertex *P* than FP_j are not considered while computing the deformation for vertex *P*. Thus, for the example taken here, weights will be computed for vertex *P* for the feature points FP_1 as well as FP_2 and FP_3, provided d_{2P} and d_{3P} are less than *d*. As a result, we have for each vertex of the mesh, a list of control points influencing it and an associated weight.

We tested the algorithm on simple meshes with different values of limits in equation 1, and different weighting functions in equation 2 and 3. The ones giving the most satisfactory results were chosen. In equation 3, we chose *sine* function as it is continuous at the minimum and maximum limits.

2.2 Deformation

Once the weights for the vertices have been computed, the mesh is ready for real-time animation. Note that *Initialization* step is computationally intensive, but carried out only once. The weights computed, take into consideration the distance of a vertex from the feature point and relative spread of the feature points around the vertex. Now, from the displacements of the feature points for animation, we calculate the actual displacement of all the vertices of the mesh. Here, we have to consider the effects caused when two or more feature points move at the same time, influencing the same vertex. We calculate the weighted sum of all the displacements caused at the point P due to all the neighbouring feature points. Let FP_i, $i=1,2,\dots,N$ be the control points influencing vertex P of the mesh. Then

1. D_i = the displacement specified for the control point FP_i
2. $W_{i,P}$ = the weight as calculated in the *Initialization* for vertex P associated with the control points
3. $d_{i,P}$ = the corresponding distance between P and FP_i.

The following equation gives the resultant displacement D_P caused at the vertex P

$$D_P = \frac{\sum_{i=0}^{N} \frac{W_{i,P} D_i}{d_{i,P}^2}}{\sum_{i=0}^{N} \frac{W_{i,P}}{d_{i,P}^2}} \qquad (5)$$

This operation is performed for every frame during the computation of the animation of the mesh.

3. ADAPTATION FOR MPEG-4 FACIAL MESH

Various muscle based models have been effectively developed for facial animation [13][16][17]. The Facial Action Coding System (Friesen, 1978) defines high level parameters for facial animation, on which several other systems are based. We use MPEG-4 facial animation standard, which defines the face object by locations of specific feature points on the facial mesh. Lavagetto *et al* have described an MPEG-4 compatible facial animation engine using a similar mesh deformation technique [7]. However, the important difference is that the wireframe semantics (the locations and the region influenced by all the feature points) have to be specified *a priori* in their method. MPEG-4 Facial Animation

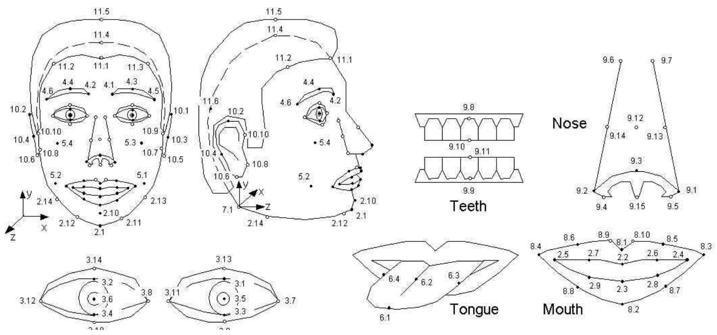

Figure 2. MPEG-4 Facial feature points

The ISO/IEC JTC1/SC29/WG11 (Moving Pictures Expert Group - MPEG) has formulated the new MPEG-4 standard. An efficient coding method has been devised within the framework of the standard for graphics models and their animation parameters specific to the model type. For face models, the Face Definition Parameters (FDPs) are defined by the locations of the feature points (*e.g.* mouth corners, eye corners, eyebrow ends *etc.*) and are used to customize a given face model to a particular face. The Facial Animation Parameters (FAPs) represent a complete set of basic facial actions and allow the representation of most natural facial expressions. All parameters involving motion are expressed in terms of the Facial Animation Parameter Units (FAPU). These correspond to fractions of distances between key facial features (e.g. the distance between the eyes). Figure 2 shows the locations of the feature points as defined by the MPEG-4 standard.

3.1 Mesh Deformation using MPEG-4 Feature Points

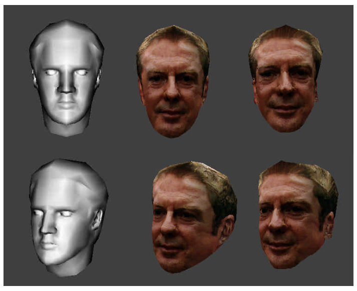

Figure 3. Morphing using deformation

Given a facial mesh, we can define the locations of the MPEG-4 feature points as per the specification, as shown in Figure 2. Also, for each feature point, we have to define the constraints as defined by the mesh deformation algorithm. Once we define this information, the facial mesh is ready to accept any FAPs and animate the face.

We also use the same deformation algorithm to deform the facial mesh in order to obtain a new face from a generic mesh. Figure 3 shows the results in two different views. The face on the left side is a generic facial mesh. The face in the middle is acquired using two orthogonal photographs of a person using the technique described in [8]. In this method, the locations of the feature points are extracted from the images and Rational Free Form Deformation (RFFD) is used to deform the generic face. Appropriate texture mapping is done to add realism. We apply the deformation algorithm explained in the previous section to the same generic face using these MPEG-4 feature points to obtain the face on the right. Thus the deformation algorithm applied for 3D morphing of generic head using MPEG-4 feature points generates satisfactory result.

4. OPTICAL TRACKING FOR ANIMATION

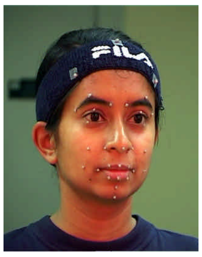

Figure 4. Placement of markers for selected MPEG-4 feature points

Facial feature tracking efforts have ranged from an ordinary video camera with coloured markers to retro-reflective markers and multiple cameras to extract directly the 3D position of the markers. We use one such commercially available system (VICON 8) to track the facial expressions and retarget the tracked features to our facial animation engine to examine the results of the deformation algorithm. We use a subset of MPEG-4 feature points corresponding to the FAP values to track the face and extract the FAPs frame by the frame. The next subsection in brief explains the algorithm for extracting the global head rotation and the calculation of the FAP values with the underlying assumptions. For the capture, we used 6 cameras and 27 markers corresponding to the MPEG-4 feature point locations. 3 additional markers are used for tracking the global orientation of the head. Figure 4 shows the placement of the feature points on the actor's face. We get the 3D trajectories for each of the marker points as the output of the tracking system.

4.1 Extracting Global Head Movements

We use 3 markers attached to the head to capture the rigid head movements (the global rotation and translation of the head). We use the improved translation invariant method [10]. Let (p_i, p_i') be the positions of the points on the surface of the rigid body, observed at two different time instants. For a rigid body motion, the pair of points (p_i, p_i') obeys the following general displacement relationship:

$$p'_i = Rp_i + t \qquad i = 1,2,\cdots,N \qquad (6)$$

R is a 3X3 matrix specifying the rotation angle of the rigid body about an axis arbitrarily oriented in the three dimensional space, whereas *t* represents a translation vector specifying arbitrary shift after rotation. Three non-collinear point correspondences are necessary and sufficient to determine *R* and *t* uniquely. With three point correspondences, we get nine non-linear equations while there are six unknown motion parameters. Because the 3D points obtained from the motion capture system are accurate, linear algorithm is sufficient for this application, instead of iterative algorithms based on least square procedure. If two points on the rigid body, p_i and p_{i+1}, undergoing the same transformation, move to p_i' and p_{i+1}' respectively, then

$$p'_i = Rp_i + t \qquad (7)$$

$$p'_{i+1} = Rp_{i+1} + t \qquad (8)$$

Subtraction eliminates translation t; using the rigidity constraints yields:

$$\frac{p'_{i+1} - p'_i}{\left|p'_{i+1} - p'_i\right|} = R\frac{p_{i+1} - p_i}{\left|p_{i+1} - p_i\right|} \qquad (9)$$

The above equation is defined as:

$$\hat{m}_i' = R\hat{m}_i \qquad (10)$$

If the rigid body undergoes a pure translation, these parameters do not change, which means the translation is invariant. After rearranging these three equations, we can solve a 3X3 linear system to get R and afterwards obtain t by substitution in equation 6. In order to find a unique solution, the 3X3 matrix of unit $\widehat{m}$ vectors must be of full rank, meaning that the three $\widehat{m}$ vectors must be non-coplanar. As a result, four point correspondences are needed. To overcome this problem of supplying the linear method with an extra point correspondence, a “pseudo-correspondence” can be constructed due to the property of rigidity. We find a third $\widehat{m}$ vector orthogonal to the two obtained from three points attached to the head. Thus, the system has lower dimension, requiring only three non-collinear rigid points. Once we extract the global head movements, the motion trajectories of all the feature point markers are compensated for the global movements, and the absolute local displacements and subsequently the MPEG-4 FAPs are calculated.

5. CONCLUSION AND FUTURE WORK

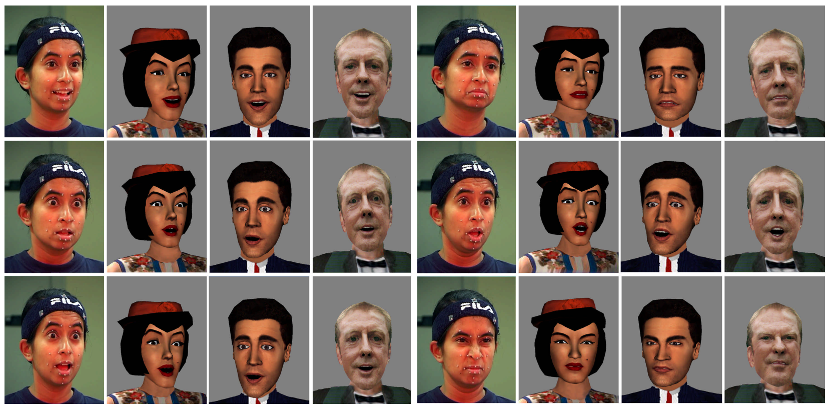

Figure 5. Facial Expressions extracted by Optical Tracking Applied to MPEG-4 Faces

Figure 5 shows the frames of animation depicting different facial expressions on the real face and three different synthetic faces. With the mesh deformation algorithm described here, we obtain a frame rate of 29 frames per second for an MPEG-4 compatible facial mesh with 1257 vertices on a 600 MHz Pentium III PC, with Matrix G400 graphics card using Open GL Optimizer for rendering. Thus, the algorithm is well suited for real time MPEG-4 compatible facial animation. We have assessed the deformation algorithm for realism by extracting the facial features from optical tracking and retargeting them to the synthetic face.

6. ACKNOWLEDGEMENTS

This work is supported by the EU ACTS SONG project. Special thanks are due to Dr. Tom Molet for his help with optical tracking system and to Chris Joslin for proof reading this paper.

REFERENCES

[1] A. Barr, "Global and Local Deformations of Solid Primitives", Computer Graphics, Vol. 18, No. 3, July 1984.

[2] V. Bearle, "A Case Study of Flexible Figure Animation", 3-D Character Animation by computer Course Notes, Siggraph'87.

[3] Y-K. Chang and A. Rockwood. "A generalizad de casteljau approach to 3d free-form deformation" Computer Graphics Proceedings of SIGGRAPH'94, pages 257--260

[4] C. Sabine. "Extended Free-Form Deformation: A Sculpting Tool for 3D Geometric modeling" Proceedings of SIGGRAPH '90, In Computer Graphics, 24, 4, pages 187--196, August 1990.

[5] E. Friesen WV (1978), Facial Action Coding System: A Technique for the Measurement of Facial Movement. Palo Alto, California: Consulting Psychologists Press.

[6] P. Kalra, A. Mangili, N. Magnenat-Thalmann, D. Thalmann, "Simulation of Facial Muscle Actions Based on Rational Free Form Deformations", Proc. Eurographics '92, Cambridge, pp. 59-69.

[7] F. Lavagetto, R. Pockaj, "The Facial Animation Engine: towards a high-level interface for the design of MPEG-4 compliant animated faces" IEEE Trans. on Circuits and Systems for video Technology, Vol. 9, no.2, March 1999.

[8] W. Lee, N. Magnenat-Thalmann, "Fast Head Modeling for Animation", Journal of Image and Vision Computing, Volume 18, Number 4, pp.355-364, Elsevier, 1 March, 2000.

[9] R. MacCracken and K. Joy, Free-form deformation with Lattices of arbitrary topology. Computer Graphics (Proc. of SIGGRAPH'96), pp. 181188, 1996.

[10] W. Martin and J. Aggarwal, "Motion Understanding Robot and Human Vision", Kluwer Academic Publishers, 1988.

[11] L. Moccozet, N. Magnenat-Thalmann, "Dirichlet Free-Form Deformations and their Application to Hand Simulation", Proc. Computer Animation `97, IEEE Computer Society, 1997, pp. 93-102.

[12] Specification of MPEG-4 standard, Moving Picture Experts Group, http://www.cselt.it/mpeg/

[13] F. Parke, "Parameterized Models for Facial Animation", IEEE Computer Graphics and Applications, Vol.2, Nuo. 9, pp 61-68, November 1982

[14] T. Sederberg and S. Parry, "Free Form Deformations of Solid Geometric Models", Computer Graphics, Vol. 20, No. 4, 1986.

[15] K. Singh, E. Fiume, "Wires: A Geometric Deformation Technique", Proc. SIGGRAPH'98, pp 405-414, 1998.

[16] D. Terzopoulos, K. Waters, "Physically Based Facial Modelling, Analysis and Animation", Journal of visualization and Computer Animation, Vo. 1, No. 2, pp 73-90, 1990.

[17] K. Waters, "A Muscle Model for Animation Three Dimensional Facial Expression", Computer Graphics, Vol. 21, No. 4, pp 17-24, July 1987.