Sketch-Based Controllers for Blendshape Facial Animation

Ozan Cetinaslan[1] and Verónica Orvalho[1] and J.P. Lewis[2]

[1]Instituto de Telecomunicações & Faculdade de Ciências, Universidade do Porto, Portugal
[2]Victoria University, New Zealand

Abstract
The blendshape approach is a widely used technique to generate realistic facial animation. However, creating blendshape facial animations using traditional weight editing tools requires either memorizing the function of a large number of parameters, or a trial-and-error search in a high-dimensional space. Direct manipulation interfaces address this problem, allowing the artist to directly move and pin manipulators placed on the surface of the face. Placing manipulators is an open-ended and slightly unnatural task for artists however. In this paper we present a sketch-based approach to this problem, inspired by artists' brush painting on canvas. In this approach the artist simply sketches directly onto the 3D model the positions of the manipulators that they feel are needed to produce particular facial expression. The manipulators activate the blendshapes in the model and allow the user to interactively create the desired facial poses by a dragging operation in screen coordinates. Our hybrid method can be used with any blendshape facial model and allows producing expeditious manipulation in an intuitive way.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques I.3.7 [Computer Graphics]: Three Dimensional Graphics and Realism—Animation

1. Introduction

Facial animation of digital characters in movies and video games has always been a popular subject for computer graphics. Blendshapes, a simple linear model of facial expression, is one of the effective and popular methods that enables the artists to create lifelike expressions with its intuitive artistic perspective. However, constructing facial animation using blendshapes takes a significant effort for realistic models that may include 100 or more blendshape targets.

Blendshape models produce facial expressions as a linear weighted sum of target faces which are an approximate independent basis of facial actions. Traditionally, these weights have been edited directly by the artists, using an interface in which each weight is presented as a slider. The direct manipulation method [LA10] offered a mathematical framework with *pin* and *drag* operation directly on the 3D facial models. Underlying the pin-drag interface is an inverse problem of determining the weights for selected point movements. Additionally, [MAO*12] offered a natural way to create, deform or control 3D models via sketching. Inspired by [LA10] and [MAO*12], we present a hybrid method that allows direct manipulation of blendshape models by using a sketched interface control system. Our method focuses on placing the manipulators on the mesh from the user sketch input. In our sketching procedure, the user can draw a simple freehand stroke onto 3D facial model at any angle and point of view independently. The method then places a "pilot" curve on the mesh which specifies the desired location of the manipulators on the 3D model. After the sketching is completed, the method creates manipulators automatically and activates all blendshape targets. The user can deform the model by dragging the manipulators. The process can be repeated until the desired facial pose for each frame of the animation sequence is obtained.

Related work is discussed in Section 2. Our approach is presented in detail in Section 3. Implementation and preliminary results can be found in Section 4. Section 5 is a summary with our conclusions and discussion of future work.

2. Related Work

All the movements of face come from a result of natural actions or emotions, for example talking, smiling, crying or even blinking. Creating realistic facial movements requires setting up complex control structures, such as joints, influence objects and blendshapes, as well as guaranteeing that

DOI: 10.2312/egsh.20151006

they all mix nicely to create appealing facial movements. This process is called *rigging* [OBP*12].

Practical and theoretical aspects of blendshape facial models are surveyed in [LAR*14]. The difficulty of editing complex blendshape facial models indirectly using the traditional slider interface was addressed in [LA10]. This difficulty was overcome by formulating direct manipulation as an inverse problem of determining the weights of blendshape for specific point movements and constraints, and solving the problem with a regularizer suitable for interactive editing. In more recent work, [SILN11] improved the regularizer for the inverse problem and presented a method for compression of complex models, allowing interactive GPU-based animation. [ATL12] offered a statistical prior algorithm that enables learning from the previous animation and allows rapid authoring of animation.

A sketching interface is a natural and useful concept for artists and animators. Building a sketching platform for an interface needs considerable accuracy. In [MAO*11], a 2D drawing region, the canvas, was augmented with a novel deformation control system that works in real-time. This allows simple, intuitive, and rapid manipulation of the 3D facial models using only simple strokes. Subsequently, [MAO*12] improved this method to allow animators to manipulate the 3D model directly in 3D space. [CJ08] describes a system to create articulation and posing of 3D facial models by using reference and target curves to find the optimum pose within the sketched input. Recently, [OBP*13] showed sketching applied not only for small deformations but also whole body bends and twist actions using a novel differential blending algorithm.

3. Method

We have created an application in Maya that allows controlling blendshapes based on our new approach. The system accepts a 3D blendshape facial model as an input in the center of local and world coordinate system of Maya. The method consists of three discrete steps: first, a free-form stroke is applied on the imaginary canvas which is a 2D drawing region aligned to the image plane. This stroke is converted to a NURBS curve and mapped on the 3D model. Second, the edit points (knots) of the curve are intersected with the closest vertices of the mesh. Finally, manipulators are constructed on the coordinates of the closest vertices and blendshape editing is activated. The user can deform the model by using these manipulators on runtime. This whole process can be repeated until the desired expression is realized.

3.1. Drawing Strokes and Mapping

The hand drawn stroke S can be defined as the set of ordered points $\{s_0, ..., s_{n-1}\}$, which represents the stroke in screen space. Originally, drawing is applied on a dynamically created virtual canvas that is oriented orthogonal to the view direction. The direction of S is determined by the user on the line of action.

S is projected from the virtual canvas onto the closest visible surface of the 3D model by a ray tracing operation (see Figure 1). After the projection, a NURBS curve C is generated on the mesh with the same length as the projected stroke. The stroked curve C may contain undesirable detail, for example, if the recording of mouse movement is uneven due to computer load. Because C will determine the position of the manipulators, we restrict its shape to a spline space. A cubic spline is selected based on the principle that it is as simple as possible while nevertheless closely approximating the surface of the 3D facial model. A lower degree curve would not lie on the surface, while a higher degree curve may incorporate unintended details of the artist's gesture. While in general the cubic spline will not lie exactly on the surface (our method can work with polygonal or spline blendshape models), this approximation is not obvious or obtrusive for the artist (Figure 1).

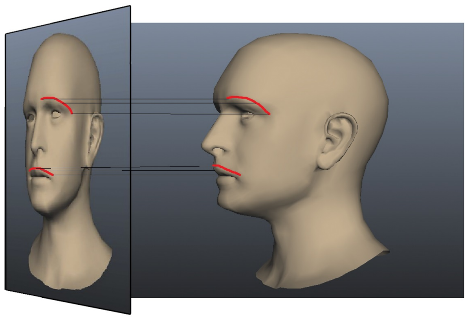

Figure 1: *A hand drawn stroke on the screen plane projected onto the visible surface of the 3D model by a ray tracing operation.*

3.2. Intersection of the Edit Points of Curve with the Closest Points on Mesh

The goal of this step is to create a physical connection between the NURBS curve and the 3D model in the scene. After the *Drawing Strokes* phase, the NURBS curve C of degree $D = 3$ is located onto the surface of the mesh and parametrized with k edit points EP, where k corresponds to D. The method creates C by selecting the EPs over the total number of point of the original S. These EPs are homogenous and located at the beginning, middle and end points of C. To create a direct correlation between C and the 3D model, we have implemented an algorithm, which intersects each EP with the closest vertex of the 3D model by using an optimized octree method embedded in Maya. Algorithm 1 describes the Points Intersection Procedure and Figure 2 illustrates a visual result.

3.3. Blendshape Weight Calculation

After *Intersection* phase, the manipulators $\{M_0, \ldots M_{N-1}\}$, which are represented as small spheres, are created on the

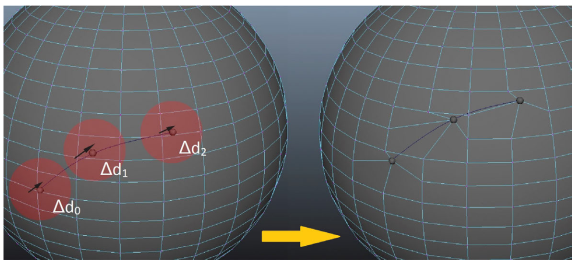

Figure 2: *Left: after a stroke operation, NURBS curve and manipulators are located on the 3D model without our Points Intersection Algorithm; Right: The manipulators location with the vertices of the 3D model with our Points Intersection Algorithm, each closest vertex moves towards to its related manipulator with $\triangle d_i$ pitch.*

Algorithm 1 Points Intersection Algorithm

```
 1: procedure INTERSECTION
 2:     for each EP in C do
 3:         $intersectPoint \leftarrow 0$
 4:         $intersector \leftarrow meshIntersector$
 5:         $matrix \leftarrow inclusiveMatrix$
 6:         $objectNode \leftarrow directedAcyclicGraphNode$
 7:         $status \leftarrow createIntersector(objectNode, matrix)$
 8:         if status = 1 then
 9:             $status \leftarrow getClosestPoint(EP_r, pointInfo)$
10:             $k \leftarrow faceID\ from\ pointInfo$
11:         for each allVertices in mesh do
12:             $vertexID \leftarrow vertexID\ from\ mesh$
13:             $faceID \leftarrow faceID\ from\ mesh$
14:             if faceID = k then
15:                 $ID_{CP} \leftarrow vertexID$
16:             else continue
17:         $setPoint(ID_{CP}, EP_r)$
```

locations of the intersected points on the 3D model automatically by our method. The user easily activates one of the M by selection and moves it in the 3D space. This is called a pin and drag operation. During the dragging action, blendshapes are already activated and ready to deform the 3D model at runtime, while other M on the 3D model remain stationary.

From the algebraic point of view, blendshape model can be defined as:

$$f = \sum_k w_k b_k \tag{1}$$

where f is a vector which includes all vertex coordinates of the model in an arbitrary order as $xyzxyz...$, w are the blendshape weights and b_k is the vector of coordinates for blendshape target k. In order to simplify the equation (1), we can also denote it as $f = Bw$. In the whole-face scheme, B includes all facial expressions. However, in delta form, the target shapes are offsets from the neutral face f_0 which also has to be added to equation (1):

$$f = Bw + f_0 \tag{2}$$

There are two significant observations which are stated by [LA10]. The first one is, if the face model includes n blendshape targets, the movements of the face are limited to a subspace of dimension n. The second one is, blendshape weights parameterize the valid shape space which means that the perceptual distance between facial expressions is approximated by the Euclidean distance in slider space. With these observations and the equation (2), we give a solution to the *direct manipulation inverse problem* to find the target poses.

In our method, some M are fixed and some are dragged. m denotes the vector of manipulators (constraints) that have been pinned and moved by the artist. The corresponding vertices of the face model f are extracted and stored in $\bar{f}$. $\bar{B}$ similarly denotes the matrix $(\bar{b}_1\bar{b}_2\ldots\bar{b}_m)$ containing the rows of each target corresponding to the constrained vertices. Obtaining the final weights, which minimize the difference between the model and m by using least squares, requires the solution of equation (3).

$$\min_w \|\bar{B}w - m\|^2 + \alpha\|w - w_0\|^2 \tag{3}$$

α is a user defined regularization parameter to keep the equilibrium during the weight update.

4. Implementation and Preliminary Results

Our method has been implemented as a plugin for Maya 2010 using C++, Python and Maya Embedded Language (MEL). For matrix calculations, the Numpy package has been used. During the implementation, the main focus was keeping the user away from the technical details of the method. In our method, equation (3) is solved in real time while a manipulator is dragged, thereby giving interactive visual feedback to the artist.

When $\alpha \approx 0$ in equation (3), haphazard shapes can result from repeated editing due to the lack of regularization of the inverse problem. To prevent this undesired deformation, keeping the α value relatively large values is essential. While the *Intersection* algorithm uses an octree for acceleration, there can still be a brief delay due to the large number of vertices and targets of the 3D model. However, this delay does not impact neither the preceding stroke nor the subsequent editing. For example, several for 3D facial models with 2422, 3234, 4092, 5565 or 6771 vertices, the creation of manipulators requires between 0.5 seconds and 4 seconds on a 4-core Intel Core i7-2600 3.4 GHz machine with 8 GB of RAM and an nVidia GTX 570 GPU. Figure 3 demonstrates some facial poses that have been created by our method.

We obtained comments from one digital artist and one professional technical director. In general, the feedback has been positive. After spending some time to gain familiarity with the system, both agreed that it is easier and faster

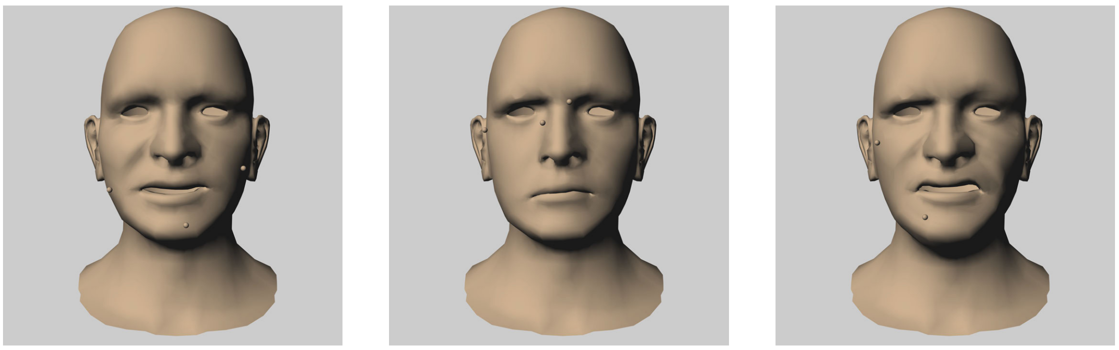

Figure 3: *Sample facial poses created using our Sketch-Based Controllers for Blendshape Facial Animation.*

to create a facial pose by using sketched manipulators that are created specifically to achieve the required editing. Another comment was that synchronized control of the blendshape sliders via the sketched manipulators had advantages in terms of both intuitive interaction and speed. A final comment cautioned that the underlying slider interface may still be needed in some cases to fine tune the final pose.

Limitations Our method has some limitations. To obtain high quality visual results, the blendshape model has to be constructed very accurately and carefully, because the facial poses are restricted to the span of the blendshape basis. This is a key advantage of our method, however it does mean that the editing inherits any peculiarities of the underlying model. For example, in our tests one of the models showed a slight movement in the upper face when the mouth is moved (this coupling is unexpected but not necessarily unrealistic). The strokes have to be applied inside the boundaries of the model and we assume that the operator has artistic skills. Finally, if the user aggressively drags the manipulator, a small delay in the resulting motion is perceptible.

5. Conclusions and Future Work

In this paper, we have presented a sketch-based approach to blendshape direct manipulation for facial animation. Our system has focused on precise interconnection between the sketching and the manipulators on a 3D mesh. This system has an intuitive interface that allows an easy and interactive procedure for locating the manipulators onto any blendshape facial model. Compared with other approaches, our work enables the user to create general poses without using the traditional slider-based editing interface while nevertheless keeping the "semantics" of a blendshape model.

While our system is meant to be used to rapidly produce static poses that serve as keyframes, in the future we plan to investigate temporal aspects of the stroke-based approach. We have not yet devoted time to speed optimizations and will pursue these. We will also investigate supporting different control objects such as lattices and physically based deformers both theoretically and in the implementation.

Acknowledgements

This work is funded by Fundação para a Ciência e Tecnologia/FCT (SFRH/BD/82477/2011), POPH/FSE program and Instituto de Telecomunicações (ref: PEst-OE/EEI/LA0008/2013 and UID/EEA/50008/2013). We especially thank Xenxo Alvarez, José Serra, Pedro Mendes and all Porto Interactive Center members.

References

[ATL12] ANJYO K., TODO H., LEWIS J. P.: A practical approach to direct manipulation blendshapes. *Journal of Graphics Tools 16*, 3 (2012), 160–176. 2

[CJ08] CHANG E., JENKINS O.: Sketching articulation and pose for facial animation. In *Data-Driven 3D Facial Animation*. Springer London, 2008, pp. 145–161. 2

[LA10] LEWIS J. P., ANJYO K.: Direct manipulation blendshapes. *IEEE Comput. Graph. Appl. 30*, 4 (July 2010), 42–50. 1, 2, 3

[LAR*14] LEWIS J. P., ANJYO K., RHEE T., ZHANG M., PIGHIN F., DENG Z.: Practice and theory of blendshape facial models. In *Eurographics STAR 2014* (2014), pp. 199–218. 2

[MAO*11] MIRANDA J. C., ALVAREZ X., ORVALHO J., GUTIERREZ D., SOUSA A. A., ORVALHO V.: Sketch express: facial expressions made easy. In *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling* (2011), SBIM '11, pp. 87–94. 2

[MAO*12] MIRANDA J. C., ALVAREZ X., ORVALHO J., GUTIERREZ D., SOUSA A. A., ORVALHO V.: Sketch express: A sketching interface for facial animation. *Computers & Graphics 36*, 6 (2012), 585–595. 1, 2

[OBP*12] ORVALHO V., BASTOS P., PARKE F., OLIVEIRA B., X. A.: A facial rigging survey. In *in Proc. of the 33rd Annual Conference of the European Association for Computer Graphics - Eurographics* (2012), pp. 10–32. 2

[OBP*13] ÖZTIRELI C. A., BARAN I., POPA T., DALSTEIN B., SUMNER R. W., GROSS M.: Differential blending for expressive sketch-based posing. In *Proceedings of the 2013 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2013), SCA '13, pp. 155–164. 2

[SILN11] SEO J., IRVING G., LEWIS J. P., NOH J.: Compression and direct manipulation of complex blendshape models. In *Proceedings of the 2011 SIGGRAPH Asia Conference* (2011), SA '11, pp. 164:1–164:10. 2