Sketching Articulation and Pose for Facial Meshes

Edwin Chang *
Brown University

Advisor: Odest Chadwicke Jenkins †
Brown University

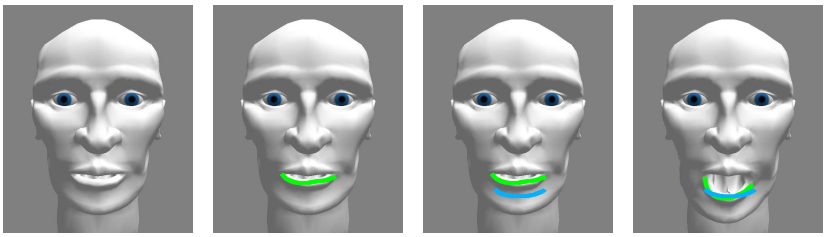


Figure 1: A reference curve (green) and target curve (blue) are sketched to pose the lower lip of an articulated mesh in our posing system.

Abstract

We present a methodology for articulating and posing meshes, in particular facial meshes, using sketching as input. Our method focuses on the use of two sketched curves, a reference and target curve. Our articulation system uses these curves to deform selected regions of a mesh in order to specify articulation. Likewise, our posing system uses reference and target curves to find an optimal pose in an articulation space of a mesh. This mesh can be articulated using our sketch-based approach or through alternative methods. Through our method, we seek to make articulation and posing more intuitive and simple for a novice to learn while also providing a method for more experienced users to prototype complex deformations.

1 Introduction

Articulating and posing meshes are both challenges that must be met in order to animate using 3D meshes. Defining the articulation of a mesh is often a tedious and complex process, requiring a user to specify several deformation variables for a desired motion. To achieve satisfactory results, a user may need to manually specify deformation settings for hundreds of vertices. Furthermore, an infinite number of plausible deformations can exist for a given mesh that range from the realistic flexing and extending of underlying muscle to cartoon squash and stretch motion. This problem is particularly difficult when defining the articulation of a facial mesh, where motion is quickly discernable as natural or unnatural to a viewer.

While the **articulation** (or rigging) of a mesh specifies the range of motion that mesh can take, an animator must then specify the **pose** of the mesh in that articulation space. Posing an articulated mesh presents a separate but related problem to articulation. To specify a pose efficiently, an animator must be provided with a set of controls that manipulate the deformation of the mesh. Such control rigs are often in the form of direct manipulation widgets or sliders that provide a puppet-like control of the mesh to the animator. Often as much time is spent setting up the user controls for an articulated mesh as specifying the mesh deformations. Professional computer animation packages, such as Alias Maya, often allow users to customize these interfaces to a high-degree, but are still usually built from a system of sliders and widgets.

*e-mail: ewchang@cs.brown.edu
†e-mail: cjenkins@cs.brown.edu

Many of the current interfaces for the above problems provide detailed control to a user, but are unintuitive both to novices and traditional animators trained with pencil and paper. A sketching interface, however, provides a familiar interface while still providing a high level of control to users. It can be particularly helpful to a novice who lacks a strong understanding of facial movement but is comfortable working with simple line drawings of a face. To traditional animators, it provides a direct correlation between hand drawn and 3D animation.

In this paper, we present two separate sketch-based processes, one for articulating a single mesh and one for posing an articulated mesh. In our articulation process, the user first selects regions of interests on the mesh and then manipulates the mesh using curves drawn in the image plane. These curves provides a simple and rapid method of defining deformations. Our posing process uses a similar approach, but does not require the user to specify regions of interest. Instead, the user first draws a reference curve on the mesh for selection followed by a target carve. The pose is then optimized over the articulation space so that the distance between these two curves is minimized. The user can introduce additional constraints to pin parts of the mesh in place. This method is particularly beneficial to novices to use as it requires no knowledge of the underlying articulation.

Both our articulation and posing methods can work in tandem, but are independent such that one can be replaced by alternative methods of articulating or posing. For example, we also use our posing method with a blend-shape articulated mesh. These methods work best with facial meshes, but are suitable for other types of meshes as well.

Our approach provides a simple sketch-based interface for users to specify deformations to a mesh and to pose those deformations. Other approaches to the these problems have been proposed, but were not capable of smooth deformation in enclosed regions, which is neccessary for facial animation.

2 Related Work

There exists much previous work in mesh articulation and deformation, as well as the closely related field of mesh editing. One typical approach for facial articulation is to create several meshes with the same topology and blend between them, i.e. a blend shape approach. While robust and granting users a high amount of control, this approach often requires many blend shapes. The blend shape process has also been combined with a skeletal approach to provide the flexibility of a skeletal system with the expressiveness of a blend shape system [Lewis et al. 2000]. Shapes have also been used as examples for a combination of shape and transform blending [Sloan et al. 2001]. We seek to maintain that level of expressiveness in our method without requiring additional shapes to blend between. We do use a simple blend shape method of 15 shapes, however, to test the ability of our posing process to work with several types of articulation systems. Our articulation system only begins with one mesh for a user to work with.

Free-Form Deformation (FFD) [Sederberg and Parry 1986] is one method that provides a wide range of possible deformation without requiring multiple shapes. Our work parallels the use of FFDs, in particular a curve-based FFD method that warps the mesh [Singh and Fiume 1998; Corrêa et al. 1998]. This type of FFD provides a method of smooth deformation that facilitates the use of curves sketched by users. Sketches have also been used to specify FFDs based on scalar field manipulation [Hua and Qin 2003] and as input to a gesture based FFD interface [Draper and Egbert 2003]. Outside of FFDs, sketches have also been used as skeletal strokes [Hsu and Lee 1994] to bend and twist 2-dimensional images.

Recent work has also focused on drawing directly onto the image plane in order to specify deformation. This poses challenges in interpreting the intent of users as well as providing a coherent translation from 2D to 3D space. This problem has also been encountered in 3D modeling using 2D sketches. One modeling method interprets 2D sketches as silhouettes to infer and construct 3D Shape [Igarashi et al. 1999]. Another solution used for mesh editing is to treat the process as an inverse NPR process [Nealen et al. 2005], where the mesh is transformed to match user-drawn contours and silhouette contours. This process is not ideal for articulation, however, as it can alter the topology of the mesh by inserting new vertices. Sketches have also been used to specify curves in a free-form skeleton method [Kho and Garland 2005], but the approach was limited to deformation in appendage-like parts of a mesh, e.g. tails, legs, or arms. We extend this approach, in particular its use of reference and target curves, to work with enclosed areas of a mesh, which is necessary for facial articulation.

Often one limitation of drawing on the image plane is that deformations remain parallel to the image plane. We approach this by constraining vertices to follow a surface similar to the method used in manipulating clothing [Igarashi and Hughes 2002], where cloth can be positioned by moving it across surface of a mesh.

Posing an articulated mesh involves its own unique challenges separate from those encountered in articulation. Control widgets are often added that allow users to interact directly with the articulation parameters. Sketching has been applied to manipulate multiple control points in order to pose the mesh [Swain and Duncan 2004], but these control points must have been previously defined by a user. Sketches have also been used to describe the motion of a figure across time rather than through individual poses [Thorne et al. 2004]. Other work has treated the posing problem as an optimization problem, attempting to determine the pose of a human figure that best matches hand-drawn sketches [Davis et al. 2003]. Our work in posing also views the problem as an optimization problem but focuses on posing articulated facial meshes. Posing a facial mesh has been approached previously using a blend shape method [Chuang and Bregler 2002], but required users to build a set of key blend shapes instead of using a previously created set. Other work has applied inverse kinematics to sets of existing blend shapes [Sumner et al. 2005], allowing users to interactively pose between the blend shapes. Our posing method also works with blend shape approaches, but is versatile enough to work with many types of articulation methods (including our own).

One of the methods of evaluation we use for our posing process involves the use of curves generated from the tracking of facial features in video. We use the eigen-points approach [Covell and Bregler 1996] in order to determine these curves. This approach uses an eigen-feature based method in order to place control points onto unmarked images, which we then use to define curves for posing.

3 Approach

While our articulation and posing methods are independent, they share a simple interaction scheme based on sketching curves. They differ in that our articulation method requires users to select regions of interest, while posing does not. In addition, while posing an articulated mesh users can specify constraints to keep parts of the mesh in place, which is unneeded when articulating but can be helpful for posing. The reason for these differences become clearer in the following explanations of our approach.

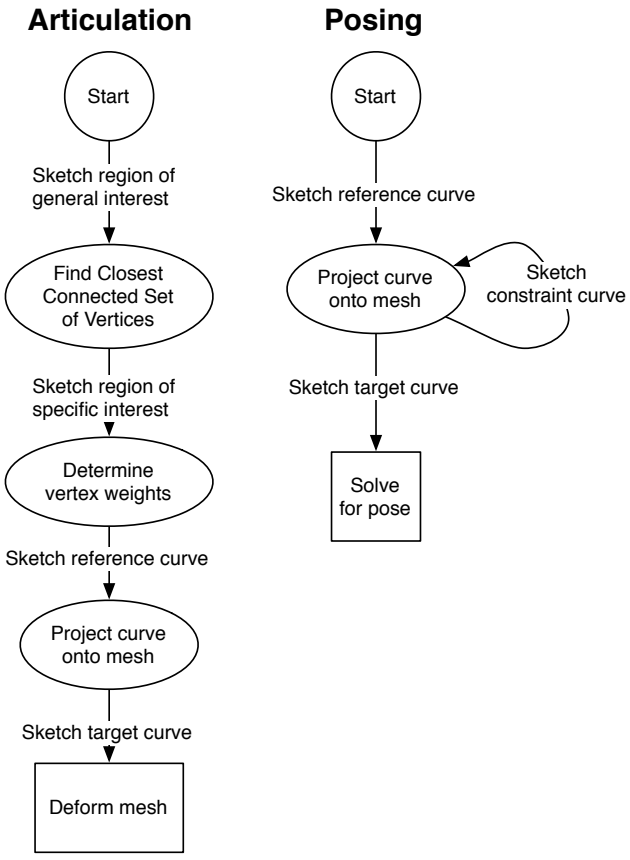


Figure 2: The flow of user control for articulation and posing

3.1 Articulation

Users specify one deformation at a time in our system in a 4-step process. Users first select a region of general interest, then a region of specific interest. Users can then draw reference and target curves to specify the deformation. Each of these deformations becomes one dimension in the articulation space. The 4 steps are pictured in Figure 3.

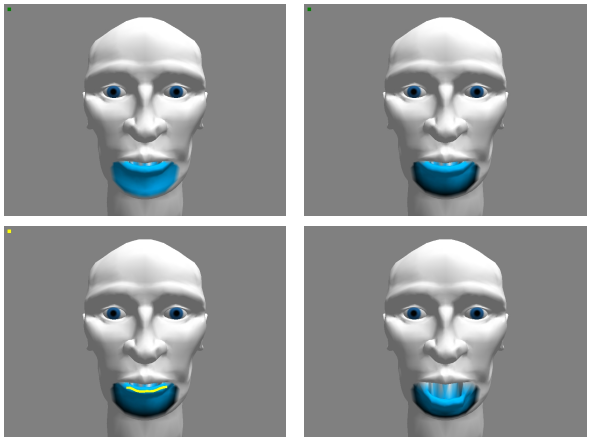


Figure 3: 1) A region of general interest is selected and then 2) a region of specific interest to specify articulation weights. 3) A reference curve and 4) target curve are then drawn to specify the deformation.

3.1.1 Regions of Interest

In the first step, the user must pick a general region on the mesh where the deformation is desired, a set of vertices V_g. To do so, the user first draws a curve C_g, an ordered set of points $\{\mathbf{g_1},...,\mathbf{g_n}\}$, to encircle the region on the image plane, selecting a set of vertices V_a. The desired set of vertices will be a subset of the set of all selected vertices ($V_g \subseteq V_a$), but often $V_g \neq V_a$ as some vertices in V_a may be behind the desired region or occluded by other parts of the mesh. In order to avoid selecting these vertices, a depth first search on V_a is used to determine connected graph subsets of vertices. The set of connected vertices containing the vertex closest to the camera and of sufficient size ($|V_k| > 10$, where $V_k \subseteq V_a$ and V_k is connected) is then chosen as V_g. Each vertex in this set ($\mathbf{v_i} \in V_g$) is then projected to the image plane in order to determine its 2D distance to the drawn curve C_g, which is then stored. We will call this distance g_i for every vertex $\mathbf{v_i}$ in V_g.

The user then encircles a region of specific interest with a new curve C_s to specify a set of vertices V_s, where $V_s \subseteq V_g$. Each vertex $\mathbf{v_i}$ in V_g is then assigned articulation weights by the following equation, where w_i is the articulation weight and c_i is the distance to the curve C_s on the image plane:

$$w_i = \begin{cases} 1.0 & \text{if } \mathbf{v_i} \in V_s, \\ \dfrac{g_i}{g_i + c_i} & \text{otherwise.} \end{cases} \quad (1)$$

In this manner, articulation weights smoothly blend off from 1.0 to 0.0 from the region of specific interest to the borders of the region of general interest. Our system displays the articulation weights to users by coloring vertices white if unselected, and blue to black from 1.0 to 0.0 (see Figure 4). There exists one restriction with this approach - g_i and c_i are 2D distances calculated on the image plane and using different camera views when selecting the two regions may result in an undesirable blend of articulation weights. This camera restriction only exists at this step.

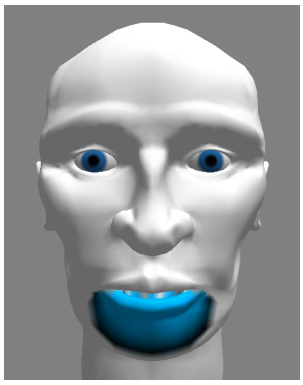


Figure 4: The blending of articulation weights between the two regions

3.1.2 Reference and Target Curves

Our system then generates a reference curve C_r, an ordered set of points $\{\mathbf{r_1},...,\mathbf{r_n}\}$, that estimates a skeletal curve for the region of specific interest. The curve is determined by ordering the vertices in V_s by the x value of each vertex to form C_r. If the difference in the minimum and maximum y values of the vertices in V_s is larger than the minimum and maximum x values, the y value of each vertex is used to order C_r instead. This curve is then smoothed through convolution with a triangle filter reference $f(j)$ across a kernel size v that is $1/3$ of the number of vertices in the curve ($v = |C_r|/3$):

$$\mathbf{r_i'} = \sum_{j=-v/2}^{v/2} f(j)\mathbf{r_{i+j}} \quad (2)$$

In some cases this estimated curve may not be satisfactory to the user, especially when the region of interest does not have a distinct curve-based feature, like the cheek of a face. If desired, the user can redraw the curve C_r on the image plane, which is then projected onto the mesh to form the reference curve. The reference curve, either estimated or not, is then slightly smoothed to account for noise (convolved with a triangle filter reference) and reparameterized to have a regular spacing. We reparameterize the line by choosing the new points by distance along the original line, where $\mathbf{r_i}'$ is the i-th of n points along the reparameterized curve:

$$\mathbf{r_i}' = C_r(\frac{i-1}{n}) \quad (3)$$

With the reference curve C_r smoothed and reparameterized in 3D space, the user can choose to move the camera and view the mesh at different angles. In order to facilitate this, the system does not depth test when rendering curves, instead overlaying them over the entire image.

In the final step, the user draws a target curve C_t, indicating how the mesh should be deformed so that the reference curve meets the target curve. The order of the points of the curve is reversed if the target curve's endpoint, q_n, is closer to the reference curve's start

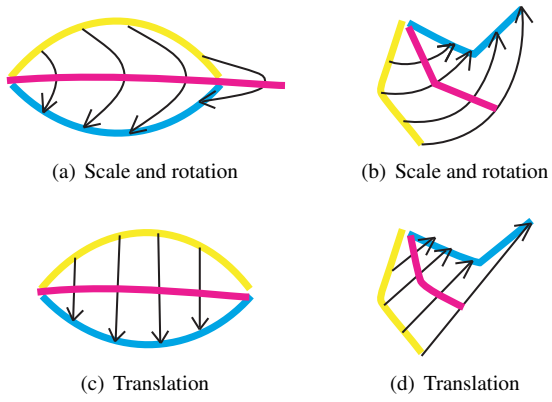
point, p_1, than the target curve's own startpoint, q_1 (i.e. reverse C_t if $|q_n - p_1| < |q_1 - p_1|$. The target curve is then reparameterized to match the number of points in the reference curve, n. The points of the target curve are then projected into 3D space by using the distances to the camera of the corresponding points on the reference curve, d_1 to d_n.

3.1.3 Curve Interpolation

Since the target and reference curves now share the same number of points, we can determine rotations between the matching line segments on the two curves by finding the cross product of the two segments and the angle between them. We will call these rotations ϕ_j for each segment j. The system stores these rotations as relative for each segment, such that each rotation assumes all rotations previous to a line segment have been applied. By keeping rotations relative, we can determine partial rotations between points on the curves when we perform the mesh deformation. The system also calculates a scale change s_i between the two segments, as the two curves may be different lengths:

$$s_i = \frac{|q_{i+1} - q_i|}{|p_{i+1} - p_i|} \tag{4}$$

With the rotations and scales for each segement of the lines, the system can then interpolate between the curves by applying a partial transformation α (where $0 \leq \alpha \leq 1$) of $\alpha\phi_j$ and αs_i to the line segments of the reference curve.



(a) Scale and rotation (b) Scale and rotation

(c) Translation (d) Translation

Figure 5: Different methods of curve interpolation

In certain situations, however, applying scale and rotation to interpolate is inappropriate. The curves in Figure 5(a) and 5(b) are interpolated using the rotation-scale method. In Figure 5(b), this works well, especially if these lines pose an appendage like a leg. In Figure 5(a), however, the curve becomes undesirably extended, which would be inappropriate if these curves were posing a blinking eyelid. For this case, we instead linearly interpolate between corresponding points on the two curves, translating the points without regard to scale or rotation of the line segments (Figure 5(c). Our system automatically chooses this method if the endpoints of the reference and target curves are within 10 pixels of each other, but also allows the user to specify otherwise.

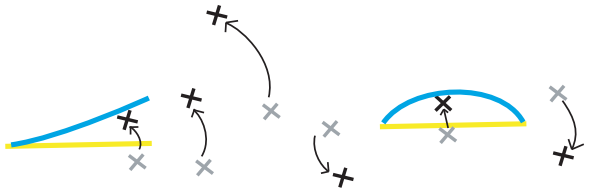
3.1.4 Mesh Deformation

Once the system has an appropriate method of interpolation between the reference and target curves, it can deform the vertices of the mesh according to those curves. Each vertex $\mathbf{v_i}$ in V_g is projected onto the reference curve to find the closest point on that curve, which is then stored as a proportional distance along the length of the entire curve, l_i, where $0 \leq l_i \leq 1$. This projection is done on the image plane in 2D space so that vertices farther from camera than other vertices still project to an appropriate reference point $\mathbf{r_i}$. The system then determines the corresponding point on the target curve, which we will call the target point $\mathbf{t_i}$, by the distance along the target curve according to l_i. We then apply the translation from the reference point to the target point ($\mathbf{t_i} - \mathbf{r_i}$) to the vertex. We must also apply a rotation transformation to the vertex centered around the target point. Since this point does not likely lie on the end of a line segment on the curve, we must calculate the rotation.

Our system first combines all the line segment rotations previous to the target point, ϕ_j from 1 to $k-1$, where the target point lies on segment k. We then apply a partial rotation of the last line segment's rotation, ϕ_k, according to the length along on that segment the target point lies, a value from 0 to 1 we will call u. We express this in the following equation, where the final rotation is ϕ_t. The rotations are centered about the target point.

$$\phi_t = (\prod_{j=1}^{k-1} \phi_j)(\phi_k u) \tag{5}$$

In order to pose between the reference and target curves, the system applies the same operations, but instead uses an interpolated curve, determined using the method described in Section 3.1.3, instead of the target curve. For similar reasons discussed concerning



(a) Deformation with rotation (b) Deformation with undesirable rotation

Figure 6: Different examples of mesh deformation with rotation

curve interpolation, rotations are not always desired in the mesh deformation. In Figure 6, a mesh deformation with rotations on three vertices is depicted in two examples. Rotations are appropriate for Figure 6(a), but less so for Figure 6(b), especially if this were deforming an eyelid. Vertices past the endpoints of the curves can move greater distances than expected due to rotation. For this reason, when curve interpolation does not use rotations, we do not apply them in mesh deformation as well.

Since deformations are specified using curves in the image plane, it can be difficult to specify deformation outside of one plane of movement. We approach this problem by allowing the user to specify simple surfaces to follow. In Figure 7, the deformation is constrained to maintain the vertices' distance from the eyeball sphere. Otherwise, the vertices move in only one direction and the eyelid intersects the eyeball.

Once all the vertices have been repositioned according to the target curve, they are returned back to their original positions according to the value of the articulation weights determined previously. The transformation is calculated as a linear translation for each vertex, where vertices with weight 0 return completely to their original position and vertices with weight 1 stay in their new position. In this

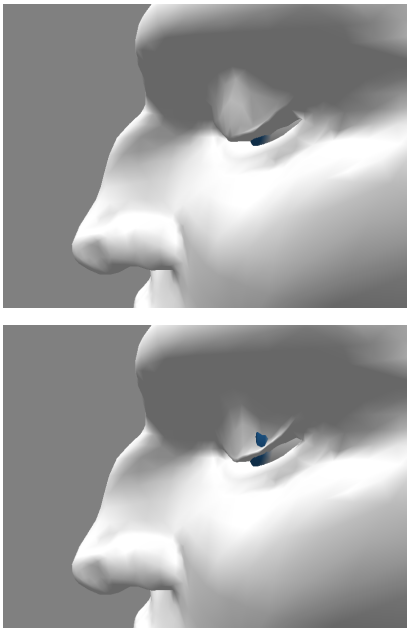


Figure 7: An eyelid deformation constrained and not constrained to follow the eyeball surface

manner we can ensure smooth deformations even when the region is enclosed by other parts of the mesh.

Multiple instances of these deformations can be applied to a mesh and overlap across vertices. In this case, deformations are applied in succession to the mesh. While this can result in some unexpected combinations of deformation, it provides a greater range of deformation that works well with our posing process.

3.2 Posing

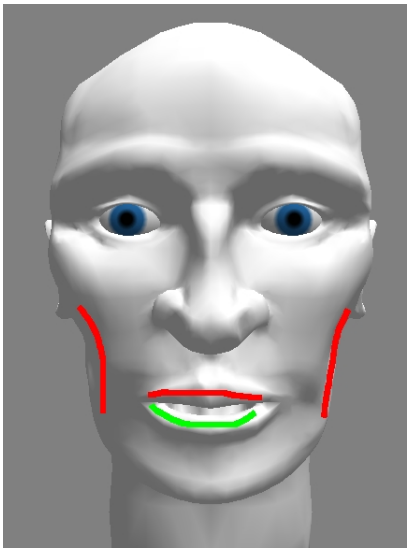


Figure 8: The posing process, with constraints in red and reference curve in green

Given an articulated mesh, posing that mesh presents its own challenges for estimating the appropriate articulation parameters. Our approach casts pose parameter estimation as an optimization problem. We apply our optimization engine to an articulated mesh from our method and a system based on blend shape interpolation.

Unlike our articulation method, the user does not need to specify any region of interest. Instead, the user first draws a reference curve C_r, which is projected onto the mesh. The user then draws a target curve C_t. As in our articulation method, we reverse the order of C_t if its endpoint is closer to the reference startpoint than its own start point. We then reparameterize the target curve to match n, the number of points in the reference curve. The target curve is then projected into 3D space using the distances from the camera along the reference curve. Our system then searches the articulation space M^d of d deformers to find an optimal pose P given by the optimal articulation parameters $\mathbf{x}$ that minimizes the distance between the reference curve, which maintains its position on the mesh, and the target curve. The distance term for the optimization is given by the following, where $\mathbf{r_i}$ and $\mathbf{t_i}$ are corresponding points on the reference and target curves for a given pose $\mathbf{x}$ in an articulation space of d dimensions.

$$E(P) = \sum_{i=1}^{n} |\mathbf{r_i} - \mathbf{t_i}| \tag{6}$$

In order to solve this optimization problem, we use the downhill simplex method [Press et al. 1992], which gives us the ability to perform optimization without the use of derivatives. Since this is the case, the optimization process does not need knowledge of the underlying articulation system and can work with any type of articulation. The downhill simplex method searches a d-dimensional space using a simplex shape of $d+1$ points that searches the space by reflecting and contracting itself until it reaches its goal (Figure 9). The optimization works best with non-hierarchical articulation (like faces, rather than arms), however, and is only efficient for a limited number of variables ($d < 20$). We propose methods to deal with this limitation in our discussion section.

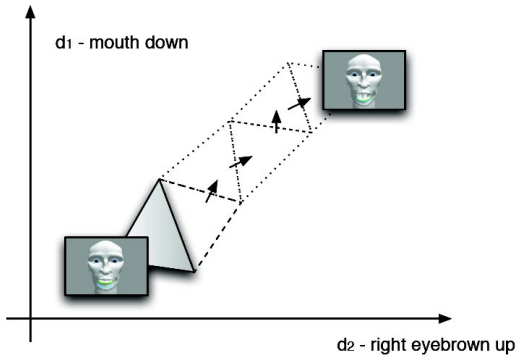


Figure 9: Searching in a two-dimensional articulation space using the downhill simplex method

Using the downhill simplex method, we determine we have reached an acceptable solution when the vector distance travelled in one iteration is less than a fractional tolerance of 0.05. After we have found this solution, we perform a cleanup stage. Since several of the articulation parameters may have had no effect on the region of interest, these parameters may have become unneccessarily changed through searching the articulation space in the optimization process. We evaluate a pose P_i for each articulation variable x_i, where x_i is set to its original value and all other variables are set to those

from $\mathbf{x_o}$, the set of articulation variables derived from optimization. If the difference between $E(\mathbf{P_i})$ and $E(P_o)$ (where P_o is the pose set by $\mathbf{x_o}$) is minimal, we return x_i to its original value.

Our system also provides a method for the user to set constraints on the mesh with additional curves in order to keep parts of the mesh in place (Figure 8). Each constraint curve K_j, a set of ordered points $\{k_1,...,k_n\}$ is projected onto the mesh. When a pose is evaluated using equation 6, the following term is also added for each constraint, where k_i' is the position of k_i in the new pose P.

$$E_j(P) = \sum_{i=1}^{n} |\mathbf{k_i'} - \mathbf{k_i}| \tag{7}$$

Constraint curves are useful as a deformer on a mesh may have a small effect on the region the reference curve lies on even though it mainly deforms a separate area. For example, a cheek deformer could slightly affect the vertices around the lips on a mesh. When the user attempts to pose the lips the cheeks could then be undesirably affected. These constraints are drawn on the mesh in the same manner the reference curve is. Previously used reference curves can also be used as constraint curves in order to keep previously specified deformation in place.

4 Results

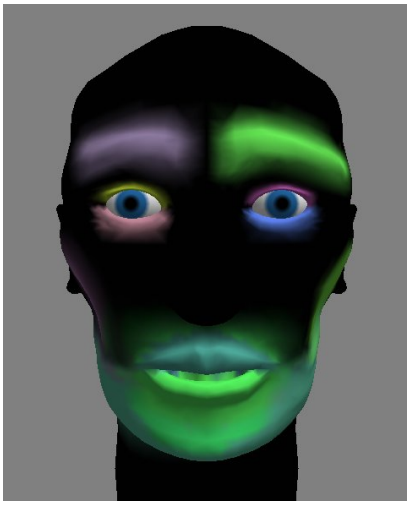


Figure 10: An articulated mesh colored according to deformer and articulation weights

We begin with the mesh of a face (seen in Figure 8) and articulate it using our system to specify a deformation for each eyelid, eyebrow, cheek, jaw, and various movements of the lips (Figure 12) for a total of 15 deformers. Figure 10 depicts these deformers as separately colored regions that fade to black according to articulation weights. Figure 11 shows some of the poses that can be acheived using this articulation. Each of these deformations was created quickly, in under 2 minutes for each. By comparison, specifying similar deformations in a blend shape approach required 10-20 minutes per shape. For eyelid deformations, we specified the deformation to follow the surface of a sphere centered at the eye. Our system also works for non-facial meshes, like the trunk of an elephant (Figure 13).

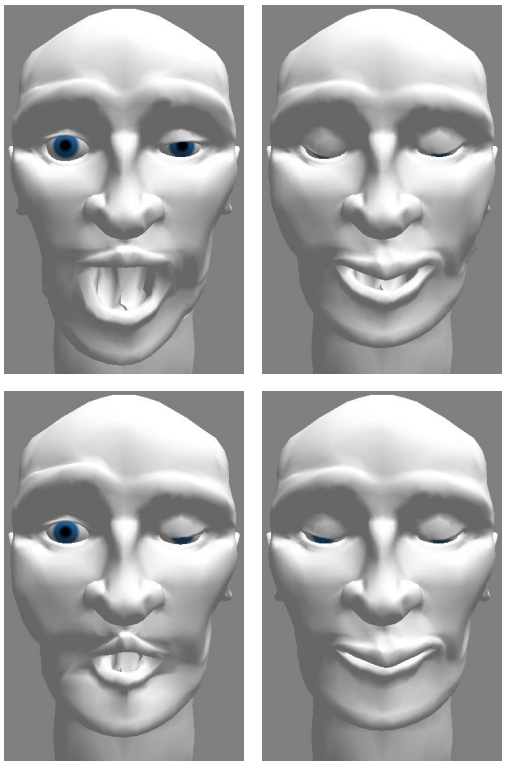


Figure 11: A sampling of poses in our articulation space

Bringing this articulated mesh into our posing system, we can pose the face using reference and target carves. We also test our posing system with a mesh articulated by a blend shape method using shapes created in Alias Maya (Figure 14) and achieve similar results in both. In the top example, we pose the mouth in two iterations, one for the upper lip and one for the lower lip. In total, with the cheek constraints, we drew 6 curves to pose the face (2 constraint curves, and 2 pairs of reference and target curves). In the lower example we posed the right eyelid and left eyebrow using 4 curves (2 pairs of reference and target curves).

4.1 Posing from Video Features

We additionally used our posing process with curves generated from tracking of facial features in video on our sketch-based articulated mesh. These curves were determined through the eigenpoints method [Covell and Bregler 1996] and follow the eyebrows, eyelids, and lips of the subject in the video. These tracked curves, while slightly noisy, remain unfiltered in our testing. Since the facial features of the subject do not match those in the 3d mesh, relative changes in the tracked curves are applied to user-drawn reference curves to create new target curves. For one curve from video C_{vf} in frame f, relative changes, $\{\mathbf{c_1},...,\mathbf{c_n}\}$, from frame to frame for each point were determined. These relative changes were then applied to a user-drawn curve C_u reparameterized to have n points, $\{\mathbf{d_1},...,\mathbf{d_n}\}$. For each frame of video a new curve C_u' was determined by applying $\mathbf{c_i}$ to every point $\mathbf{d_i}$. The change $\mathbf{c_i}$ was also scaled up in order to account for difference in length between C_{v0} and C_u:

$$\mathbf{d_i}' = \mathbf{d_i} + \mathbf{c_i}\frac{|C_u|}{|C_{v0}|} \tag{8}$$

While the limited articulation of the mesh does not fully match the range of expression in the human face, the posing process works well at capturing the motion of the face across frames (Figure 15).

The optimization process for posing requires many iterations before convergence and results in a pause in the posing system after drawing the target curve. On a AMD XP 2000+ processor, this pause is under 5 seconds for the blend shape method and under 10 seconds for our articulation method. The optimization takes longer for our articulation method because it takes slightly longer to pose than the blend shape method. From pose to pose this time is small (the mesh can be posed at ($\sim$50 fps), but is still longer than the blend shape method ($\sim$90 fps).

5 Discussion

Our implementation allows users to quickly sketch out a wide range of articulations for a mesh. Several additions could be added to the system to allow for more complex movements, such as combining separately defined deformations, but our work focuses on establishing a simple system suited to novices or prototyping deformations for more complex systems. By combining deformations, however, we would be able to achieve deformation with similar results to pose space deformation [Lewis et al. 2000].

We also maintain a level of control in our deformations comparable to blend shape approaches (Figure 14). Furthermore, we do not face the limitations blend shapes have, such as the issues linear blending between shapes can cause. For example, it is difficult to have rotational movement with blend shapes, like an eye blink or jaw movement. Our method can recreate these motions. With our method, we are able to apply the strengths of free form deformation to enclosed areas of the mesh while maintaining smooth deformation.

Our posing process is likewise easy to use and requires little to no training or knowledge of the articulation system. Through sketching curves to pose the mesh, the user has intuitive control over the articulation space while unaware of the actual articulation parameters.

The optimization required for posing is not instantaneous, however, and lacks the interactive speeds of other methods. The tradeoff for compatibility with any articulation system is a speed loss. Further limitations involve the limit of number of variables the optimization process can deal with. The downhill simplex method is only effective to under 20 variables and a large number of variables will further slow down the optimization. As many professional facial animation systems often have several hundred controls, this method may be impractical. We can reduce the problem, however, by limiting the articulation search space only to those articulation variables that affect the reference curve. If the search space still remains overly large, our method can be used in stages, first posing articulation variables that have a large effect and then smaller variables in greater detail. Another possible approach would be to use dimension reduction methods on the articulation space, like principal component analysis.

6 Conclusion

In this paper, we presented a sketch-based method of preparing a mesh for animation in two processes - articulation and posing. Our system focused on facial animation, but was adept at defining other kinds of motion as well. This system was simple to use and provided the freedom for a user to specify many kinds of deformation. Furthermore, the posing process was flexible enough to work with other kinds of articulation, including blend shape interpolation.

7 Acknowledgements

We thank John F. Hughes for his comments and advice on our work. Special thanks also go to the Brown Graphics Group, in particular Olga Karpenko and Tomer Moscovich for initial discussions on this work.

References

CAPELL, S., GREEN, S., CURLESS, B., DUCHAMP, T., AND POPOVIĆ, Z. 2002. Interactive skeleton-driven dynamic deformations. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, 586–593.

CHUANG, E., AND BREGLER, C. 2002. Performance driven facial animation using blendshape interpolation. Tech. rep., Stanford University Computer Science.

CORRÊA, W. T., JENSEN, R., THAYER, C., AND FINKELSTEIN, A. 1998. Texture mapping for cel animation. In *Proceedings of ACM SIGGRAPH 1998*, 435–446.

COVELL, M., AND BREGLER, C. 1996. Eigen-points. In *Proceedings of 3rd IEEE International Conference on Image Processing*, 471–474.

DAVIS, J., AGRAWALA, M., CHUANG, E., POPOVIĆ, Z., AND SALESIN, D. 2003. A sketching interface for articulated figure animation. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 320–328.

DRAPER, G. M., AND EGBERT, P. K. 2003. A gestural interface to free-form deformation. In *Proceedings of Graphics Interface 2003*, 113–120.

HSU, S. C., AND LEE, I. H. H. 1994. Drawing and animation using skeletal strokes. In *Proceedings of ACM SIGGRAPH 1994*, 109–118.

HUA, J., AND QIN, H. 2003. Free-form deformations via sketching and manipulating scalar fields. In *Proceedings of the eigth ACM symposium on Solid modeling and applications*, 328–333.

IGARASHI, T., AND HUGHES, J. F. 2002. Clothing manipulation. In *Proceedings of the 15th annual ACM symposium on User interface software and technology*, 91–100.

IGARASHI, T., MATSUOKA, S., AND TANAKA, H. 1999. Teddy: a sketching interface for 3d freeform design. In *Proceedings of Graphics Interface 2003*, 113–120.

KHO, Y., AND GARLAND, M. 2005. Sketching mesh deformations. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games*, 1142–1147.

LEWIS, J., CORDNER, M., AND FONG, N. 2000. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 165–172.

NEALEN, A., SORKINE, O., ALEXA, M., AND COHEN-OR, D. 2005. A sketch-based interface for detail-preserving mesh editing. In *Proceedings of ACM SIGGRAPH 2005*, 1142–1147.

PRESS, W. H., FLANNERY, B. P., TEUKOLSKY, S. A., AND VETTERLING, W. T. 1992. *Numerical Recipes in C - The Art of Scientific Programming*. 408–412.

SEDERBERG, T. W., AND PARRY, S. R. 1986. Free-form deformation of solid geometric models. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, 151–160.

SINGH, K., AND FIUME, E. 1998. Wires: a geometric deformation technique. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, 405–414.

SLOAN, P.-P. J., ROSE, C. F., AND COHEN, M. F. 2001. Shape by example. In *Proceedings of the 2001 symposium on Interactive 3D Graphics*, 135–143.

SUMNER, R. W., ZWICKER, M., GOTSMAN, C., AND POPOVIĆ, J. 2005. Mesh-based inverse kinematics. In *Proceedings of ACM SIGGRAPH 2005*, 488–495.

SWAIN, M., AND DUNCAN, B. 2004. Sketchpose: Artist-friendly posing tool. SIGGRAPH 2004 Sketch.

THORNE, M., BURKE, D., AND VAN DE PANNE, M. 2004. Identifying and sketching the future: Motion doodles: an interface for sketching character motion. In *Proceedings of ACM SIGGRAPH 2004*, 424–431.

Figure 12: A few of the deformations created in the sketch-based articulation system

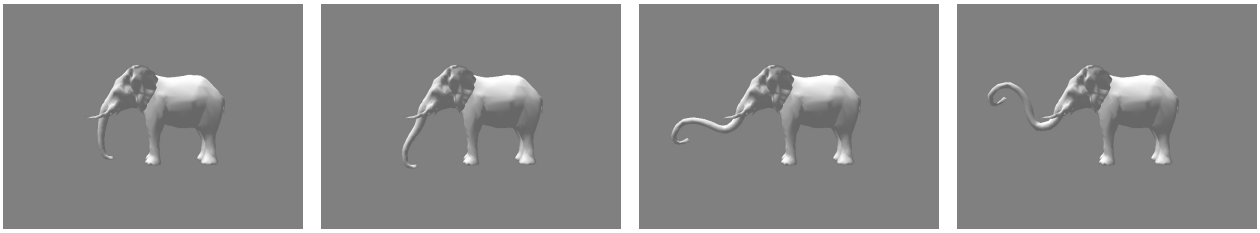


Figure 13: Deformation of an elephant's trunk using the articulation system

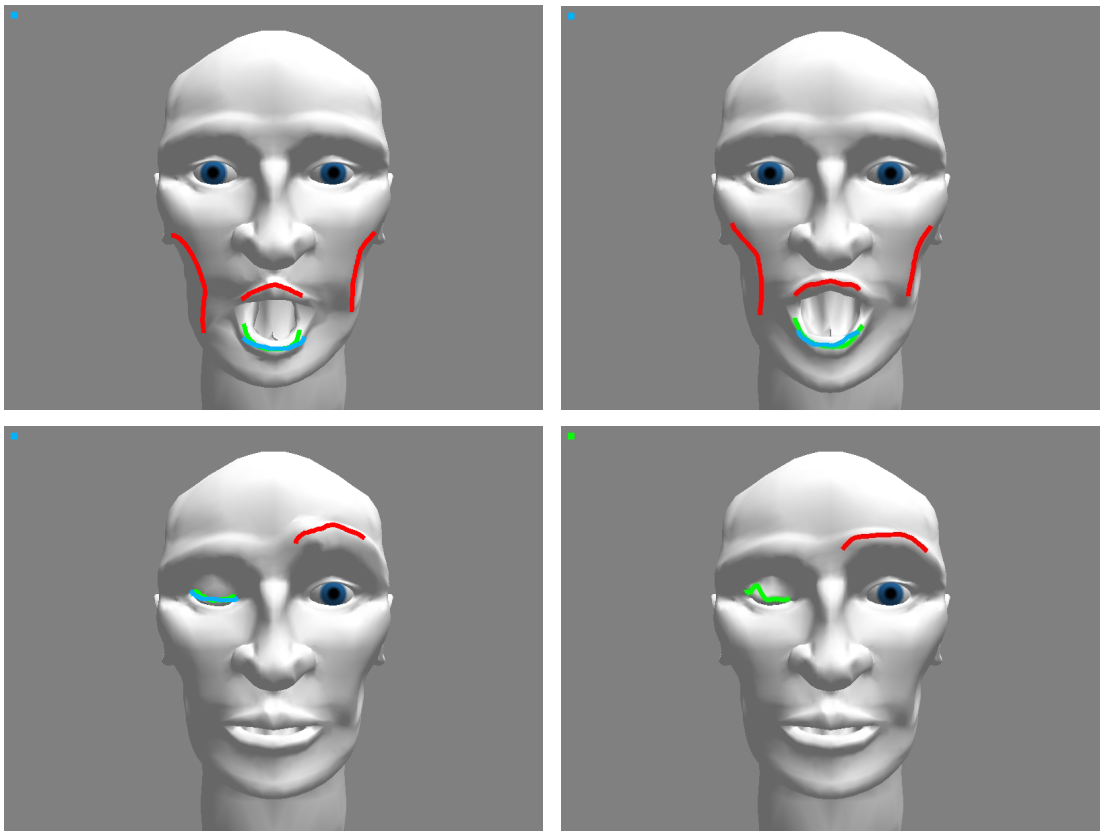


Figure 14: Posing using similar curves on a sketch-based articulation (left) and a blend shape articulation (right)

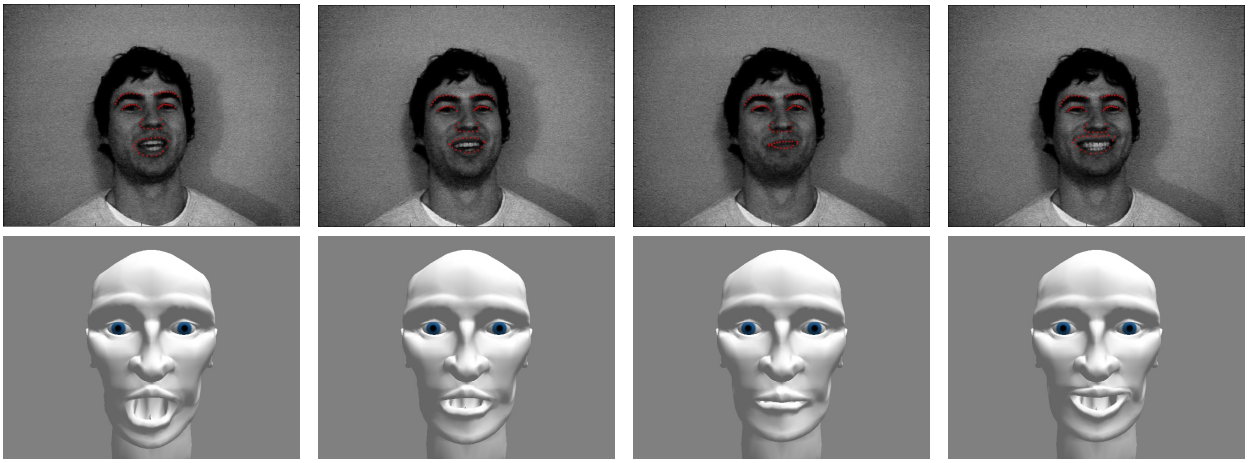


Figure 15: Posing using curves from tracking of facial features in video