Pyramid Coordinates for Morphing and Deformation

Alla Sheffer
University of British Columbia
sheffa@cs.ubc.ca

Vladislav Kraevoy
University of British Columbia
vlady@cs.ubc.ca

Abstract

Many model editing operations, such as morphing, blending, and shape deformation, require the ability to interactively transform the surface of a model in response to some control mechanism. For most computer graphics applications, it is important to preserve the local shape properties of input models during editing operations. Our work introduces the first, to our knowledge, mesh editing technique that explicitly preserves local shape properties. The method is based on a local shape representation, which we refer to as pyramid *coordinates. The pyramid coordinates capture the local shape of the mesh around each vertex and help maintain this shape under various editing operations. They are based on a set of angles and lengths relating a vertex to its immediate neighbors. This representation is invariant under rigid transformations. Using pyramid coordinates, we introduce a new technique for mesh deformation and morphing based on a small number of user-specified control vertices. Our algorithm generate natural looking deformations and morphing sequences in seconds with minimal user interaction.*

1 Introduction

Mesh editing techniques are at the heart of many geometry processing applications in computer graphics and geometric modeling. Good editing tools must be intuitive, easy to use, robust and efficient. Most importantly, they should provide natural looking resultant models while requiring only minimal user interaction. While it is somewhat difficult to quantify the intuitiveness or natural look of edited models, the consensus seems to be that the models should preserve as much as possible their local and global shape properties (distances, angles, etc.) [1].

This paper focuses on three main types of editing operations:

- *Deformation*: Mesh deformation is based on vertex relocation. Hence, its basic component is the displacement of a single vertex followed by the redefinition of the model geometry around this vertex in response to the displacement. The geometry redefinition should preserve the local shape properties in the displaced region and provide a smooth transition between it and the rest of the model. Complex deformations can typically be expressed through a combination of vertex relocations [2]. In addition to modeling, deformation techniques are also commonly used in animation, where forward animation sequences are constructed by continuous deformation.

- *Morphing*: Morphing is one of the basic and most popular computer graphics applications. Morphing algorithms create a smooth transition in time between multiple input models. The primary challenge of all morphing algorithms is to generate intermediate models that retain the appearance and properties of the source models.

- *Blending*: Many operations applied to multiple models can be viewed as a combination of global or local blending. Blending is often viewed as the generation of a single intermediate model in a morphing sequence. It is somewhat more challenging than morphing, as the interpolation throughout the model is non-linear in time. In the case of local blending, parts of the input models are preserved in their entirety, connected by smooth transition regions.

1.1 Previous Work

Due to recent developments in digital geometry processing, mesh editing has been the subject of increasing attention in recent years. Much of the research in this area has focused on subdivision and multiresolution editing techniques (e.g. [3, 4, 5]). These methods are very popular, but have several drawbacks. The multiresolution tools combine simplification with subdivision surface reconstruction as prerequisites for the editing, and are thus non-trivial to implement. The required preprocessing is time consuming. More importantly, the majority of multiresolution morphing techniques, such as [5], interpolate absolute coordinates leading to visible artifacts in some morphing sequences [1].

Global structure information on the models, such as a skeleton, can facilitate the editing operations. As such, skeleton extraction is the first step of many model deformation and animation algorithms (e.g. [6]). The skeleton is then modified according to user specifications. The desired global shape deformation is then obtained by reconstructing the shape corresponding to the modified skeleton. Although theoretically existent for any model, skeletons are generally hard to construct. Boolean editing opera-

Proceedings of the 2nd International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'04)
0-7695-2223-8/04 $ 20.00 IEEE

tions usually require skeletons with identical connectivity. The construction of such skeletons in 3D remains, to our knowledge, an open problem.

Morphing or blending algorithms commonly begin by constructing a common connectivity for the two or more geometries at hand [1]. Most methods then proceed to linearly interpolate the geometry to generate intermediate models . As a consequence, in many cases the resulting morphing sequences do not preserve the shape properties of the input models. Instead of gradually transforming into each other, features such as arms can sometimes disappear and grow out again. More sophisticated trajectory computation methods in 2D [7, 8] utilize compatible triangulations of the input models. Alexa et al. [8] suggested a technique for computing trajectories in 3D using compatible tetrahedralizations. However, the construction of compatible tetrahedralizations remains an open problem, so this approach remains impractical in 3D.

Gregory et al. [9] subdivide the models into patches and allow the user to define the trajectories of the patch corner vertices. The trajectories of the remaining vertices are computed by interpolating the corner trajectories. The method interpolates absolute coordinates often leading to unnatural artifacts [1].

The drawbacks of absolute coordinates techniques led to the introduction of morphing and deformation techniques based on local coordinates [10, 11]. The Laplacian coordinates [10] of each vertex are defined as a displacement vector between the average of the neighbor vertices and the actual 3D position of the vertex. These coordinates are not invariant under rotation and scaling. Alexa [10] uses Laplacian coordinates to generate smooth translational local morphs and deformations. However, the technique introduces visible artifacts (Figures 3 and 6) when the deformations or trajectories contain rotation or scaling. Sorkine et al. [11] extend the use of Laplacian coordinates by linearly approximating local rotation. However, the examples demonstrated in the paper contain a very small rotational component.

1.2 Contribution

As mentioned above, a major criterion in evaluating an editing technique is its ability to preserve the global and local shape properties of the original model. People are typically more sensitive to local distortion than global distortion [12]. Hence, for most computer graphics applications, the preservation of local shape properties during editing operations is of major significance. Our work introduces the first, to our knowledge, mesh editing technique that explicitly preserves local shape properties.

Our method is based on a local shape representation, which we refer to as *pyramid* coordinates. The pyramid coordinates capture the local shape (lengths and angles) of the mesh around each vertex and help maintain this shape under various editing operations. In contrast to Laplacian coordinates [10, 11] they are invariant under rigid transformations.

Based on this representation, we introduce algorithms for mesh deformation, morphing, and blending which preserve the shape properties of the input models. The proposed deformation method is based on a small number of control vertices or handles defined by the user, combined with a user-prescribed region of influence. Based on this minimal interface, our deformation method computes natural looking deformed models even under severe deformation. Our morphing procedure generates intermediate models which interpolate the shape properties of the input models. The algorithm supports the introduction of user defined trajectories for a number of control vertices. The technique is particularly well suited for local morphing operations where only a part of the model is modified. As opposed to many other existing methods, it generates a natural and smooth transition between dynamic and static parts of the model. The proposed editing algorithms are robust, efficient, and extremely simple to implement. In contrast to multiresolution or skeleton based techniques, they do not require any complex preprocessing. Another advantage of our editing approach is that although it does not explicitly prevent model self-intersections, the shape preservation property drastically reduces the risk of local self-intersection.

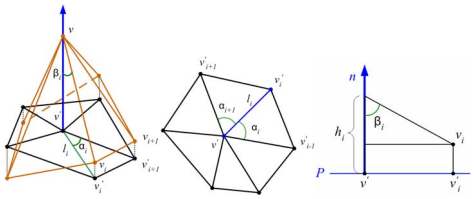
The rest of the paper is organized as follows. Section 2 introduces the pyramid coordinates and explains how to convert between the pyramid representation and Euclidean 3D coordinates. Sections 3 describes algorithms for computing morphing trajectories and model blending using pyramid coordinates. Section 4 explains the model deformation technique. Section 5 contains examples of using the different algorithms and compares the results to previous techniques. Finally, Section 6 summarizes the paper's contribution and points areas of future research.

2 Pyramid Coordinates

Pyramid coordinates have been designed to capture the shape of the mesh around each vertex. Therefore they measure the set of angles and lengths uniquely relating a vertex to its immediate neighbors (Figure 1).

2.1 Coordinates

Let v be a mesh vertex in 3D and let $v_1, v_2, ..., v_m$ be its neighboring vertices. Given the normal $n = (n_x, n_y, n_z)$ at v, we define the projection plane P as



(a) (b) (c)

Figure 1: (a) Pyramid coordinates: (b) tangential components in the projection plane P; (c) normal component β.

$$P = n_x x + n_y y + n_z z + d, \;\; d = -\sum_{i=1}^{m} n \cdot v_i. \qquad (1)$$

We define v' and v'_i to be the projections of v and v_i to P, respectively. The description of the vertex with respect to its neighbors consists of:

1. a set of angles α_i between the projected edges $\langle v', v'_i \rangle$ and $\langle v', v'_{i+1} \rangle$ (Figure 1 (b));
2. a set of angles β_i between n and the edges $\langle v, v_i \rangle$ (Figure 1 (c));
3. a set of projected edge lengths $l_i = \|v' - v'_i\|$.

Pyramid coordinates can be viewed as a combination of tangential (α and l) and normal (β) components. Given a 3D model, the coordinates are uniquely defined. By definition, both lengths and angles are invariant under rigid transformations. Hence, pyramid coordinates have this desired property.

2.2 Reconstruction

Given the pyramid coordinates, we can reconstruct the 3D mesh by explicitly solving the non-linear system defined by a set of equations linking the angles and lengths to the actual 3D coordinates. However this approach will render the algorithm impractical. Instead, we developed an efficient iterative Gauss-Seidel reconstruction procedure, described below, that takes advantage of the reproduction property of the mean value coordinates [13] to drastically simplify the necessary computations. The reproduction property implies that given the angles α_i and the lengths l_i computed from v' and v'_i, we can use the mean value weights

$$w_i = \frac{tan(\alpha_i/2) + tan(\alpha_{i+1}/2)}{l_i} \qquad (2)$$

to obtain the original position of v' from v'_i:

$$v' = \frac{1}{\sum_{i=1}^{m} w_i} \sum_{i=1}^{m} w_i v'_i.$$

The reproduction property holds even if some angles α_i are negative, which will happen when v' is outside the kernel of the polygon formed by v'_i. Thus, we obtain a linear formulation linking the position of a vertex in the projection plane to those of the neighbors (note that the weights remain constant throughout the reconstruction procedure). In case the neighbor vertices are not in the original positions, using these weights to position v' minimizes the angular distortion of the projected mesh [13].

We use this formulation of v' as a function of v'_i, to obtain the 3D coordinates of the mesh vertices from the pyramid coordinates:

1. For each vertex v recompute the position of v from the neighbor vertices:
   (a) Compute the current normal n at v.
   (b) Compute the projection plane P (Equation 1) and project the neighbor vertices of v to P.
   (c) Set the position of v' to
   $$v' = \frac{1}{\sum_{i=1}^{m} w_i} \sum_{i=1}^{m} w_i v'_i, \qquad (3)$$
   where w_i are the mean value weights derived from α_i and l_i (Equation 2).
   (d) To derive the position of v given v', calculate a set of offsets h_i along n using the angles β_i:
   $$h_i = \|v' - v'_i\| \cot(\beta_i) + (v_i - v'_i) \cdot n. \qquad (4)$$
   Finally, compute the new position of v by offsetting v' by the average of h_i along n as follows:
   $$v = v' + n \frac{1}{m} \sum_{i=1}^{m} h_i. \qquad (5)$$
2. Repeat until convergence.

The reconstruction procedure uses edge lengths only inside the mean value weights. Those weights are designed to minimize angular distortion. Therefore, under the existing formulation, angles are sometimes preserved at the expense of stretch. To account for stretch, we scale the weights w_i:

$$w_i = w_i \frac{\|v' - v'_i\|}{l_i}.$$

Scaling by the ratio of current edge length (in the projection plane) to the optimal length, has the effect of pulling edge lengths closer to the optimal. A similar idea was proposed by Belyaev et al. [14], who used the triangle stretch for scaling the weights. In addition to better length preservation, the use of a stretch component in the vertex placement computation drastically speeds up the convergence of the reconstruction procedure, especially for large deformations. When a control vertex is moved to a new location, it affects both the shape and the stretch of adjacent

triangles. Therefore, taking the stretch into account increases the magnitude of each vertex relocation step in the iterative procedure above.

The reconstruction formulation is invariant under rigid transformations and scaling. Therefore, to accurately reconstruct the original model we need to fix three vertices so as to eliminate the extra degrees of freedom. After fixing the vertices, the controlled reconstruction procedure described below is applied.

2.3 Controlled Reconstruction

Our mechanism for model editing is based on prescribing different positions to a set of *control vertices* V_c. Given user prescribed positions for V_c the positions of the remaining vertices are computed by iterating over the following scheme:

1. For each vertex v, if v is not in V_c, update the position of v using the pyramid coordinates as described in Section 2.2.
2. For each control vertex v in V_c, compute the height offsets h_i (Equation 4) and update the positions of the neighbor vertices:
$$v_i = v'_i + n\frac{1}{m}\sum_{i=1}^{m} h_i.$$
This sets the distance between v and the projection plane P to the average of the h_i values.
3. Repeat until convergence.

The neighbor position update for control vertices in step 2 preserves the normal distance between each control vertex and its neighbors (Figure 1(c)). The tangential displacement of the control vertices is typically small, since it is reflected in the neighbor vertex placement. However, the normal distance has very minor influence on the neighbor placement and so we preserve it explicitly as described in step 2 above.

Two reconstruction examples are shown in Figure 2. Both reconstruct the triceratops model (Figure 2(a)) from the same initial guess. However the first model has 3 fixed control vertices, while the second has 6. For both models the initial positions (and normals) of the vertices were set to a projection of the original model to the unit sphere (Figure 2(b)). Figures 2(c) and (d) show the reconstructed models after 1000 iterations. The L_2 distances between the vertices of the original mesh and the reconstructed models are 3.6% and 1.6%, respectively (as a percentage of the bounding box diagonal). The geometric Laplacian norm [12] which measures local fidelity is 0.03% and 0.02%, respectively.

The controlled reconstruction procedure can be used for both morphing and deformation as described below.

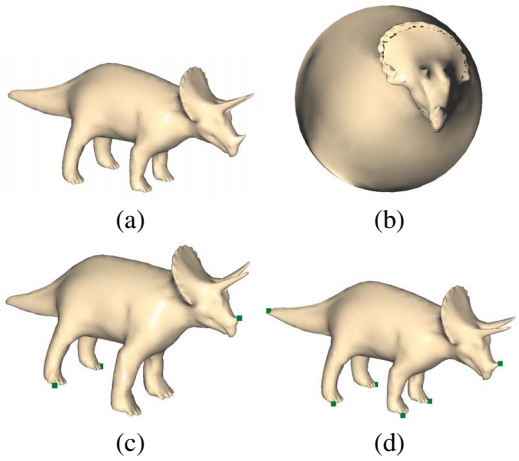


Figure 2: Geometry reconstruction (after 1000 iterations, 14 sec): (a) original model; (b) input model; (c) 3 control vertices; (d) 6 control vertices.

3 Morphing

A typical morphing algorithm consists of three main stages: computing a bijective mapping between the models; remeshing the models with a common connectivity; and computing the trajectory for each vertex in the mesh between its source and target positions. There are numerous algorithms for performing the mapping and the remeshing, as reviewed by Alexa [1]. This paper focuses on the final stage of the procedure, the trajectory computation. Our method generates intermediate models based on interpolated pyramid coordinates. It can also take into account user-prescribed trajectories of a number of control vertices. The prescribed trajectory of a control vertex is a curve in space-time defined over [0,1].

For simplicity, the description of the morphing algorithm is limited to two inputs. The extension to a larger number of inputs is straightforward. We refer to the input source and target meshes as S and T, respectively. The preprocessing stage of the algorithm computes the pyramid coordinates of S and T: projected face angles α^S and α^T, normal-edge angles β^S and β^T, and projected edge lengths l^S and l^T. For each time-frame, the algorithm computes the pyramid coordinates given the frame's time t (in [0,1]) by linearly interpolating between source and target values:

$$\begin{aligned} \alpha &= \alpha^S(1-t) + \alpha^T t, \\ \beta &= \beta^S(1-t) + \beta^T t, \\ l &= l^S(1-t) + l^T t. \end{aligned} \tag{6}$$

The control vertex positions are now set as defined by the trajectories. The remaining vertex positions are computed by the controlled reconstruction procedure using the

pyramid coordinates (Section 2.3). The initial guess for the placement is the vertex positions from the previous time-frame. Those positions are typically close to the expected positions in the current time step. Therefore, the reconstruction procedure converges very fast for each time-frame.

In global model blends a new model is a weighted average of the inputs. To blend two models, we apply the same procedure as for generating one time-frame in a morphing sequence, with different values of t per vertex, if necessary.

3.1 Local Blending and Morphing

For local morphs and blends, special care must be taken to generate a smooth transition between the blended/morphed regions and the rest of the model.

Blending: The input to the local blending procedure consists of two mesh models S and T (with the same connectivity), and a blend function B defined over the mesh vertices. $B(v)$ is 0 if the geometry at v should come from S, and 1 otherwise. To generate a seamless shape transition, we first smooth B using standard Laplacian smoothing. Now, the value of B at each vertex is used to assign the vertex coordinates. Instead of simply blending the 3D vertex coordinates, as in standard linear blending, we use the blending function to set the pyramid coordinates. For each vertex v and a neighbor vertex v_i, we set:

$$\begin{aligned}\alpha_i &= \alpha_i^S(1-B(v_i)) + \alpha_i^T B(v_i),\\ \beta_i &= \beta_i^S(1-B(v_i)) + \beta_i^T B(v_i),\\ l_i &= l_i^S(1-B(v_i)) + l_i^T B(v_i).\end{aligned}$$

Note that we use the blend function of the neighbor vertex to reflect its desired pyramid coordinates. Figure 3 compares the result of applying our blending procedure to linear blending and blending using Laplacian coordinates [10].

Morphing: For local morphing, we first use the local blending procedure above to redefine the target pyramid coordinates. We then use the redefined coordinates to generate the intermediate 3D meshes using the regular morphing procedure.

4 Deformation

Another important application of the controlled shape reconstruction procedure is model deformation. A major difference between morphing and deformation is that in the case of deformation the ideal pyramid coordinates for the constructed model are not known. It is clear that once a model is deformed, the original parameters cannot and should not be preserved in their entirety. Therefore, we apply the following procedure that adopts shape parameters as it proceeds, based on the deformation at hand. To do this, we define three sets of pyramid coordinates: current (α, β and l); source (α^S, β^S, and l^S); and target (α^T, β^T, and l^T). To define the deformation, we need to define two types of regions of influence around the control vertices: *geometric influence* regions and *shape influence* regions. Geometric influence regions determine whether the actual 3D position of the vertex changes. Shape influence regions define whether the local shape, i.e. the pyramid coordinates, of the vertex change. By definition, the geometric influence region contains the shape influence region. For most of our examples, the region sizes are identical.

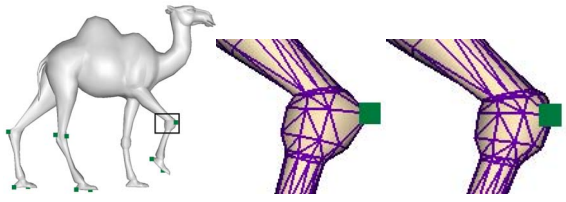
The steps of the deformation algorithm are:

1. Compute the pyramid coordinates of the input model. Set both the source and current coordinates to the computed coordinates.

2. Compute a shape influence function I for each vertex. Initialize $I(v)$ to 1 at the control vertices and 0 everywhere else. Perform a number of standard Laplacian smoothing iterations on the value of I. The region where $I(v) > 0$ defines the shape influence region. The number of smoothing iterations is based on the region size prescribed by the user.

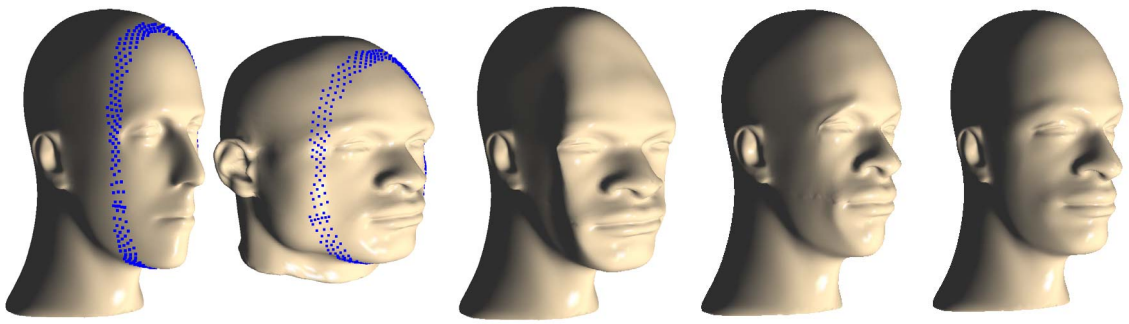
3. Set the geometric region of influence. Set the influence function $G(v)$ to 1 in the affected region and to 0 outside it. Typically, $G(v)$ is set to 1 if $I(v) > 0$ and to 0 otherwise.

4. Given the new locations of the control vertices, apply the controlled reconstruction procedure (Section 2.3), using the current pyramid coordinates α, β, and l. Do not recompute the coordinates for vertices outside the geometric region of influence (with $G(v) = 0$). After the reconstruction terminates, the typical result has a well shaped mesh everywhere except in the vicinity of the control vertices (Figure 5).



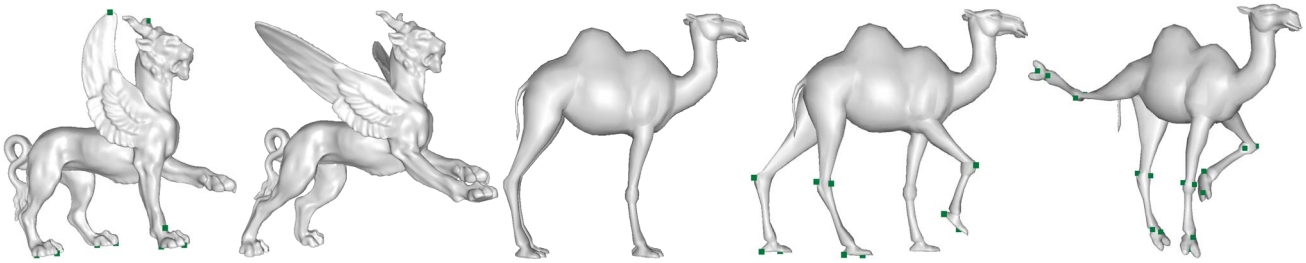
(a) (b) (c)

Figure 5: Bending the camel's knee (Figure 4 (d)) : (b) after one global iteration (using the original pyramid coordinates); (c) after three global iterations (including coordinate recomputation).



(a) (b) (c) (d) (e)

Figure 3: Face-off (transferring the face of (b) to head (a)): (a) & (b) Original models (dots indicates the boundary of the two blended regions). (c) Linear blend. (d) Blend using Laplacian coordinates. Note the discontinuity around the face, particularly noticeable on the brow and chin. (e) Blend using pyramid coordinates.



(a) (b) (c) (d) (e)

Figure 4: Deformations: (a),(b) feline; (c)-(e) camel; (d) walk; (e) ballet dance.

5. Compute the pyramid coordinates of the reconstructed mesh and assign them to the target parameters α^T, β^T, and l^T.
6. Recompute the current pyramid coordinates from the source and target coordinates based on the shape influence function. For a vertex v and a neighbor vertex v_i, set the corresponding pyramid coordinates of v to

$$\begin{aligned} \alpha_i &= \alpha_i^S(1-I(v_i)) + \alpha_i^T I(v_i), \\ \beta_i &= \beta_i^S(1-I(v_i)) + \beta_i^T I(v_i), \\ l_i &= l_i^S(1-I(v_i)) + l_i^T I(v_i). \end{aligned}$$

   Similarly to blending, we use the shape influence function of the neighbor vertex to reflect its desired pyramid coordinates.
7. Repeat the procedure starting from step 3. Typically, 2 to 3 iterations are sufficient to obtain visually pleasing results.

The deformation procedure provides very well-shaped and intuitive results. In the examples throughout the paper, despite the large deformations performed, the pyramid coordinates of the deformed models remain very close to those of the inputs (Table 1). This indicates that our deformation procedure preserves the local shape of the models.

5 Results

The figures throughout the paper demonstrate the results of applying our technique for different editing operations. To measure the level of shape preservation, we compute the L_2 distance between the pyramid coordinates (α, β, l) of the edited mesh and the optimal pyramid coordinates (α_O, β_O, l_O):

$$\begin{aligned} D_\alpha &= ((\sum(\alpha-\alpha_O)^2)/NumAlpha)^{1/2}/2\pi \\ D_\beta &= ((\sum(\beta-\beta_O)^2)/NumBeta)^{1/2}/\pi \\ D_l &= ((\sum(l-l_O)^2)/NumL)^{1/2}/BBoxDiag \end{aligned}$$

The scaling above is performed to normalize the distortion values to $[0,1]$ interval. For deformation, the original pyramid coordinates are viewed as optimal. For blending and morphing, we use the interpolated values as the optimal coordinates. The results for some of the examples in the paper are summarized in Table 1.

Figure 4 showcases our deformation algorithm. Both models undergo large deformations using only a small

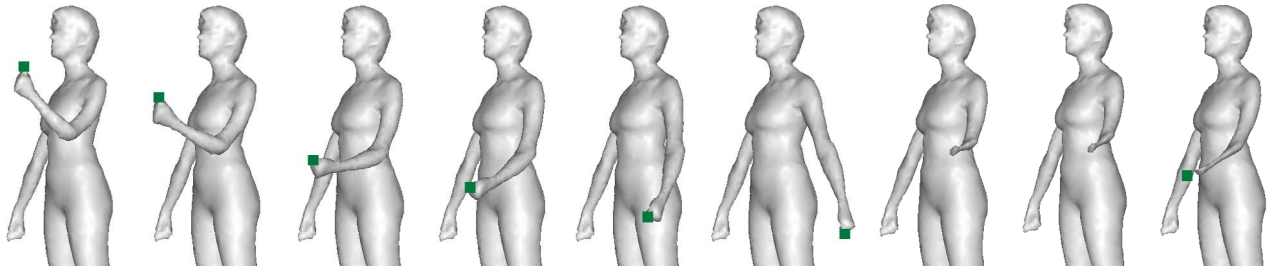


Figure 6: Morphing comparison. (a) Pyramid coordinates. The single control vertex follows a user prescribed trajectory. (b) Linear morphing (50%). (c) & (d) Morphing using Laplacian coordinates (50%) without (c) and with (d) trajectory specification.

number of control vertices. The resulting models closely preserve the shape of the original (Table 1). Figure 8 demonstrates the effect on deformation of changing the radius of the region of influence around a control vertex. As demonstrated by these examples, our method can handle significantly larger and more complex deformations than most previous techniques.

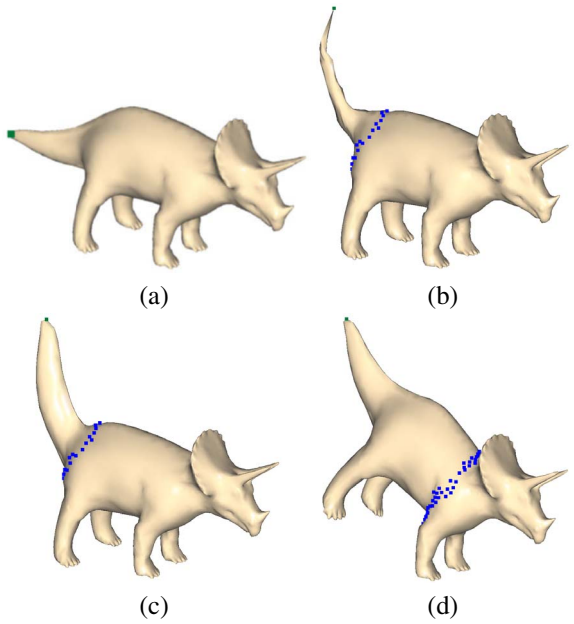


Figure 8: Deformation of the triceratops model (a) using one control vertex and different ROI sizes (indicated by the dots). In all the models the control vertices positions are identical. (b) Deformation using Laplacian coordinates. (c) Same deformation, using pyramid coordinates. (d) Deformation using pyramid coordinates with larger region of influence.

Figure 6 compares our morphing procedure to linear and Laplacian [10] morphs. Our method generates a natural trajectory for the entire arm, including smoothly folding it at the elbow. The shape and size of the arm are preserved throughout the entire morphing sequence. This is achieved using a prescribed trajectory for a single control vertex at the thumb. As expected, linear morphing performs poorly (Figure 6(b)). Without trajectory specification, Laplacian morphing produces very similar results to the linear morph ((Figure 6(c)). With a specified trajectory, Laplacian morphing performs somewhat better, but the arm remains drastically shrunken. Laplacian coordinates are not invariant under rotation and so they generate artifacts for rotational morphs such as the one in our example. Figure 7 shows a more complex morphing example where a cow is morphed into a bull. The morphing uses 8 trajectories defining the transition for the legs, tail, horns, and nose. Given the need to rotate those components, both linear and Laplacian morphing exhibit similar artifacts as in the female example (Figure 6).

Figure 3 shows an example of blending using pyramid coordinates. The result, Figure 3(e), is a seamless blend. On the same example, linear morph performs extremely poorly (Figure 3(c)). Laplacian coordinates blend performs somewhat better, but has a visible transition discontinuity between the blended parts (Figure 3(d)).

6 Conclusions

We introduce a new, robust method for mesh editing based on a novel local representation, the pyramid coordinates. Our representation captures the local shape properties of the mesh and is invariant under rigid transformations. Using the pyramid coordinates, we developed mesh editing operations which preserve the shape properties of the input models. As a result, our deformation method generates well shaped models even under extremely severe deformations. Our morphing procedure generates intermediate models which interpolate the shape properties of the input

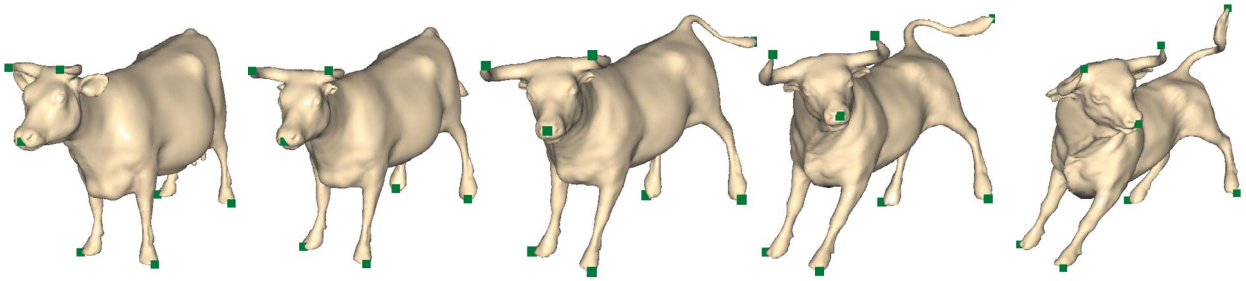


Figure 7: Turning a cow into a bull. Note the preservation of local details and the smooth rotation of the head and tail.

Model	Size (#vert.)	# control vertices	D_α	D_β	D_l	Runtime (sec)
Heads blend	3816	-	0.024	0.005	0.001	6
Feline	49,864	9	0.04	0.015	0.1%	161
Walking camel	4884	10	0.054	0.08	0.3%	3
Dancing camel	4884	15	0.095	0.067	0.5%	6
Female	8074	1	0.017	0.006	0.03%	–
Cow/bull	21,610	8	0.037	0.019	0.15%	–
Triceratops	7998	1				
(Pyramid coord.)			0.038	0.012	0.36%	0.3
(Laplacian coord.)			0.09	0.04	0.5%	

Table 1: Model statistics. For the morph examples (Figures 6 and 7) the statistics measure the difference between the 50% interpolation of the source and target pyramid coordinates and the actual coordinates of the morphed model at $t = .5$.

meshes. The procedure supports user prescribed control vertex trajectories. Using those trajectories, the method generates intuitively looking morphing sequences. All of the proposed editing operations require minimal user interaction. No segmentation of the model into patches, subdivision hierarchy construction, or any additional model information is required. The method is fast and simple to implement. These properties make pyramid coordinates based editing an effective tool for geometry processing and animation. Future research directions include speeding up the method using multiresolution approach and the application of pyramid coordinates to other editing operations such as pasting or high frequency detail transfer.

References

[1] M. Alexa, "Recent advances in mesh morphing," *Computer Graphics Forum*, vol. 21, no. 2, pp. 173–196, 2002.

[2] G. H. Bendels and R. Klein, "Mesh forging: editing of 3d-meshes using implicitly defined occluders," in *Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pp. 207–217, Eurographics Association, 2003.

[3] D. Zorin, P. Schröder, and W. Sweldens, "Interactive multiresolution mesh editing," *Computer Graphics*, vol. 31, no. Annual Conference Series, pp. 259–268, 1997.

[4] L. Kobbelt, J. Vorsatz, and H.-P. Seidel, "Multiresolution hierarchies on unstructured triangle meshes," *Computational Geometry*, vol. 14, no. 1-3, pp. 5–24, 1999.

[5] I. Guskov, W. Sweldens, and P. Schröder, "Multiresolution signal processing for meshes," in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pp. 325–334, ACM Press/Addison-Wesley Publishing Co., 1999.

[6] S. Yoshizawa, A. G. Belyaev, and H.-P. Seidel, "Free-form skeleton-driven mesh deformations," in *Proceedings of the eighth ACM symposium on Solid modeling and applications*, pp. 247–253, ACM Press, 2003.

[7] V. Surazhsky and C. Gotsman, "Morphing stick figures using optimized compatible triangulations," in *Proceedings of the 9th Pacific Conference on Computer Graphics and Applications*, p. 40, IEEE Computer Society, 2001.

[8] M. Alexa, D. Cohen-Or, and D. Levin, "As-rigid-as-possible shape interpolation," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pp. 157–164, ACM Press/Addison-Wesley Publishing Co., 2000.

[9] A. D. Gregory, A. State, M. C. Lin, D. Manocha, and M. A. Livingston, "Feature-based surface decomposition for polyhedral morphing," in *Proceedings of the fifteenth annual symposium on Computational geometry*, pp. 415–416, ACM Press, 1999.

[10] M. Alexa, "Local control for mesh morphing," in *Proceedings of the International Conference on Shape Modeling & Applications*, p. 209, IEEE Computer Society, 2001.

[11] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Roessl, and H.-P. Seidel, "Laplacian surface editing," in *Proceeding of the Eurographics/ACM SIGGRAPH symposium on geometry processing, to appear*, 2004.

[12] O. Sorkine, D. Cohen-Or, and S. Toledo, "High-pass quantization for mesh encoding," in *Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pp. 42–51, Eurographics Association, 2003.

[13] M. S. Floater, "Mean value coordinates," *Comput. Aided Geom. Des.*, vol. 20, no. 1, pp. 19–27, 2003.

[14] S. Yoshizawa, A. G. Belyaev, and H.-P. Seidel, "A fast and simple stretch-minimizing mesh parameterization," *Accepted for Shape Modeling International*, 2004.