The Visual Computer

Differential coordinates for local mesh morphing and deformation

Marc Alexa

Technische Universität Darmstadt, GRIS,
Rundeturmstr. 6, 64283 Darmstadt, Germany
E-mail: alexa@gris.informatik.tu-darmstadt.de

Published online: 14 February 2003

Mesh vertices are usually represented with absolute coordinates. In some applications, this leads to problems for local operations because of global misalignment. We investigate the idea of describing mesh geometry in a differential way. These differential coordinates describe local properties of the geometry rather than the absolute position in space. The main application discussed here is the insertion of shape features from one mesh into another, given the meshes have the same connectivity. We regard this as local control over mesh morphing. Differential coordinates also prove useful for free-form modeling of meshes.

Key words: Meshes – Morphing – Local frames – Free form deformation

1 Introduction

Meshes have become a widespread and popular representation of models in computer graphics. A fundamental modeling operation is to change the local shape of a mesh. Here, we consider two ways of altering the mesh geometry: first, mesh morphing adapted to modify the shape only locally; and second, free-form deformation.

Mesh morphing, typically, aims at gradually transforming one given mesh into another. In this work, we describe techniques to add space–time controls to the morph, so as to morph only parts of the meshes or different features at different instances of time.

Mesh morphing is commonly understood as a three step process [2]. The first two steps result in one mesh connectivity with two geometries attached to the vertices. Using this description it should be possible to change only parts of the geometry from source to target. This enables a user to insert features from one mesh into another.

Free-form deformation allows users to freely transform parts of a shape. We believe that an easy way to control the deformation is to specify only some scattered vertices, while the general shape of the mesh stays as close as possible to the original, however, given the additional constraints.

In both instances, absolute Euclidean coordinates cause problems. For local mesh morphing the problem is due to the fact that corresponding features might not have the same position in space and, thus, interpolation of absolute coordinates could lead to undesirable effects. This problem is illustrated in Fig. 1. The shapes on the left are the source and target geometry of one mesh. The idea is to locally change the geometry of the baby's face so that the nose takes the shape of the boy's. Locally interpolating vertex coordinates leads to the shape depicted middle right, which is clearly not usable. Note that the faces are overall aligned in space and that the misalignment of the noses results from different relative positions in the faces.

For free-form deformation, it would be desirable that coordinates capture the local shape rather than global position. This is because deformations typically change the global shape. If coordinates were local, they would not be affected by global deformations and, thus, they would be useful in solving the fitting problem posed by the user's contraints.

For these reasons, we propose differential coordinates, which describe the relation of vertices to their local neighborhood. This allows inserting shape features from one mesh into another or placing global

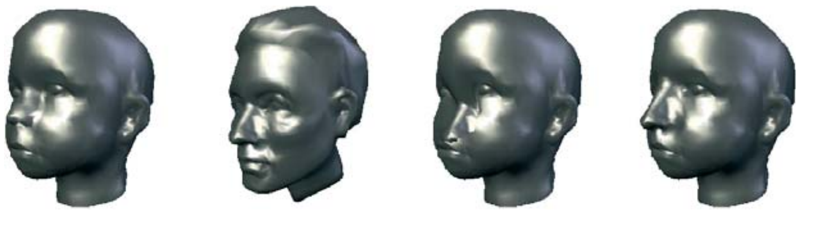


Fig. 1. Given a mesh with two geometries so that corresponding features (eyes, ears, nose, mouth, etc.) are represented by the same vertices, if one feature (in this example the nose) is morphed towards the target geometry in absolute coordinates, different positions in space lead to undesirable effects. The shape on the right shows a more pleasing result achieved with the methods presented here

constraints on some mesh vertices while simply fitting the differential coordinates.

2 Local mesh morphing framework

A mesh $\mathcal{M}$ is described by the connectivity K of its vertices, edges, and faces and geometric positions V of the vertices. The geometric realization ϕ_V of the mesh maps $\{V, K\}$ to the set of all points comprising the mesh and, thus, describes the shape of the object represented.
Given two meshes $\mathcal{M}_0$ and $\mathcal{M}_1$ the goal of classical mesh morphing is to produce a family of meshes $\mathcal{M}(t), t \in [0, 1]$ so that $\phi(\mathcal{M}_0) = \phi(\mathcal{M}(0))$ and $\phi(\mathcal{M}_1) = \phi(\mathcal{M}(1))$. The general idea for achieving this goal is to generate one mesh topology that can be deformed to the shapes of the source and target shapes: $\mathcal{M}(t) = \{V(t), K\}$. To generate this family of shapes, three steps are necessary:

1. Finding a correspondence between the meshes. More specifically, computing maps ψ_0, ψ_1 so that $\psi_0(V_0) \in \phi(\mathcal{M}_1)$ and $\psi_1(V_1) \in \phi(\mathcal{M}_0)$, yielding a barycentric coordinate for each vertex with respect to a simplex in the other mesh. This step is typically performed by using 2D parameterizations of the meshes. Particularly important is the alignment of automatically detected or user-specified features of the meshes.
2. Generating a new, consistent mesh topology K together with two geometric positions $V(0), V(1)$ for each vertex so that the shapes of the original meshes are reproduced. This is typically done so that both vertex sets are contained in the new vertex set, both edge sets are contained in the new edge set, and new vertices are generated where edges from different meshes cross.
3. For classical mesh morphing, the goal in this step is to create paths $V(t), t \in]0, 1[$ for the vertices. In our context, however, we intend to compute vertex positions from the original geometric positions $V(0), V(1)$, where t varies among the vertices.

In the following, we briefly discuss each of the above-mentioned steps. For the first two, recent work is reported; for the third, basic approaches are explained. A more detailed state-of-the-art approach can be found in [2].

2.1 Correspondence of shapes

Finding a 2D parameterization of the vertex-edge graphs of the input meshes results in a correspondence between the surfaces. We distinguish the parameterization of genus 0 meshes in a suitable parameter domain and the isomorphic dissection of arbitrary meshes into patches homeomorphic to a disk, which are parameterized independently in the plane.
Several techniques for computing the paramterizations of topological disks [8, 9, 31] and topological spheres [1, 5, 14, 18, 20, 28] exist. Embedding topological disks is important for morphing techniques that generate isomorphic dissections of several meshes [4, 7, 12, 15, 21, 32] with the help of the user.
Feature alignment is important to yield pleasing morphing results. Features could be either selected by the user or shape features (such as curvature and normals). Specifically, most dissection methods force the user to select corresponding features on

the shapes and allow for fine-grained control. Point-to-point correspondence can also be achieved with warping methods [1, 6, 32].

2.2 Representation mesh

Given the correspondence information generated in the first step, a mesh is to be produced that contains vertices, edges, and faces from both meshes. This is done by either overlaying the meshes in the parameter domain or by generating a multiresolution representation.

If the parameter domain is the plane, the overlay problem is known as planar map overlay, and several algorithms from the computational geometry literature are applicable. For non-planar parameter domains, works on morphing offer special solutions [1, 18].

Multiresolution models (e.g. [22]) have recently been advocated in the literature [21, 25] for the generation of morphable models. The idea is to remesh the shape starting from an irregular coarse base mesh with regular connectivity. Thus, the mesh topology is given by the refinement operator while the geometry is given by the original meshes.

2.3 Classical vertex paths

After the computation of one mesh topology K and two mesh geometries represented by vertex coordinates $V(0)$ and $V(1)$, it remains to compute vertex coordinates for the intermediate shapes. For a global transformation, a set of vertex coordinates $V(t), t \in]0, 1[$ has to be generated, where t is mentally connected to time and sometimes called *transition parameter*.

A simple choice is linear interpolation [12, 18, 32]. A rigid [5, 6] or affine [1] transform prior to linear vertex interpolation yields better results.

A more general solution is to interpolate an intrinsic description of the boundary. This approach has been successfully applied to polygons [26]. However, the extension to meshes is difficult [29].

Skeleton-based shape interpolation suggests that reasonable vertex paths cannot be found by just analyzing the boundary of a shape [27]. However, the extension to 3D is yet to come. Another way of taking the interior of a shape into account is to generate isomorphic simplicial complexes of the shapes and compose optimal simplex morphs to yield vertex paths [3].

2.4 Spatially non-uniform transition states

In this work, we introduce the idea of local morph control to mesh morphing. Local morph control is well known in image morphing as transition control. Defining the transition of each pixel can be done very similarly to warping the images; for example, the transition could be specified for only few points and scattered-data interpolation techniques are used to calculate transition information for all points of the images [23].

Local control requires describing the transition with more than a single scalar t. A general solution for the mesh setting is to assign a transition parameter to each vertex. We call the vector of per-vertex transition parameters *transition state* T. If the per-vertex transition parameter is applied to a vertex representation, T has to be a diagonal matrix containing the parameters as diagonal elements.

The idea of the transition state is to apply each transition parameter to the vertex representation independently. Linear interpolation of absolute vertex coordinates would be represented as $(I - T)V(0) + TV(1)$, yet this would lead to the problems already mentioned and depicted in Fig. 1. From the techniques for path generation only [3] seems to generate a shape representation suited for spatially non-uniform interpolation (namely, affine transforms per simplex). However, this approach tends to be numerically difficult to handle for large meshes.

For the special case of implementing one feature from one mesh into another, Kanai et al. [16] attach an affine transform to the feature. However, if more than one feature is present and some features overlap, this is not possible anymore. Here, we want the shape to be completely defined by the transition state (and not by the definition of a feature), as this might be an interesting description for shapes.

2.5 More than two meshes

The conceptual extension of the framework to more shapes is straightforward. Given meshes $\mathcal{M}_i = (V_i, K_i)$, a common topology K together with vertex sets $V(\mathbf{e_i})$ is established. The vertex sets form a base of a space, which is reflected by using canonical base vectors $\mathbf{e_i}$ as indices. A morphed shape $(V(\mathbf{s}), K)$ is represented by a vector $\mathbf{s} = (s_0, s_1, \dots)$, reflecting the shares of the meshes $\mathcal{M}_0, M_1, \dots$

Not all techniques presented in this framework are equally suited to being extended to more meshes. The correspondence procedure discussed in Sect. 2.1 seems to be relatively easy to extend. All meshes are embedded in the given parameter domain, which leads to barycentric representation of the original vertices. If each set of original vertices V_i needs to be mapped to all other meshes $\mathcal{M}_j, i \neq j$, the complexity would grow quadratically with the number of meshes. However, this is not necessary if a remeshing strategy is used to generate a consistent mesh topology (see Sect. 2.2, last paragraph). This procedure generates the same set of vertices over all shapes, thus, the complexity is linear to the number of meshes times the number of vertices used in the remesh, which is the best we can expect. So, the best way to generate the set $\{(V(\mathbf{e_i}), K\}$ is to embed all meshes in a common parameter domain (spherical or piecewise linear) and then remesh to the desired accuracy.

The transition state description now extends to compute combinations of several vertex vectors. Global linear vertex combination is easily extended:

$$V(\mathbf{s}) = \sum_i s_i V(\mathbf{e_i}) \tag{1}$$

The extension of transition states, where each vertex has its own transition parameter, leads to a vector of matrices; that is, each scalar s_i is replaced by a matrix S_i.

3 Mesh deformation approach

Mesh deformation, as it is viewed here, is simply another point of view of the framework described above. For local morphing, intrinsic constraints should be used; while for free-form deformation, we use explicit constraints.

More specifically, we wish the user to specify the absolute position of several mesh vertices. We use the differential representation of the mesh to compute a global least-squares fit constrained to the user specified vertices.

Current mesh deformation techniques typically use multiresolution modeling techniques [8, 13, 19, 24, 33]. The current level controls the effect of modifying the position of a particular vertex. Modifications on a coarse level affect large regions of the shape; while modifications on a detailed level change the shape only locally.

In our approach, the region of influence is implicitly defined by the set of constrained vertices. Each constraint is exactly satisfied, which means that the influenced region could be easily defined as the set of non-constrained vertices.

4 Differential representation of a mesh

The main idea of this work is to represent vertex coordinates with respect to their neighbors in the mesh. Given a vertex and a neighborhood $\mathcal{N}(i)$, the position should be described relative to the positions of vertices in this neighborhood. Here, we mostly consider the neighborhood $\mathcal{N}(i)$ to be a one-neighborhood ring, i.e. $\mathcal{N}(i) = \{j|(i, j) \in K\}$ (see Fig. 2). Nevertheless, the techniques presented are generally applicable to any shape of sufficiently large neighborhood.

Further, the representation of a vertex should be linear in the absolute coordinates. This is because both the transformation from absolute to relative coordinates, and vice versa, should be numerically stable to compute. A linear mapping leads to solution of linear systems for each of the transforms, which is, compared to other possibilities, the simplest possible solution.

The relative representation aims at making the shape of the mesh invariant to translation or, ideally, invariant under affine transforms. If a vertex were represented in the affine space of its neighbors, invariance under affine transforms would trivially follow. Floater and Gotsman have shown how to use such representations to morph planar triangulations [10]. The extension to triangulations in $\mathbb{R}^3$ is difficult because vertices of the neighborhood are not necessarily affinely independent in $\mathbb{R}^3$. We introduce a translation-invariant scheme and briefly elaborate on the affinely independent scheme in the following sections.

4.1 Laplacian representation

A simple scheme that achieves invariance under translation works as follows: Let $\mathbf{v_i}$ be the vertex position to be represented. Compute the center of mass of the neighbors

$$\bar{\mathbf{v}}_\mathbf{i} = \sum_{j \in \mathcal{N}(i)} \frac{1}{|\mathcal{N}(i)|} \mathbf{v_j}, \tag{2}$$

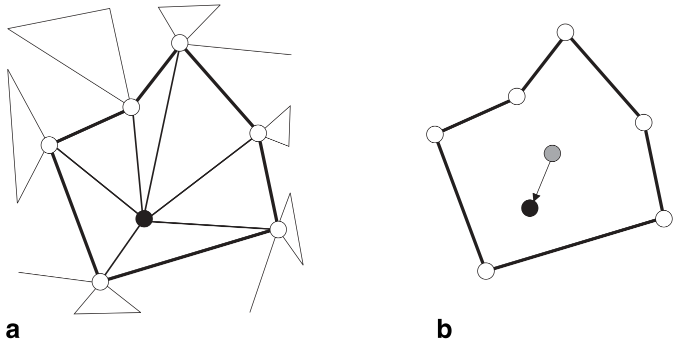


Fig. 2. A vertex (*black*) and its neighborhood ring. In Laplacian coordinates a vertex is represented by the difference to the centroid of its neighbors

and let the new representation be the difference of this center of mass to the original position:

$$\mathbf{w_i} = \mathbf{v_i} - \bar{\mathbf{v}}_\mathbf{i}. \tag{3}$$

For an illustration assuming $\mathcal{N}(i)$ to be a one-neighborhood see Fig. 2. Note that this representation is robust in the sense that it does not suffer from degeneracies in the input points. In particular, the neighbors of $\mathbf{v_i}$ might be in non-convex positions, coplanar, collinear, or even collapsed to one point. Their center of mass is always defined and, thus, the new representation.

If we write all vertices as a vector, the forward transformation (from absolute to relative coordinates) can be represented in matrix form. Let A be the adjacency matrix of the mesh and D be a diagonal matrix with $d_{ii} = 1/|\mathcal{N}(i)|$. The transform is then represented by $L = I - DA$. Note that L is a Laplacian of the mesh [30]. This is an important observation as it generalizes the approach to shape representations other than meshes, such as parametric or implicit functions.

The backward transformation (from relative to absolute coordinates) is, by construction, not unique. It should be uniquely determined up to a translation. This means, $L \in \mathbb{R}^{m \times m}$ should have rank $m - 1$, which is indeed so. Note that DA is a stochastic as well as a normal matrix. A stochastic matrix has an eigenvalue of 1. In addition, the eigenvectors of normal matrices form a basis of full rank, meaning that all eigenvalues have multiplicity 1. It follows that DA has exactly one eigenvalue of 1 and, thus, L has exactly one eigenvalue of 0.

4.2 Affine independent representation

Representing a vertex $\mathbf{v_i}$ with respect to its neighborhood $\mathcal{N}(i)$ so that the representation is invariant under affine transforms is straightforward. We have to solve

$$\mathbf{v_i} = \sum_{j \in \mathcal{N}(i)} w_i(j) \mathbf{v_j}, \tag{4}$$

so that

$$\sum_{j \in \mathcal{N}(i)} w_i(j) = 1. \tag{5}$$

This could be rewritten into one linear system using homogenous coordinates. The system is usually overdetermined, so we might additionally try to minimize the sum of squared weights $w_i(j)$. Nevertheless, the vertices in $\mathcal{N}(i)$ might not form a basis of $\mathbb{R}^3$; in which case, no solution exists. A simple way to solve the system in any case is SVD [11]. This would automatically minimize the squares of the weights in the overdetermined case, as well as providing the best possible answer in the case where no exact solution exists. In this way, one could define the forward transformation.

However, if one neighborhood $\mathcal{N}(i)$ is not a base of $\mathbb{R}^3$, this transform loses information about the shape. In this case the necessary displacement could be added using a Laplacian coordinate. We have experimented with this idea and found it numerically difficult to solve. The problem is to differentiate clearly between cases of degeneration and usable configurations. The problem is delicate because local degradation has global effects.

For this reason, we decided to use the simpler but more stable Laplacian scheme, which is not invariant under rotation, scaling, and shearing. Yet, it proved sufficient for the applications intended here.

4.3 Solving for absolute coordinates

Solving the equation $LV = W$ for V is not possible in a naive way for two reasons. First, L is singular; and second, typical meshes will induce matrix dimensions that make the use of explicit techniques prohibitive.
The first problem is easy to overcome. It was already shown that the solution is specified up to a translation. This means fixing one arbitrary vertex will lead to a linear system of equations with full rank.
A practical solution of the resulting linear system should account for the following conditions:

- L is very large and sparse, which prohibits explicit matrix techniques.
- The equation $LV = W$ has to be solved three times, i.e. for the x, y, and z vectors.
- In practice, good approximate solutions are known for V, as morphing changes the shape gradually and smoothly from one state to another. Knowing good approximate solutions calls for iterative matrix methods.
- For free-form modeling, a least-squares fit is required.

Iterative schemes like Jacobi or Gauss–Seidel iteration have the advantage that they are easily implemented on the existing mesh data structures. In addition, they also work stably in the case of least-squares fitting.

5 Application to mesh morphing

The main idea of this work is to morph by linearly interpolating Laplacian coordinates rather than absolute coordinates. Since Laplacian coordinates are linear in absolute coordinates, morphing the whole shape (i.e. all vertices have the same transition parameter) will be the same in absolute and Laplacian coordinates. Yet, if the desired transitions are different for subsets of vertices, interpolating Laplacian coordinates yields more reasonable results.
More precisely, the approach works as follows. Given a mesh topology K and several geometries V_i. The topology defines a forward transform L which allows computing Laplacian coordinates W_i for each geometry. A transition state described by a set of transition state matrices T_i defines the transformed geometry V' by combining the Laplacian coordinates according to the transition state and then solving for absolute coordinates:

$$V' = L^{-1}\left(\sum_i T_i W_i\right). \tag{6}$$

In the following, we apply this approach to several examples, where morphable meshes have been generated with the techniques described in [1]. For simplicity in the following discussion, we assume to morph only between two shapes, so a single matrix T is enough to describe the factors $T_0 = (I - T)$ and $T_1 = T$. A morph, or transition, would be defined as matrices T changing over time. In practice, one would define several key frames in terms of matrices. Interpolation of key frames is done in matrix space as interpolation in absolute coordinates has to be avoided.
Obviously, it is impossible to specify transition matrices by hand. We either need a user interface or automatic methods.

5.1 GUI for defining transition states

The basic requirement for a GUI is to make it easy to define a region of interest (ROI). A ROI is a part of the shape's boundary, for which the transition state will be modified. Modification is performed relative to the current overall transition state of the shape.
We have found it very straightforward to specify a ROI by two rings of mesh edges, where the outer ring bounds a region which contains the inner ring. These two rings divide the mesh into three parts. The region inside the inner ring is subject to the modifications defined by the user, while the region outside the outer ring stays unchanged. The region between the rings smoothly adapts between changed and unchanged region and could be empty.
The boundaries could be chosen by "drawing" on the shape. In our implementation, we allow the user to pick a vertex as the center of interest and to specify two radii which define the rings. The rings are found as sets of vertices with the specified distance using Dijktstra's algorithm starting from the center of interest.
To quantify the necessary modifications to the per-vertex transition parameters, we define a distance

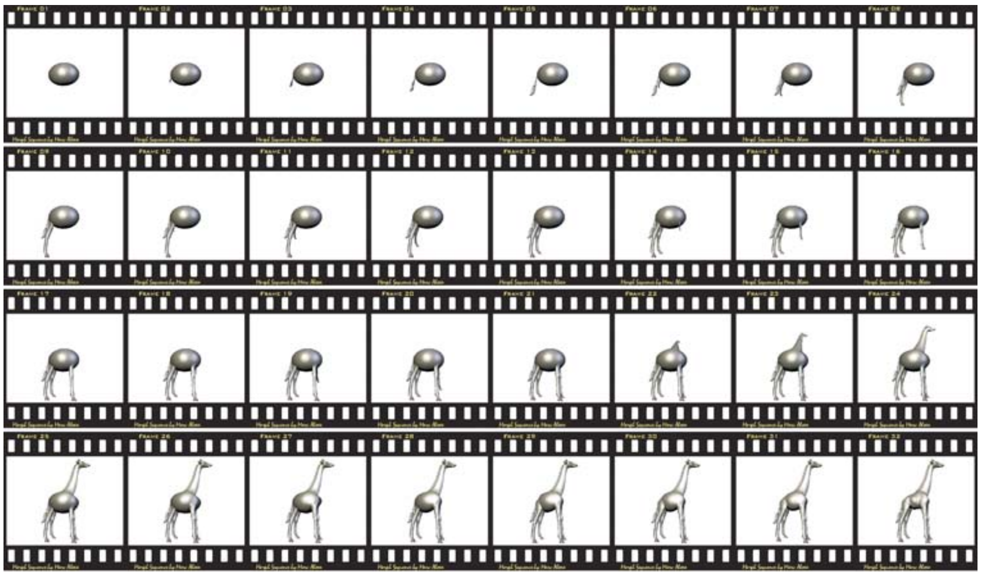


Fig. 3. A morph sequence from an egg to a model of a giraffe generated from several transition states. Transitions states were defined using ROIs, where each ROI corresponds to either the tail, one of the legs, the neck including the head, or the body. Eight transition states were defined, letting the different parts of the body pop out one after the other

value d, which is 0 inside the inner boundary, 1 outside the outer boundary, and represents the distance from the boundaries between them. Again, Dijkstra's algorithm is used to compute the distances.

Assume the user specifies a new transition state $\tilde{T}$ for the ROI. The new overall transition state T' is defined based on the current transition state T as

$$T' = (1-d)\tilde{T} + dT. \tag{7}$$

In a typical modification process, the user selects several ROIs and changes their respective transition states one after the other.

This technique was used for generating a morph sequence from the shape of an egg to the shape of a giraffe (see Fig. 3). The idea was to let parts of the body pop out of the egg one after the other. This was achieved by defining ROIs for the different parts of the body and using them to specify transition states. A smooth transition sequence was produce by interpolating the key transition states.

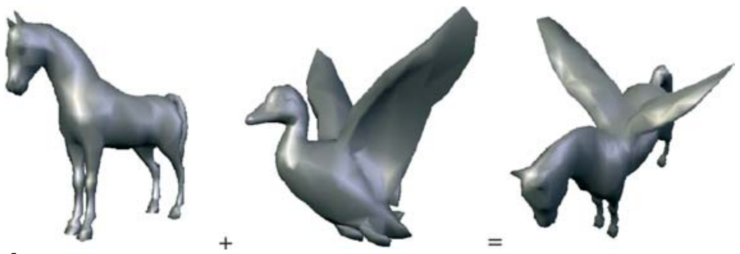
An explicit example of modeling using spatial morph control is depicted in Fig. 4. The intent was to model Pegasus, i.e. a horse with wings. One easily finds polyhedral models of horses and animals with wings. Using ROIs around the wings it is easy to define a transition state which yields the desired result.

In these examples, morphing is performed between rather different shapes. An interesting application arises from locally morphing among different version of the same shape. In particular, think of different versions representing only parts of the spectrum of the mesh. The eigenvectors of the Laplacian L form a basis, which is the equivalent to a Fourier basis of a mesh [17]. This basis could be exploited to define several band-limited versions of the shape. Locally morphing among these shapes has the effect of a local spectral filter.

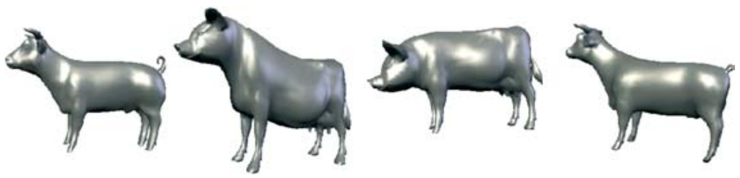
5.2 Spatially dependent transition states

Instead of defining the transition state of the shape explicitly using ROIs, the transition state could be defined implicitly. In particular, the transition state might be a function of absolute coordinates. This makes several interesting and important effects possible, e.g. a plane cutting the shape into parts with different transition states.

The definition of a transition state from absolute coordinates would involve a function that maps points



4



5

Fig. 4. Using spatial control for modeling. Pegasus is modeled by morphing between models which comprise its features. Local morph control is used to define a transition state so that only the wings of the duck appear on the horse
Fig. 5. Defining transition states using spatial constraints. Here, a plane is used to split the shape into two parts corresponding to one of the two source shapes

in $\mathbb{R}^3$ to weights defining how to combine Laplacian coordinates in this point. However, using Laplacian coordinates, the global positioning of the shapes is lost. We, therefore, define that the source shapes as well as the generated shape have their centers of mass at the origin of $\mathbb{R}^3$. Nevertheless, in the general case, the system of equations to be solved will be non-linear.

We resort to a simple heuristic to define a transition state. The absolute coordinates of vertices for each of the source shapes define a transition state. These transition states are averaged to yield the final transition state.

This definition has the advantage that smooth changes of the spatial distribution lead to smooth changes in the shape. For example, assume the user wants to move a plane through a shape so that the two parts correspond to two source shapes. The heuristic given above assures that moving the plane will grow one region and shrink the other, as desired.

Several examples of defining transition states from absolute coordinates are shown in Fig. 5. A morphable model of a cow/pig is cut by a plane in two positions. Rather than a sequence, several models are presented showing that spatial morph control could be used for modeling. One can imagine how many different creatures could be modeled with virtually no effort using this approach.

5.3 Automatic transition sequences

The basic idea is to start the transition at one or several points of the shape and then to "grow" regions representing the target shape until the shape is "covered". In this process, the graph representing the mesh can be exploited.

Nice effects result from "flooding" the mesh, i.e. traversing the graph breadth-first and setting each visited vertex to its target coordinate.

Spatial effects can be achieved as was described in the previous section.

6 Application to free-form modeling

As has been mentioned, the idea for free-form modeling using differential coordinates is to least-squares fit the original representation given a set of constraints in terms of absolute vertex coordinates. Given a mesh (K, V), this defines the forward transform L and, thus, the Laplacian representation W. We assume the user constrains a set of vertices $\mathbf{v_i}$.

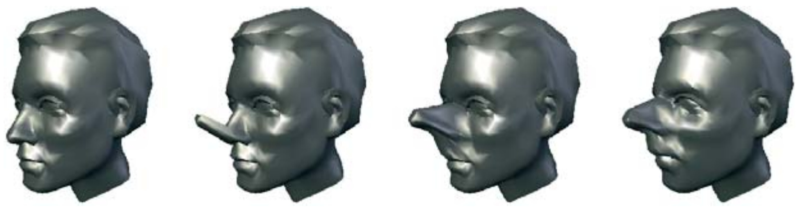


Fig. 6. Using Laplacian coordinates for free-form modeling. The model of a boy's face is deformed. First, just the tip of the nose is displaced. By defining a set of free vertices, which are relaxed to least-squares fit their Laplacian coordinates, different parts of the face could be affected by the displacement

The idea of a least-squares fit V' is to minimize the squares of actual Laplacian coordinates; that is,

$$(LV' - W)^2 \tag{8}$$

is minimized, which is computed by setting the gradient over the free vertices to 0. This leads to a linear system in the free vertices. In practice, we use, again, a relaxation method to solve for the solution vector, however, with the constrained vertices unchanged.
In order to specify fixed vertices, we found it convenient to use ROIs, again defined by an inner and outer ring. Vertices outside the outer ring are fixed to their original position. Vertices inside the inner boundary can be moved by the user. The remaining vertices are free and will be computed using the linear least-squares approach.
An examples is depicted in Fig. 6. In the face of a young boy the nose tip is displaced. The remaining vertices are relaxed to approximately fit their Laplacian coordinates using varying sets of free vertices to achieve different effects.

7 Conclusions

We have demonstrated the use of differential mesh coordinates. The representation eases local mesh modeling tasks. Specifically, Laplacian coordinates are introduced, which seem to be well suited for the task of modeling. The advantages of Laplacian coordinates are that they are independent of transformations of the shape and are rigorously defined (i.e. tolerate degeneracies in the mesh).
However, Laplacian coordinates are sensitive to scaling and rotation of the shape. This could be a problem if corresponding regions of shapes have different size or orientation despite the fact that shapes are overall reasonably aligned. A representation of coordinates as an affine sum of neighboring vertices would be insensitive to affine transforms, however, has be proven to be numerically difficult to handle. In the future, we will investigate this problem and try devise an affine invariant representation scheme.

Acknowledgements. Thanks to Daniel Cohen-Or, Craig Gotsman, Jarek Rossignac, Luis Velho for their stimulating work and helpful discussions.

References

1. Alexa M (2000) Merging polyhedral shapes with scattered features. Vis Comput 16(1):26–37
2. Alexa M (2002) Recent advances in mesh morphing. Comput Graph Forum 21(2):173–196
3. Alexa M, Cohen-Or D, Levin D (2000) As-rigid-as-possible shape interpolation. In: Proceedings of SIGGRAPH 2000. ACM SIGGRAPH, New York, pp 157–164
4. Bao H, Peng Q (1998) Interactive 3D morphing. Comput Graph Forum 17(3):23–30
5. Cohen-Or D, Carmel E (1998) Warp-guided object-space morphing. Vis Comput 13(9-10):465–478
6. Cohen-Or D, Solomovici A, Levin D (1998) Three-dimensional distance field metamorphosis. ACM Trans Graph 17(2):116–141
7. DeCarlo D, Gallier J (1996) Topological evolution of surfaces. In: Proceddings of Graphics Interface '96, Canadian Information Processing Society/Canadian Human-Computer Communications Society, pp 194–203
8. Eck M, DeRose T, Duchamp T, Hoppe H, Lounsbery M, Stuetzle W (1995) Multiresolution analysis of arbitrary meshes. In: Proceedings of SIGGRAPH 95. ACM SIGGRAPH, New York, pp 173–182
9. Floater MS (1997) Parametrization and smooth approximation of surface triangulations. Comput Aided Geom Des 14(3):231–250
10. Floater MS, Gotsman C (1999) How to morph tilings injectively. J Comput Appl Math 101:117–129

11. Golub GH, Van Loan CF (1989) Matrix computations, 2nd ed. Johns Hopkins series in the mathematical sciences, vol 3. Johns Hopkins University Press, Baltimore, Md.
12. Gregory A, State A, Lin M, Manocha D, Livingston M (1998) Feature-based surface decomposition for correspondence and morphing between polyhedra. In: Computer Animation '98, pp 64–71
13. Guskov I, Sweldens W, Schröder P (1999) Multiresolution signal processing for meshes. In: Proceedings of SIGGRAPH 99. ACM SIGGRAPH, New York, pp 325–334
14. Kanai T, Suzuki H, Kimura F (1998) Three-dimensional geometric metamorphosis based on harmonic maps. Vis Comput 14(4):166–176
15. Kanai T, Suzuki H, Kimura F Metamorphosis of arbitrary triangular meshes. IEEE Comput Graph Appl 20(2):62–75
16. Kanai T, Suzuki H, Mitani J, Kimura F (1999) Interactive mesh fusion based on local 3D metamorphosis. In: Graphics Interface '99, Canadian Information Processing Society/Canadian Human-Computer Communications Society, pp 148–156
17. Karni Z, Gotsman C (2000) Spectral compression of mesh geometry. In: Proceedings of SIGGRAPH 2000. ACM SIGGRAPH, New York, pp 279–286
18. Kent JR, Carlson WE, Parent RE (1992) Shape transformation for polyhedral objects. Comput Graph (Proceedings of SIGGRAPH 92) 26(2):47–54
19. Kobbelt L, Campagna S, Vorsatz J, Seidel H-P (1998) Interactive multi-resolution modeling on arbitrary meshes. In: Proceedings of SIGGRAPH 98. ACM SIGGRAPH, New York, pp 105–114
20. Lazarus F, Verroust A (1997) Metamorphosis of cylinder-like objects. J Vis Comput Anim 8(3):131–146
21. Lee A, Dobkin D, Sweldens W, Schröder P (1999) Multiresolution mesh morphing. In: Proceedings of SIGGRAPH 99. ACM SIGGRAPH, New York, pp 343–350
22. Lee AWF, Sweldens W, Schröder P, Cowsar L, Dobkin D (1998) Maps: multiresolution adaptive parameterization of surfaces. In: Proceedings of SIGGRAPH 98. ACM SIGGRAPH, New York, pp 95–104
23. Lee S-Y, Chwa K-Y, Shin SY, Wolberg G (1995) Image metamorphosis using snakes and free-form deformations. In: Proceedings of SIGGRAPH 95. ACM SIGGRAPH, New York, pp 439–448
24. Lounsbery M, DeRose TD, Warren J (1997) Multiresolution analysis for surfaces of arbitrary topological type. ACM Trans Graph 16(1):34–73
25. Praun E, Sweldens W, Schröder P (2001) Consistent mesh parameterizations. In: Proceedings of SIGGRAPH 2001. ACM SIGGRAPH, New York, pp 179–184
26. Sederberg TW, Gao P, Wang G, Mu H (1993) 2D shape blending: An intrinsic solution to the vertex path problem. In: Proceedings of SIGGRAPH 93. ACM SIGGRAPH, New York, pp 15–18
27. Shapira M, Rappoport A (1995) Shape blending using the star-skeleton representation. IEEE Comput Graph Appl 15(2):44–50
28. Shapiro A, Tal A (1998) Polyhedron realization for shape transformation. Vis Comput 14(8-9):429–444
29. Sun YM, Wang W, Chin FYL (1997) Interpolating polyhedral models using intrinsic shape parameters. J Vis Comput Anim 8(2):81–96
30. Taubin G (1995) A signal processing approach to fair surface design. In: Proceedings of SIGGRAPH 95. ACM SIGGRAPH, New York, pp 351–358
31. Tutte WT (1963) How to draw a graph. Proc Lond Math Soc 13:743–768
32. Zöckler M, Stalling D, Hege H-C (2000) Fast and intuitive generation of geometric shape transitions. Vis Comput 16(5):241–253
33. Zorin D, Schröder P, Sweldens W (1997) Interactive multiresolution mesh editing. In: Proceedings of SIGGRAPH 97. ACM SIGGRAPH, New York, pp 259–268

MARC ALEXA leads the project group 3D graphics computing within GRIS, Technische Universität Darmstadt. He received his MS and PhD degrees in computer science with honors from TU Darmstadt. His research interests include shape modeling, transformation, and animation, as well as conversational user interfaces and information visualization.