COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds 2005; **16:** 499–508
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cav.92

Animating Geometrical Models

Mesh morphing using polycube-based cross-parameterization

By Zhengwen Fan, Xiaogang Jin, Jieqing Feng and Hanqiu Sun*

In this paper, we propose a novel mesh morphing approach based on polycubic cross-parameterization. We compose parameterizations over the surfaces of the polycubes whose shape is similar to that of the given meshes. Because the polycubes capture the large-scale features, we can easily preserve the shape of the models, mapping legs to legs, head to head, and so on. For the finer features that are not reflected by the shape of the polycubes, we split the polycubes into matching patches and optimize them to get a low-distortion bijection that satisfies user-prescribed constraints. Our approach works well for meshes with arbitrary genus as long as the polycubes capture this feature and transfers texture seamlessly. We can also build maps with singularities between models with different genus.
Copyright © 2005 John Wiley & Sons, Ltd.

KEY WORDS: mesh morphing; cross-parameterization; polycube maps; texture transfer

Introduction

Mesh morphing,[1] which transforms one mesh into another smoothly, has been widely applied to various aspects of computer graphics industry. To generate a pleasing morph sequence, a correspondence (or a cross-parameterization, i.e. a one-to-one and onto mapping between two surfaces)[2] between the models is usually required, which is typically done by finding a common parameter domain for the surfaces. The cross-parameterization of two surface meshes is a fundamental step in many other geometry processing algorithms, such as mesh editing,[3] shape blending, deformation transferring and surface detail transferring (textures, normal maps, etc).[4,5] To compose the correspondence map between the given models, the user may specify corresponding features on both meshes, and the cross-parameterization should preserve these constraints.

*Correspondence to: Xiaogang Jin, State Key Lab of CAD&CG, Zhejiang University, Hangzhou, 310027, P.R. China.
E-mail: jin@cad.zju.edu.cn

Contract/grant sponsor: 973 Program; contract/grant number: 2002CB312101.
Contract/grant sponsor: National Natural Science Foundation of China; contract/grant numbers: 60273054 and 60340440422.
Contract/grant sponsor: Fok Ying Tung Education Foundation; contract/grant number: 91069.
Contract/grant sponsor: Specialized Research Fund for the Doctoral Program of Higher Education; contract/grant number: 20020335070.

Related Work

Planar Parameterization

The traditional parameter domain is a planar region. Representing an entire surface requires that it be cut into one or more disk-like charts, where each chart is parameterized independently. Planar parameterization for 3D meshes has received a lot of attention over the past several years.[6]

Several researchers introduce methods which satisfy positional constraints. Levy suggests a method to incorporate soft positional constraints into the formulation of the parameterization problem.[7] Desbrun *et al.* use Lagrange multipliers to add 'hard' constraints to the parameterization formulation.[8] Kraevoy *et al.* present a robust planar solution satisfying all constraints exactly.[9]

Spherical Parameterization

For closed, zero-genus surfaces, the sphere is a natural parameterization domain,[10–12] since it does not require cutting the surface into disks. But the distortion may be

Copyright © 2005 John Wiley & Sons, Ltd.

difficult to control. Alexa presents a warping scheme over the sphere,[10] but the scheme does not always satisfy all constraints. Praun *et al.* use their spherical parameterization technique to create a morph between two models,[12] but they have not addressed the problem of feature correspondence.

Simplicial Parameterization

A more general approach is to parameterize the models over a common base mesh.[2,5,13–16] This approach splits the meshes into matching patches with an identical inter-patch connectivity. After the split, each set of matching patches is parameterized on a common convex planar domain. One advantage of this approach is that it naturally supports feature correspondence by using feature vertices as corners of the matching patches. The main challenges in simplicial parameterization are that constructing identical inter-patch connectivities is complex and the patch structure severely restricts the freedom of the parameterization. In,[2] Kraevoy and Sheffer introduce a new method for computing shape preserving cross-parameterizations and compatible remeshing. Compared to previous methods, this one is significantly less dependent on the shape of the patches.

Polycube Maps

Tarini *et al.*[17] introduce PolyCube-Maps, a generalization of the well-known cube-map mechanism, for seamless texture mapping with low distortion. They use the surface of a polycube whose shape is similar to that of the given mesh as the texture domain. The special structure of this surface allows efficiently storing and accessing the texture information. Polycube maps have all the advantages of a seamless mapping, including mip-mapping and mesh independence. This approach is simple enough to be implemented in currently available programming graphics hardware.

We extend this framework to cross parameterization of morphing by parameterizing meshes on their polycubes and then constructing a correspondence through both polycubes. Using polycubes as the common parameter domain of cross-parameterization has the following advantages:

- *Seamless texture*. Although the parameterization of a mesh over a triangulated base complex is seamless, difficulties arise when the vertices of a mesh triangle have parameter values on different domain triangles and their linear interpolation is a secant triangle that falls outside the surface on which the colour information is defined. It is well known that seams cause mesh dependence, inadequate filtering and wasted texture memory. Polycube maps are truly seamless texture mapping techniques which can handle the problem of secant triangles by simply projecting them onto the parameter domain and support mip-mapping because each squarelet of polycube consists of $S * S$ texels where the size S is a power of two. So using the polycube as the common parameter domain leads to seamless texture transferring between two or more models.
- *Low distortion*. Using a polycube surface as the parameterization domain instead of a flat domain or simple spherical shapes helps to reduce the overall distortion because the polycube has a shape similar to that of the mesh.
- *Automatic large scale feature correspondence*. Because the shape of both polycubes is similar, we can directly build the correspondence of matching faces of the polycubes.

Direct Inter-Surface Mapping

Unlike previous approaches which compose parameterizations of both meshes over an intermediate domain, Schreiner *et al.* directly create and optimize a continuous map between the meshes.[18] To generate a smooth cross-parameterization, they use a symmetric, stretch based relaxation procedure, which trades high computational complexity for quality of the mapping.

Contribution

In this paper we propose a new method for mesh morphing using polycube-based cross-parameterization. Its main contributions are:

- Polycube-based cross-parameterization, to preserve common large-scale features straightforwardly.
- Directly composing matching patches on the polycubes without tracing a set of paths on the meshes, to reduce the computational complexity dramatically.
- Sufficient optimization of patch layouts to provide quick convergence to a good solution. This framework of creating and optimizing corresponding patches between two meshes can also be used in constructing constrained texture maps, which speeds up the computation and obtains low-distortion results.

Copyright © 2005 John Wiley & Sons, Ltd.

- Building maps with singularities to allow morphing between models with different topologies. User input is required to associate topological features and introduce singularities on some of the meshes.

All above mentioned cross-parameterization methods are typically used to construct a continuous bijective map between two meshes of the same topology because continuity precludes maps between surfaces with different genus or number of boundaries. Volume-based methods allow models to change topology easily during morphing.[19–21] Cohen-Or *et al.*[20] interpolate the distance field of two objects to obtain the intermediate objects. Breen *et al.* express the interpolation of two shapes (represented as level set models) as a process where one shape deforms to maximize its similarity with another shape.[21] However, all volumetric morphing methods have their limitations:

— Creating distance transformations from 3D polygonal meshes and extracting a polygonal iso-surface from each volume produced by the morphing process are time consuming.
— The basic volumetric representation can produce aliasing artifacts on objects that have regions of high curvature since the accuracy is restricted by the sampling resolution of the volume.
— The volumetric morphing methods are only useful for solid (i.e. closed) objects and cannot be used to morph open shell-like surfaces.

By merging the topology of both polycubes, we can build maps and so allow morphing between models with different genus, which is a desirable effect in many applications.

Algorithm Overview

Definitions

- A *polycube* is a shape composed of axis-aligned unit cubes that are attached face to face (see Figure 1, left). For more detail information on polycube maps we refer the interested reader to Tarini *et al.*[17]
- A *polycube face* is a rectangular sub-part of the polycube surface whose four corner vertices are specified by the user (see Figure 1, right).
- A *path* connecting two features is represented by the shortest possible path along the surface of the polycube. To simplify the implementation, only two features lying on the same plane or adjacent planes on the polycube can be connected by a path.
- A *patch layout* of a polycube induced by a set of feature vertices is a partition of the polycube into simply connected, non-overlapping triangular patches where the boundary of each patch is defined by non-intersecting paths on the polycube between feature vertices. To simplify the embedding, we limit a patch to cover two adjacent planes on the polycube, at most, so that each patch can be easily developed into a planar triangle.

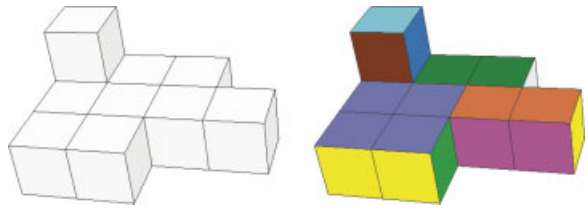


Figure 1. A polycube that consists of 10 cubes(left) and the partition of its surface into faces (right). The rectangular regions with different colour correspond to different faces specified by the user.

Algorithm Stages

The input to the algorithm consists of two manifold meshes M_0 and M_1 and the corresponding sets of matching feature vertices V_0 and V_1, typically selected by the user. The main steps of our algorithm can be summarized as follows:

- *Construction of polycube maps, respectively.* The first step is to define a polycube for each mesh. In order to get the best results, both polycubes should be similar and resemble the shape of the given meshes. Once the polycubes are defined, we compute and optimize mapping between models and their polycubes independently using the method presented in (Figure 2).[17]
- *Cross-parameterization.* Construction of polycube maps provides two parameterizations F_0 and F_1 between the meshes M_0 and M_1 and their polycubes P_0 and P_1. If the polycubes capture all the features the user wants to preserve, we can directly map each face in P_0 to the matching face in P_1. Since both polycube maps are optimized, respectively and the shape of the corresponding faces is similar, we obtain the cross-parameterization with low distortion

Copyright © 2005 John Wiley & Sons, Ltd.

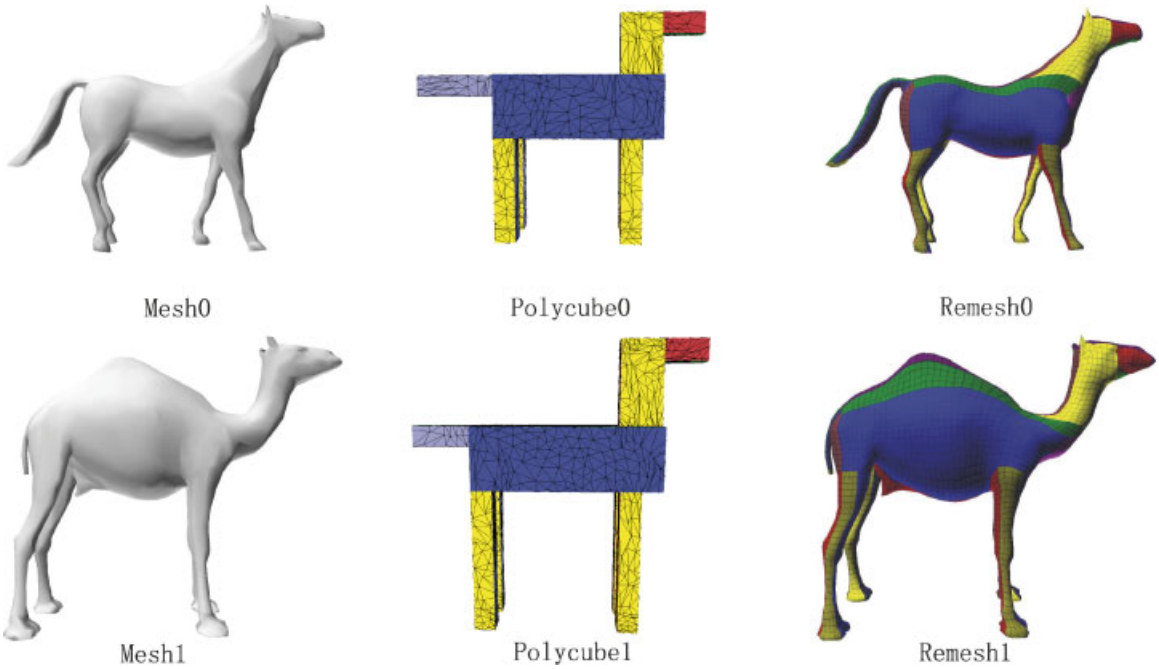


Figure 2. Models, polycubic parameterizations and their remeshes.

(Figure 2). For the finer features that are not reflected by the shape of the polycubes, we split the polycubes into matching patches based on the feature vertices V_0 and V_1 and then optimize the patch layouts. By mapping each patch in P_0 to the corresponding patch in P_1, we obtain the initial cross-parameterization F.

- *Compatible remeshing.* After the initial cross-parameterization is computed, the models need to be represented by compatible meshes, that is meshes with identical connectivity, based on the cross-parameterization.

  We uniformly sample P_0 to construct a semi-regular mesh R_0. By embedding R_0 to P_1 we get a compatible mesh E_0 where E_0 closely approximates the geometry of M_1. Because of the distortion between the patches, flip may occur in E_0 if a triangle spans more than one patch. So the algorithm checks and corrects the flip. Finally, the algorithm smoothes the embedding E_0 in little time and we obtain a bijective mapping with low distortion.

  Of course, we can also adopt other methods for compatible remeshing, such as the approach presented in[2] which first remeshes the target model with the connectivity of the source mesh and then improves the geometry approximation by vertex relocation and refinement.

- *Interpolation of shapes.* After remeshing, the models are interpolated to produce the morphing sequence, using any standard interpolation scheme.[1]

In the following, the construction of the common patch layouts and the embedding are explained in details. Then the mapping between the models with different genus is described. Finally, the results are demonstrated and future works are suggested.

Constructing Patch Layouts

If the shape of the polycubes does not capture all the features that the user wants to preserve, we need to split the polycubes into matching patches based on the feature vertices. Unlike previous approaches, we directly operate on the polycube and avoid the complicated tracing paths on the mesh.

Construct Initial Patch Layouts

Firstly, we add as many paths as possible without violating a number of validity conditions as suggested,[2] such as, the new path must not intersect existing paths in either of the patch layouts L_0 and L_1; the new path must have the same cyclical order around the end

Copyright © 2005 John Wiley & Sons, Ltd.

vertices; the new path must not block necessary future paths, etc. If user-specified features (hard constraints) are not enough to partition both polycubes into triangular patches, additional feature vertices (soft constraints) are added simultaneously on both polycubes by the algorithm until the polycubes are partitioned completely. Since we will optimize the patch layouts later, the initial positions of the added soft constraints in both polycubes are not critical. At the end, we have a topologically identical partition of both polycubes into triangular patches, and each patch spans only one or two adjacent planes of the polycube (Figure 4(b)).

Optimize Patch Layouts

Since a good patch layout can greatly decrease the distortion in the parameterization, we optimize the patch layouts in the following steps subject to the criteria of metric distortion presented by Degener *et al.*[22] which mediates between angle and area deformation of the matching patches:

- Symmetrically optimize the positions of each soft constraints on both polycubes to reduce the distortion between the matching patches while maintain the hard constraints in place to satisfy user-specified correspondences.
- Iteratively refine the patch layouts. Select the worst pair of matching patches from a priority queue sorted by patch energy and add Steiner vertices at the midpoint of their longest pair of corresponding paths simultaneously. The Steiner vertices split original two adjacent patches into four small patches (Figure 3).

We optimize the positions of the Steiner vertices on both polycubes symmetrically to minimize the distortion of the four patches for several iterations. If the distortion of the four new patches is smaller than that of the two original patches, we update the patch layouts. Otherwise, the Steiner vertices are discarded. The insertion is repeated on the path of each patch until the distortion is sufficiently low or until a fixed number of Steiner vertices are inserted. Since the optimization is performed only on the polycube and thus it is quick, we can add as many as needed Steiner vertices to reduce the distortion of the patch layouts without changing the mesh connectivity.

During the optimization, the algorithm guarantees the resulting patch layouts remain valid, that is, have no patches spanning more than two adjacent planes. After the optimization is completed, each patch in polycube P_0 is similar in shape to its counterpart in polycube P_1 (Figure 4(c)).

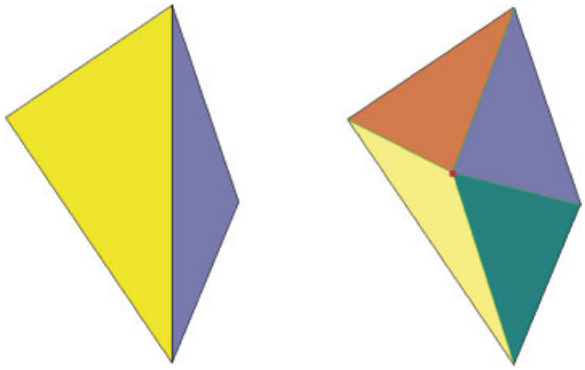


Figure 3. Before and after Steiner vertex inserted on a path.

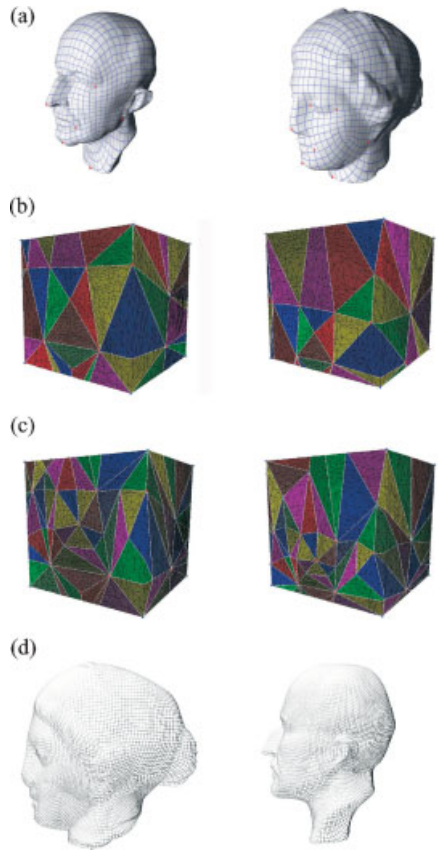


Figure 4. Patch layouts construction. (a) models with feature vertices (red dots), (b) initial patch layouts (hard constraints are red and soft constraints are blue), (c) optimized patch layouts, (d) embedding based on the otimized patch layouts.

Copyright © 2005 John Wiley & Sons, Ltd.

Embedding

Given the patch layouts L_0 and L_1, the algorithm maps each patch b_0 in L_0 to the matching patch b_1 in L_1, mapping the corner vertices to the corresponding corners and using barycentric coordinates for the interior. Thus, for any position on the surface of polycube, we have two geometric positions: v_0 on M_0 and v_1 on M_1.

Although the distortion between matching patches is low after the optimization of patch layouts, there still may exist a few flips if a triangle spans more than one patch. So the algorithm checks flips and simply averaging the embedding parameters of 1-ring vertices is enough to correct them because flips seldom occur.

It can be seen from Figure 4(d) that the parameterization is smooth within each patch, but not across some patch boundaries. So we use the smoothing algorithm in[2] to reduce the distortion further by letting vertices migrate from one patch to another and thus modifying the current mapping from the mesh to the polycube, which ensures the mapping is smooth across patch boundaries and whose distortion varies gradually over the entire surface. Since the patch layouts have been optimized first, the smoothing need only be repeated few times to achieve global smooth effect.

Building Maps Between Models with Different Topologies

We can also build maps with singularities between surfaces with different genus. First, we construct polycube maps, respectively as mentioned previously. Second, matching pairs of faces are assigned by the user. Finally the topology of two polycubes are merged (Figure 5(d)). Each point on the merged polycube surface would lead to two coordinates: one source position on M_0 and one target position on M_1.

For faces that only exist in P_0, the user needs to assign their target positions and their disappearing time in terms of the desired morphing effect. Similarly, for faces that only exist in P_1, the user needs to assign their source positions and their appearing time.

For example, face-1 and face-2 of the merged polycube of the torus/genus-2 models only exist in P_0. We assign their target positions as bilineary interpolation of the target positions of the four edges e_1, e_2, e_3, e_4. When face-1 morphs to meet face-2, both faces disappear at the same time. For face-3, face-4, face-5 and face-6 that only exist in P_1, we assign source positions of edges e_5, e_6, e_7, e_8 to them, respectively, that means each face is initially compressed into a curve on M_0.

Take the torus/tanglecube as another example (Figure 6). Their merged polycube is the same as the polycube of the tanglecube model. All faces that do not exist in P_0 are also initially compressed into curves on M_0 and gradually stretch to their target positions on M_1 during morphing (Figure 7).

Experimental Results

Figure 7 demonstrates blending, morphing and texture transferring performed using our cross-parameterization. By texturing the compatible meshes, the texture can be used throughout a morphing sequence.

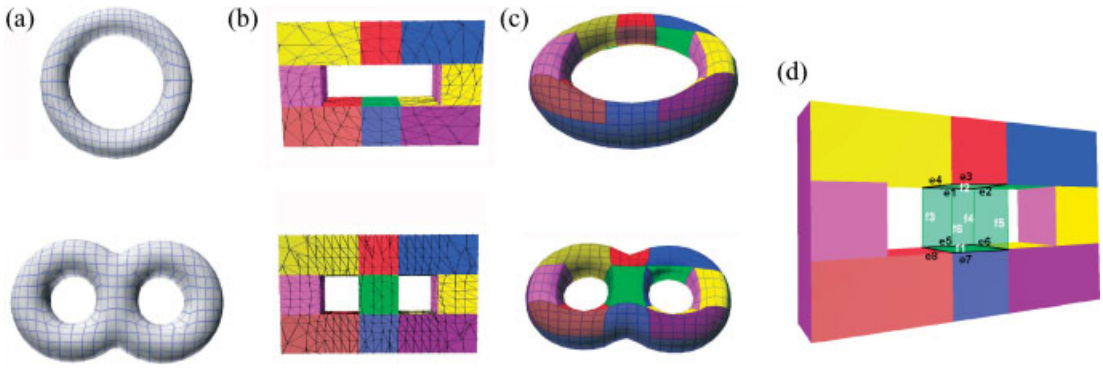


Figure 5. Merging polycubes between torus and genus-2 models. (a) models M_0, M_1, (b) P_0, P_1, (c) remeshes R_0, R_1, (d) merged polycube.

Copyright © 2005 John Wiley & Sons, Ltd.

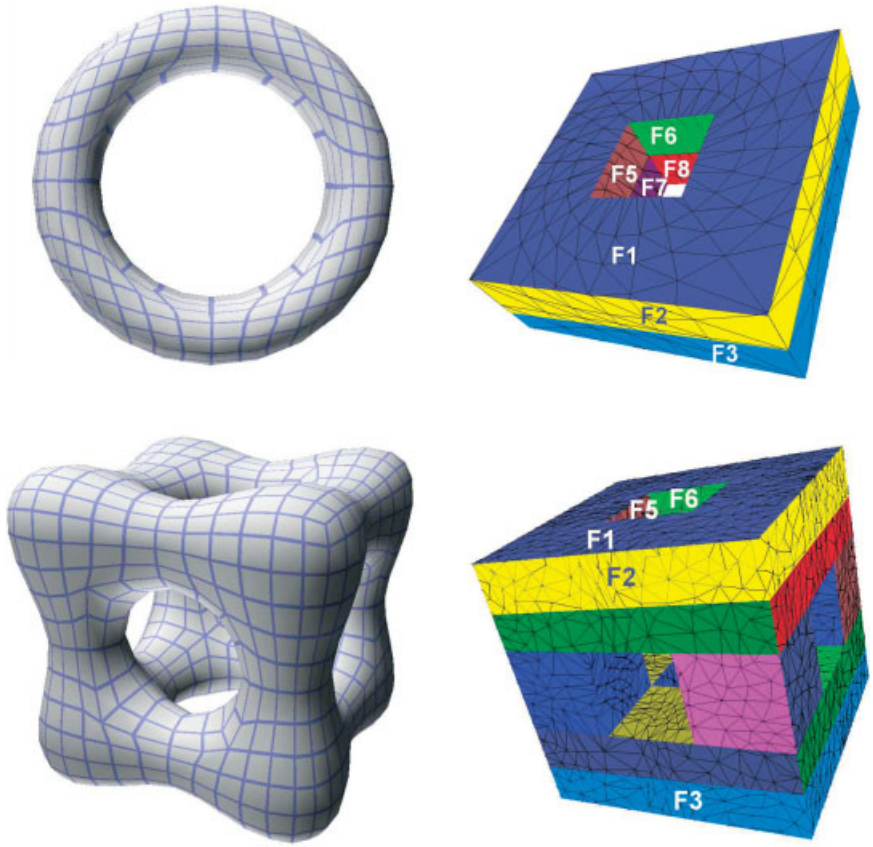


Figure 6. Merging polycubes between torus and tanglecube models.

Some cross-parameterization statistics (after remeshing) are shown in Table 1. For models with different genus, only faces exist on both polycubes are taken into account. Area and angle distortions are measured by integrating and normalizing the values $\delta_1\delta_2 + 1/(\delta_1\delta_2)$ and $\delta_1/\delta_2 + \delta_2/\delta_1$, respectively, where δ_1 and δ_2 are the singular values of the Jacobian matrix J_ϕ.[6,22] The L_2 stretch is measured as described in Sander *et al.*[23] For all measures, the optimal value is 1. Clearly, the distortion is highly dependent on the level of similarity between the models.

Conclusions

In this paper, we have proposed a novel mesh morphing approach based on polycubic cross-parameterization. The main advantage of polycube is seamless texture mapping. Thanks to that, we can transfer texture between models seamlessly. Because the polycubes capture the large scale features, we can easily preserve the shape of models. The smoothing mechanism of patch layouts we have proposed significantly reduces the mapping distortion and improves the speed of optimization. Our approach can also build maps with singularities between models with different topological genus and produce a reasonable morphing result.

Obviously, polycubic cross-parameterization also has limits in the scope of its applicability. If the geometry of two models is quite different, the polycubes cannot be matched easily. Also, if the geometry or topology of a mesh is too complex, an appropriate polycube would consist of so many cubes that the size of the corresponding 3D look-up table of the polycube would soon exceed the texture memory.

Our future research will explore ways to determine appropriate polycubes with minimal or no user

Copyright © 2005 John Wiley & Sons, Ltd.

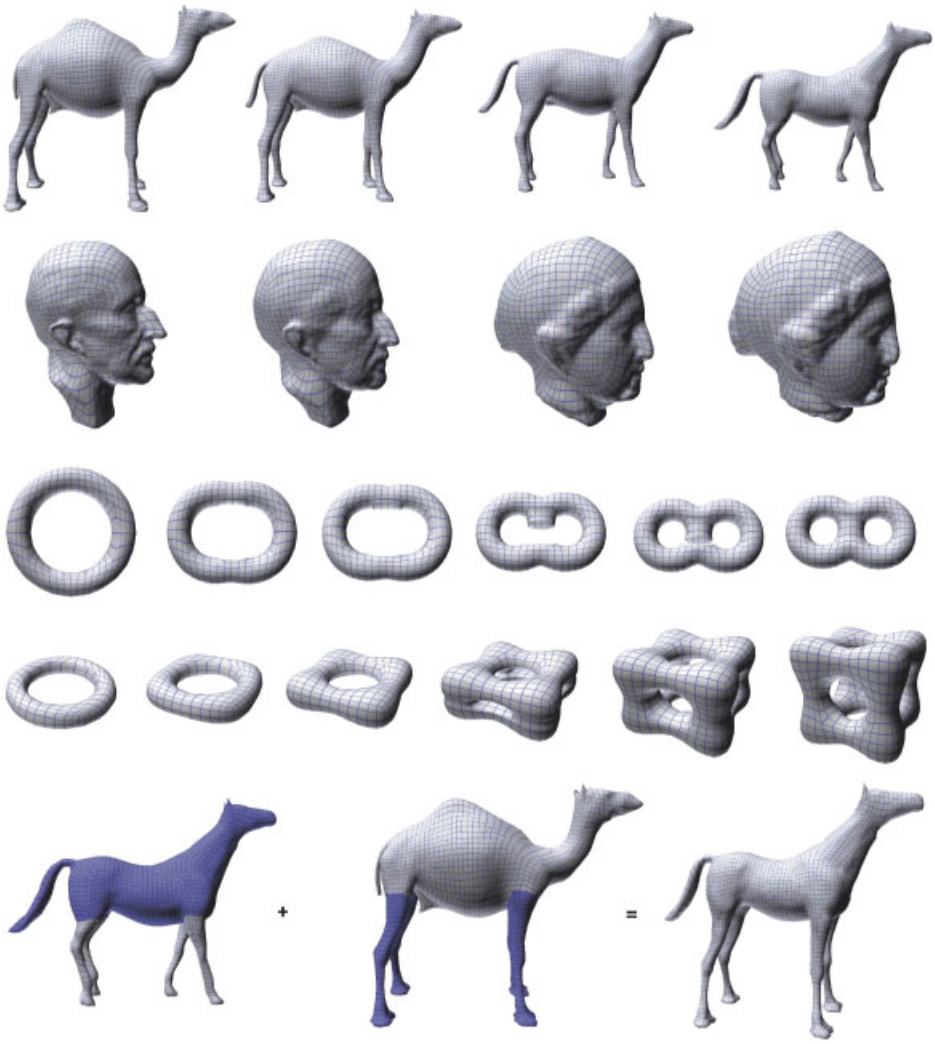


Figure 7. Morphing and blending of different models.

Models (source/target)	E_{angle}	E_{area}	L_2 stretch
Camel/horse	1.25	1.11	0.79
Venus/Maxplanck	1.12	1.07	0.85
Torus/genus-2 model	1.07	1.02	0.89
Torus/tanglecube	1.08	1.04	0.87

Table I. Parameterization statistics

intervention and to speed-up polycube maps with hierarchical methods.

ACKNOWLEDGEMENTS

This work was supported by 973 Program (No. 2002CB312101), National Natural Science Foundation of China (No. 60273054, 60340440422), Fok Ying Tung Education Foundation (No. 91069) and Specialized Research Fund for the Doctoral Program of Higher Education (No. 20020335070). Models are courtesy of Cyberware, Max Planck Institut für Computer Graphik , and Robert Sumner and Jovan Popović from the Computer Graphics Group at MIT.

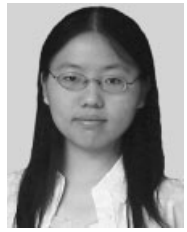
References

1. Alexa M. Recent advances in mesh morphing. *Computer Graphics Forum* 2002; **21**(2): 173–196.
2. Kraevoy V, Sheffer A. Cross-parameterization and compatible remeshing of 3d models. *ACM Transactions on Graphics* 2004; **23**(3): 861–869.
3. Biermann H, Martin I, Bernardini F, Zorin D. Cut- and paste editing of multiresolution surfaces. *ACM Transactions on Graphics* 2002; **21**(3): 312–321.

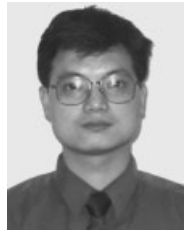
Copyright © 2005 John Wiley & Sons, Ltd.

4. Sumner R, Popović J. Deformation transfer for triangle meshes. *ACM Transactions on Graphics* 2004; **23**(3): 399–405.
5. Praun E, Sweldens W, Schröder P. Consistent mesh parameterizations. In *Proceedings of ACM SIGGRAPH 2001*; pp. 179–184, 2001.
6. Floater M, Hormann K. Surface parameterization: a tutorial and survey. In *Advances in Multiresolution for Geometric Modelling*, Dodgson NA, Floater S, Sabin MA (eds). Springer-Verlag: Heidelberg, 2004; 157–186.
7. Lévy B. Constrained texture mapping for polygonal meshes. In *Proceedings of ACM SIGGRAPH 2001*, 2001; pp. 417–424.
8. Desbrun M, Meyer M, Alliez P. Intrinsic parameterizations of surface meshes. *Computer Graphics Forum* 2002; **17**(2): 167–174.
9. Kraevoy V, Sheffer A, Gotsman C. Matchmaker: constructing constrained texture maps. *ACM Transactions on Graphics* 2003; **22**(3): 326–333.
10. Alexa M. Merging polyhedral shapes with scattered features. *The Visual Computer* 2000; **16**(1): 26–37.
11. Gotsman C, Gu X, Sheffer A. Fundamentals of spherical parameterization for 3d meshes. *ACM Transactions on Graphics* 2003; **22**(3): 358–363.
12. Praun E, Hoppe H. Spherical parameterization and remeshing. *ACM Transactions on Graphics* 2003; **22**(3): 340–349.
13. Lee A, Dobkin D, Sweldens W, Schröder P. Multiresolution mesh morphing. In *Proceedings of ACM SIGGRAPH 1999*, 1999; pp. 343–350.
14. Guskov I, Vidimce K, Sweldens W, Schröder P. Normal meshes. In *Proceedings of ACM SIGGRAPH 2000*, 2000; pp. 95–102.
15. Michikawa T, Kanai T, Fujita M, Chiyokura H. Multiresolution interpolation meshes. In *Proceedings of the 9th Pacific Graphics International Conference*, 2001; pp. 60–69.
16. Khodakovsky A, Litke N, Schröder P. Globally smooth parameterizations with low distortion. *ACM Transactions on Graphics* 2003; **22**(3): 350–357.
17. Tarini M, Hormann K, Cignoni P, Montani C. Polycube-maps. *ACM Transactions on Graphics* 2004; **23**(3): 853–860.
18. Schreiner J, Prakash A, Praun E, Hoppe H. Inter-surface mapping. *ACM Transactions on Graphics* 2004; **23**(3): 870–877.
19. Lerios A, Garfinkle C, Levoy M. Feature-based volume metamorphosis. In *Proceedings of ACM SIGGRAPH 1995*, 1995; pp. 6–11.
20. Cohen-Or D, Levin D, Solomovici A. Three dimensional distance field metamorphosis. *ACM Transactions on Graphics* 1998; **17**(2): 116–141.
21. Breen D, Whitaker R. A level-set approach for the metamorphosis of solid models. *IEEE Trans. on Visualization and Computer Graphics* 2001; **7**(2): 173–192.
22. Degener P, Meseth J, Klein R. An adaptable surface parameterization method. In *Proceedings of the 12th International Meshing Roundtable*, 2003; pp. 201–213.
23. Sander PV, Snyder J, Gortler S, Hoppe H. Texture mapping progressive meshes. In *Proceedings of ACM SIGGRAPH 2001*, 2001; pp. 409–416.

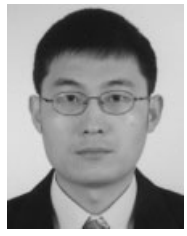
Authors' biographies:



Zhengwen Fan is a graduate student of the State Key Lab of CAD&CG, Zhejiang University. She received her B.Sc. degree in Information Science in 2000 from HuaZhong Normal University. Her research interesets include computer animation and surface parameterization.



Xiaogang Jin is a professor of the State Key Lab of CAD&CG, Zhejiang University. He received his B.Sc. degree in Computer Science in 1989, and M.Sc. degree and his PhD in Applied Mathematics in 1992 and 1995,respectively, all from Zhejiang University. His research interests include implicit surface modeling, space deformation, computer animation and realistic image synthesis.



Jieqing Feng is a professor at the State Key Lab of CAD&CG, Zhejiang University, People's Republic of China. He received his B.Sc. degree in Applied Mathematics from the National University of Defense Technology in 1992, PhD in Computer Graphics from Zhejiang University in 1997. His research interests include space deformation, computer-aided geometric design and computer animation.

Copyright © 2005 John Wiley & Sons, Ltd.

Hanqiu Sun received her B.S. in electrical engineering from Huazhong University of Science and Technology, China. She received her M.S. in Electrical Engineering from University of British Columbia and Ph.D. degree in Computer Science from University of Alberta, Canada. She then worked at University of Alberta and University of Winnipeg as a lecturer and later as an assistant professor till 1996. Dr. Sun then joined the Computer Science and Engineering Department of CUHK. Dr. Sun has published more than 100 refereed technical papers. Her current research interests include interactive animations, virtual & augmented reality, hypermedia, realistic haptics simulation.

Copyright © 2005 John Wiley & Sons, Ltd.