Eurographics Symposium on Geometry Processing (2004)
R. Scopigno, D. Zorin, (Editors)

Connectivity Transformation for Mesh Metamorphosis

Minsu Ahn[1], Seungyong Lee[1,2], and Hans-Peter Seidel[2]

[1]Pohang University of Science and Technology, Korea
[2]Max-Planck-Institut für Informatik, Germany

Abstract

In previous mesh morphing techniques, the vertex set and connectivity of an in-between mesh are fixed and only the vertex positions are interpolated between input meshes. With this restriction, to accurately represent both source and target shapes, an in-between mesh should contain a much larger number of vertices than input meshes. This paper proposes a novel approach for mesh morphing, which includes connectivity changes in a metamorphosis. With the approach, an in-between mesh contains only the vertices from the input meshes and so the in-between vertex count does not exceed the sum of source and target vertex counts. The connectivity changes are realized by a sequence of edge swap operations, determined by considering the geometric errors from the input meshes. Experimental results demonstrate that the proposed approach generates almost same in-between shapes as the metamesh-based approach with a much smaller number of vertices.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling

1. Introduction

Metamorphosis, commonly referred to as morphing, deals with the fluid transformation from one object into another. 3D mesh morphing handles two input polyhedral objects and generates an animation in which the source mesh gradually changes to the target through in-between meshes. With applications such as special effects and shape design, successful techniques have been developed for mesh morphing [LV98].

Most of previous mesh morphing techniques are based on the construction and interpolation of a metamesh. A metamesh is created by merging the vertex and edge sets of input meshes on a common domain, such as a sphere [KPC92, Ale00], 2D polygon [KSK97, BP98, GSL*98], and base mesh [LDSS99]. An in-between mesh can be generated by interpolating the vertices of a metamesh between the source and target positions.

Another interesting approach for mesh morphing is remeshing-based, where the input meshes are resampled to have the same structure [PSS01, MKFC01]. Vertex positions are obtained from the input mesh surfaces with a common mesh parameterization, while the edges are determined by recursive subdivision of a common base mesh. As in the metamesh-base approach, the interpolation of vertex positions generates a sequence of in-between meshes.

The vertices of a mesh and its connectivity represented by the edges are both important for representing the shape approximated by the mesh. The vertices are sample points of the shape and the connectivity determines the interpolation of the sample points for non-sampled parts. In both the metamesh- and remeshing-based approaches, the connectivity of an in-between mesh is fixed in a morphing sequence. Only the vertex positions can change to interpolate the shapes of source and target meshes. This restriction inevitably incurs the drawback that an in-between mesh contains a much larger number of vertices than input meshes.

In a morphing sequence, an in-between mesh reflects more the source shape at the beginning and the target at the end. To precisely represent both the source and target shapes with a fixed connectivity, a metamesh should contain the source and target edges as well as source and target vertices. Also, to merge the source and target edges, a metamesh must include the intersection points of the edges as new additional vertices. In the case of the remeshing-based approach, the vertices and edges of source and target meshes are not

directly used. However, to accurately approximate both the source and target shapes with a fixed vertex set and connectivity, we may need many subdivision levels, which results in a large vertex count of an in-between mesh.

In this paper, we propose a novel approach for mesh morphing, where a morphing sequence includes connectivity transformation and an in-between mesh contains only the vertices from the source and target meshes. Since no additional vertices are introduced, the vertex count of an in-between mesh does not exceed the sum of those of the input meshes. The connectivity transformation is realized by a sequence of edge swap operations, where the sequence is determined by considering the geometric errors of an in-between mesh from the source and target. In a metamorphosis, the connectivity transformation allows an in-between mesh to contain the connectivity similar to the source at the beginning and the target at the end. Hence, we can effectively represent the source and target shapes throughout a metamorphosis with a much smaller number of vertices than previous approaches.

1.1. Related work

Hoppe *et al.* [HDD*93] proved that the connectivity can be transformed among two homeomorphic meshes with a sequence of three edge transformations: edge collapse, edge split, and edge swap. However, they did not present an algorithm to derive the edge transformation sequence that realizes the connectivity transformation.

Hanke and Ottmann [HOS96] presented an algorithm to derive an edge swap sequence that changes a 2D triangulation into another one with the same set of vertices. They also showed that the number of edge swaps required for the connectivity change is bounded by the number of intersections among the edges of the two triangulations. The algorithm and the upper bound provide a theoretical basis for the correctness of the connectivity transformation algorithm proposed in this paper.

Ahn and Lee [AL02] proposed a mesh morphing technique that considers connectivity transformation. However, the technique simplifies input meshes in a metamorphosis, which may incur the loss of source and target details in an in-between mesh. Furthermore, the technique performs a sequence of edge swaps in the source mesh to create an edge of the target mesh. In this approach, the edge swaps for creating a target edge may increase the number of edge intersections between the source and target so that a larger number of edge swaps may be needed to create other target edges later. Hence, this approach may produce an unnecessarily large number of edge swaps in the connectivity transformation.

In a recent technique of Lin and Lee [LL04], the vertices and edges of the target mesh are incrementally created in the source mesh to transform the connectivity in a morphing sequence. However, similar to [AL02], the technique creates a target edge by applying a sequence of edge swaps to the source mesh and does not try to reduce the number of edge swaps used for the connectivity transformation. Hence, the technique has the same drawback as [AL02] in that unnecessarily many edge swaps may be performed in a metamorphosis.

1.2. Contributions

In this paper, to realize mesh morphing with connectivity changes, we present two algorithms;

connectivity transformation: Given two input meshes, we first convert the meshes to have the same number of vertices while preserving the shapes. Then, the connectivity of the converted source mesh is transformed to the converted target by an edge swap sequence. We reduce the number of edge swaps for connectivity transformation by performing an edge swap only when it decreases the number of intersections between the source and target edges. To generate a smooth metamorphosis, geometric errors from the input meshes are also considered in determining the edge swap sequence. Experimental results in Sec. 5 show that the number of edge swaps produced by our algorithm is smaller than that from the technique in [LL04].

connectivity interpolation: After we obtain the edge swap sequence, an animation can be generated by applying geomorphs [Hop96] to the sequence while interpolating the vertex positions from the source to the target. However, in this case, the connectivity of an in-between mesh always changes only at a single edge that is currently being swapped. To avoid this limitation, we rearrange the edge swaps by considering the dependency among them and perform the independent edge swaps simultaneously with geomorphs. In contrast, in [LL04], connectivity interpolation is not considered and edge transformations are performed in sequence one by one.

2. Preliminaries

A mesh is a polygonal approximation of an object, which consists of connectivity and geometric information. In this paper, we only consider triangular meshes. A triangular mesh M can be represented by a pair (P,K), where P is a set of 3D positions of the vertices and K is an abstract simplicial complex that specifies the connectivity information [Spa66].† In this paper, we use edge swap (*eswap*) as the elementary operation for mesh connectivity transformation (see Fig. 1). We assume that two input meshes are homeomorphic to each other and do not consider a genus change in a metamorphosis.

† In the literature, K is often referred to the topological information because it contains the vertex set as well as the connectivity between vertices.

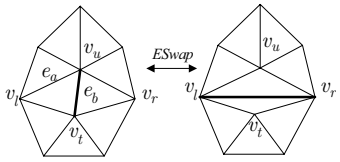


Figure 1: *Edge swap operation*

2.1. Problem definition

Given the source mesh, $M_S = (P_S, K_S)$, and the target mesh, $M_T = (P_T, K_T)$, the problem of mesh morphing is to determine an in-between mesh, $M_I(r) = (P_I(r), K_I(r))$, for a transition rate r between 0 and 1, where 0 and 1 imply the source and target meshes, respectively. In both the metamesh-based and remeshing-based approaches, the in-between connectivity $K_I(r)$ is fixed regardless of r, while the vertex positions $P_I(r)$ are interpolated between P_S and P_T with respect to r. These approaches are quite effective in representing a shape blended from M_S and M_T but they have to pay the cost of a complicated connectivity of $M_I(r)$. In contrast, in this paper, $K_I(r)$ is allowed to change depending on r and $M_I(r)$ is required to effectively represent a blended shape with a simpler connectivity than in the previous approaches.

2.2. Preprocessing for vertex mapping

Let M_S and M_T be the given source and target meshes. To control the shape of an in-between mesh, a user specifies the corresponding feature vertices on M_S and M_T. The vertex mapping between M_S and M_T can be obtained by embedding M_S and M_T onto a common domain. For the embedding, although other approaches (e.g., [LDSS99]) can be used, we adopt the spherical embedding proposed by Alexa [Ale00]. With the embedding, the vertices of M_S and M_T have the corresponding positions on the surfaces of the M_T and M_S, respectively. Note that, in this paper, the embedding is used only for determining the vertex mapping and no metamesh is constructed.

The corresponding feature vertices of M_S and M_T are mapped onto the same position on the common domain and represented by one vertex in an in-between mesh. Similarly, we can merge source and target vertices that are close to each other on the common domain. This vertex merging reduces the number of vertices in an in-between mesh without much degrading the in-between shape quality.

For vertex merging, we adopt the technique proposed by Jeong *et al.* [JYL*03]. The technique merges the mutually closest vertex pairs in the source and target meshes, where the merged position is the middle of the original vertex positions. The vertex merging is not allowed when it introduces a fold-over in the source or target embedding. Additional relaxation on the embedding is performed to reduce the distortions introduced in the vertex merging.

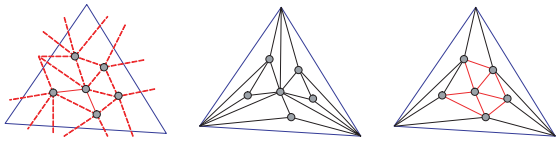
3. Connectivity Transformation

Let n_s and n_t be the vertex counts of M_S and M_T, respectively. Let n_c be the number of merged vertices by vertex merging mentioned in Sec. 2.2. To perform a metamorphosis, we first convert M_S and M_T to M'_S and M'_T, respectively. The vertex counts of M'_S and M'_T are the same as $n_s + n_t - n_c$ and the vertices have an 1-to-1 correspondence between M'_S and M'_T. However, M'_S has the same shape as M_S and so contains a different connectivity from M'_T, which has the same shape as M_T. Then, we obtain a sequence of *eswap* operations that changes the connectivity of M'_S into that of M'_T. With the *eswap* sequence, we can generate a metamorphosis from the source shape to the target in which the vertex count of an in-between mesh is constantly $n_s + n_t - n_c$. Note that $n_s + n_t - n_c$ is the minimal number of vertices required for representing the details of both M_S and M_T at the same time.

3.1. Input mesh conversion

In this section, we explain how to convert M_S into M'_S. In the same way, M_T can be converted to M'_T.

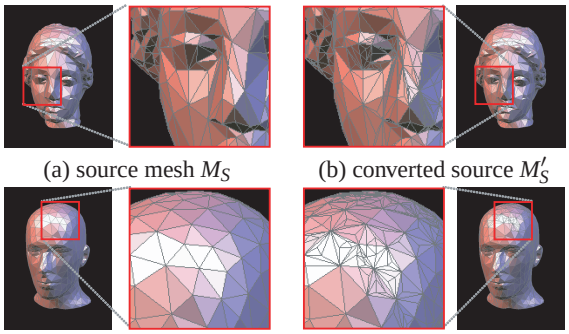
The vertices of M_T can be mapped onto the surface of M_S with the vertex mapping determined in Sec. 2.2 (see Fig. 2(a)). For a target vertex v_t, we first find the face f_s of M_S which contains the mapped position of v_t. Then, a vertex v_s is created at the mapped position and connected to the three vertices of f_s. By repeating this process for each vertex of M_T, we can obtain a mesh M''_S that contains all the source and target vertices. Although its vertex set and connectivity differ from M_S, M''_S has the same shape as M_S (see Fig. 2(b)).



(a) (b) (c)

Figure 2: *Embedding of vertices onto the other mesh: (a) original configuration of target vertices mapped onto a source triangle; (b) result of simple embedding; (c) enhanced result after edge swaps*

However, the edges in M''_S connecting the vertices mapped from M_T may much differ from the edges in M_T. To reduce the difference, we perform *eswap* operations on the edges in M''_S. An edge e''_s of M''_S is swapped if the *eswap* reduces the number of edge intersections between M''_S and M_T on the common embedding. To preserve the shape of M_S, no *eswap* is allowed for the original edges of M_S. We consider each edge e''_s of M''_S in sequence and perform an edge swap until no more edge swap is possible. As a result of the *eswap* operations, we obtain the converted mesh M'_S, which contains as many as possible of the target edges among the vertices mapped from M_T (see Fig. 2(c)). Fig. 3 shows an example of input mesh conversion.



(a) source mesh M_S (b) converted source M'_S

(c) target mesh M_T (d) converted target M'_T

Figure 3: *Input mesh conversion: Input meshes M_S and M_T are converted to M'_S and M'_T, respectively, where the vertices of M'_S and M'_T have an 1-to-1 correspondence.*

3.2. Transformation sequence computation

We can obtain an *eswap* sequence for the connectivity transformation from M'_S to M'_T by adapting the algorithm proposed by Hanke and Ottmann [HOS96]. Although the algorithm is developed for 2D triangulations, it also works for 3D meshes if we check the edge intersections on the common domain for embedding. However, the algorithm does not consider the geometry of triangulations at all because it was aimed to minimize the number of *eswap* operations.

To reflect the geometry changes in the connectivity transformation from M'_S to M'_T, we define an error metric for *eswap*. Let e_s be an edge of M'_S. When e_s is swapped to change M'_S, a geometric error from M'_S happens around e_s. We measure the error by the shortest distance in 3D between e_s and the edge created by swapping e_s. The error of *eswap* performed in the middle of the connectivity transformation can be defined in a similar way. Let e_s be an edge created by applying several *eswap* operations to an edge of M'_S. Then, e_s may intersect several edges in M'_S, where the intersections are checked on the common domain for embedding. We define the error of e_s from M'_S as the maximum of the shortest distances from e_s to the intersecting edges. For *eswap* performed on e_s, we define the error as the difference between the errors of e_s before and after *eswap*. The error for *eswap* performed on an edge e_t from M'_T is defined in a similar way.

We have investigated other possible error metrics for *eswap*, such as an error volume. However, it is difficult to compute the error volume from M'_S after several *eswap* operations are performed on an edge of M'_S. Hence, we chose the distance based metric, which approximates the error volume.

With the error metric for *eswap*, the connectivity transformation from M'_S to M'_T can be obtained by the following algorithm;

1. For each edge e of M'_S and M'_T, we first check if there exists the corresponding edge in M'_T and M'_S, respectively. If not, we compute the swapping error of e and insert e into a priority queue Q with the computed error as the priority.
2. An edge e with the smallest error is extracted from Q. Suppose that e is from M'_S. The case that e has come from M'_T can be handled in a symmetric way. We test if the intersection count of e with the edges of M'_T decreases by swapping e.

3a. If yes, we perform *eswap* on e to update M'_S. After *eswap*, if e corresponds to an edge e' of M'_T in Q, we remove e' from Q. If there exists no such edge in Q, we insert e into Q with the updated error.

3b. If no, e is put back to Q but not extracted again from Q until one of its four neighbor edges is swapped. The neighborhood change will make the result of *eswap* on e varied from the previously tested one.

4. We repeat from Step 2 until no edge remains in Q.

In the algorithm, the given M'_S and M'_T are gradually transformed to each other in connectivity as *eswap* operations are performed. The priority queue Q contains the edges of M'_S and M'_T for which the 1-to-1 correspondence has not been established yet. When the algorithm terminates, the transformed M'_S and M'_T have the same connectivity, which we can consider as an in-between connectivity of the given M'_S and M'_T. The algorithm generates two *eswap* sequences, one for M'_S and the other for M'_T. The *eswap* sequence from M'_S to M'_T can be obtained by concatenating the sequence for M'_S with the reverse of the sequence for M'_T.

In Step 3a of the algorithm, if *eswap* on e generates a triangle flip in the embedding of M'_S, we do not perform the *eswap* and e is put back to Q as in Step 3b. Although some edges of M'_S and M'_T can be inserted into Q more than once, we can guarantee the termination of the algorithm by adapting the proof in [HOS96]. In practice, only a small portion of the processed edges, about 2% in our experiments, are put back into Q, which does not degrade the performance of the algorithm.

The algorithm performs *eswap* operations for M'_S and M'_T in the increasing order of errors from M'_S and M'_T, respectively. In the final concatenated sequence, an *eswap* for M'_S with a smaller error comes earlier, while a smaller error *eswap* for M'_T is around the end. Hence, when we perform the *eswap* sequence, the in-between shape gradually changes from M'_S and M'_T.

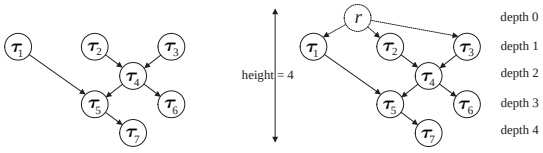
4. Connectivity Interpolation

To generate an in-between mesh M_I, we should interpolate the connectivity between the converted source and target meshes, M'_S and M'_T, with the *eswap* sequence determined in Sec. 3. If we simply apply the *eswap* operations one by one, the connectivity of M_I will locally change on the swapped edge. To obtain global connectivity changes of M_I, we determine the time interval for each *eswap* in the sequence. A

time interval specifies the part of a morphing sequence during which an *eswap* should be performed with a geomorph. At time r, the *eswap* operations whose time intervals contain r simultaneously happen on different parts of $M_I(r)$.

4.1. Time interval calculation

To determine the time interval for each *eswap*, we consider the partial order existing in the *eswap* sequence. For example, if an edge e_a is created by an *eswap* operation τ_a and is a neighbor of an edge e_b to be swapped by *eswap* τ_b (see Fig. 1), τ_b cannot be performed until τ_a has been finished. To represent the partial order, we construct a *transformation dependency graph* D (see Fig. 4(a)). In the graph D, each node is an *eswap* and a directed edge represents a precedence relationship between two *eswap* operations.



(a) dependency graph D (b) graph D' with the root

Figure 4: *Transformation dependency graph*

To construct the graph D, we use the precondition of *eswap* determined by edges. An *eswap* can be applied to an edge e only if the four edges surrounding e are active in the current mesh M (see Fig. 1). Hence, if an *eswap* τ_a creates one of the four surrounding edges of another *eswap* τ_b, D includes a directed edge from τ_a to τ_b.

To specify the time interval for an *eswap* τ, we determine the *start time* and *end time* of τ using the graph D. We first convert the graph D into a graph D' with a root, where the root is connected to all nodes in D with no incoming edges (see Fig. 4(b)). Then, we can define the depth of a node τ in D' as the length of the longest path from the root to τ.

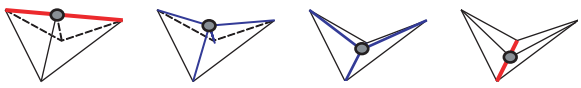
The end time of a node τ_e in D' with no outgoing edges should be 1, i.e., the end of metamorphosis, because no other *eswap* is performed after τ_e. For a node τ in D' with an outgoing edge whose depth is d, let d_e be the maximum of the depths among the descendants of τ. Then, we set the end time of τ as d/d_e because τ is the d-th node on the length d_e path in D' from the root to a node having no outgoing edges. The start time of a node τ_b in D' which is connected to the root is 0, i.e., the beginning of metamorphosis, because no other *eswap* is performed before τ. The start time of a node τ with an incoming edge is determined as the maximum of the end times of the parent nodes of τ. That is, τ starts as soon as its preceding *eswap* operations have been finished.

4.2. In-between mesh generation

To generate an in-between mesh $M_I(r)$ at time r, we first sequentially apply the *eswap* operations whose end times are less than r to the converted source mesh M'_S. Let M' be the resulting mesh. Then, $M_I(r)$ is obtained by applying geomorphs to M' with the *eswap* operations whose time intervals contain r. The vertex positions during the geomorphs are determined by linear interpolation with the vertex mapping obtained in Sec. 2.2. The *eswap* operations whose start times are greater than r have no effect on $M_I(r)$.

When we make an animation of a metamorphosis between input meshes, we should generate a sequence of in-between meshes by incrementally increasing the time. In this case, we do not need to sequentially apply the *eswap* operations to M'_S for each generation of an in-between mesh. Instead, the in-between mesh computed up to the current time can be incrementally updated to obtain the in-between mesh at the next time step.

The original geomorphs proposed by Hoppe [Hop96] handle edge collapse and vertex split transformations. The concept of a geomorph can easily be extended to *eswap*. Fig. 5 illustrates the process.



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

Figure 5: *Geomorph for eswap operation: (a) a vertex is created at the center of the edge to be swapped; (b) & (c) the vertex is moving during the geomorph; (d) the vertex is removed and the edge has been swapped*

5. Experimental Results

We have implemented the proposed mesh morphing technique on a Pentium IV 2.5 GHz PC. Fig. 6 shows metamorphosis examples. In each example, an in-between mesh contains only the vertices from the input meshes and so has a much simpler structure than previous techniques. For example, in [LDSS99], the triangle count of a metamesh is five to ten times larger than the bigger one of input meshes. In contrast, Table 1 shows that the vertex count of an in-between mesh is less than the sum of the vertex counts of input meshes. Fig. 6 demonstrates that the proposed technique generates visually pleasing metamorphoses with the smaller numbers of vertices.

Table 1 summarizes the statistics of the morphing examples. For each example, feature vertices were specified by the user to establish the feature correspondence between input meshes. In Table 1, the size of an in-between mesh does not include the vertices temporarily introduced for geomorphs.

In Table 1, we can see that the number of edge swaps required for connectivity transformation is quite smaller than the edge count of an in-between mesh. Note that the edge swaps are applied to the converted source and target meshes, M'_S and M'_T, which have the same number of edges as an

in-between mesh. When we compare the results with the related work [LL04], we can see that our approach generates smaller number of edge swaps. In Table 1 of this paper, the number of edge swaps is about a half of the edge count of the bigger input mesh. In Table 1 of [LL04], the edge swap count is similar to the bigger edge count.

Table 1 also shows the height of a dependency graph constructed for connectivity interpolation described in Sec. 4. The height depends on the geometry and connectivity differences of input models. However, the height is always quite smaller than the number of edge swaps, which implies many edge swaps can be performed simultaneously in connectivity interpolation.

Table 2 shows the computation times for the steps of the proposed approach. After the edge swap sequence for connectivity transformation has been obtained, the time intervals of the edge swaps are computed only once. With the time intervals, an in-between mesh $M_I(r)$ can be generated for any rate r. Note that, e.g., $M_I(0.7)$ takes more time to generate than $M_I(0.4)$ because the edge swap sequence should always be performed from the beginning. However, this restriction does not much increase the morphing time.

As mentioned in Sec. 4.2, when we generate a sequence of in-between meshes, the current in-between mesh can be efficiently obtained by updating the one at the previous time step. This property is verified by the sixth column in Table 2, which gives the computation time for generating 100 in-between meshes for a metamorphosis. Note that the time for 100 in-between meshes is much less than 100 times the average time for generating a single in-between mesh.

The right part of Table 2 shows the sizes of metameshes and the shape differences of in-between meshes among the proposed and metamesh-based approaches. In Table 2, we performed the vertex merging mentioned in Sec. 2.2 before a metamesh was constructed. Although the vertex merging may reduce the size of a metamesh, the edge intersections still makes the vertex count of a metamesh two or three times larger than the in-between vertex count of the proposed approach. The in-between shape errors in Table 2 were measured by the Metro tool [CRS98]. From the small error values, we can confirm that the proposed approach produces almost same in-between shapes as the metamesh-based approach with a much smaller number of vertices.

6. Discussion and Future Work

The algorithm proposed by Hanke and Ottmann [HOS96] provides a theoretical basis for the connectivity transformation algorithm presented in Sec. 3. However, the algorithm of Hanke and Ottmann deals with 2D triangulations with boundaries, while in this paper, the algorithm is applied to the spherical embedding of triangular meshes. Since a sphere with a hole can be projected onto a plane, we expect that the results of Hanke and Ottmann also hold for the case of spherical embedding. Although this is not a rigorous proof, in our experiments, the connectivity transformation algorithm in Sec. 3 always worked correctly.

In the algorithm presented in Sec. 3, an edge swap is performed only when it decreases the number of intersections between the source and target edges. As shown by Hanke and Ottmann [HOS96], this strategy helps a lot in reducing the number of edge swaps required for connectivity transformation. However, the resulting number of edge swaps may not be minimal because we consider the mesh geometry in determining the priority of edge swaps. The minimal number can be achieved when we ignore the mesh geometry and consider only the number of edge intersections in determining the priority. On the other hand, the priority with a geometric error is very important to generate smooth shape changes in a metamorphosis. Hence, the algorithm in Sec. 3 provides a nice compromise between the connectivity and geometry changes in computing a morphing sequence.

The current implementation of the proposed approach cannot handle morphing between non-zero genus objects. This drawback is totally because we currently use spherical embedding for common parameterization, not due to a fundamental limitation of the approach. With parameterization such as used in [LDSS99], the approach can immediately be applied to non-zero genus objects. The only component to be revised is the intersection test routine for the edges embedded on the common domain. In the current implementation for spherical embedding, edge intersection is tested by adapting the signed area to handle spherical arcs. Edge intersection tests for the other types of common domains can be implemented in a similar way.

In the proposed approach, the connectivity of an in-between mesh dynamically changes in a metamorphosis. Although the dynamic connectivity is an important property to generate a simpler in-between mesh, it may make it difficult to fully utilize the power of a graphics card with a GPU. In contrast, the static connectivity of metamesh or remeshing based approach allows the vertex position interpolation to be accelerated by vertex programming on a GPU. Also, with the static connectivity, shape blending between more than two meshes would be easier than with the dynamic connectivity. However, the simplicity of in-between meshes from the proposed approach will still be useful when high quality rendering is applied to a metamorphosis in an animation. To better optimize the proposed approach, effective handing of dynamic mesh connectivity on a GPU will be an interesting research direction.

The proposed approach uses only edge swaps for connectivity transformation. If vertex splits and edge collapses are included as well, the number of vertices in an in-between mesh can be further reduced. Recently Lin and Lee [LL04] proposed a morphing technique in which vertices are gradually removed or created in a metamorphosis. However, the

technique does not consider the geometric errors from the input meshes and may generate a large number of edge transformations. An interesting future work is to develop a morphing technique that provides a nice control of geometric errors and minimizes the number of transformations for in-between meshes with dynamically varying vertex and edge sets. We are also interested in investigating a connectivity transformation technique that can handle genus changes in a metamorphosis.

Acknowledgments

The authors would like to thank Radek Grzeszczuk for the horse, star, and turtle models and the anonymous reviewers for their helpful comments. The skull, mannequin head, and Venus and rabbit models are courtesy of Headus (www.headus.com.au), the University of Washington, and Cyberware, respectively. This work was supported in part by the Korea Ministry of Education through the Brain Korea 21 program and the Game Animation Research Center.

References

[AL02] AHN M., LEE S.: Mesh metamorphosis with topological transformations. In *Proc. Pacific Graphics 2002* (2002), IEEE Computer Society Press, pp. 481–482. 2

[Ale00] ALEXA M.: Merging polyhedral shapes with scattered features. *The Visual Computer 16*, 1 (2000), 26–37. 1, 3

[BP98] BAO H., PENG Q.: Interactive 3D morphing. *Computer Graphics Forum (Proc. of Eurographics '98) 17*, 3 (1998), 23–30. 1

[CRS98] CIGNONI P., ROCCHINI C., SCOPIGNO R.: Metro: measuring error on simplified surfaces. *Computer Graphics Forum 17*, 2 (1998), 167–174. 6, 8

[GSL*98] GREGORY A., STATE A., LIN M., MANOCHA D., LIVINGSTON M.: Feature-based surface decomposition for correspondence and morphing between polyhedra. *Proc. Computer Animation '98* (1998). IEEE Computer Society Press. 1

[HDD*93] HOPPE H., DEROSE T., DUNCHAMP T., MCDONALD J., STUETZLE W.: *Mesh Optimization*. Tech. Rep. TR 93-01-01, University of Washington, 1993. 2

[Hop96] HOPPE H.: Progressive meshes. *Computer Graphics (Proc. SIGGRAPH '96)* (1996), 99–108. 2, 5

[HOS96] HANKE S., OTTMANN T., SCHUIERER S.: The edge-flipping distance of triangulations. *Journal of Universal Computer Science 2*, 8 (1996), 570–579. 2, 4, 6

[JYL*03] JEONG E., YOON M., LEE Y., AHN M., LEE S., GUO B.: Feature-based surface light field morphing. *Proc. of Pacific Graphics '03* (2003), 215–223. 3

[KPC92] KENT J. R., PARENT R. E., CARLSON W. E.: Establishing correspondences by topological merging: a new approach to 3-d shape transformation. *ACM Computer Graphics (Proc. SIGGRAPH '92) 26*, 2 (1992), 47–54. 1

[KSK97] KANAI T., SUZUKI H., KIMURA F.: 3D geometric metamorphosis based on harmonic map. In *Proc. of Pacific Graphics '97* (1997), pp. 97–104. 1

[LDSS99] LEE A. W., DOBKIN D., SWELDENS W., SCHRÖDER P.: Multiresolution mesh morphing. *ACM Computer Graphics (Proc. SIGGRAPH '99)* (1999). 1, 3, 5, 6

[LL04] LIN C.-H., LEE T.-Y.: Metamorphosis of 3D polyhedral models using progressive connectivity transformations. *IEEE Trans. Visualization and Computer Graphics* (2004), to appear. http://couger.csie.ncku.edu.tw/∼tonylee/papers/IEEE_TVCG_PM_2004.pdf. 2, 6

[LV98] LAZARUS F., VERROUST A.: Three dimensional metamorphosis: A survey. *The Visual Computer 14*, 8/9 (1998), 373–389. 1

[MKFC01] MICHIKAWA T., KANAI T., FUJITA M., CHIYOKURA H.: Multiresolution interpolation meshes. In *Proc. Pacific Graphics 2001* (2001), IEEE Computer Society Press, pp. 60–69. 1

[PSS01] PRAUN E., SWELDENS W., SCHRÖDER P.: Consistent mesh parameterizations. In *SIGGRAPH 2001, Computer Graphics Proceedings* (2001), pp. 179–184. 1

[Spa66] SPANIER E. H.: *Algebraic Topology*. McGraw-Hill, 1966. 2

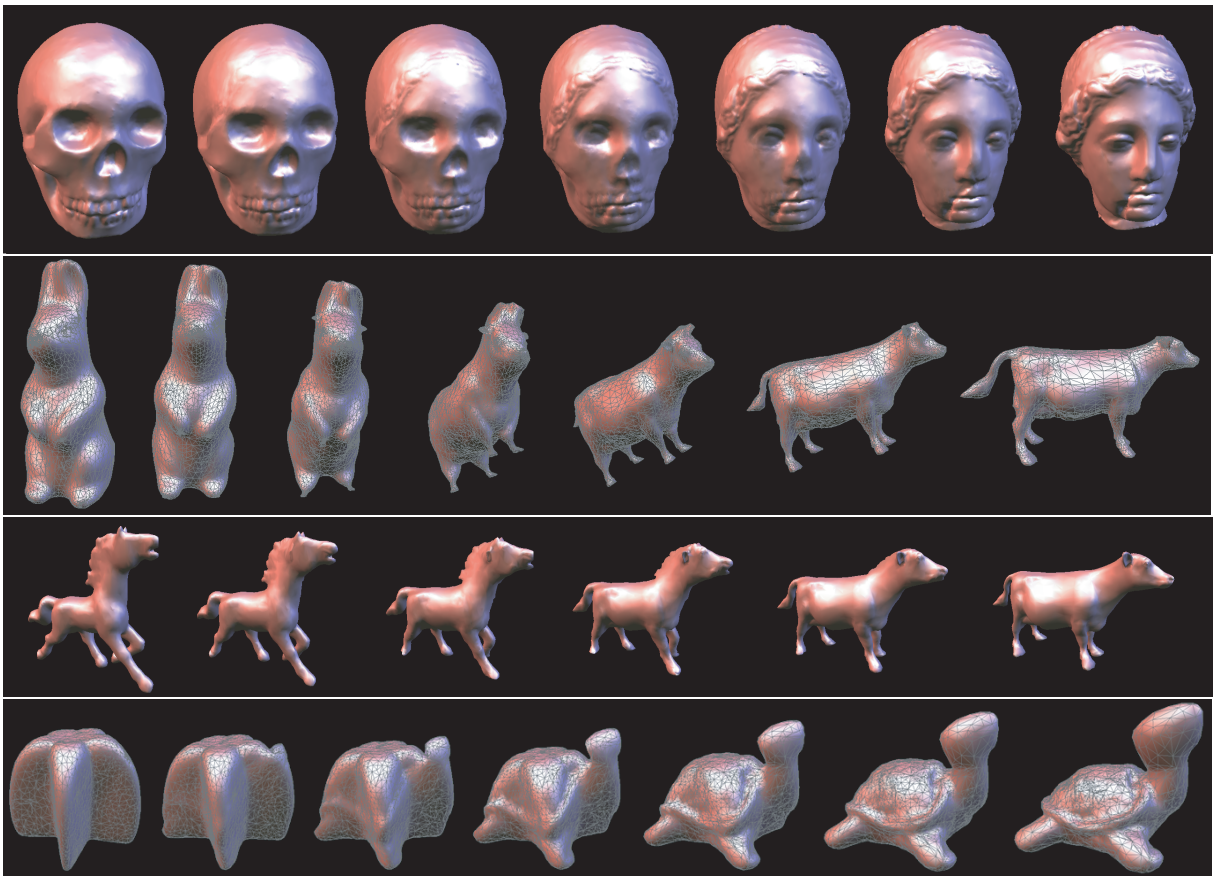


Figure 6: *Morphing gallery: From top to bottom, morphing from Skull to Venus, from a rabbit to a cow, from a horse to a cow, and from a star to a turtle (see also Color Plate)*

models	feature points	merged vertices	# vertices / # edges			# of *eswap*	dependency graph height
			M_S	M_T	M_I		
skull-venus	14	26,219	55,718 / 167,148	39,802 / 119,400	69,301 / 207,897	67,911	35
rabbit-cow	16	1,201	2,004 / 6,006	2,577 / 7,725	3,380 / 10,134	3,913	42
horse-cow	34	1,683	3,503 / 10,503	2,577 / 7,725	4,397 / 13,185	5,512	45
star-turtle	7	1,203	2,491 / 7,467	1,850 / 5,544	3,138 / 9,408	3,796	50

Table 1: *Statistics of metamorphosis examples: The number of edge swaps required for connectivity transformation is quite smaller than the edge count of an in-between mesh. The small heights of the dependency graph implies that many edge swaps can be performed simultaneously in connectivity interpolation.*

models	input conversion	connectivity transform	time intervals	M_I generation		metamesh	% L^2 error	
				average	100 frames	# vertices / # edges	average	max
skull-venus	23.47s	12.08s	9.59s	1.91s	76.41s	169,878 / 509,628	0.0014	0.0008
rabbit-cow	0.61s	0.47s	0.50s	0.10s	3.25s	8,008 / 24,018	0.069	0.110
horse-cow	0.73s	0.64s	0.61s	0.12s	4.05s	11,486 / 34,452	0.053	0.085
star-turtle	0.48s	0.42s	0.41s	0.08	2.95s	7,947 / 23,835	0.036	0.055

Table 2: *Timing data and comparison with a metamesh: Computation time was measured in seconds on a Pentium IV 2.5Hz PC. The L^2 error was measured by the Metro tool [CRS98] in the percentage of the diagonal length of the bounding box.*