academicJournals

Vol. Vol. 9(13), pp. 302-308, 16 July, 2014
DOI: 10.5897/IJPS2014.4157
ISSN 1992 - 1950
Article Number: 9BE0C0946337

http://www.academicjournals.org/IJPS

International Journal of Physical Sciences

Full Length Research Paper

An efficient technique for morphing zero-genus 3D objects

A. Elef, M. H. Mousa* and H. Nassar

Department of Computer Science, Faculty of Computers and Informatics, Suez Canal University, Egypt.

Received 30 April, 2014; Accepted 10 July, 2014

In this paper, we present an algorithm to morph a zero-genus mesh model to a topologically equivalent one based on spherical parameterization, as it is the natural parameterization method for this kind of objects. Our algorithm starts by normalizing the two objects to the cube of unity, as a preprocessing step. Then, the two normalized models are parameterized onto a common spherical domain. We reposition the points of the objects on the sphere in accordance to the relative areas of their triangles. Repositioning on the sphere prevents point clustering and overlapping during the matching process. Experimental results are presented to demonstrate the efficiency of the algorithm.

Key words: 3D morphing, zero-genus mesh, spherical parameterization, nearest neighbor matching.

INTRODUCTION

Shape changing has gained attention due to the attraction of scenes people see on the screen. Indeed, morphing is derived from the biological term “metamorphosis”, meaning the change in form and often habits of the individual during normal development after the embryonic stage. Therefore, it suits very well modeling computer animation techniques dealing with the design of algorithms changing one object into another over time (Sompagdee, 2009). In other terms, morphing is an interpolation technique used to create from two objects a series of intermediate objects that change continuously to make a smooth transition from the source object to the target object. Morphing has been done in two dimensions by varying the values of the pixels of one image to make a different image, or in three dimensions by doing the same. We are presenting here a new type of morphing, which is applied to the geometry of three dimensional models, creating intermediate 3D objects which can be translated, rotated, scaled, and zoomed into.

The history of morphing can be categorized into two categories: 2D and 3D morphing. 2D morphing seems to have reached its goals in finding the solutions for the transformations as well as feature handling, while 3D morphing is still far behind 2D success (Sompagdee, 2009).

Two-dimensional morphing

In general, 2D morphing techniques require a lot of manual processes. They can be classified into two categories: image-based and geometric-based.

*Corresponding author. E-mail: amira_mohamed@ci.suez.edu.eg, mohamed_mousa@ci.suez.edu.eg, nassar@ci.suez.edu.eg

Image-based morphing

Image-based approaches are used enormously in the entertainment industry. Algorithms here are generally composed of three major steps: control, warping and cross dissolving. For controlling the morphing process, these algorithms rely heavily on the experience and knowledge of animators to define correspondence points. Good results can be obtained by using images captured from similar angles and positions regarding the lack of the depth information for generating intermediate objects. Beier and Neely (1992) proposed the line segment feature based method. Where, animators have less work but the complexity of calculation is increased. Lee et al. (1995) used moving curves called snakes in order to capture features accurately and fasten the process of feature defining.

Geometric-based morphing

For geometric–based approaches, 2D polygons are given as input. Sederberg and Greenwood (1992) proposed a minimization method without user interaction. New vertices are added to the polygon that has less number of vertices. All possible paths are calculated but only the best path, the one that gives a minimum amount of work or less shrinkage, is selected. Gao and Sederberg (1998) improved the method to measure the amount of work as well as the way to find the least work path. Shapira and Rappoport (1995) embedded skeletons inside each polygon by decomposing the polygon into star-shaped pieces with a star origin inside each piece then they connected those star origins. For interpolation, skeletons are interpolated and star pieces are unfolded.

Three-dimensional morphing

Three- dimensional morphing algorithms transform a 3D model into another. The complexity of an algorithm depends on many factors, such as: the object representation, genus, and convexity. The simplest morphing can be done when the initial and final shapes are convex and similar in their geometry and topology. Usually, 3D methods have restrictions on object models and how models are represented. Concave objects are more difficult to morph than convex ones. Objects with a different genus, e.g. with holes or with a closed surface are very complicated to transform. Without user intervention, it is not possible to get the desirable results. The existing solutions are categorized by object representations as follow (Sompagdee, 2009):

(i) Polygonal-based representation: The objects are represented by their boundaries. This representation is commonly used and is easy to obtain.

(ii) Volumetric representation: The 3D models are described either by their geometric primitives or by volumes (volumetric data sets). A volumetric representation is ideal for modeling the behavior of objects with complex interior structures. In particular, this representation helps overcome the limitation on model types.

CONTRIBUTION

Our approach is focused on creating the series of intermediate objects, using spherical parameterization as a common domain of the source and target zero-genus objects. This parameterization domain is the natural domain to use, given when our object is a sphere, and as such- makes the mapping step easier.

To create this series of intermediate objects, we start by parameterizing the source and target objects on the sphere. We use progressive meshes (Hoppe, 1996; Zhou et al., 2004) in combination with a local smoothing strategy (Shen and Makedon, 2006) to find the final spherical parameterization. Then we create the point-to-point correspondence between the objects using the AABB tree search technique (CGAL, 2010). The spherical parameterization which we have used makes the mapping step easier since the objects are mapped to their natural parameterization domain.

Initial spherical parameterization

Parameterization of 3D mesh data is important in such applications as, texture mapping, remeshing and morphing. Closed manifold genus-0 meshes are topologically equivalent to a sphere; hence this is regarded as the natural parameterization domain for it. Parameterizing a triangle mesh onto a sphere means assigning a 3D position on the unit sphere to each of the mesh vertices, such that the spherical triangles induced by the mesh connectivity are not too distorted and do not overlap. Satisfying the non-overlapping requirement is the most difficult and critical component of this process. Moreover, it is usually an expensive optimization procedure for large meshes. Here, we describe the spherical parameterization algorithm we use, which is based on the algorithm proposed in (Praun and Hoppe, 2003; Zhou et al., 2004; Shen and Makedon, 2006). The later algorithm incorporates a local parameterization scheme into the progressive mesh representation (Hoppe, 1996). This reduces the complexity of global optimization for large meshes.

Given a triangle mesh M, the problem of spherical parameterization is to form a continuous invertible mapψ from the unit sphere to the mesh. That's, $\psi: S^2 \rightarrow M$. The map is specified by assigning for each mesh vertex v a parameterization $\psi^{-1}(v) \in S^2$. Therefore, each mesh

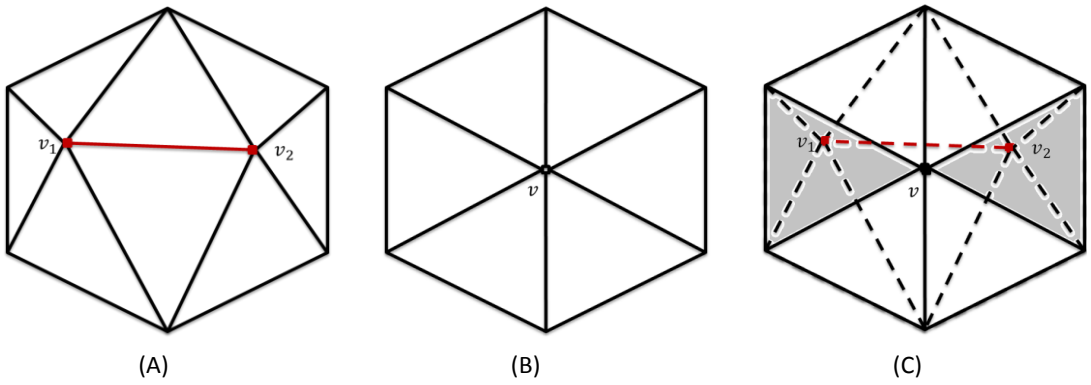

Figure 1. An edge collapse operation. (A) (Left) the edge v to be collapsed. (B) (Middle) The mesh after the edge contraction. (C) (Right) Parameterizing the deleted vertices using the created triangles.

edge is mapped to a great circle arc, and each mesh triangle is mapped to a spherical triangle bounded by these arcs. The spherical parameterization algorithm consists of two major steps:

(i) A progressive mesh representation, $\mathcal{M} = \{M^0, M^1, \cdots, M^k = M\}$, with embedded local parameterization information is generated from the original mesh M. This is performed by iterating the successive edge collapse operations until the current simplified mesh becomes a convex polyhedron M^0. The obtained polyhedron is considered as the base mesh of $\mathcal{M}$. For each edge collapse, the two decimated vertices are parameterized over the resulting simplified mesh. The local parameterization information is recorded in $\mathcal{M}$.
(ii) Suppose that the centroid of the base mesh M^0 is the center of the unit sphere. The projection of the vertices of M^0 onto the considered sphere produces an initial spherical mesh. Starting from this initial spherical mesh, the sequence of vertex split operations, the inverse of the edge collapse, in $\mathcal{M}$ is performed progressively. For each vertex split operation, the two split vertices are positioned on the unit sphere using the recorded connectivity and embedded parameterization information.

The key point of the progressive mesh hierarchical structure is the choice of the order of the edges to be collapsed. Here, we use the selection strategy proposed by Garland and Heckbert (1997) to determine the edge collapse order and to position the newly created vertices. In the classical progressive mesh representation (Hoppe, 1996; Zhou et al., 2004), the geometrical and topological information of each removed vertex are recorded in a vertex splitting operation during the decimation process. Thus the vertex can be completely recovered during progressive refinement process. However, this information is not suitable to reposition the recovered vertex on the unit sphere when the same vertex split sequence is performed on the corresponding spherical mesh. In what follows, we present how to reposition the recovered vertex according to the relative position with respect to its first order neighborhood in the original mesh. As shown in Figure 1 (left and middle), edge $e = v_1v_2$ is collapsed and a new vertex v is created. Let $star(v)$ be first order neighborhood of the vertex v. Using the MAPS algorithm (Lee et al., 1998), $star(v)$ is flattened into a planar region Ω. The vertices v_1 and v_2 are embedded into Ω and find the triangles T_1 and T_2 containing v_1 and v_2 respectively. The barycentric coordinates $(\alpha_i, \beta_i, \gamma_i)$, $i = 1,2$ of the embedding of v_1 and v_2 inside T_1 and T_2 respectively. Therefore, v_1 and v_2 can be locally parameterized with respect to the containing triangles T_1 and T_2 using the barycentric coordinates $(\alpha_i, \beta_i, \gamma_i)$. The local parameterization information of the two decimated vertices, together with all collapse information of the edge, is recorded in a vertex split operation for later reconstruction on the sphere. The final form of the progressive hierarchy of the given mesh is $\mathcal{M} = \{M^0, sp^1, sp^2, \cdots, sp^k\}$, where M^0 is the convex base mesh and the sp^i are the ordered split operations. That is, $M^i = M^0, sp^1, \cdots, sp^i$.

Starting from this base spherical mesh which is generated by projecting the convex base mesh M^0 onto the unit sphere, the vertex split operations in $\{sp^1, sp^2, \cdots, sp^k\}$ are performed progressively to simultaneously recover the original mesh and construct the spherical parameterization. Let M^i be the recovered mesh after performing the $i$ths vertex split operation, M_s^i be its corresponding spherical mesh. The $(i+1)$ths vertex split operation is then performed as follows. With the containing triangles T_1, T_2 and barycentric coordinates $(\alpha_i, \beta_i, \gamma_i)$ of the two new vertices v_1' and v_2' retrieved from sp^{i+1}; two new vertices v_1' and v_2' are inserted into M_s^{i+1} with the same connectivity as in M^{i+1} and positioned on the unit sphere by:

$$v_1' = Normalize(\alpha_1 p_1^1 + \beta_1 p_1^2 + \gamma_1 p_1^3)$$
$$v_2' = Normalize(\alpha_2 p_2^1 + \beta_2 p_2^2 + \gamma_2 p_2^3)$$

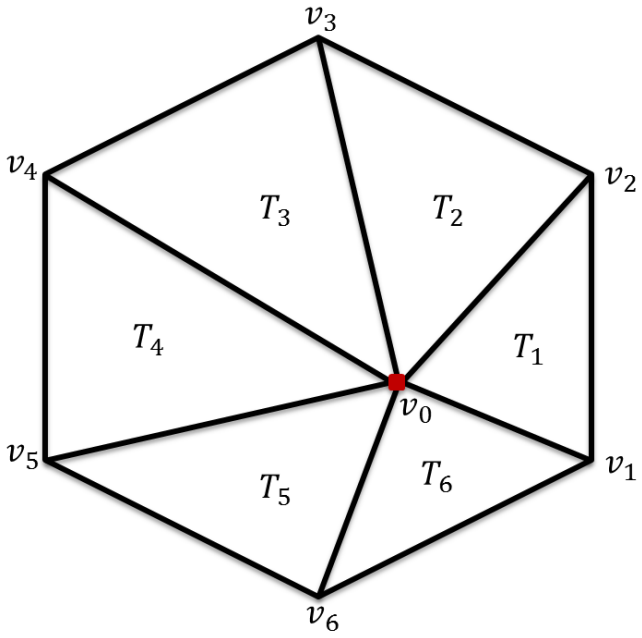

Figure 2. The local neighborhood $star$ projected on the 2D plane.

Where (p_1^1, p_1^2, p_1^3) and (p_2^1, p_2^2, p_2^3) are the triangles in M_s^i corresponding to T_1 and T_2 respectively.

Local spherical improvement

The local spherical improvement uses iterations overthe vertices of the initial parameterization on the unit sphere (Shen and Makedon, 2006). At each vertex, it tries to reposition in the local neighborhood $star$ to gradually improve the mapping quality. The triangles composing the local neighborhood $star$ is piece-wise linear. Therefore it can be projected on the 2D plane while preserving the relative area of each spherical triangle. Given that the vertex to reposition, Figure 2 shows an example of the projection of $star$ on a 2D plane. Supposing that $v_i = (x_i$, the signed area of the triangle can be computed by:

$$A_1 = \frac{1}{2}\left((x_2 - x_1)(y_0 - y_1) - (x_0 - x_1)(y_2 - y_1)\right)$$

Thus, replacing A_1 with A_{ideal} and treating $x_0$and y_0 as the only unknowns in the previous equation, we can formulate a system of linear equations using all the triangles in the spherical$star(v_0)$and solve it in the least squares sense to locate a new center vertex position. This new center position minimizes the square sum of the relative area differences between $star(v_0)$ on the object and $star(v_0)$ on the sphere S^2. Note that only the center position is concerned here. This indicates that the border of spherical $star(v_0)$is fixed,and consequently that. the total area of the spherical $star(v_0)$cannot be changed. Therefore, each triangle in the spherical $star(v_0)$ aims to achieve not its correct absolute area but the correct area relative to the other triangles in $star(v_0)$. For example, Let$T_1, T_2, \cdots, T_6$ be the set of projected triangles depicted in Figure 2 and $T_1', T_2', \cdots, T_6'$ be the corresponding triangles on the object. If we use $A(\cdot)$ to denote the area of a triangle, the relative area of T_i' can be given by:

$$T_i = \frac{A(T_i')}{\sum_{j=1}^{6} A(T_j')}$$

In order to preserve this relative area, on the 2D project should have an ideal area of,

$$A_i = \frac{A(T_i')}{\sum_{k=1}^{6} A(T_k')} A_{Total}$$

Where A_T, is the total area of the 2D projection of the parameter submesh. To calculate the new location (x) for the parameter submesh center on its 2D projection, the following system of linear equations is formulated:

$$\begin{aligned}
(x_2 - x_1)(y - y_1) - (x - x_1)(y_2 - y_1) &= 2A_1 \\
(x_3 - x_2)(y - y_2) - (x - x_2)(y_3 - y_2) &= 2A_2 \\
(x_4 - x_3)(y - y_3) - (x - x_3)(y_4 - y_3) &= 2A_3 \\
(x_5 - x_4)(y - y_4) - (x - x_4)(y_5 - y_4) &= 2A_4 \\
(x_6 - x_5)(y - y_5) - (x - x_5)(y_6 - y_5) &= 2A_5 \\
(x_1 - x_6)(y - y_6) - (x - x_6)(y_1 - y_6) &= 2A_6
\end{aligned}$$

The new center location $(x$ is obtained by solving this linear system in a least squares sense.

MESH MAPPING

Now that we have the source and the target 3D objects parameterized on the sphere, the objective here is to find the point-to-point correspondence between the two parameterized objects. According to the number of points in the two objects, we have two kinds of mesh mapping:

(i) M to M mapping: The two input objects have the same number of points. We use the nearest neighbor algorithm (Manolis et al., 1997) to find the point to point correspondence. To avoid the case that multiple points are matched to a single point, we associate with each point an attribute, valued by true or false, to indicate whether that point is selected before or not.

(ii) M to N mapping: The source and the target objects have not the same number of points. Without loss of generality, assume that the source object has more points than the target object. To overcome this inequality problem we subdivide the target object using the subdivision algorithm (Kobbelt, 2000) such that the two

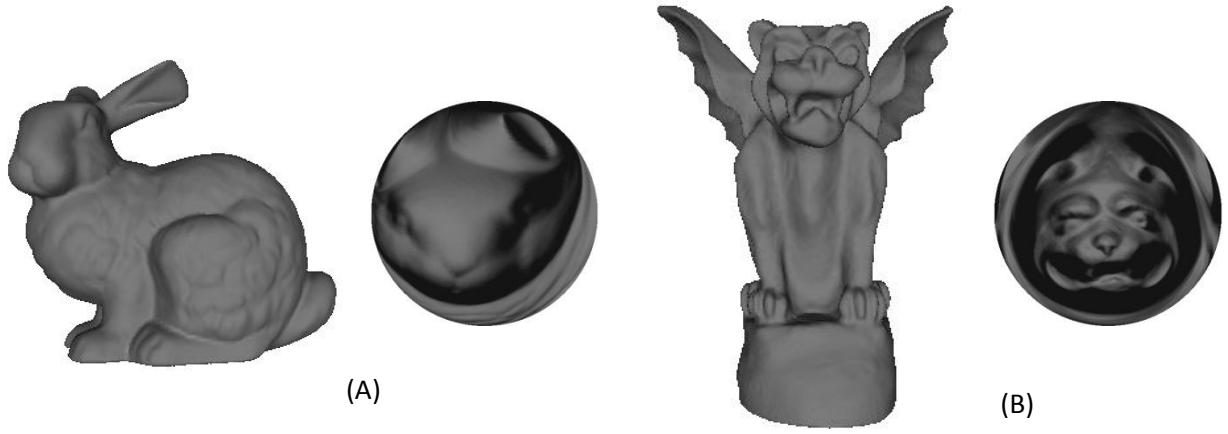

Figure 3. (A) Left two images. The input triangular mesh of the Stanford Bunny (39 k points, 70 k triangles) and, its spherical parameterization, (B) Right two images. The target triangular mesh of the Gargoyle (100 k points and 200 k triangles) and its spherical parameterization.

objects have an equal number of points. We fall again in the first kind-- M to M mapping. It can be seen that M to N mapping is more costly than M to M mapping, simply since we apply M to M mapping in addition to the subdivision step.

M to N mapping has the disadvantage of being time consuming. We use an enhanced implementation of nearest neighbor searching called AABB tree (CGAL, 2010). The AABB tree features a static data structure and algorithms to perform efficient intersection and distance queries against sets of finite 3D geometric objects. Using this static data structure, we build a tree with the vertices of the target object. We then use the tree to find for each point of the source object the corresponding nearest neighbor from the tree.

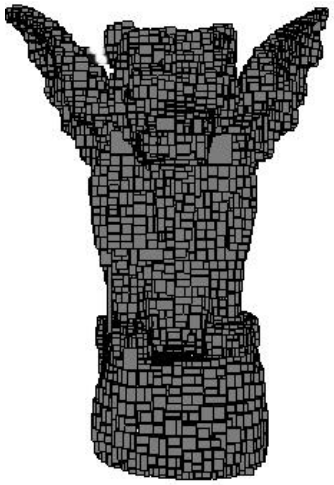

Figure 4. The AABB tree for the set of the spherical triangles of the gargoyle.

RESULTS

The methods described in the preceding sections have been implemented in C++ and using CGAL the Computational Geometry Algorithms Library (CGAL, 2010). The experiments are carried out on a PC with a 2.8 GHZ dual-core processor and 2GB of memory. The input objects are in the form of triangular meshes. Before constructing the spherical parameterization of our objects, we apply the following two preprocessing steps:

(1) Normalize the input models to a cube of unityto get a smooth and a robust transition between the input and output objects. This normalization of each object is performed separately. For each object, the centroid of the object is translated to the origin of the coordinates. Then, calculate the distance, of the farthest point with respect to the origin. Finally, divide the coordinates of each point, $p = (p_x, p_y, p_z)$, by d to get the normalized point $p_{Norm} = \left(\frac{p_x}{d}, \frac{p_y}{d}, \frac{p_z}{d}\right)$.

(2) If the objects are not sufficiently sampled, apply surface subdivision to enhance the mapping between the input and output objects.

Figure 3 shows the spherical parameterization of two triangular meshes the input object is the Stanford bunny and the target object is the Gargoyle. In our experiment, the parameterization time is 79 s for the Gargoyle (100 k points and 200 k triangles). Once the spherical parameterization is constructed for the input and target objects, we construct the AABB tree for the set of spherical triangles of the target object (the gargoyle, in our case) for the mapping step, as shown in Figure 4.

Figure 5. The sequence of morphing the stanford bunny to the gargoyle.

Each point in the spherical parameterization of the Bunny is matched with the closest spherical point in the AABB tree.

Once the matching step is performed, a linear interpolation is performed between the set of points of the Bunny and the corresponding points in the Gargoyle. This interpolation creates the set of frames to be visualized as the final morphing animation. Figure 5 shows the series of the frames that creates the morphing of Bunny-Gargoyle.

CONCLUSIONS

In this paper, we propose a novel technique for 3D mesh morphing capable to interpolate between arbitrary zero-genus objects. The technique can be presented as an animation by creating a series of intermediate objects using the spherical parameterization as a common domain of the source and target objects. We converted both objects into the same spherical domain with suitable modifications to reposition of the points on the sphere. This repositioning prevents the morphing from creating overlapping or clustering of points on the sphere during the mapping step.Then we carry out mesh mapping to realize the morphing process and this happened by using the point-to-point correspondence between the objects of the AABB tree search algorithm. The spherical parameterization which is used to make the mapping step easy since the objects are mapped to their natural parameterization domain.

Conflict of Interest

The author(s) have not declared any conflict of interest.

REFERENCES

Beier T, Neely S (1992). Feature-Based Image Metamorphosis. Proceedings of SIGGRAPH 92:35-42.

CGAL (2010). Computational Geometry Algorithms Library, http://www.cgal.org.

Garland M, Heckbert PS (1997). Surface simplification using quadric error metrics. Proceedings of the 24th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co. 209-216.

Gao P, Sederberg TW (1998). “A Work Minimization Approach to Image Morphing,” Visual Computer, pp. 390-400.

Hoppe H (1996). Progressive meshes. Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, ACM. pp. 99-108.

Kobbelt L (2000). sqrt(3)-subdivision. Computer Graphics (Proc. SIGGRAPH '00). 34:103-112.

Lee AWF, Sweldens W, Schroder P, Cowsar L, Dobkin D (1998). MAPS: multiresolution adaptive parameterization of surfaces. Proceedings of the 25th annual conference on Computer graphics and interactive techniques, ACM: pp. 95-104.

Lee S, Chwa K, Shin SY, Wolberg G (1995). “Image Metamorphosis Using Snakes and Free-form Deformations. Proceedings of SIGGRAPH 95.

Manolis K, Hoomanvassef MDB, Krzysztof G (1997). "3D MOrphing." MIT project (6.837) http://web.mit.edu/manoli/morph/www/morph.html.

Praun E, Hoppe H (2003). "Spherical parametrization and remeshing." ACM. Trans. Graph. 22(3):340-349.

Sederberg TW, GreenWood E (1992). A Physically Based Approach to 2D Shape Blending. Proceedings of SIGGRAPH 92. In Computer Graphics Proceedings, Annual Conferences Series.

Shapira M, Rappoport AA (1995). "Shape Blending Using Star-Skeleton Representation." IEEE Computer Graphics and Applications. pp. 44-50.
Shen L, Makedon F (2006). "Spherical mapping for processing of 3D closed surfaces." Image Vision Computing. 24(7):743-761.
Sompagdee P (2009). Survey of Morphing. C. S. Department, Thammasat University. pp.1-8.
Zhou K, Bao H, Shi J (2004). "3D surface filtering using spherical harmonics." Computer-Aided Design. 36(4):363-375.