DOI: 10.3745/KIPSTA.2009.16-A.2.61

이동 최소 자승법 기반의 빠르고 강체성이 유지되는 3차원 형상 변형 기법

이 정[†] · 김 창 헌[††]

요 약

본 논문에서는 이동 최소 자승법을 기반으로 이미지에 나타나는 객체의 강체 변형을 근사함으로써 자연스러운 변형 결과를 획득할 수 있는 빠른 속도의 3차원 형상 변형 기법을 제안한다. 본 연구에서는 이동 최소 자승법을 강체변형에 맞게 수정하여 각각의 점들이 이동되는 최적의 위치를 계산하는데 소요되는 계산량을 감소시키면서 변형된 결과의 강체성도 그대로 유지하고 있다. 복잡한 기하 형상이라도 점이나 타원형 핸들의 조작을 통해 쉽고 직관적이며 상호작용이 가능한 속도로 변형이 가능하다.

키워드: 형상 변형, 강체 변형, 이동 최소 자승법

Fast and Rigid 3D Shape Deformation Based on Moving Least Squares

Lee Jung[†] · Kim Chang hun[††]

ABSTRACT

We present a fast 3D shape deformation method that achieves smoothly deformed result by approximating a rigid transformation based on moving least squares (MLS). Our modified MLS formulation reduces the computation cost for computing the optimal transformation of each point and still keeps the rigidity of the deformed results. Even complex geometric shapes are easily, intuitively, and interactively deformed by manipulating point and ellipsoidal handles.

Keywords: Shape Deformation, Rigid Transformation, Moving Least Squares

1. Introduction

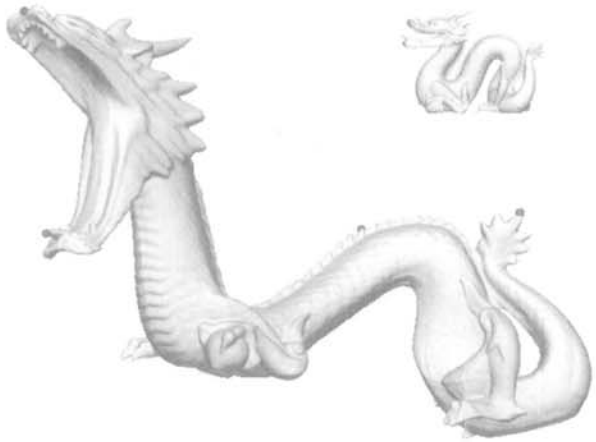
Shape deformation has been an active research topic in computer graphics. 3D shapes can be deformed generally by defining some handles, such as control points and line segments, displacing these handles, and transforming the target surfaces/polygons appropriately according to the displacement of the handles. There are two important issues in shape deformation. One is how to maintain the rigidity between the original and the transformed one and the other is the computation overhead for computing the optimal position of the transformed shape.

As-rigid-as-possible deformation methods are suggested to minimize the distortion in a deformed shape. One appealing approach for achieving as-rigid-as-possible deformation is based on moving least squares (MLS) formulation, which deforms a shape by computing a locally optimal transformation at each vertex of the shape. The MLS approach has some nice properties. It generates smoothly deformed result that is interpolated from user-specified control handles and keeps the sense of rigidity. However, finding such optimal rigid transformation is a non-linear process. So it takes considerable computation time per vertex.

We present a 3D shape deformation method that achieves smoothly deformed result by approximating a rigid transformation based on MLS. (see Fig. 1) Our approximation method is considerably faster than the standard MLS formulation (by orders of magnitude) and also keeps the rigidity of the deformed results.

The point control has been the most popular way of shape deformation due to its simplicity. We discuss the

† 정 회 원 : 고려대학교 BK21 산업단 연구교수
†† 정 회 원 : 고려대학교 정보통신공학부 교수
논문접수 : 2008년 12월 4일
수 정 일 : 1차 2009년 1월 28일
심사완료 : 2009년 2월 2일



(Fig. 1) A dragon model deformed by our method (437,645 vertices. 5 point handles)

use of ellipsoid control as deformation metaphor. It provides flexibility and convenience by specifying rotation and directional scaling as well as position displacements.

2. Related Work

As mentioned above, the researches on shape deformation have focused on two major issues: retaining the visual quality of the original shape and reducing computation costs for interactive manipulation.

Alexa [1] showed that differential coordinates can represent the local features of the shapes independently of their absolute coordinates. Laplacian coordinates [11, 19, 22], gradient vectors [21], and pyramid coordinates [15] are examples of exploiting differential coordinates. The goal of distortion minimization under large deformation can be achieved by formulating differential coordinates in a rotation-invariant way [11, 12]. As-rigid-as-possible transformation minimizes the global distortion by yielding a locally-optimal rigid transformation for each triangle or tetrahedron of the shape [2, 8].

Most numerical solutions using differential coordinates require a linear system, which encodes the relation between a point and its neighbors. Since the time complexity of solving the linear system is O(n2), the computation overhead problem occurs as the number of points increases. The size of the matrix can be effectively reduced by bounding the region of interests [11, 19]. Multiresolution methods maintain a mesh surface at various resolutions and reduce the computational cost by exploiting the coarse-to-fine hierarchical structure [4, 9, 24]. A multi-grid method is a state-of-the-art technology for solving linear systems. Recently, the multi-grid method has been shown to be useful for deforming large surface and volume meshes [18].

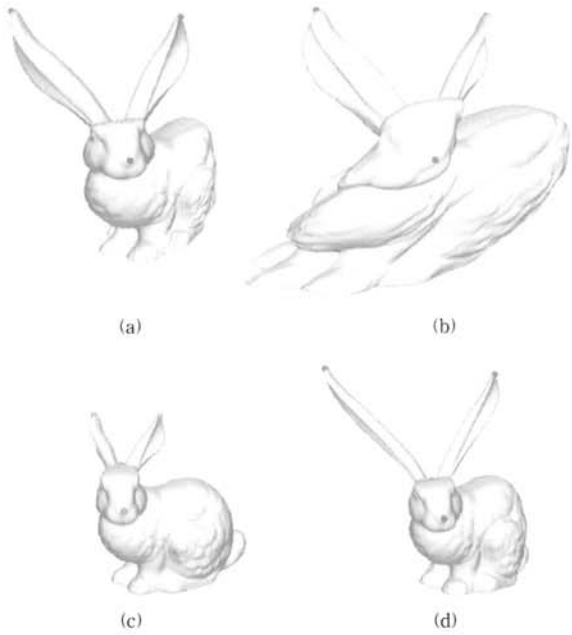
The quality of deformation is closely related to the choice of deformation metaphors. Freeform deformation (FFD) warps the shape using enclosing lattices [14, 17]. A skeleton structure can be exploited for deforming articulated shapes [10, 20]. Twisters [13] and swirl sweepers [3] enable extremely large deformations by controlling the position and orientation of the handles. Deformation methods using radial basis functions can edit the arbitrary region of the space with control points and curves [5]. Carefully-designed implicit vector fields can be employed for volume-preserving deformation [6].

Our approach is a generalization of the image deformation method using MLS proposed by Schaefer et al. [16]. A big advantage of MLS is its capability of selecting the type of transformations to be used for deformation. In 2D space, locally optimal transformations of any of three popular types (affine, similarity, and rigid) can be formulated in a closed, linear form. Generalizing this formulation for 3D shapes[7, 23] is relatively easy with affine transformations. However, MLS using similarity and rigid transformations is not easily generalized for 3D shapes and ends up with non-linear equations. This non-linear formulation is computationally expensive for interactive applications because it should be solved for every point in the shape. Our method uses a linear approximation that can be evaluated very efficiently, while retaining desired properties of MLS. Our method generates smoothly deformed results those are interpolated from the given control points (see Fig. 2).

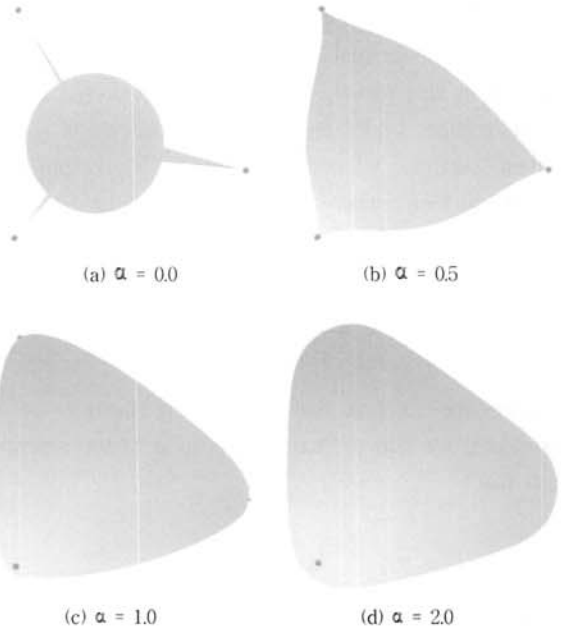
Zhu and Gortler [23] proposed MLS-based deformation technique, but they used only point handles for deformation. The point handle is a powerful way of deformation, because it is very intuitive and simple to manipulate. But the displacement of a point handle is isotropically propagated to neighboring points. The iso-surface from a point handle forms a sphere. So some weird and unexpected results are often occurred. To avoid such a problem, we also consider an ellipsoidal handle that has anisotropic iso-surfaces.

3. Deformation Using Control Points

A 3D shape is defined by a set of points V which describes spatial positions of points in $\mathbb{R}^3$. The shape is deformed by specifying displacement of control points on the shape. Let $P = \{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_m\}$ be a set of control points and $Q = \{\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_m\}$ be the displaced position of the control points. An affine transformation $f(\mathbf{x}) = \mathbf{x}R + T$ maps a point to its deformed position. Function $f_V(\mathbf{x})$ that



(Fig. 2) Variously deformed bunny model. An original shape (a) is deformed by using rigid (b), similarity (c), and affine (d) transformations



(Fig. 3) The results by changing parameter α

transforms a point v is defined such that it minimizes

$$E_v = \sum_i w_i \|\mathcal{F}_v(\mathbf{p}_i) - \mathbf{q}_i\|^2, \tag{1}$$

where the weight w_i of the i^{th} control point is inversely proportional to the distance between v and $\mathbf{p}_i$:

$$w_i = \frac{1}{\|\mathbf{p}_i - \mathbf{v}\|^{2\alpha}}. \tag{2}$$

Parameter α determines how much the control point affects its neighboring points (see Fig. 3).

Let $\mathbf{p}^* = \sum_i w_i \mathbf{p}_i / \sum_i w_i$ and $\mathbf{q}^* = \sum_i w_i \mathbf{q}_i / \sum_i w_i$ be the weighted centroids of P and Q, respectively. Equation (1) can be simplified by removing the translation term $T = \mathbf{q}^* - \mathbf{p}^* R$ (see Schaefer et al. [2006] for details). Then, the deformed position $\mathbf{v}'$ for a point $\mathbf{v}$ is computed by $\mathbf{v}' = (\mathbf{v} - \mathbf{p}^*)R + \mathbf{q}^*$ and Equation (1) becomes

$$E_v = \sum_i w_i \|\hat{\mathbf{p}}_i R - \hat{\mathbf{q}}_i\|^2. \tag{3}$$

where $\hat{\mathbf{p}}_i = \mathbf{p}_i - \mathbf{p}^*$ and $\hat{\mathbf{q}}_i = \mathbf{q}_i - \mathbf{q}^*$. Assuming that $\mathcal{F}_v(\mathbf{x})$ is an affine transformation, we can determine an optimal 3x3 matrix R that minimizes Equation (3) by a straightforward generalization of the image deformation method from Schaefer et al. [16].

3.1 Rigid Deformations

Suppose that $\mathcal{F}_v(\mathbf{x})$ is a rigid transformation, that is, $R^T R = I$. Let R_i be a 3x3 orthogonal matrix that transforms $\hat{\mathbf{q}}_i$ to $\hat{\mathbf{p}}_i$. The rotation axis of R_i is $\hat{\mathbf{u}}_i = \hat{\mathbf{p}}_i \times \hat{\mathbf{q}}_i / \|\hat{\mathbf{p}}_i \times \hat{\mathbf{q}}_i\|$ and its angle of rotation is $\theta_i = \cos^{-1}(\hat{\mathbf{p}}_i \cdot \hat{\mathbf{q}}_i / \|\hat{\mathbf{p}}_i\| \|\hat{\mathbf{q}}_i\|)$. The equation (3) becomes

$$\begin{aligned} E_v &= \sum_i w_i \|\hat{\mathbf{p}}_i R - \hat{\mathbf{p}}_i R_i\|^2 \\ &= \sum_i w_i \|\hat{\mathbf{p}}_i\|^2 \|R - R_i\|^2 \end{aligned} \tag{4}$$

This equation can be approximated by replacing R and R_i by $\log(R)$ and $\log(R_i)$, respectively. The approximated one is equivalent to

$$\tilde{E}_v = \sum_i \mu_i \|\mathbf{r} - \mathbf{r}_i\|^2, \tag{5}$$

where rotation vector $\mathbf{r}_i = \theta_i \hat{\mathbf{v}}_i \in \mathbb{R}^3$ and weight $\mu_i = w_i \|\hat{\mathbf{p}}_i\|^2$. The solution that minimizes the approximated function is

$$\mathbf{r} = \frac{\sum_i \mu_i \mathbf{r}_i}{\sum_i \mu_i}. \tag{6}$$

We can convert $\mathbf{r}$ into a skew-symmetric matrix. The exponential of the skew-symmetric matrix is the approximation of the optimal rotation R at a point $\mathbf{v}$.

3.2 Similarity Deformations

Similarity deformation is a subset of affine transformations that include uniform scaling followed by rigid transformation. Under the similarity deformation, the deformed position $\mathbf{v}'$ for a point $\mathbf{v}$ is computed by $\mathbf{v}' = s(\mathbf{v} - \mathbf{p}^*)R + \mathbf{q}^*$, where s is a scalar scaling factor. We determine the scaling factor to minimize

$$E_v = \sum_i w_i \|\hat{\mathbf{p}}_i\|^2 \|sR - s_i R_i\|^2. \tag{7}$$

Since the rotation and the scaling factors are independent, we can evaluate the scaling factor separately such that

$$s = \frac{\sum_i \mu_i s_i}{\sum_i \mu_i}, \tag{8}$$

where $s_i = \|\hat{\mathbf{q}}_i\| / \|\hat{\mathbf{p}}_i\|$.

4. Ellipsoidal Handles

The ellipsoidal handle is specified by its two focal points $\mathbf{p}_i^s$ and $\mathbf{p}_i^e$. The distance from the ellipsoidal handles to any point v on the shape is computed by elliptic distance, which is the sum of distances to two focal points.

$$\|\mathbf{p}_i^s - \mathbf{v}\| + \|\mathbf{p}_i^e - \mathbf{v}\| = 2a, \tag{9}$$

where $2a$ is the length of the ellipsoid along the major axis (see Fig. 4).

In order to compute the deformation by ellipsoidal handles, we map any point v on the space to a certain point $\mathbf{p}_i^v$ between two focal points. This mapping should be smooth to produce smooth deformation.

The projected point $\mathbf{p}_i^v$ is treated as if it is a point handle for any given v. The ellipsoidal handle can be considered as a point handle that moves between two focal points depending on the position of a shape point to be deformed. The position of $\mathbf{p}_i^v$ is

$$\mathbf{p}_i^v = \mathbf{o} + \frac{c}{a}(\mathbf{v}_p - \mathbf{o}), \tag{10}$$

where o is the center of the ellipsoid, 2c is the distance between two focal points, and $\mathbf{v}_p$ is the projection of v onto the straight line passing through two focal points. The corresponding deformed position $\mathbf{q}_i^v$ is determined such that the inbetweening ratio is preserved:

$$\frac{\|\mathbf{p}_i^v - \mathbf{p}_i^s\|}{\|\mathbf{p}_i^v - \mathbf{p}_i^e\|} = \frac{\|\mathbf{q}_i^v - \mathbf{q}_i^s\|}{\|\mathbf{q}_i^v - \mathbf{q}_i^e\|}. \tag{11}$$

The rotation r_i can be computed as if a point handle $\mathbf{p}_i^v$ is displaced to its deformed position $\mathbf{q}_i^v$ as explained in Section 3.1. The displacement of an ellipsoidal handle includes rotation as well as translation. The rotation r_i^e transforms the direction between two focal points $\overline{\mathbf{p}_i^s \mathbf{p}_i^e}$ to the direction $\overline{\mathbf{q}_i^s \mathbf{q}_i^e}$ at their deformed positions. Then, the deformation by a set of ellipsoidal handles is

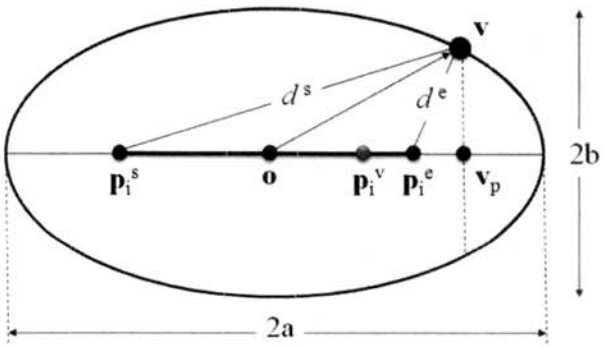
$$\mathbf{r} = \frac{\sum_i \mu_i (r_i + r_i^e)}{\sum_i \mu_i}, \tag{12}$$

which replaces Equation (6) in Section 3.1.

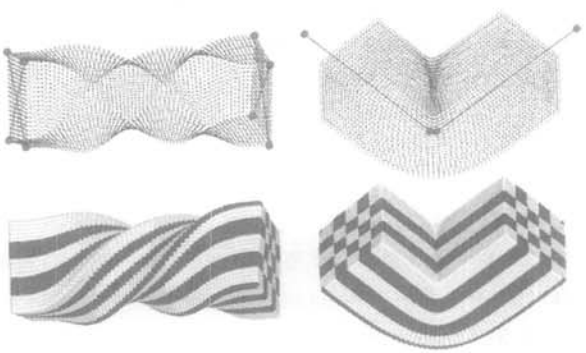
From Equation (12), the ellipsoidal handle enables us to manipulate shapes in complex ways, such as bending and twisting. (see Fig. 5)

5. Topological Distance

The use of Euclidean distances often causes serious artifacts by disregarding the connectivity and topology of the target shape. This is a common problem occurred in space-based deformation approaches.



(Fig. 4) Ellipsoidal handle



(Fig. 5) The bending and twisting of the point-sampled box using ellipsoidal handles

The influence of the control point $\mathbf{p}_i$ that is topologically far from v should be reduced even though it is spatially close to the point. (see Fig. 6) Therefore, we build the weight function in terms of the shortest path distance d_i between v and $\mathbf{p}_i$ over the surface mesh or through the volume of the object to overcome this problem.

$$w_i = \frac{1}{(d_i)^{2\alpha}} \tag{13}$$

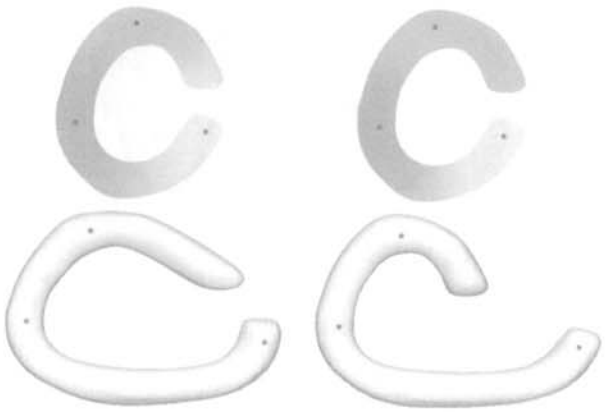
Since weights depend on points in the undeformed state, the shortest distances can be precomputed. We use the Bellman-ford algorithm to compute the shortest distances between control points and all points of the shape. The time complexity of the Bellman-ford algorithm is $O(n_p \cdot e)$, where n_p is the number of points and e is the number of connected neighbors at each point. The memory storage of $O(n_p \cdot n_h)$ is required to maintain the precomputed distances, where n_h is the number of control handles.

6. Experiment Results

We developed an interactive shape deformation system using C++, MFC and the STL library, and it was tested on an Intel Core2Quad 2.40 GHz CPU with 2 GB RAM.

From user's point of view, our algorithm is divided into three steps. These steps are summarized in <Table 1> Step 2 in the table is a preprocessing phase. While the user drags a control handle with input devices at runtime, the system updates the deformed shape V interactively by executing Step 3.

<Table 2> shows that the portion of preprocessing time



(Fig. 6) Comparison between spatial (left) and topological (right) distance weights

〈Table 1〉 The computation sequence

	Euclidean points	*Topological points*	*Ellipsoids*
Step 1: A shape is loaded			
Input :	V		
Step 2: Control handles are created or removed			
Input :	$P = \{\mathbf{p}_i\}$		
	-	$\{d_i\}$	-
	-	-	$\{\mathbf{p}_i^{\nu}\}$
	$\{w_i\}$ by Eq(2)	$\{w_i\}$ by Eq(13)	$\{w_i\}$ by Eq(2)
	$\{\mu_i\}$		
	$\mathbf{p}^*$		
	$\{\hat{\mathbf{p}}_i\}$		
Step 3: Control handles are dragged			
Input :	$Q = \{\mathbf{q}_i\}$		
	-	-	$\{\mathbf{q}_i^{\nu}\}$
	$\mathbf{q}^*$		
	$\{\hat{\mathbf{q}}_i\}$		
	$\{\mathbf{r}_i\}$		
	v'		
Output :	V'		

〈Table 2〉 Computation costs. EP: Euclidean distance point handles, TP: Topological distance point handles, and E: ellipsoidal handles

Model	Bunny	Dragon	Armadillo	
Points	35947	35820	20002	
Handle	5EP	5TP	5TP	5E
Step 2 {di}	-	65.4	24.8	-
Step 2	25.9	27.6	21.2	31.0
Total	25.9	93.0	46.0	31.0
Step 3	45.7	46.3	25.7	27.6
Unit time	2.5e-4	2.58e-4	2.5e-4	2.7e-4
Pre-proc.	36%	66%	64%	53%

in the total computation time. The performance is mainly affected by the type of control handles. The use of topological distances requires more computation at Step 2 because the shortest distances should be computed at the preprocessing phase.

<Table 3> shows that the computation time is linearly proportional to the numbers of points n_p and the control handlesn_h. The unit time is the runtime computation cost divided by $n_p * n_h$. This linear runtime computation cost allows us to determine the maximum number of points that can be interactively manipulated at a desired frame rate. For example, if we want to manipulate a shape with 5 control points at 25 FPS, the maximum number of

〈Table 3〉 Computation costs depending on the numbers of points and control handles

#points(p)	#handle(h)	#time	Unit time
10,000	5	11.9	2.38e-4
20,000	5	24.6	2.46e-4
40,000	5	50.6	2.53e-4
80,000	5	99.8	2.49e-4
80,000	10	190.7	2.38e-4
80,000	15	281.4	2.34e-4
3,000,000	5	3273.0	2.18e-4

points is about 32,000.

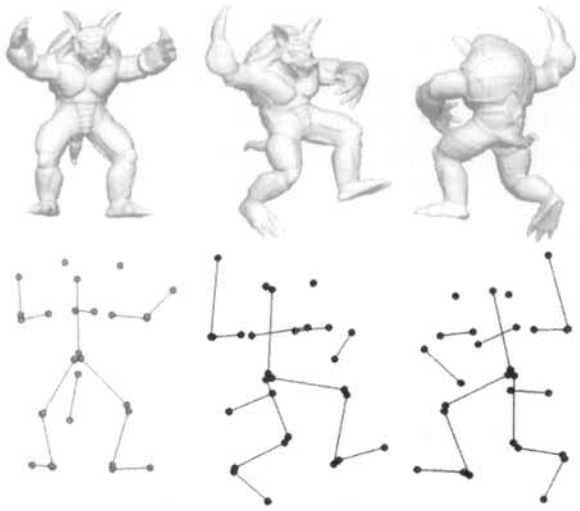
The memory requirement is also linearly proportional to the numbers of points and the control handles. We must maintain $\{w_i\}$ and $\{\hat{\mathbf{p}}_i\}$ for each control handle. Scalar values are represented as 4 byte floating points, and a point in 3D space has x, y and z elements. The total memory requirement is $24n_p + 16n_pn_h$.

7. Discussion

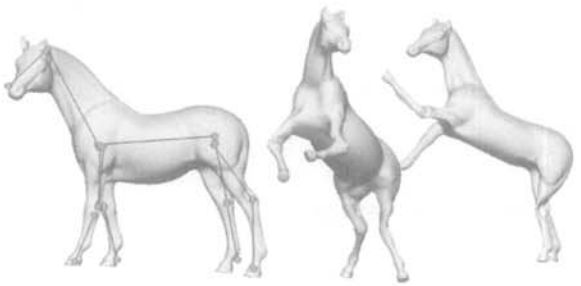
We present a fast 3D shape deformation method that approximates MLS. Our method inherits the nice properties of MLS-based deformation techniques. Our result is smoothly deformed and keeps the rigidity of the shape. The control handles are interpolated according to the various types of transformations.

Many shape deformation methods has suffered from solving $n \times n$ linear systems. LU decomposition is the most popular method for solving the linear system, because it is computed at a preprocessing phase and then the linear system can be solved efficiently at runtime using forward and backward substitutions. The time and space complexity of solving a linear system are $O(n_p^2)$. This complexity is much heavier than ours, $O(n_p \cdot n_h)$, because n_h is usually much smaller than n_p. Shi et al. [18] compared the performance of several linear system solvers. They reported that the fastest multi-grid solver takes about 40,000ms for manipulating 3.44e6 points and 740MB memory storage is required. Our method takes just 3,273 ms with 5 control handles for manipulating 3e6 points <see Table 3> and 537 MB memory storage is required. Our method shows the much better performance than the previous methods.

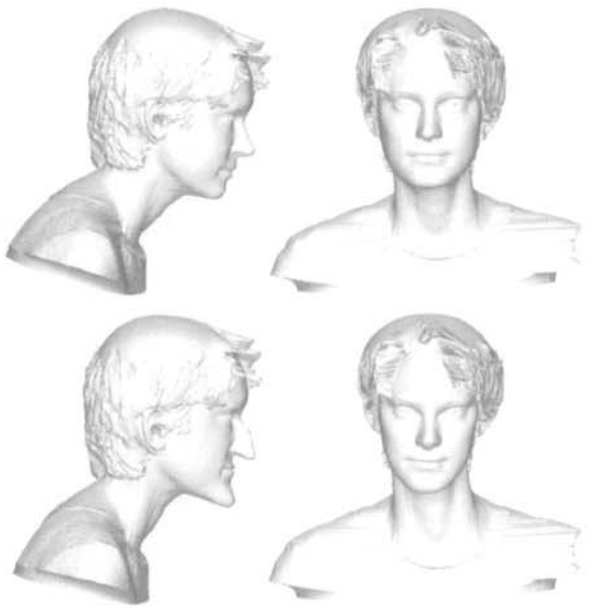
The dragon model in (Fig. 1) consists of 437,645 points and 871,414 faces. The computation cost (68ms) for deforming the model is relatively smaller than the computation cost (153ms) for rendering the model. So we can interactively manipulate it. The (Fig. 7-9) show us the various deformed results by our method. Especially, (Fig. 7) shows the character animation results by specifying and manipulating handles like its skeleton.



(Fig. 7) Deformed armadillo model
(20,002 points. 13 ellipsoidal handles and 2 point handles)



(Fig. 8) Deformed horse model
(48,485 points. 13 ellipsoidal handles)



(Fig. 9) Facial deformation with a set of point handles

References

[1] Alexa, M., "Differential Coordinates for Local Mesh Morphing and Deformation," The Visual Computer 19, 2, 105-114, 2003.

[2] Alexa, M., Cohen-Or, D., and Levin, D., "As-rigid-as-possible Shape Interpolation," In Proceedings of SIGGRAPH '00, 157-164, 2000.

[3] Angelidis, A., Cani, M.-P., Wyvill, G., and King, S. "Swirling-Sweepers: Constant Volume Modeling, In Computer Graphics and Applications," 12th Pacific Conference on (PG'04), 10-15, 2004.

[4] Botsch, M. and Kobbelt, L. "Multiresolution Surface Representation Based on Displacement Volumes," Computer Graphics Forum 22, 3, 483-491, 2003.

[5] Botsch, M. and Kobbelt, L. "Real-time Shape Editing Using Radial Basis Function," Computer Graphics Forum 24, 3, 611-621, 2005.

[6] Funck,W., Theisel,H., and Seidel,H.-P. "Vector Field Based Shape Deformations," ACM Transaction on Graphics 25, 3, 1118-1125, 2006.

[7] Horn, B.K.P., Hilen, H.M., and Negahdaripour, S. "Closed Form Solution of Absolute Orientation Using Orthonormal Matrices," Journal of the Optical Society of America. A, Optics and Image Science 5, 7, 1127-1135, 1988.

[8] garashi, T., Moscovich, T., and Hughes, J. F. "As-rigid-as-possible Shape Manipulation," ACM Transaction on Graphics 24, 3, 1134-1141, 2005.

[9] Kobbelt, L., Compagna, S., Vorsatz, J., and Seidel, H.-P. "Interactive Multi-resolution Modeling on Arbitrary Meshes," In Proceedings of ACM SIGGRAPH '98, 105-114, 1998.

[10] Lewis, J.P., Cordner, M., and Fong, N. "Pose Space Deformation: a Unified Approach to Shape Interpolation and Skeleton-driven Deformation," In Proceedings of ACM SIGGRAPH '00, 165-172, 2000.

[11] Lipman, Y., Sorkine, O., Cohen-Or, D., Levin, D., Rossl, C., and Seidel, H.-P. "Differential Coordinates for Interactive Mesh Editing," In Proceedings of Shape Modeling International, 181-190, 2004.

[12] Lipman, Y., Sorkine, O., Levin, D., and Cohen-Or, D. "Linear Rotation-invariant Coordinates for Meshes," ACM Transaction on Graphics 24, 3, 479-487, 2005.

[13] Llamas, I., Kim, B., Gargus, J., Rossignac, J., and Shaw, C. "Twister: a Space-warp Operator for the Two-handed Editing of 3D Shapes," ACM Transaction on Graphics 22, 3, 663-668, 2003.

[14] MacCracken, R. and Joy, K. "Free-form Deformations with Lattices of Arbitrary Topology," In Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, ACM Press, New York, 181-188, 1996.

[15] Sheffer, A. and Kraevoy, V. "Pyramid Coordinates for Morphing and Deformation," In Proceedings of the 2nd International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT '04), 68-75, 2004.

[16] Schaefer, S., McPhail, T., and Warren, J. "Image Deformation Using Moving Least Squares," ACM Transaction on Graphics 25, 3, 533-540, 2006.

[17] Sederberg, T. and Parry, S. "Free-form Deformation of Solid Geometric Models," In Proceedings of ACM SIGGRAPH '86, 151-160, 1986.

[18] Shi, L., Yu, Y., Bell, N., and Feng, W.-W. "A Fast Multigrid Algorithm for Mesh Deformation," In ACM Transaction on Graphics 24, 3, 1108-1117, 2006.

[19] Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rossl, C., and Seidel, H.-P. "Laplacian Surface Editing," In Proceedings of Eurographics Symposium on Geometry Processing, 179-188, 2004.

[20] Yoshizawa, S., Belyaev, A.G., and Seidel, H.-P. "Free-form Skeleton-driven Mesh Deformations," In Proceedings of the 8th ACM Symposium on Solid Modeling and Applications 2003, 247-253, 2003.

[21] Yu, Y., Zhou, K., Xu, D., Shi, X., Bao, H., Gui, B., and Shum, H.-Y. "Mesh Editing with Poisson-based Gradient Field Manipulation," ACM Transaction on Graphics 23, 3, 641-648, 2004.

[22] Zhou, K., Huang. J., Snyder, J., Liu, X., Bao, H., Gui, B., and Shum, H.Y. "Large Mesh Deformation Using the Volumetric Graph Laplacian," ACM Transaction on Graphics 24, 3, 496-503, 2005.

[23] Zhu, Y. and Gortler, S.J. "3D Deformation Using Moving Least Squares," Harvard Computer Science Technical Report: TR-10-07, 2007.

[24] Zorin, D., Schröder, P., and Sweldens, W. "Interactive Multiresolution Mesh Editing," In Proceedings of ACM SIGGRAPH 1997, 259-269, 1997.

이 정

e-mail : airjung@gmail.com

2000년 고려대학교 컴퓨터학과(학사)

2002년 고려대학교 컴퓨터학과(이학석사)

2006년 고려대학교 컴퓨터학과(이학박사)

2006년 고려대학교 BK21 산업단 연구교수

2006~2008년 삼성전자 통신연구소 책임연구원

2008년~현 재 고려대학교 BK21 산업단 연구교수

관심분야 : 컴퓨터 그래픽스, 컴퓨터 비전

김 창 헌

e-mail : chkim@korea.ac.kr

1979년 고려대학교 경제학과(학사)

1987년 한양대학교 전산학과(이학석사)

1993년 Univ. of Tsukuba 전자정보학과 (이학박사)

1979년~1987년 한국과학기술연구소(KIST) 연구원

1987년~1995년 한국과학기술원 시스템공학연구소 책임연구원

1989년~1990년 일본 동경공업대학 객원교수

2003년~2004년 UCLA 객원교수

2005년~2007년 고려대학교 정보통신대학 학장

1995년~현 재 고려대학교 정보통신대학 정보통신공학부 교수

2008년~현 재 한국컴퓨터그래픽스학회 회장

관심분야 : 컴퓨터 그래픽스, 컴퓨터 비전, 유체 시뮬레이션