A Laplacian Based Approach for Free-Form Deformation of Sparse Low-degree IMplicit Surfaces

Yutaka Ohtake
RIKEN
VCAD Modeling Team

Takashi Kanai
The University of Tokyo
Graduate School of Arts and Sciences

Kiwamu Kase
RIKEN
VCAD Modeling Team

Abstract

Sparse Low-degree IMplicit (SLIM) surface [11] is a recently developed non-conforming surface representation. In this paper, a method for free-form deformation of SLIM surfaces is presented. By employing a Laplacian based mesh deformation technique, a global deformation is realized as movements of the centers of the local function supports. A graph connectivity for defining the Laplacians is simply created using inclusion of other centers in the supports. By following the global deformation according to the movements of the centers, we update each local function and its support size via several times of local least square fittings by using the new positions of the neighboring centers. Since the additional computations for updating local functions are very computationally cheap, the proposed SLIM surface deformation achieves an interactive surface deformation similarly to Laplacian based mesh deformation techniques. Further, because a SLIM surface requires a smaller number of elements than a mesh for representing the same geometrical details, the computational effort for a global Laplacian based deformation is dramatically reduced.

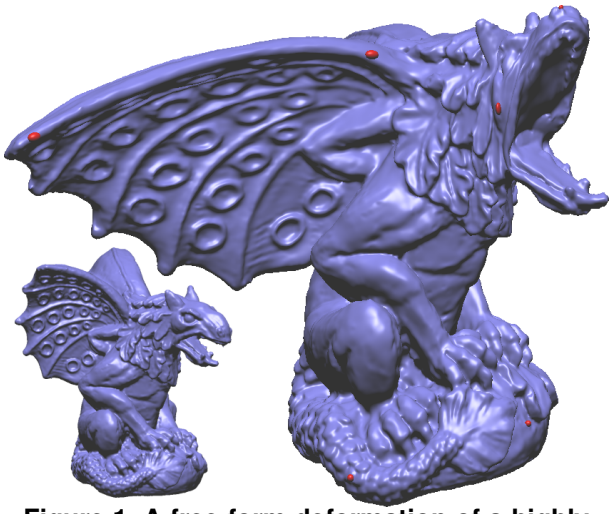

Figure 1. A free-form deformation of a highly detailed model. Left: The gargoyle model represented as a SLIM surface (69K quadratic primitives). Right: A deformed model. The red ellipsoids indicate control points.

1. Introduction

Sparse Low-degree IMplicit (SLIM) surface [11] is a recently developed non-conforming surface representation. A SLIM surface consists of a set of spherically supported quadratic/cubic polynomial implicit functions. Using SLIM surfaces, we can achieve an economical surface representation and high-quality rendering with nice derivative estimations on a surface. Fig. 2 compares rendering qualities of the Stanford bunny mesh and its approximation by a SLIM surface. We can clearly see an advantage of the SLIM surface rendering [11] over the conventional triangle mesh with Phong shading. Further the radius of each spherical support can be adaptively defined according to an approximation error between the local implicit surface and the original mesh vertices as shown in Fig. 3.

In this paper, we propose a novel method for free-form deformation of SLIM surfaces. We specifically consider the application of so called the *surface deformation* based on differential coordinates [2, 18, 9, 20, 16, 21, 10, 22], which is recently an active research topic in mesh modeling, to SLIM surfaces. In those approaches, the local details of a surface are first encoded as implicit and rotation-invariant representations by using differential coordinates such as Laplacians. According to the deformation of a sparse set of control points specified by the user, a surface in the ROI (region of interest) is reconstructed so as to preserve the details of a surface. Consequently, a detail-preserving deformation of surfaces is achieved compared to the classical *spatial deformation* techniques ([5], [15], etc.).

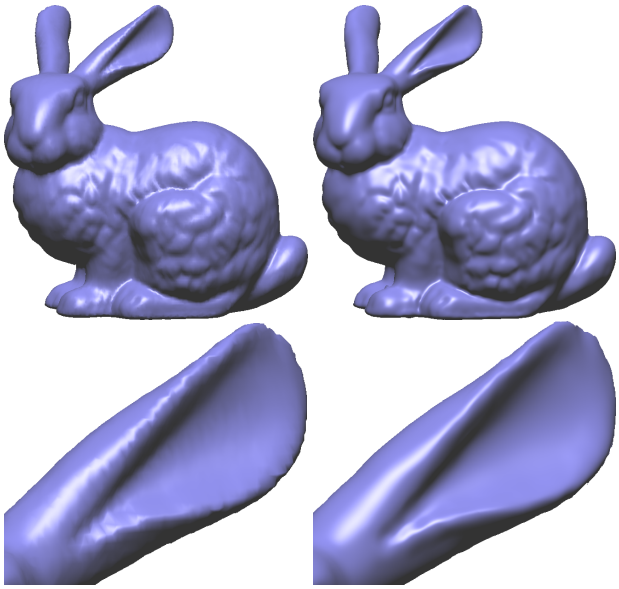

Figure 2. Left: the Stanford bunny mesh (35K triangles) rendered by Phong shading. Right: A SLIM surface approximation of the mesh vertices (8K quadratic primitives). While almost the same geometrical details are approximated, the polygonal artifacts are eliminated.

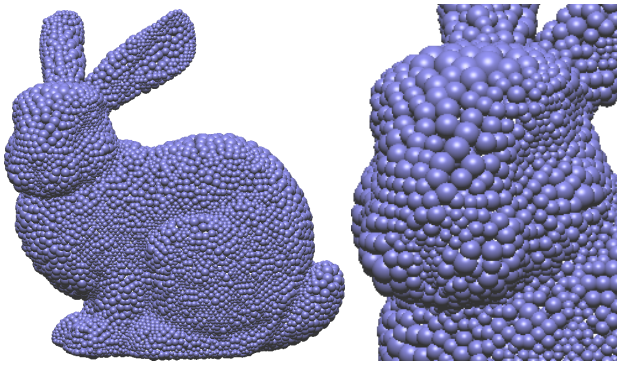

Figure 3. Adaptively sized spherical supports of the SLIM surface shown in the right image of Fig. 2. The actual spheres are twice as large as the spheres in the images.

Our contribution here is the introduction of such the surface deformation to SLIM surfaces. Our approach can be positioned as the primary approach of the surface deformation for point set surfaces. All of previously proposed approaches for deforming point set surfaces [13, 1] or implicit surfaces [4, 3] were considered with only classical spatial deformations. In such spatial deformations, the surface details are generally lost especially during the large deformations.

In the deformation, we should consider the several technical issues: First, a SLIM surface does not have the connectivity information between neighbor points. To apply the surface deformation technique, we then have to construct such connectivity before the deformation. Second, a SLIM surface has a set of local supports. During the deformation of a surface, incremental updates of such supports have to be considered.

Here we divide the computation of surface deformation for SLIM surfaces into two optimization problems: One is the *global* optimization of the center positions of supports and the propagation of the local transformations. The other is the *local* optimization of each implicit function in local supports.

Our global/local optimization method can be regarded as a simple multi-resolution technique. The rough geometry of a surface, which is represented by the center positions of supports, is deformed by a global optimization (a Laplacian based approach). Each local optimization deforms the detailed geometry which is encoded by the polynomial implicit function. The Laplacian based global optimization is efficiently performed after factorizing the Laplacian matrix [17]. Our local optimization for each local function consists of small linear least square fittings. Thus we achieve interactive surface deformations.

The most important benefit of the proposed method is demonstrated in Fig. 1. The gargoyle model is originally represented by over a million of triangles (863K vertices). By using Laplacian based mesh deformation techniques, a global deformation of such a highly detailed geometry is a challenging task because it is hard to factorize a large Laplacian matrix even if the matrix is highly sparse. In contrast, a SLIM surface requires only 69K quadratic primitives for representing almost the same geometrical details as shown in the left image of Fig. 1. This fact means that the size of the Laplacian matrix is much smaller, then the computational effort for factorizing Laplacian matrix is dramatically reduced. In other words, comparing with mesh-based approaches we can handle more highly detailed geometry using our SLIM-based deformation technique.

2. Free-form deformation of SLIM surfaces

A SLIM surface is defined as a set of compactly supported implicit surfaces each of which is composed of the following local parameters of *function supports* and a polynomial implicit function. (see also the right image of Fig. 4):

- Center position $\mathbf{c} = (c^x, c^y, c^z)^T$ of a local support.
- Local orthogonal frame $(\mathbf{d}^u, \mathbf{d}^v, \mathbf{d}^w)$ for a local support. The unit vector $\mathbf{d}^w$ indicates the normal direction of

the local function. The tangential directions are represented by the unit vectors $\mathbf{d}^u$ and $\mathbf{d}^v$. The local coordinates (u,v,w) for a point $\mathbf{x}$ are calculated by $u(\mathbf{x}) = \mathbf{d}^u \cdot (\mathbf{x}-\mathbf{c})$, $v(\mathbf{x}) = \mathbf{d}^v \cdot (\mathbf{x}-\mathbf{c})$, and $w(\mathbf{x}) = \mathbf{d}^w \cdot (\mathbf{x}-\mathbf{c})$.

- Radii of the ellipsoidal support, r^u, r^v, and r^w in the directions $\mathbf{d}^u$, $\mathbf{d}^v$, and $\mathbf{d}^w$, respectively. Given a point $\mathbf{x}$, we can check whether $\mathbf{x}$ is included in the support by $1-(u(\mathbf{x})/r^u)^2-(v(\mathbf{x})/r^v)^2-(w(\mathbf{x})/r^w)^2 > 0$.

- Local implicit function $f(\mathbf{x})$ whose zero-level represents a local approximation of the surface. It is a bivariate height function in implicit form, i.e. $f(\mathbf{x}) = w(\mathbf{x}) - h(u(\mathbf{x}),v(\mathbf{x}))$. For example, $h(u,v) = Au^2 + Buu + Cv^2 + Du + Ev + F$ in the quadratic primitive case.

In the original SLIM surfaces [11], spherically supported implicit functions are used. In this work, we use ellipsoidal supports because deforming a SLIM surface causes local stretches in the tangential directions as shown in the left image of Fig. 4.

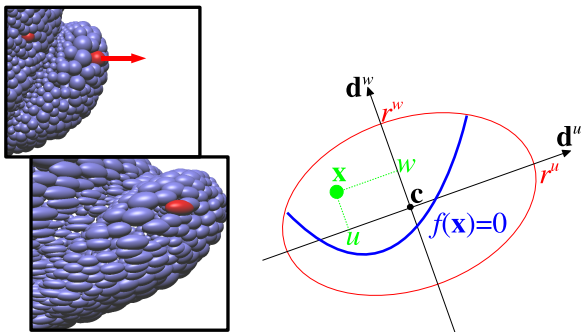

Figure 4. Left: A deformation of the bunny's tail. Right: The notations for an ellipsoidally supported function. The direction $\mathbf{d}^v$ is omitted in the image.

Our deformation consists of the following four steps. Fig. 5 shows how each step affects a SLIM surface.

1. Creation of a neighbor graph of the centers. The centers are connected using inclusion of other centers in the supports ((b) in Fig. 5).

2. Global deformation realized by the movements of the centers by employing a Laplacian based mesh deformation technique ((c) in Fig. 5).

3. Update the local frame and radii of each function support. The shape of the support is locally deformed according to the movements of its neighboring centers ((d) in Fig. 5).

4. Re-fitting each of the local implicit functions. The function is also locally deformed according to the movements of their neighboring centers ((e) in Fig. 5).

2.1. Creating a neighbor graph

We firstly create a graph by connecting the centers $\mathscr{C} = \{\mathbf{c}_i | 1 \le i \le N\}$. This graph is used in the both global and local deformations. Two centers $\mathbf{c}_{i_1}$ and $\mathbf{c}_{i_2}$ are connected by an edge if either $\mathbf{c}_{i_1}$ is included in the i_2-th support or $\mathbf{c}_{i_2}$ is included in the i_1-th support. More formally, the conditions can be written as the following inclusion tests of ellipsoids;

$$1-(u_{i_2}(\mathbf{c}_{i_1})/r^u_{i_2})^2-(v_{i_2}(\mathbf{c}_{i_1})/r^v_{i_2})^2-(w_{i_2}(\mathbf{c}_{i_1})/r^w_{i_2})^2 > 0$$
$$\text{or } 1-(u_{i_1}(\mathbf{c}_{i_2})/r^u_{i_1})^2-(v_{i_1}(\mathbf{c}_{i_2})/r^v_{i_1})^2-(w_{i_1}(\mathbf{c}_{i_2})/r^w_{i_1})^2 > 0.$$

In order to perform efficient range queries, the coordinates of the centers are sorted using a *k*d-tree.

2.2. Global deformation by movements of centers

Using the graph created in the above procedure, we can define a graph Laplacian $\Delta(\mathbf{c})$ at a center $\mathbf{c}$.

$$\Delta(\mathbf{c}) = \mathbf{c} - \frac{1}{\sum_j \omega_j} \sum_j \omega_j \mathbf{c}_j,$$

where $\mathbf{c}_j$ is a neighboring center of $\mathbf{c}$ and ω_j is a weight for $\mathbf{c}_j$. In our method equal weights are used, i.e. $\omega_j = 1$. An alternative choice of the weighting scheme is to take an inverse of the edge length [7] which is effective in the case the graph vertices are distributed irregularly. Since the centers of a SLIM surface are regularly distributed, equal weights even work well. Once a graph Laplacian is defined, techniques developed for mesh deformations can be adapted to move the center positions. For the Laplacian based deformation technique, we employ a combination of two methods proposed in [9] and [21].

In our experimental system, a user specifies a set of control points $\mathscr{D} = \{\mathbf{c}_k | 1 \le k \le M\}$ which is a subset of $\mathscr{C}$ (the index k for $\mathscr{D}$ indicates a different enumeration from i for $\mathscr{C}$) and their desired positions $\{\mathbf{p}_k | 1 \le k \le M\}$. Additionally, as demonstrated in Fig. 6 local transformations denoted by $\{\mathbf{T}_k | 1 \le k \le M\}$ are specified at the control points. Each local transformation is propagated to all centers, then at each center the propagated transformations are averaged. We denote the averaged transformation at each center $\mathbf{c}_i$ as a 3×3 matrix $\mathbf{R}_i$.

Similarly to [9], we solve the following global least square fitting in order to obtain new center positions $\tilde{\mathscr{C}} = \{\tilde{\mathbf{c}}_i | 1 \le i \le N\}$.

$$\sum_{i=1}^{N} \|\Delta(\tilde{\mathbf{c}}_i) - \mathbf{R}_i \Delta(\mathbf{c}_i)\|^2 + \sum_{k=1}^{M} (W_k \|\tilde{\mathbf{c}}_k - \mathbf{p}_k\|)^2 \to \min, \quad (1)$$

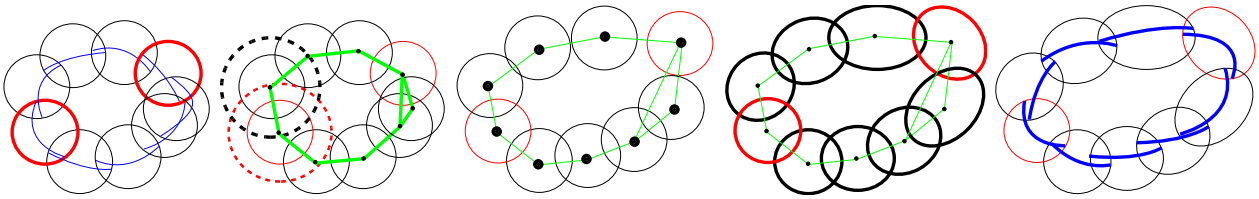

Figure 5. Algorithm overview. The drawn supports are smaller than the exact ones in order to simply the images. The exact supports are shown in the dashed lines in the second image. (a) A user selects two control points which are the centers of the supports colored in red. (b) The centers are connected by the green edges using inclusion of other centers in the supports. (c) The center positions are moved by a Laplacian based approach. (d) The shapes of the supports are updated by the movements of their neighboring centers. (e) The local implicit functions are also updated.

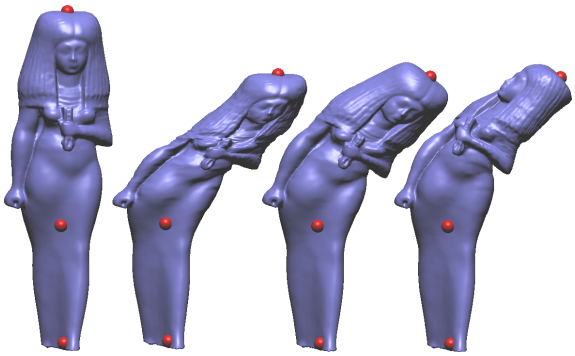

Figure 6. An example of local transformations in the SLIM-based deformations. A control point on the top is moved as shown in the middle left image. In the right two images, rotations are specified at such a control point.

where W_k is a weight for the k-th control point. In our experiments we set $W_k = N/M$ which equalizes the first and second terms of (1).

By applying the normal equation method [14], (1) can be solved as a linear least square problem, i.e. solving $N \times N$ sparse linear systems. The sparsity is due to the local support of Δ. The matrix depends on only the graph connectivity and the choice of $\mathscr{D}$. Thus, if we factorize the matrix once, the factorization can be reused during the changes of $\{\mathbf{p}_k\}$ and $\{\mathbf{T}_k\}$. The new center positions can be quickly obtained by three times of back-substitutions (x, y, and z components). As recommended in [17], we use TAUCS library [19] which includes very efficient factorization algorithms.

To propagate $\{\mathbf{T}_k\}$, we use the method proposed in [21]. In their method, a harmonic map denoted by $\{h_i^m | 1 \le i \le N\}$ is created for each control point $\mathbf{c}_m$. The values of the harmonic map are used as the weights in propagating $\mathbf{T}_m$. The harmonic map takes maximum at $\mathbf{c}_m$ ($h_m^m = 1$) and minimum at the other control points ($h_k^m = 0$ for $m \neq k$). Fig. 7 shows an example of the harmonic map from the control point on the shoulder to the other two control points. Notice that we also have the other two harmonic maps (from the head and from the tail) which are not shown in Fig. 7.

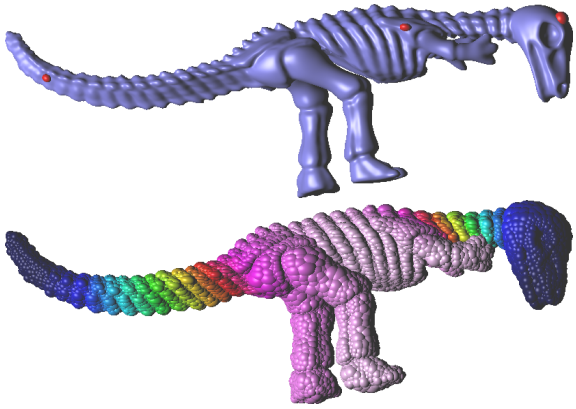

Figure 7. Top: A deformed dinosaur model. Bottom : The scalar field for propagating the local transformation from the control point on the shoulder. Pink corresponds to the highest weight and dark-blue corresponds to the lowest weight.

In our implementation, each harmonic map is computed by solving the following optimization problem.

$$\sum_{i=1}^{N} \Delta(h_i^m)^2 + \sum_{k=1}^{M} \left(W_k \left(h_k^m - \delta_{mk}\right)\right)^2 \rightarrow \min, \qquad (2)$$

where δ_{mk} equals to 1 if $m = k$, otherwise 0. After solving (2) M times, M harmonic maps are created. At each center $\mathbf{c}_i$, the scalars $\{W_m h_i^m | 1 \le m \le M\}$ are assigned. Each scalar $W_m h_i^m$ are used as the weight for $\mathbf{T}_m$ in the averaging procedure. As claimed in [21], to solve (2) we can share the same sparse matrix used for solving (1), thus the additional computations are only M times of back-substitutions.

Note that the formulation (2) is slightly different from [21]. Actually, solving (2) does not create an exact har-

monic map since the conditions $\{\Delta(h_k) = 0\}$ at the control points break the harmonic property. Our scalar field has no guarantee that the values of the map is bounded by $[0,1]$. For resolving this problem, we simply truncate the values to $[0,1]$.

2.3. Updating local frame and support size

Using the new center positions, we update the local frame $(\mathbf{d}^u, \mathbf{d}^v, \mathbf{d}^w)$ and the support radii r^u, r^v, and r^w of each function support. To update such local parameters at $\mathbf{c}$, only the movements of the local neighbors, i.e. $\{\mathbf{c}_j\}$ and $\{\tilde{\mathbf{c}}_j\}$, are used.

In normal direction The normal direction $\mathbf{d}^w$ is updated by $\tilde{\mathbf{d}}^w = (\mathbf{J}^{-T}\mathbf{d}^w)/\|\mathbf{J}^{-T}\mathbf{d}^w\|$ where the 3×3 matrix $\mathbf{J}$ is the Jacobian matrix at $\mathbf{c}$ and $-T$ means the transposed inverse matrix. An approximation of $\mathbf{J}$ is obtained by the following least square fitting.

$$\sum_j \|\mathbf{J}(\mathbf{c}_j - \mathbf{c}) - (\tilde{\mathbf{c}}_j - \tilde{\mathbf{c}})\|^2 \to \min \qquad (3)$$

After a simple calculation, the solution of (3) is directly obtained by $\mathbf{J}^{-T} = \mathbf{A}\mathbf{B}^{-1}$ where two 3×3 matrices are given by

$$\mathbf{A} = \sum_j (\mathbf{c}_j - \mathbf{c})(\tilde{\mathbf{c}}_j - \tilde{\mathbf{c}})^T, \quad \mathbf{B} = \sum_j (\tilde{\mathbf{c}}_j - \tilde{\mathbf{c}})(\tilde{\mathbf{c}}_j - \tilde{\mathbf{c}})^T.$$

The matrix inversion of $\mathbf{B}$ is not possible if $\tilde{\mathbf{c}}$ and all $\{\tilde{\mathbf{c}}_j\}$ lie on the same plane. On such a flat region, a new normal direction $\tilde{\mathbf{d}}^w$ can be simply computed by the covariance analysis of $\mathbf{B}$ [8].

The support radius in the normal direction is kept as $\tilde{r^w} = r^w$ since the local movements of the centers mainly consist of the local stretching in the tangential directions.

In tangential directions It is possible to update the remaining two tangential directions and their support radii using $\mathbf{J}$ as demonstrated in [6]. In their case very accurate $\mathbf{J}$ can be obtained in the analytical way since an analytical formula of the spatial deformation is provided. In our case $\mathbf{J}$ is numerically estimated by solving (3). According to our experiments, our $\mathbf{J}$ is sometimes not accurate enough to use for updating the radii in the tangential directions. Thus, we perform another fitting as follows.

The boundary of the old support on the tangent plane can be written as the zero-level of $\phi(u,v) = 1 - (u/r^u)^2 - (v/r^v)^2$. Similarly, the boundary of the new support is the zero-level of $\tilde{\phi}(\tilde{u},\tilde{v}) = 1 - (\tilde{u}/\tilde{r^u})^2 - (\tilde{v}/\tilde{r^v})^2$. As illustrated in Fig. 8, our fitting is aimed to keep the values of ϕ at each of the old center positions as much as possible at the new center positions. Thus the following least square fitting is applied.

$$\sum_j \left(\phi(u(\mathbf{c}_j), v(\mathbf{c}_j)) - \tilde{\phi}(\tilde{u}(\tilde{\mathbf{c}}_j), \tilde{v}(\tilde{\mathbf{c}}_j))\right)^2 \to \min \qquad (4)$$

In order to solve the above problem as a linear least square problem, $\tilde{\phi}$ is rewritten as $\tilde{\phi}(s,t) = \alpha s^2 + 2\beta st + \gamma t^2$. The temporal tangential coordinates (s,t) are computed by $s(\mathbf{x}) = \mathbf{d}^s \cdot (\mathbf{x} - \tilde{\mathbf{c}})$ and $t(\mathbf{x}) = \mathbf{d}^t \cdot (\mathbf{x} - \tilde{\mathbf{c}})$ where two orthogonal unit vectors $\mathbf{d}^s$ and $\mathbf{d}^t$ are perpendicular to $\tilde{\mathbf{d}}^w$.

After solving (4) with respect to α, β, and γ, we analyze them by the following eigenvalue decomposition.

$$\begin{pmatrix} \alpha & \beta \\ \beta & \gamma \end{pmatrix} = \begin{pmatrix} \xi_1^s & \xi_1^t \\ \xi_2^s & \xi_2^t \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \begin{pmatrix} \xi_1^s & \xi_2^s \\ \xi_1^t & \xi_2^t \end{pmatrix}$$

The two unit eigenvectors (ξ_1^s, ξ_1^t) and (ξ_2^s, ξ_2^t) indicate two principal orthogonal axes of the ellipse $\tilde{\phi} = 0$. Thus, the new tangential directions are given by $\tilde{\mathbf{d}}^u = \xi_1^s \mathbf{d}^s + \xi_1^t \mathbf{d}^t$ and $\tilde{\mathbf{d}}^v = \xi_2^s \mathbf{d}^s + \xi_2^t \mathbf{d}^t$. The eigenvalues λ_1 and λ_2 give the squared inverse of radii of $\tilde{\phi} = 0$. So, we get the new support radii as $\tilde{r^u} = 1/\sqrt{\lambda_1}$ and $\tilde{r^v} = 1/\sqrt{\lambda_2}$.

If we fail to solve (4) or either $\lambda_{1,2}$ is negative (very rare case in our experiments), then we switch the fitting procedure from an ellipse to a circle, i.e. $\phi(\tilde{u},\tilde{v}) = 1 - \alpha\tilde{u}^2 - \alpha\tilde{v}^2$. Fitting a circle is never failed unless the all neighboring centers $\{\tilde{\mathbf{c}}_j\}$ lie on the $\tilde{w}$-axis.

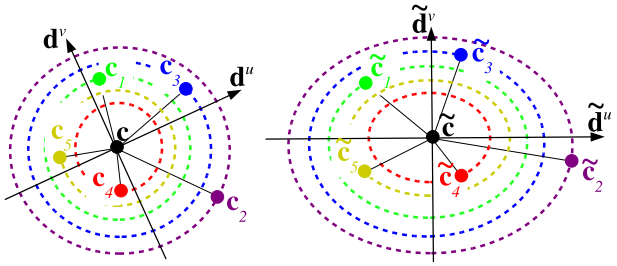

Figure 8. Updating the support on the tangent plane (4). The levels of the $\phi(u,v)$ at each of the old centers $\{\mathbf{c}_j\}$ (right) should be preserved at the new centers $\{\tilde{\mathbf{c}}_j\}$ (left).

2.4. Local polynomial fitting

Using the neighboring new and old center positions, we re-fit the local polynomial implicit function $f(\mathbf{x})$. As illustrated in Fig. 9, the idea is essentially same as (4). We try to keep the function values $\{f(\mathbf{c}_j)\}$ at the new centers $\{\tilde{\mathbf{c}}_j\}$.

$$\sum_j \left(\tilde{f}(\tilde{\mathbf{c}}_j) - f(\mathbf{c}_j)\right)^2 + \left(\tilde{f}(\tilde{\mathbf{c}}) - f(\mathbf{c})\right)^2 \to \min \qquad (5)$$

Since $f(\mathbf{c}) \neq 0$ in general, $\mathbf{c}$ is included in (5). Solving (5) requires a 6×6 matrix inversion in the quadratic primitive

case. We use SVD [14] for computing the matrix inversion since the matrix is sometimes singular.

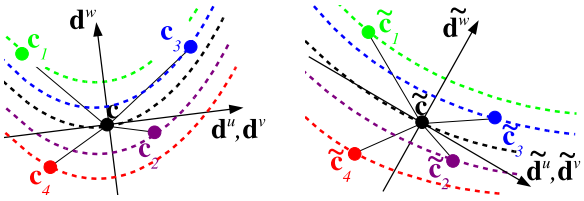

Figure 9. Updating the local implicit function (5). The levels of the $f(\mathbf{x})$ at each of the old centers $\{\mathbf{c}_j\}$ (right) should be preserved at the new centers $\{\tilde{\mathbf{c}}_j\}$ (left).

3. Results and discussion

3.1 Implementation summary

Fig. 10 shows a sequence of deformations via several interactions of a user. In the following, we explain user's interaction steps and the corresponding computations which are performed in each step.

Step 1: A neighbor graph is created. Then the user selects several control points from the centers.

Step 2: After deciding a choice of control points, the matrix factorization for solving (1) and (2) is performed. Then, the weights for averaging the local transformations are computed by solving (2) M times.

Step 3: The user changes the positions and the local transformations of the control points. Simultaneously with the user's control, the new center positions are computed by solving (1) after averaging the local transformations. Then, we solve (3), (4), and (5) at each of the local supports.

Step 4: When the user confirms the deformation, we go back to the step 1 for applying another deformation with a different choice of control points.

In step 3, solving (5) seems to be a little bit expensive comparing with the others, however an interactive updating is possible in deforming relatively small SLIM models, for example the bunny and dinosaur. For large models like in Fig. 1, solving (5) is omitted during the user moves a control point or local transformation. In this case all coefficients are set to zero that means we set $f(\mathbf{x}) = 0$ as the tangent plane (flat primitive). The left image of Fig. 11 shows a rendering result during the user's change of a control. If the user stops the change, the high quality geometry is rendered after solving (5).

In our current system, specifying ROI is not supported. So all elements of a SLIM surface participate in the computation. The fixed parts under a deformation are roughly pined by a few control points as shown in Fig.10. Of course the deformation using ROI seems to work as well as mesh deformation case.

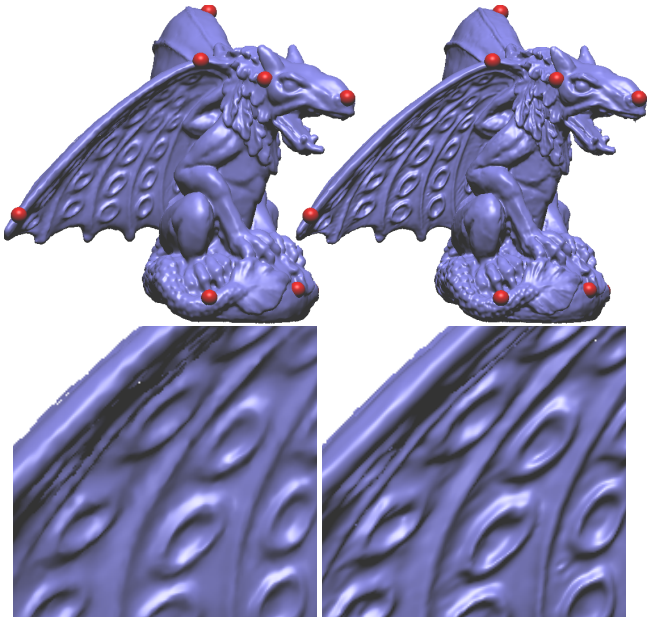

Figure 11. Left : the model rendered using the tangent planes by omitting local fitting (5). The details are lost in the rendering, but quick response to user's control. Right : the model rendered using the quadratic primitives.

3.2 Timing

The timing results are reported in Table 1. In our experiments we used a standard 3.0GHz Pentium 4 PC with 2GB RAM. In all results presented in this paper we used quadratic primitive based SLIM models. The choice of the polynomial degree affects only the time for solving (5).

Since the cost of propagating the local transformations is proportional to the number of control points, we report two cases in deformations of the armadillo model (corresponding to (a)–(c) and (c)–(e) in Fig. 10).

Rendering SLIM surfaces proposed in [11] is too time-consuming for interactive movements of control points. For example, creating an image of the Armadillo model with 512×512 pixels takes about 1 second. To save the rendering time during a user's movement of a control point, we decrease the support size and omit the blending intersections. The resulting rendering time is less than 0.1 second, thus the overhead of rendering a SLIM surface is relatively

Figure 10. A sequence of deformations via several interactions of the user. (a) The user selected control points. (b) The user deformed the left leg. (c) After deforming the right leg, the control points were released. (d) With new control points, the user deformed the arms. (e) The head was also deformed.

small. Once the user releases the control point, the full rendering procedure is performed.

3.3 Comparison with mesh deformation

In Fig. 12 we compare a mesh-based deformation and our SLIM-based deformation. For the mesh-based deformation, the optimization (1) was solved with respect to the mesh vertex positions instead of the center positions of a SLIM surface.

According to our experiments large global distortions were observed in the mesh-based deformations. On a SLIM surface, the local details are encoded by the local implicits and the support center positions represent only a simplified and smooth geometry. Thus, the Laplacian vector field on the support centers is very smooth and can be accurately integrated in a numerical way. In contrast, the Laplacian vector field on a dense and detailed mesh is not so smooth since the Laplacians contain the local detail information. We guess that an accurate integration of such a complicated vector field is a numerically difficult task.

Another benefit to use a SLIM surface for solving (1) is that relatively less constraints are required to successfully compute the matrix factorization. We think this is simply caused by the fact that the number of support centers is much less than the number of mesh vertices. As demonstrated in Fig. 13, we could deform the dinosaurs model represented as a SLIM surface using only a few control points. In our experiments, the matrix factorization for the dinosaurs mesh using such a few number of control points was failed due to a numerically ill-conditioned matrix.

As reported in the last two rows on Table 1, SLIM surfaces are effective to save the computational time in solving (1). However, a total time to update a shape for each movement of a control point in SLIM-based deformations is almost the same as that in mesh-based deformations. In SLIM-based deformations, we must perform additional computations: estimating the new local frames, radii, and

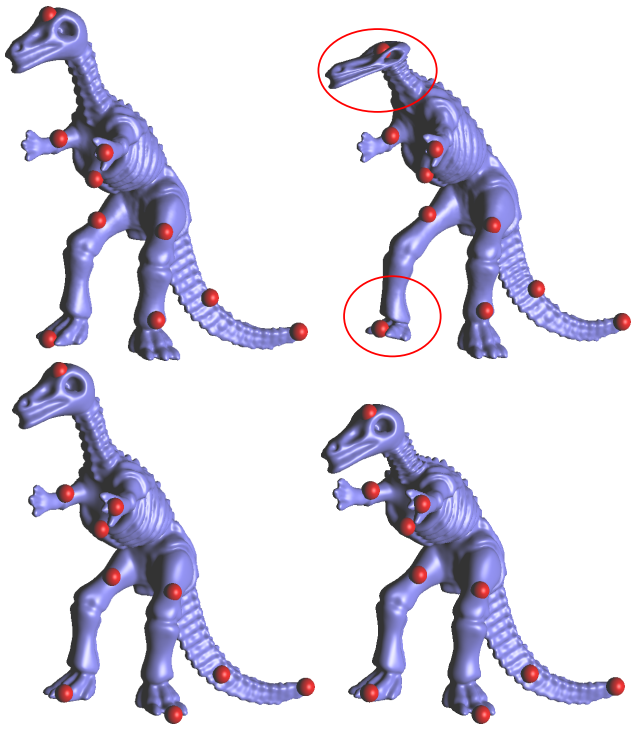

Figure 12. A comparison of a mesh-based deformation (top) and our SLIM-based deformation (bottom). By setting a similar set of control points, the point on the top of a head was slightly moved down. On the head and right leg, unwanted large global distortions were observed in the mesh-based deformation. In contrast, such distortions did not happen in the SLIM-based deformation.

local implicits. However, these computations are localized at each primitive since only the neighboring centers are used. Thus, it is easy to accelerate SLIM-based deformations by parallelizing the local computations using a multi-core processor or GPU with which standard PCs are

Model	N	M	Graph creation	Factorization for (1) (2) + Solve (2)	Average $\{\mathbf{T}_k\}$ + Solve (1)	Solve (3) (4)	Solve (5)
Dinosaurs	9K	3	0.088 sec.	0.91 sec.	0.062 sec.	0.031 sec.	0.19 sec.
Armadillo	25K	8	0.28 sec.	3.6 sec.	0.18 sec.	0.094 sec.	0.50 sec.
	25K	11	0.44 sec.	4.6 sec.	0.24 sec.	0.094 sec.	0.50 sec.
Gargoyle	69K	8	1.4 sec.	13 sec.	0.77 sec.	0.29 sec.	1.5 sec.
Dinosaurs (SLIM)	9K	10	0.088 sec.	1.1 sec.	0.047 sec.	0.031 sec.	0.19 sec.
Dinosaurs (mesh)	56K	10	–	4.2 sec.	0.27 sec.	–	–

Table 1. Timing results.

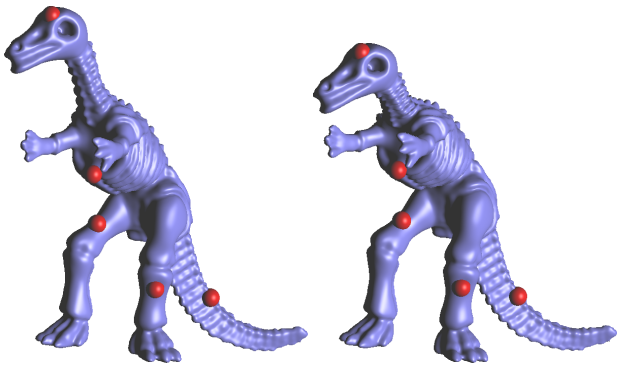

Figure 13. Similar SLIM-based deformation with the bottom images in Fig. 12 using more sparsely placed control points.

equipped nowadays.

3.4 Sharp feature preservation

One good property of the mesh-based Laplacian deformations is the preservation of sharp features. This good property is also kept in our SLIM-based deformations as demonstrated in the top images of Fig. 14. Instead of using mesh connectivity, an artificial neighboring graph on a SLIM surface causes no problem even near the sharp features.

In our current SLIM surface, only a polynomial implicit can be assigned to each local support. This restriction slightly smooths out sharp features due to the blending of overlapped implicits as shown in the bottom images in Fig. 14. As indicated in [12], we need to assign several implicits with Boolean operations in order to represent exact sharp features. Even if several implicits are assigned to a local support, our proposed polynomial updating (5) seems to work well. We just need to apply the refitting procedure to each local implicit separately.

3.5 Problems and future work

We encountered the following problems and will try to solve them in the future.

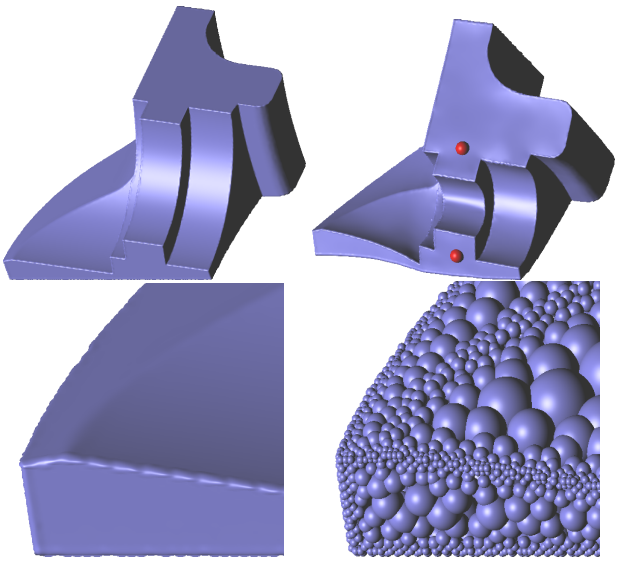

Figure 14. Top : SLIM-based deformation of an object with sharp features. Bottom : sharp features represented by our current SLIM surface.

- As shown in Fig. 15, highly stretching might create small holes. Of course we did not encounter this problem in typical deformations. This problem means ellipsoidal supports can not capture a large amount of deformations. We think this problem can be solved by fissions of elements similarly to [13].

- Topological changes are not possible. To allow the topological changes during deformations we need dynamic changes of the graph structure and local boolean operations for eliminating the redundant parts of a surface. Using Laplacian based approach is not suited for this purpose because changing the connectivity of the graph invokes re-factorization of the Laplacian matrix. Spatial deformation techniques are more suitable for solving this problem. Further our local optimization method can also be combined with spatial deformation methods.

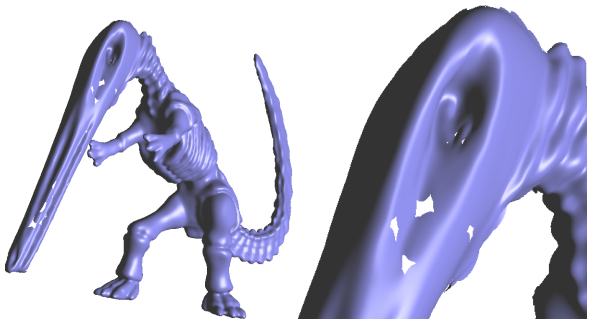

Figure 15. Highly stretching might create small holes.

Acknowledgments

We would like to thank Alexander Belyaev and Shin Yoshizawa for their valuable and constructive comments. We are grateful to anonymous reviewers for their useful comments and suggestions. The models are courtesy of VCG-ISTI via the AIM@SHAPE Shape Repository (gargoyle), the Stanford 3D Scanning Repository (bunny, armadillo), and Cyberware (dinosaur). This study was supported in part by Industrial Technology Research Grant Program in '04 from New Energy and Industrial Technology Development Organization (NEDO) of Japan.

References

[1] B. Adams and P. Dutré. Interactive boolean operations on surfel-bounded solids. *ACM Transaction on Graphics (Proc. SIGGRAPH 2003)*, 22(3):651–656, 2003.

[2] M. Alexa. Differential coordinates for mesh morphing and deformation. *The Visual Computer*, 19(2):105–114, 2003.

[3] A. Angelidis, M.-P. Cani, G. Wyvill, and S. King. Swirling-sweepers: Constant-volume modeling. In *Proc. 12th Pacific Conference on Computer Graphics and Applications*, pages 10–15. IEEE CS Press, Los Alamitos, CA, 2004.

[4] A. Angelidis, G. Wyvill, and M.-P. Cani. Sweepers: Swept user-defined tools for modeling by deformation. In *Proc. International Conference on Shape Modeling and Applications 2004*, pages 63–73. IEEE CS Press, Los Alamitos, CA, 2004.

[5] A. H. Barr. Global and local deformations of solid primitives. In *Computer Graphics (Proc. SIGGRAPH 84)*, pages 21–30. ACM Press, New York, 1984.

[6] M. Botsch and L. Kobbelt. Real-time shape editing using radial basis functions. *Computer Graphics Forum (Proc. Eurographics 2005)*, 24(3):611–621, 2005.

[7] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. *Computer Graphics (Proc. SIGGRAPH 1999)*, 33(Annual Conference Series):317–324, 1999.

[8] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *Computer Graphics (Proc. SIGGRAPH 1992)*, pages 71–78, 1992.

[9] Y. Lipman, O. Sorkine, D. Cohen-Or, D. Levin, C. Rössl, and H.-P. Seidel. Differential coordinates for interactive mesh editing. In *Proc. International Conference on Shape Modeling and Applications 2004*, pages 181–190. IEEE CS Press, Los Alamitos, CA, 2004.

[10] Y. Lipman, O. Sorkine, D. Levin, and D. Cohen-Or. Linear rotation-invariant coordinates for meshes. *ACM Transaction on Graphics (Proc. SIGGRAPH 2005)*, 24(3):479–487, 2005.

[11] Y. Ohtake, A. Belyaev, and M. Alexa. Sparse low-degree implicit surfaces with applications to high quality rendering, feature extraction, and smoothing. In *Proc. 3rd Eurographics / ACM SIGGRAPH Symposium on Geometry Processing*, pages 149–158. Eurographics Association, Aire-la-Ville, Switzerland, 2005.

[12] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H.-P. Seidel. Multi-level partition of unity implicits. *ACM Transactions on Graphics (Proc. SIGGRAPH 2003)*, 22(3):463–470, 2003.

[13] M. Pauly, R. Keiser, L. P. Kobbelt, and M. Gross. Shape modeling with point-sampled geometry. *ACM Transaction on Graphics (Proc. SIGGRAPH 2003)*, 22(3):641–650, 2003.

[14] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1993.

[15] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. In *Computer Graphics (Proc. SIGGRAPH 86)*, pages 151–160. ACM Press, New York, 1986.

[16] A. Sheffer and V. Kraevoy. Pyramid coordinates for morphing and deformation. In *Proc. 2nd International Symposium on 3D Data Processing, Visualization and Transmission*, pages 68–75. IEEE CS Press, Los Alamitos, CA, 2004.

[17] O. Sorkine. Laplacian mesh processing. In *EUROGRAPHICS 2005 State-of-The-Art-Report*, 2005.

[18] O. Sorkine, Y. Lipman, D. Cohen-Or, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Proc. 2nd Eurographics / ACM SIGGRAPH Symposium on Geometry Processing*, pages 179–188. Eurographics Association, Aire-la-Ville, Switzerland, 2004.

[19] S. Toledo. A library of sparse linear solvers, version 2.2. Tel Aviv University, September 2003. `http://www.tau.ac.il/ stoledo/taucs/`.

[20] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.-Y. Shum. Mesh editing with poisson-based gradient field manipulation. *ACM Transaction on Graphics (Proc. SIGGRAPH 2004)*, 23(3):644–651, 2004.

[21] R. Zayer, C. Rössl, Z. Karni, and H.-P. Seidel. Harmonic guidance for surface deformation. *Computer Graphics Forum (Proc. Eurographics 2005)*, 24(3):601–609, 2005.

[22] K. Zhou, J. Huang, J. Snyder, X. Liu, H. Bao, B. Guo, and H.-Y. Shum. Large mesh deformation using the volumetric graph laplacian. *ACM Transaction on Graphics (Proc. SIGGRAPH 2005)*, 24(3):496–503, 2005.