Subspace Gradient Domain Mesh Deformation

Jin Huang* Xiaohan Shi* Xinguo Liu Kun Zhou
Li-Yi Wei Shang-Hua Teng† Hujun Bao* Baining Guo Heung-Yeung Shum
Microsoft Research Asia *Zhejiang University †Boston University

Abstract

In this paper we present a general framework for performing constrained mesh deformation tasks with gradient domain techniques. We present a gradient domain technique that works well with a wide variety of linear and nonlinear constraints. The constraints we introduce include the nonlinear volume constraint for volume preservation, the nonlinear skeleton constraint for maintaining the rigidity of limb segments of articulated figures, and the projection constraint for easy manipulation of the mesh without having to frequently switch between multiple viewpoints. To handle nonlinear constraints, we cast mesh deformation as a nonlinear energy minimization problem and solve the problem using an iterative algorithm. The main challenges in solving this nonlinear problem are the slow convergence and numerical instability of the iterative solver. To address these issues, we develop a subspace technique that builds a coarse control mesh around the original mesh and projects the deformation energy and constraints onto the control mesh vertices using the mean value interpolation. The energy minimization is then carried out in the subspace formed by the control mesh vertices. Running in this subspace, our energy minimization solver is both fast and stable and it provides interactive responses. We demonstrate our deformation constraints and subspace deformation technique with a variety of constrained deformation examples.

Keywords: nonlinear constraints, skeletal control, volume preservation, projection constraint.

1 Introduction

Recent years have witnessed significant progress in gradient-domain mesh deformation techniques [Sorkine et al. 2004; Yu et al. 2004; Zhou et al. 2005; Lipman et al. 2005; Nealen et al. 2005]. These techniques have several attractive properties, including the abilities to preserve surface details during deformation and to produce visually pleasing results by amortizing distortions throughout the mesh. However, existing gradient-domain techniques are not effective at performing *constrained deformation* tasks. For example, it is desirable to preserve the volume when deforming an incompressible object. Also when working with a digital character, it is important to maintain the straightness and length of the limbs following the underlying skeleton [Lander 1998]. Unfortunately, all these are extremely difficult to accomplish with existing gradient domain techniques.

In this paper we present a general framework for performing constrained deformation tasks with gradient domain techniques. We introduce a number of deformation constraints and present a gradient domain technique that works well with a wide variety of linear and nonlinear constraints. The constraints we introduce include *the volume constraint* for volume preservation, *the skeleton constraint* for skeleton-based deformation, and *the projection constraint* for easy manipulation of the mesh without frequently switching between multiple viewpoints. Among these constraints the projection constraint is linear, whereas the volume and skeleton constraints are nonlinear.

*This work was done while Jin Huang and Xiaohan Shi were visiting students at Microsoft Research Asia.

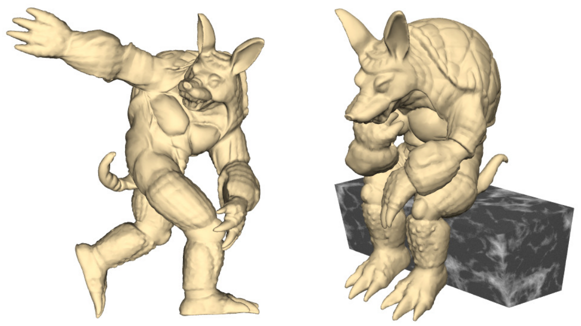


Figure 1: *Deformation examples generated by our system. The rigidity of limb segments is maintained by our skeleton constraint, whereas the body volume is exactly preserved by our volume constraint.*

Nonlinear deformation constraints present special challenges to gradient domain techniques. Indeed, we are not aware of any work on gradient domain deformation that involves nonlinear constraints. The only constraint that has appeared in previous related work is the position constraint [Sorkine et al. 2004], which is a linear constraint. The difficulty with nonlinear constraints is understandable: most existing gradient domain techniques cast mesh deformation as a linear least-squares energy minimization problem, and the inclusion of nonlinear constraints would immediately make the problem nonlinear.

The subspace deformation technique we derive in this work can handle nonlinear constraints and still achieve interactive performance. Our technique casts mesh deformation as a nonlinear least-squares energy minimization problem and solves the problem using an iterative algorithm. In theory the nonlinear least squares formulation allows us to put any nonlinear constraints in the deformation energy. In practice, however, we must carefully select the constraints that go into the energy if we are to expect a manageable computational cost for energy minimization. We include a nonlinear constraint in the energy only if the constraint is quasi-linear. Intuitively, a quasi-linear constraint is one that almost behaves like a linear constraint. It turns out that many nonlinear constraints in mesh deformation behave this way. For nonlinear constraints that are not quasi-linear, we treat them as hard constraints and solve them using Lagrange multipliers. Because solving hard constraints with Lagrange multipliers is costly, the number of such constraints should be kept to a minimum.

Even with a carefully formulated deformation energy and hard constraints, we still run into serious problems with slow convergence and numerical instability when minimizing the energy using iterative algorithms. In fact, the stability problem is often so severe that the iterations do not converge. To address this problem, our technique first builds a coarse control mesh around the original mesh. We then project the deformation energy and the hard constraints onto the control mesh vertices using mean value interpolation [Ju et al. 2005; Floater et al. 2005], and perform the energy minimization in the subspace formed by control mesh vertices. Since the number of vertices in the control mesh is much smaller than that of the original mesh, the problem size at each iteration becomes much smaller in the control mesh subspace. Furthermore, the smoothness of the mean value coordinates leads to fast and stable convergence of our iterative algorithm. This is because the mean value interpolation essentially smoothes out the nonlinearity of the nonlinear component of the deformation energy and improves the matrix condition number of the linear component of the energy. Deformation examples generated by our system are shown in Figure 1.

An additional advantage of our subspace technique is that it can easily handle real-world mesh output by commercial modelers, including meshes having non-manifold features and disconnected components. Such meshes are usually troublesome for existing gradient-domain techniques as they require a "clean" manifold mesh.

It is important to note that, even without nonlinear constraints, gradient domain mesh deformation is a nonlinear problem. This is because local Laplacian coordinates must be transformed according to the global orientation of the deformed mesh, and the local transformations create a nonlinear term in the deformation energy. Existing techniques convert this nonlinear term into a linear one by either heuristically approximating [Lipman et al. 2004; Zhou et al. 2005] or linearizing [Sorkine et al. 2004] the local transformations. The price for employing these heuristically schemes is suboptimal deformation results. In our work no such approximation and linearization is used and the local transformations are calculated accurately.

2 Previous Work

Mesh Deformation Multi-resolution techniques [Zorin et al. 1997; Kobbelt et al. 1998; Guskov et al. 1999; Botsch and Kobbelt 2003] can preserve surface details by decomposing a mesh into several frequency bands. A deformed mesh is obtained by first manipulating the base mesh and later adding back the high frequency details as displacement vectors. Because these displacements are processed independently, these techniques may produce artifacts in highly deformed regions.

Gradient domain techniques [Alexa 2003; Lipman et al. 2004; Yu et al. 2004; Sorkine et al. 2004; Zhou et al. 2005; Lipman et al. 2005; Nealen et al. 2005] cast mesh deformation as an energy minimization problem. The energy function contains both the term for detail preservation and the term for position constraints. The detail-preserving term is nonlinear because it involves both the differentials for local details and the local transformations which are position dependent. For computational efficiency, existing techniques convert this nonlinear term into a linear one by various approximations including local linearization [Sorkine et al. 2004], interpolation from handles [Zhou et al. 2005], and heuristic reconstruction [Lipman et al. 2004].

Sheffer et al. [Sheffer and Kraevoy 2004] proposed a rotation invariant shape representation, called pyramid coordinates, based on a set of angles and lengths relating a vertex to its immediate neighbors. Au et al. [Au et al. 2005] proposed to use the curvature normal of the unknown deformed mesh as the Laplacian differential coordinates for computing the deformation. Surface details are preserved by re-scaling the curvature normal to the same length as the Laplacian coordinates of the original mesh.

Deformation Constraint Mesh deformation by manually manipulating individual vertices is impractical, and various deformation handles and constraints have been proposed [Milliron et al. 2002]. By fitting some kind of control handles to the original mesh, the user only needs to manipulate the control handles and the mesh will deform accordingly. Common deformation handles include the control grids in free-form deformation (FFD) [Sederberg and Parry 1986; Coquillart 1990; MacCracken and Joy 1996], control curves [Singh and Fiume 1998], and control points [Hsu et al. 1992].

For animating articulated figures, a common control mechanism is skeleton structures. Proper deformations could be achieved by first configuring the skeletons followed by skin deformation conforming to the underlying skeletons. However, it can be challenging to perform the follow-up skin deformation, as various artifacts could happen such as joint collapsing or candy-wrapping effects [Kavan and Zara 2005]. These artifacts can be greatly reduced by accurate anatomical simulation [Wilhelms and Gelder 1997] or by example-based synthesis [Lewis et al. 2000; Kry et al. 2002; Sumner et al. 2005]. In particular, [Sumner et al. 2005] is robust enough to be applied to non-manifold or disconnected mesh surfaces.

Volume preservation is another common constraint. [Rappaport et al. 1996] introduces tri-variate tensor product free-form solids with the volume-preserving property. [Hirota et al. 1999] preserves volumes via a multi-level lattice representation. These techniques do not take into account preserving surface details. [Zhou et al. 2005] uses the volumetric graph Laplacian to preserve surface details while minimizing apparent volume changes. However, this technique cannot preserve volumes exactly.

3 Overview

A mesh is represented as a tuple (K, X), where K encodes the connectivity of the *simplicial complex* containing the vertices, edges, and triangles, and $X = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^t$, $\mathbf{x}_i \in R^3$, represents the positions of mesh vertices.

Deformation with Nonlinear Constraints Using the Laplacian coordinates, we can formulate mesh deformation as solving the following unconstrained energy minimization problem

$$\text{minimize} \quad \frac{1}{2}\sum_{i=1}^{m} ||f_i(X)||^2,$$

where $f_1(X) = \mathcal{L}X - \hat{\delta}(X)$ is for reconstructing X from its Laplacian coordinates $\hat{\delta}(X)$ [Sorkine et al. 2004] and $\mathcal{L}$ is the Laplacian operator matrix. $f_i(X),\ i > 1$, represent various deformation constraints. With nonlinear constraints the above is a nonlinear least squares problem [Madsen et al. 2004].

For convenience we regard $\mathcal{L}X = \hat{\delta}(X)$ as a constraint as well and call it the Laplacian constraint. Unlike most existing techniques, we do not convert $\mathcal{L}X - \hat{\delta}(X)$ to a linear function. Instead we derive a novel non-linear formulation of $\hat{\delta}(X)$ for exact evaluation of $\hat{\delta}(X)$. Our formulation of $\hat{\delta}(X)$ is based on the cotangent form introduced in [Desbrun et al. 1999].

We divide the set of constraints into two classes, *soft* and *hard constraints*. A soft constraint is included as a term in the deformation energy, whereas a hard constraint is handled using Lagrange multipliers [Madsen et al. 2004]. With the hard constraints our energy minimization becomes a constrained nonlinear least squares problem, which is usually solved using iterative techniques. In order to ensure that this nonlinear problem can be efficiently and robustly solved, we need to carefully select soft constraints and reduce the number of hard constraints.

We allow a nonlinear constraint to be a soft constraint only if it is quasi-linear. Intuitively a quasi-linear constraint is in some sense "almost linear": it can be written as $AX = b(X)$, where A is a constant matrix and $b(X)$ is a vector function whose Jacobian is "very small" (to be defined more precisely in Section 4.5). The Laplacian and skeleton constraints are examples of quasi-linear

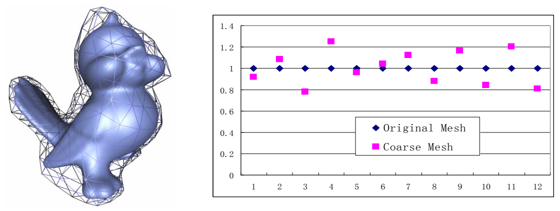


Figure 2: *Volume change plots. Left: the original + control meshes. Right: the volume change plots during a typical user interaction. Note that the volume is preserved on the original mesh, not on the coarse control mesh.*

constraints. Since all nonlinear constraints in the energy function are quasi-linear, our energy minimization problem can be written as

$$\text{minimize } ||LX - b(X)||^2 \quad \text{subject to} \quad g(X) = 0, \tag{1}$$

where L is a constant matrix and $g(X) = 0$ represents all hard constraints.

Because solving hard constraints with Lagrange multipliers is costly, we save the hard constraints for those with low-dimensional restriction (such as the volume constraint) and nonlinear constraints that are not quasi-linear.

In our current system, the soft constraints include a Laplacian constraint for retaining surface details, a skeleton constraint for maintaining rigidity, and a position constraint for user manipulation in 3D object space; the hard constraints include a volume constraint for preserving volumes and a projection constraint for user manipulation in 2D screen space.

Subspace Deformation When solving Equation 1 with iterative methods we run into serious problems with slow convergence and numerical instability. Often the stability problem is so severe that the iterations do not converge. Through theoretical analysis and experiments we found that the two dominating causes for the instability are the large condition number $\kappa(L^t L)$ of the matrix $L^t L$ and the nonlinearity of $b(X)$. Our subspace deformation technique is designed to address these issues.

The subspace method first builds a coarse control mesh around the original mesh (e.g., Figure 2, Figure 10 and Figure 15). The deformation energy and the hard constraints are then projected onto the control mesh vertices using mean value interpolation [Ju et al. 2005; Floater et al. 2005]. Let the control mesh vertices P be related to original mesh vertices X through $X = WP$. After projection we perform energy minimization in the control mesh subspace as follows:

$$\begin{aligned} &\text{minimize} && ||(LW)P - b(WP)||^2 \\ &\text{subject to} && g(WP) = 0. \end{aligned} \tag{2}$$

Since the number of vertices in P is much smaller than the number of vertices in X, the linear systems we solve at each iteration are relatively small. Furthermore, using the smoothness of the mean value coordinates we can show that, for a properly constructed control mesh, $\kappa((LW)^t(LW))$ has magnitudes smaller than $\kappa(L^t L)$ and the nonlinearity of $b(WP)$ is significantly reduced from that of $b(X)$. Our experiments indicate that the subspace method provides a numerically robust scheme for solving the deformation problem in Equation 1.

Most importantly, our technique does not simply apply constraints and solve the deformation on the coarse mesh P and interpolate back the results to the original mesh X; this naive approach would certainly not preserve mesh properties on the original mesh. Instead, as shown in Equation 2, we apply all constraints to the original mesh X and we only project the variables of the resulting constraints equations into the subspace formed by the coarse mesh P. Specifically, in Equation 2, our Lagrange term $g(WP) = 0$ allows us to satisfy our hard constraints exactly in the original mesh X, even though the equation variable is expressed in P. Similarly, the $LWP - b(WP)$ term allows us to enforce our soft constraints on the original mesh. For example, our volume constraint encoded in the Lagrangian term allows us to preserve volume in the original mesh, even though volume in the coarse control mesh is not preserved, as demonstrated in Figure 2. A more complex example for preserving both volume and surface details is shown in Figure 9.

4 Deformation Energy and Constraints

In this section we present our deformation energy and introduce several linear and nonlinear constraints. First we describe a novel formulation of the energy for reconstructing mesh vertex positions from the Laplacian coordinates. This formulation allows us to properly handle local transformations without resorting to heuristic approximations. Then we introduce two nonlinear deformation constraints: the skeleton constraint and the volume constraint. We also describe the projection constraint, which we found extremely handy for user interaction. Finally, we formulate the mesh deformation as a constrained nonlinear least-squares problem.

4.1 Laplacian Reconstruction

An essential step of gradient-domain deformation is the reconstruction of the mesh vertex positions X from their Laplacian coordinates $\hat{\delta} = \hat{\delta}(X)$. This reconstruction is done in the least-squares sense by imposing the Laplacian constraint $\mathcal{L}X = \hat{\delta}(X)$. As mentioned, this is a nonlinear constraint because $\hat{\delta}(X)$ includes the effects of local transformations.

We present a non-linear formulation of the Laplacian coordinates that allows us to evaluate $\hat{\delta}(X)$ exactly rather than through approximation. Our formulation of $\hat{\delta}(X)$ is based on the *cotangent* form as introduced in [Desbrun et al. 1999], from which we make use of the following observations: a) the Laplacian is a discrete approximation of the curvature normal, and b) the cotangent form Laplacian lies exactly in the linear space spanned by the normals of the incident triangles.

Consider an inner vertex $\mathbf{x}_i$ on the original mesh. Let $\mathbf{x}_{i,1}, \ldots, \mathbf{x}_{i,n_i}$ be the adjacent vertices and $\{T_{ij} = \triangle(\mathbf{x}_i, \mathbf{x}_{i,j-1}, \mathbf{x}_{i,j})\}_{j=1}^{n_i}$ be the incident triangles ($\mathbf{x}_0 = \mathbf{x}_{i,n_i}$ for notational convenience). From observation b), there exists a set of coefficients μ_{ij} such that

$$\delta_i = \sum_{j=1}^{n_i} \mu_{ij} \left((\mathbf{x}_{i,j-1} - \mathbf{x}_i) \otimes (\mathbf{x}_{i,j} - \mathbf{x}_i) \right) \tag{3}$$

where δ_i is the differential coordinate of vertex $\mathbf{x}_i$ on the original mesh, $\otimes$ denotes the cross product of two vectors in R^3, and the term $(\mathbf{x}_{i,j-1} - \mathbf{x}_i) \otimes (\mathbf{x}_{i,j} - \mathbf{x}_i)$ indicates the normal of triangle T_{ij}. Note that $\{\mu_{ij}\}$ remains invariant with respect to rigid rotation of the mesh.

We now describe how to solve the set of coefficients μ_{ij} for δ_i. Let A_i be the $3 \times n_i$ matrix whose j-th column is $(\mathbf{x}_{i,j-1} - \mathbf{x}_i) \otimes (\mathbf{x}_{i,j} - \mathbf{x}_i)$, and let μ_i be $(\mu_{i,1}, \ldots, \mu_{i,n_i})^t$. Then $\delta_i = A_i \mu_i$. This system is under-constrained when there are more than three incident triangles. One way to solve μ_i is to compute A_i^+, the pseudo inverse of A_i, through singular value decomposition (SVD) and set $\mu_i = A_i^+ \delta_i$. This is equivalent to finding a solution of Equation 3 that minimizes $||\mu_i||$.

Let $d_i(X) = \sum_j \mu_{ij} \left((\mathbf{x}_{i,j-1} - \mathbf{x}_i) \otimes (\mathbf{x}_{i,j} - \mathbf{x}_i) \right)$ be our new representation of Laplacian. Since $\{\mu_{ij}\}$ remains constant, it is easy to show that $d_i(X) = R_i \delta_i$ when the 1-ring neighborhood undergoes a local rotation R_i. Hence, $d_i(X)$ provides a rotation-invariant representation for the Laplacian differential coordinates and the deformed mesh should best maintain this invariant. Let $\hat{\gamma}_i = ||\delta_i||$ and $\gamma_i = ||d_i(X)||$. When the mesh is deformed, we

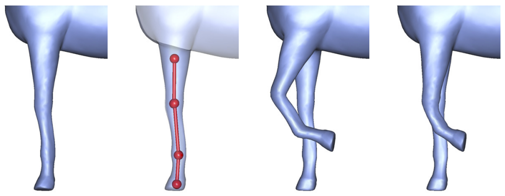


Figure 3: *Deformation with the skeleton constraint. From left to right: the un-deformed horse fore-leg, virtual skeleton (red segments), and deformation with and without the skeleton constraint.*

constrain the target differential coordinate to the direction of $d_i(X)$ while keeping its original length, i.e.,

$$\hat{\delta}_i(X) = \frac{\hat{\gamma}_i}{\gamma_i} d_i(X). \tag{4}$$

Note that even though our Laplacian formulation bears resemblance to [Sheffer and Kraevoy 2004] which is also nonlinear, they are mathematically different; in particular, we use cotangent weights while [Sheffer and Kraevoy 2004] used 2D mean-value weights. The major reason we choose a cotangent form over other representations is due to its curvature approximation property [Desbrun et al. 1999].

4.2 Skeleton Constraint

In deforming articulated figures, it is a common requirement to constrain parts of the model to be unbendable. For example, it is desirable that the fore-legs of a horse as illustrated in Figure 3 remain rigid. This effect can be achieved by skeleton-based deformation, which is widely used by artists [Lander 1998]. To enable skeleton-based deformation with gradient-domain techniques, we introduce a new type of nonlinear constraint, the skeleton constraint.

Let us illustrate the skeleton constraint with a simple scenario. As shown in Figure 4, suppose we have part of the unbendable mesh (circled by a dashed curve) and we would like to add a skeleton segment. The user simply specifies a virtual skeleton segment $\overline{\mathbf{ab}}$, and along it our algorithm automatically distributes a set of sample points $\{\mathbf{s}_i\}_{i=0}^{r}$, where $\mathbf{s}_0 = \mathbf{a}$ and $\mathbf{s}_r = \mathbf{b}$. The value of r is determined such that the distance between two adjacent samples equals the average edge length of the unbendable part.

During deformation, we would like to preserve both the straightness and the length, $\hat{\rho}$, of $\overline{\mathbf{ab}}$:

$$\left\{ \begin{array}{l} (\mathbf{s}_i - \mathbf{s}_{i-1}) - (\mathbf{b} - \mathbf{a})/r = 0 \\ \qquad i = 1, 2, \ldots, r, \\ ||\mathbf{b} - \mathbf{a}|| = \hat{\rho}. \end{array} \right. \tag{5}$$

We represent each sample point (including $\mathbf{a}$ and $\mathbf{b}$) as a linear combination of the mesh vertices: $\mathbf{s}_i = \sum_j k_{ij}\mathbf{x}_j$, where k_{ij} are some constant coefficients. Substituting $\mathbf{s}_i$ in Equation 5 with these linear representations, we have the following constraints:

$$\left\{ \begin{array}{rcl} \Gamma X & = & 0 \\ ||\Theta X|| & = & \hat{\rho} \end{array} \right. \tag{6}$$

where Γ is a constant $r \times n$ matrix with $(\Gamma)_{ij} = (k_{ij} - k_{i-1,j}) - \frac{1}{r}(k_{rj} - k_{0j})$, and Θ is a row vector with $(\Theta)_j = k_{rj} - k_{0j}$.

The coefficients k_{ij} are computed as the mean value coordinates [Ju et al. 2005] with respect to the constrained part of the mesh. Since [Ju et al. 2005] requires a closed mesh, we close the two open ends of the constrained segment by adding as two virtual vertices (c_1 and c_2 in Figure 4) the centroids of the boundary curves of the open ends.

Figure 3 demonstrates the importance of the skeleton constraint for maintaining rigid body parts. As shown, a horse leg deformed without any skeleton constraint looks quite unnatural.

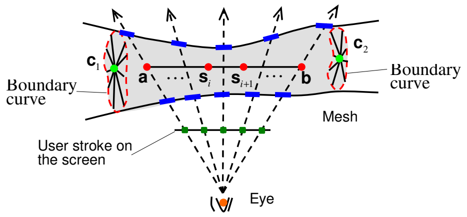


Figure 4: *Skeleton constraint specification. Line segment* $\overline{\mathbf{ab}}$*: constraint bone segment. Dark-green squares: pixels under the user stroke. Blue segments: ray intersections with the mesh. Light-green dots: virtual vertices to close the two open mesh boundaries.*

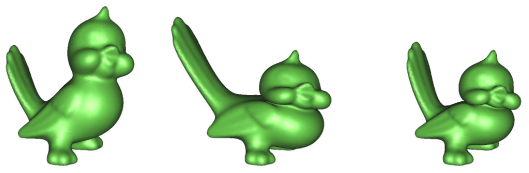


Figure 5: *Deformation with (middle) and without (right) the volume constraint. The original model is on the left.*

Skeleton Specification Here, we describe the implementation details for specifying skeleton constraints on the unbendable mesh part. As shown in Figure 4, the user simply draws a stroke over the target region (dark-green) and our algorithm will automatically construct the skeleton segment and the associated constrained region (gray), as described below.

For each user stroke pixel (dark-green), we construct a ray connecting the stroke pixel and the eye point. We then compute the first two intersections (blue) of each such stroke ray with the mesh; essentially these two intersections reside on the front and back sides of the target mesh segment. We construct $\overline{\mathbf{ab}}$ as the line segment approximately in the middle of the front and back intersections via a simple least squares fitting.

To determine the constrained region, we first place a plane perpendicular to $\overline{\mathbf{ab}}$ at each of its end vertices. These two planes serve as boundary planes for the constrained region. We then determine the middle portion of the constrained region by growing outward the intersection triangles (blue) computed in the previous step until there is no gap between them.

4.3 Volume Constraint

We introduce a new volume constraint to exactly preserve the total volume of the mesh. In the following, we assume the mesh is a closed 2D manifold.

The total signed volume of a mesh can be computed using their vertex positions: $\psi(X) = \frac{1}{6}\sum_{T_{ijk}}(\mathbf{x}_i \otimes \mathbf{x}_j) \cdot \mathbf{x}_k$, where each $T_{ijk} \in K$ is a triangle formed by vertices i, j, and k. Judging by this, our volume constraint can be easily represented by

$$\psi(X) = \hat{v} \tag{7}$$

where $\hat{v}$ denotes the total volume of the original un-deformed mesh.

We handle Equation 7 as a hard constraint in our system. Due to the use of hard constraints, our technique is able to preserve volume exactly; if we were to use a soft constraint as in [Zhou et al. 2005], we could only reduce apparent volume change but not exactly preserve the volume.

Figure 5 demonstrates our volume preserving deformation effects on a bird model; notice that our technique preserves volume on the original mesh exactly, as illustrated in Figure 2. Also, even though Figure 5 only presents an example for whole-mesh volume preservation, our volume constraints can be applied to local body parts as well. For example, by incorporating only triangles of a human's forearm in Equation 7, we could preserve volume for this specific body part.

4.4 Projection Constraint

The projection constraint is similar to the position constraint for the purpose of user manipulation, but is imposed in the 2D screen space rather than in 3D. The position constraint, which enforces a vertex to move to a specific 3D position, is useful when the target 3D position is given. During a typical interactive deformation session through a 2D GUI (graphic user interface), the target 3D position is usually not given and the user often needs to control the shape of the mesh by dragging a surface point to a desired 2D location on the screen. This is when the projection constraint becomes most useful.

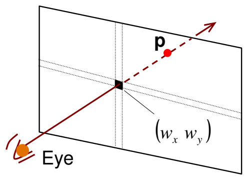


Figure 6: *Projection constraint. The projection of a 3D point P is constrained to (w_x, w_y) but free to move long the ray connecting the eye point and (w_x, w_y).*

Let $\mathbf{p} = Q_p X$ be an arbitrary point on the mesh (not necessarily a vertex), written as a linear combination of mesh vertex positions X via a constant matrix Q_p. The projection constraint requires that $\mathbf{p}$ move to a new 3D position whose 2D projection is located at a user-specified target position (w_x, w_y) on the screen (see Figure 6). Let M be the model view matrix which maps a point from the object space into the eye space, and f be the focal length of the viewing camera. The projection of $\mathbf{p}$ in the window's coordinate system can be computed as

$$\left(f\frac{M_x^r\mathbf{p} + M_x^t}{M_z^r\mathbf{p} + M_z^t},\ f\frac{M_y^r\mathbf{p} + M_y^t}{M_z^r\mathbf{p} + M_z^t} \right) \tag{8}$$

where the superscripts in M^r and M^t indicate the rotational and translational parts of M, and the subscripts in $M_{x,y,z}$ indicate the corresponding rows for the individual components. Since the target position in the screen space is (w_x, w_y), the above equation would lead to

$$\begin{aligned} \left(fM_x^r - w_x M_z^r\right) Q_p X &= -fM_x^t + w_x M_z^t, \\ \left(fM_y^r - w_y M_z^r\right) Q_p X &= -fM_y^t + w_y M_z^t. \end{aligned}$$

We can rewrite the above two equations as a single constraint:

$$\Omega X = \hat{\omega}, \tag{9}$$

where Ω is a constant $2 \times 3n$ matrix and $\hat{w}$ is a constant column vector.

Figure 7 shows an example of typical user interactions using the projection constraint. For the dinosaur model shown in Figure 7(a), the user simply drags a foot, the tail, and the head in the front view and obtains the result shown in Figure 7(b). For evaluating the deformation, we show Figure 7(a) and Figure 7(b) from a side view in Figure 7(c) and (d). We can see that the deformation result looks fairly natural. The projection constraint automatically adjusts the depth value of the manipulated foot, head and tail for better preserving the shape and surface details.

4.5 Constrained Nonlinear Least Squares

Now we are ready to formulate our nonlinear least-squares problem in Equation 1 using the traditional position constraint $\Phi X = \hat{V}$, as well the Laplacian, skeleton, volume, and projection constraints we introduced above.

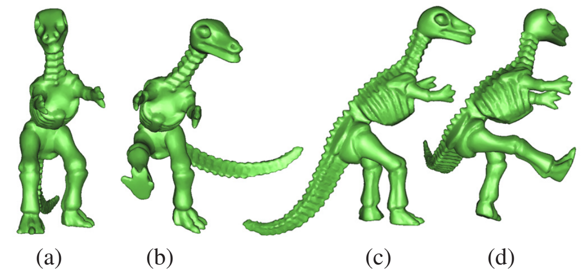


Figure 7: *User interaction via our projection constraint. From the original model view in (a), the user drags the dinosaur head and tail into the new pose shown in (b). Our projection constraints will automatically compute an optimal deformation, as illustrated from another view of the same deformation, where (c) is a side view of (a), and (d) is a side view of (b).*

Following the discussion in Section 3, we classify volume and projection constraints as hard constraints since they have a low-dimensional restriction. For the projection constraint, even though it is linear, its coefficient matrix Ω depends on (w_x, w_y), which changes whenever the user moves the target position. If we treat the projection constraint as a soft constraint, then Ω will be part of L which prevents us from pre-factorizing $L^t L$ for acceleration, as discussed in Section 5. For this reason, we handle the projection constraint as a hard constraint.

For the Laplacian, skeleton and position constraints, since their potential number of equations are large, it would be expensive to handle them as hard constraints in Equation 1. The position constraint can be easily treated as a soft constraint. As for the Laplacian and skeleton constraints, it turns out that they are quasi-linear constraints. A quasi-linear constraint is one that can be written as $AX = c(X)$, where A is a constant matrix and $c(X)$ is a vector function whose Jacobian is small, i.e., $||J_c|| \ll ||A||$.

Summarizing the above discussions, we have the following matrices and vector functions for Equation 1:

$$L = \begin{pmatrix} \mathcal{L} \\ \Phi \\ \Gamma \\ \Theta \end{pmatrix},\ b(X) = \begin{pmatrix} \hat{\delta}(X) \\ \hat{V} \\ 0 \\ \hat{\rho}\frac{\Theta X}{||\Theta X||} \end{pmatrix} \text{ and } g(X) = \begin{pmatrix} \Omega X - \hat{\omega} \\ \psi(X) - \hat{v} \end{pmatrix},$$

where $\Phi X = \hat{V}$ indicates the position constraint and the other symbols correspond to constraints defined earlier in this section.

Note that we represent the skeleton length constraint as $\Theta X = \hat{\rho}(\Theta X/||\Theta X||)$ even though it is equivalent to the simpler form $||\Theta X|| = \hat{\rho}$. The reason for this unusual representation is that it maintains the block structure of matrix L, such that we can solve the deformation by $n \times n$, instead of $3n \times 3n$, linear systems for the x, y, z components in Equation 12 and Equation 13.

5 Subspace Deformation Solver

We present a subspace iterative solver for mesh deformation as formulated in Equation 1. The ability to effectively combine the energy minimization for mesh deformation with the subspace reduction technique is the key to the development of our fast and high quality deformation algorithm. In addition to this algorithmic development and its implementation, we provide some analysis of one of the most important aspects of this subspace reduction technique — the improvement of convergence and numerical stability.

We first give an iterative algorithm based on the Gauss-Newton method in Section 5.1, and analyze its numerical stability in Sections 5.2 and 5.3. We then apply a subspace technique to develop a robust deformation solver.

5.1 The Gauss-Newton Formulation

Following the Gauss-Newton method [Steihaug 1995], we linearize $f(X) \equiv LX - b(X)$ at each iteration as

$$f(X+h) \approx l(h) \equiv f(X) + (L - J_b(X))h, \tag{10}$$

where $J_b(X)$ is the Jacobian of b. At each iteration we solve

$$\text{minimize } \frac{1}{2}||l(h)||^2 \quad \text{subject to } g(X+h) = 0. \tag{11}$$

By locally linearizing $g(X+h) \approx g(X) + J_g(X)h$ and applying Lagrange multipliers [Madsen et al. 2004] with Newton's method, we can express the local update that minimizes Equation 11 as:

$$\begin{aligned} h &= -(J_f^t J_f)^{-1}\left(J_f^t f + J_g^t \lambda\right) \\ \lambda &= -(J_g (J_f^t J_f)^{-1} J_g^t)^{-1}\left(g - J_g (J_f^t J_f)^{-1} J_f^t f\right) \end{aligned} \tag{12}$$

where $J_f \equiv J_f(X) = L - J_b(X)$ and $J_g \equiv J_g(X)$. Thus, starting from an initial X_0, we can solve Equation 1 iteratively by computing the update h_k from Equation 12 (assuming $X = X_{k-1}$) and then setting $X_k = X_{k-1} + \alpha h_k$, where α is a small constant that can be found by line search.

5.2 Numerical Considerations

Each Gauss-Newton step requires the calculation of:

a. $J_b(X)$, $J_g(X)$, $f(X)$, and $g(X)$, and

b. $(J_f^t J_f)^{-1}$, $J_g(J_f^t J_f)^{-1} J_g^t$ and $J_g(J_f^t J_f)^{-1} J_f^t f(X)$.

Since we only use a small number of hard constraints, the dominant computation of a) is the formation of J_b. Note that each volume and projection constraint adds one and two hard constraints, respectively. With s hard constraints and a mesh with n vertices, $J_g(X)$ is an $s \times n$ matrix and $J_g(J_f^t J_f)^{-1} J_g^t$ is an $s \times s$ matrix. $(J_f^t J_f)^{-1} J_g^t$ and $(J_f^t J_f)^{-1} J_f^t f(X)$ can be formed by solving $s+1$ linear systems with matrix $(J_f^t J_f)$ and hence Equation 12 calls for the solution of $(s+2)$ such linear systems.

We have $(J_f^t J_f) = L^t L - (L^t J_b + J_b^t L - J_b^t J_b)$, and $(L^t L)$ stays constant during the deformation. When the condition number $\kappa = \kappa((L^t L)^{-1}(J_f^t J_f))$ is small, which is the case if $||J_b||$ is much smaller than $||L||$, we can pre-compute the Cholesky factorization of $(L^t L)$ and apply the conjugate gradient (CG) method with $(L^t L)$ as a preconditioner to solve the linear systems. CG terminates with an ϵ-precise solution in $O(\kappa^{1/2} \log(1/\epsilon))$ iterations.

We can further eliminate the computation of $J_b(X)$, a costly step for large meshes, when $||J_b|| \ll ||L||$, by simplifying Equation 10 as $l(h) \equiv f(X) + (L - J_b(X))h \approx f(X) + Lh$. The resulting Gauss-Newton method is commonly referred to as the inexact Gauss-Newton method in the literature [Steihaug 1995] and has the same updates as Equation 12, but with J_f replaced by L.

We can use the pre-computed factorization of $L^t L$ to directly solve all the linear systems defined by $(L^t L)$. Thus we only have to factorize $(L^t L)$ once for a given set of soft constraints.

5.3 Convergence and Stability

Even for medium size meshes the above methods sometimes experience slow convergence and instability. When the mesh resolution increases, the instability could prevent the iterations from converging unless much smaller steps are taken. In addition, it becomes more costly for each iteration due to the increased sizes of L, J_b, J_f and J_g.

We analyze the factors that affect convergence. Following [Steihaug 1995], the local convergence of the Gauss-Newton method depends on the spectral radius of

$$-\left(J_f^t(X^*) J_f(X^*)\right)^{-1} \sum_{i=1}^{m} H_i(X^*) f_i(X^*),$$

	$\lvert X\rvert$	$\lvert P\rvert$	$\frac{\kappa(W^t \mathcal{L}^t \mathcal{L} W)}{\kappa(\mathcal{L}^t \mathcal{L})}$	$\frac{\lvert\lvert J_b^t J_b\rvert\rvert}{\lvert\lvert \mathcal{L}^t \mathcal{L}\rvert\rvert}$	$\frac{\lvert\lvert W^t J_b^t J_b W\rvert\rvert}{\lvert\lvert W^t \mathcal{L}^t \mathcal{L} W\rvert\rvert}$
Dino.	10k	159	5.3e-7	1.6e-1	1.6e-4
Armad.	30k	220	2.9e-7	1.5e-1	7.5e-5

Table 1: *Comparisons of condition numbers and Jacobian magnitudes.*

where X^* is a nearby local minimum of $||f(X)||$, H_i is the Hessian of the i^{th} component f_i of f. Thus, numerical stability depends on two key factors: the finite condition number (the ratio of the largest and the smallest non-zero eigenvalues) $\kappa(J_f^t(X^*) J_f(X^*))$ and the nonlinearity $\sum_{i=1}^{m} H_i(X^*) f_i(X^*)$ of $f(X)$. Note that the nonlinearity of $f(X)$ is that of $b(X)$.

The nonlinearity of $b(X)$ also limits the step size. As shown in [Kaporin and Axelsson 1994] the limiting step size from a point X along the normalized direction h is the largest positive number δ satisfying

$$2||f(X)|| \cdot \left\| \delta^2 \sum_{i=1}^{m} h^t H_i(X^*) h \right\| \leq \delta(1-\delta)||J_b(X)h||.$$

Moreover, the accuracy of the inexact Gauss-Newton method depends on $\kappa(L^t L)$. Suppose it takes k steps to converge, then we have $(L^t L) X_k = b(X_{k-1})$. The backward-error due to dropping $J_b(X_{k-1})$ is $||(L^t L)^{-1} J_b(X_{k-1})(X_k - X_{k-1})|| \leq \kappa(L^t L)||J_b(X_{k-1})(X_k - X_{k-1})||$, which could be significant if $\kappa(L^t L)$ is large.

5.4 Subspace Deformation

Now we present our subspace technique for robust mesh deformation. This technique significantly reduces the size of the linear systems at each iteration. More importantly, it enables us to improve the numerical stability of our non-linear deformation algorithm. The design of the subspace deformation solver is based on the following observations: (1) the key in gradient domain deformation is to deform the low frequency coarse shape while maximally preserving the high frequency features such as surface and skeleton details. Thus, in the view of spectral analysis via singular value decomposition, the deformation is mostly performed in a subspace defined by low frequency features. (2) If the subspace deformation formulation is robust and only involves a small number of variables, then the (inexact) Gauss-Newton method can converge rapidly and hence we can meet the interactive deformation requirements.

Thus, the first essential step in subspace deformation is to determine a quality subspace and its parametrization. Ideally, one can use spectral analysis to capture the subspace of low frequency features: Consider a deformation $X = X_0 + D$, where X_0 denotes the original mesh position and D is the desired displacement of the deformation. Let $\mathcal{L}$ be the Laplacian matrix. Then the changes in the differential coordinates is $\mathcal{L}X - \mathcal{L}X_0 = \mathcal{L}D$. Let $\mathcal{L} = USV^t$ be the SVD of $\mathcal{L}$, and $D = \sum_j d_j V_j$ be an expansion using the singular vectors V_j in V. Then $||\mathcal{L}D||^2 = \sum_j (d_j s_j)^2$, where s_j are the j^{th} singular values. In order to preserve the high frequency surface details, D should lie in a subspace formed by the set of singular vectors with small singular values. So one can form a reduced subspace in which energy minimization is performed in the subspace formed by the singular vectors in V with small singular values.

In practice, it could be expensive to compute the SVD of $\mathcal{L}$ for large meshes. We have found that mesh simplification provides an efficient alternative for subspace formation: We achieve these two conditions above by creating a coarse control mesh that surrounds the original one and reasonably approximates the shape of the original mesh, and using the numerically stable mean value interpolation [Ju et al. 2005] to project the high frequency details onto the control mesh to constrain the deformation of control vertices in the low-frequency subspace.

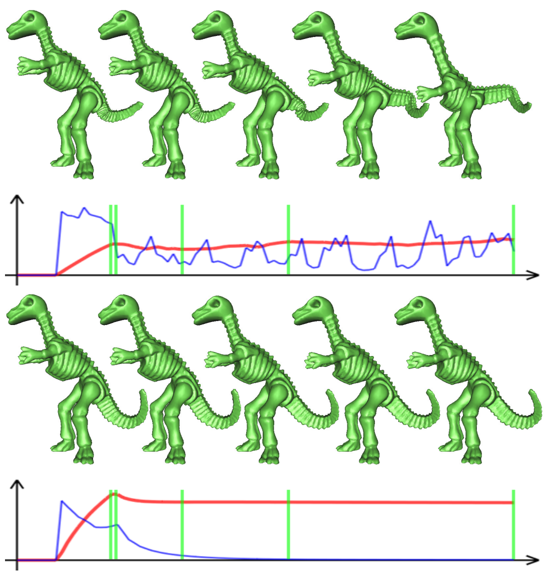


Figure 8: *Stability comparison betweed the original (top) and subspace (bottom) solvers. The thin-blue curves indicate step size while the thick-red curves indicate energy. The green bars indicate the locations of the deformation poses.*

We build the coarse control mesh by first applying the progressive convex hull construction algorithm in [Sander et al. 2000]. If the original mesh is closed, the resulting control mesh is also closed and contains the original one in its interior. Otherwise, in order to form a closed control mesh, we then fill in the open regions of the coarse mesh with extra triangles. Since the mean value coordinate is proportional to the distance reciprocal, we shift the control vertices along the normal direction by some user-specified offset to achieve a smooth transformation between the control mesh and the original mesh. Finally, we can also adjust some control vertices for more effective deformation control.

Let P be the vectors representing the locations of the control mesh vertices. Let W be the matrix that interpolates the original mesh from the control mesh, i.e., $X = WP$, using the mean value interpolation method [Ju et al. 2005]. The deformation energy and constraints from the original mesh are then projected to the control mesh by substituting $X = WP$ in Equation 1. We thus formulate the deformation in terms of subspace vertices P as in Equation 2. We apply the (inexact) Gauss-Newton method to solve Equation 2. For example, the update of the inexact Gauss-Newton method is

$$\begin{aligned} h_P &= -\left(W^tL^tLW\right)^{-1}\left(W^tL^tf + (J_gW)^t\lambda\right) \\ \lambda_P &= \quad -\left((J_gW)(W^tL^tLW)^{-1}(J_gW)^t\right)^{-1} \\ &\qquad \left(g - (J_gW)(W^tL^tLW)^{-1}W^tL^tf\right). \end{aligned} \tag{13}$$

The mean value interpolation is well-defined and smooth for all points. By its linear precision property, $X = WP$ holds before deformation and a smooth change of P induces a smooth change of X during deformation.

This subspace deformation is more robust than that in Equation 12 as $\kappa(W^tL^tLW)$ is much smaller than $\kappa(L^tL)$, as shown in Table 1. Note that $\kappa(L^tL)$ is dominated by $\kappa(\mathcal{L}^t\mathcal{L})$, where $\mathcal{L}$ is the surface Laplacian matrix. We can analyze this improvement in two stages. Let $\mathcal{L}'$ be the surface Laplacian matrix of the control meshes. First, we note that $\kappa((\mathcal{L}')^t\mathcal{L}')$ is much smaller than $\kappa(\mathcal{L}^t\mathcal{L})$. Suppose $\theta_{\min}$ and $\theta'_{\min}$ are respectively the smallest angle in the original and control mesh. Following [Shewchuk 2002], $\kappa(\mathcal{L}^t\mathcal{L})$ and $\kappa((\mathcal{L}')^t\mathcal{L}')$ are respectively proportional to $(|X|/\sin(\theta_{\min}))^2$ and $(|P|/\sin(\theta'_{\min}))^2$. As $|P|$ is much smaller than $|X|$ and $\theta'_{\min}$ is usually better than $\theta_{\min}$, thus $\kappa((\mathcal{L}')^t\mathcal{L}')$ usually is much smaller than $\kappa(\mathcal{L}^t\mathcal{L})$. Secondly, we found in experiments that $\kappa(W^t\mathcal{L}^t\mathcal{L}W)$ is usually smaller than $\kappa((\mathcal{L}')^t\mathcal{L}')$ when the coarse mesh reasonably approximates the shape of the original mesh. Therefore, we have $\kappa(W^tL^tLW) \ll \kappa(L^tL)$.

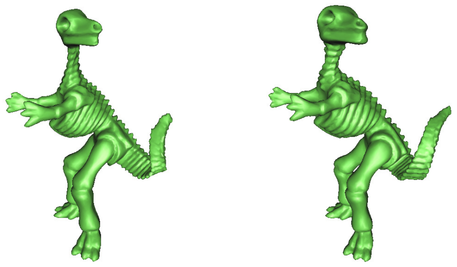


Figure 9: *Comparison of naive interpolation and our subspace method. Left: deformation result generated by naive interpolation, as described in Section 3. Notice the unnatural volume shrinkage around the head and neck. Right: deformation result by our subspace method.*

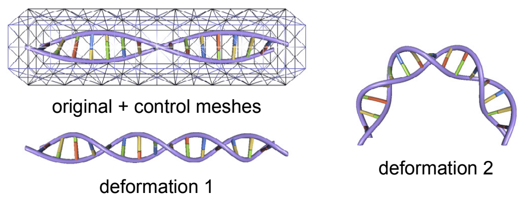


Figure 10: *Multi-component mesh deformation. The DNA sequence contains 32 disjointed components indicated by different colors.*

Note that the rows of W are the mean value coordinates of the original mesh vertices in terms of the control vertices. Geometrically, the mean value contribution of a point $\mathbf{p}$ on the control surface to a mesh vertex $\mathbf{x}$ is proportional to $1/||\mathbf{p} - \mathbf{x}||$. Since the control surface has a reasonable distance to the original mesh, the mean value coordinates of nearby mesh vertices are smoothly distributed. The continuous transformation of the control mesh also greatly restricts and reduces the non-linearity of the quasi-linear constraints $LX = b(X)$. As confirmed by our experiments shown in Table 1, $||W^tJ_b^tJ_bW|| \ll ||W^t\mathcal{L}^t\mathcal{L}W||$ (Note that $\mathcal{L}X = \hat{\delta}(X)$ represents the dominating quasi-linear constraint).

In Figure 8, we show an example comparing the stabilities of a direct solver and our subspace solver. As we can see, the subspace solver converges much faster than the direct solver.

As discussed in Section 3, our subspace solver preserves constraints on the original, instead of the control, mesh. Figure 9 demonstrates a complex example for preserving both volume and surface details; note that our subspace technique generates superior deformation results than naive interpolation.

A bonus of using a control mesh in the subspace solver is that it allows us to easily handle non-manifold surfaces or objects with multiple disjoint components. We simply ignore the non-manifold vertices in the surface detail energy term, and for objects consisting of multiple components, we create a single control mesh as the envelope of all the components so that they can be deformed together. In Figure 10, we show an example of a multi-component mesh, which is difficult to deform via previous differential domain techniques. See the accompanying video for the deformation process.

6 Results

We have built an interactive deformation system based on the constraints and the subspace solver presented above. With our system, the user can simply select groups of vertices as the control handles and drag them to new positions for deformation. The user can also apply position or projection constraints on the center or at all points of the handle; if only the center point is constrained, the

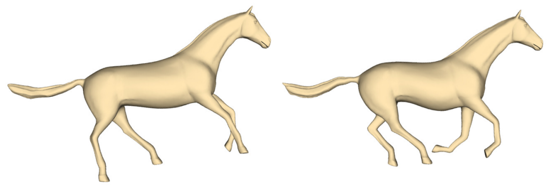
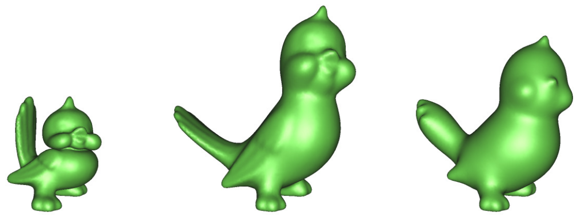


Figure 11: *Deformation examples with the skeleton constraint.*



(a) 60% volume (b) 160% volume (c) 160% volume +edge length

Figure 12: *Deformation example with volume manipulation.*

handle has an extra degree of freedom for rotation around the central point. For Laplacian and volume constraints, no manual handle is required because both constraints are usually applied uniformly over the entire mesh. The response time of the deformation system depends on two factors: N_t the number of the iterations needed for convergence and Δ_t the computation cost of each iteration. N_t is the most important factor but also the hardest to quantify because it varies significantly depending on many factors such as the shape of the model, the type of constraints, and the locations of the constraints. For the models we have experimented with, the average N_t is about 15.

The per-iteration cost Δ_t is much easier to quantify. It mainly depends on $|X|$, $|P|$ and the number of constraints. The most expensive steps are computing $W^t\left(L^t b(X)\right)$, $X = WP$ and $b(X)$. As the number of hard constraints increases, solving the linear systems defined by $W^t L^t L W$ becomes more expensive. On average, the cost of an iteration is proportional to $|X| \times |P|$. Please refer to Table 2 for detailed timing and mesh statistics for all demos shown in this paper.

Below, we demonstrate deformation effects achievable in our system. Please also refer to the accompanying video for a live recording of these effects. In Figure 11, we demonstrate the expressive power of our skeleton constraint for bending horse legs. It takes about 20 minutes of user time per frame to obtain these results.

In addition to volume preservation, our volume constraint also allows user-controllable volume changes by proper scaling of the right-hand-side term in Equation 7. In Figure 12(a,b), we show two deformation results for volume decreasing and increasing. In Figure 12(c), we add a skeleton constraint on each edge of the mesh in order to achieve the ballooning effect.

In Figure 13, we demonstrate deformations of a Santa model which has multiple disjoint components and non-manifold vertices. The model has more than 40 disjoint components and about 24k vertices. In the video, we show a walking sequence of the Santa model, which is produced by first generating 8 key poses using our deformation technique, followed by mesh interpolation [Zhou et al. 2005] of these key-frames to produce the whole animation sequence.

Figure 14 demonstrates the effect of control mesh density on the quality of our deformation results. As shown, our technique produces good quality even with fairly coarse control meshes. Of course, when the control mesh is too coarse, our technique may still produce grid artifacts as shown in the bottom case.

Multi-resolution Acceleration Multi-resolution methods can be utilized to accelerate gradient-domain techniques for very large

Figure 13: *Deformation examples of Santa.*

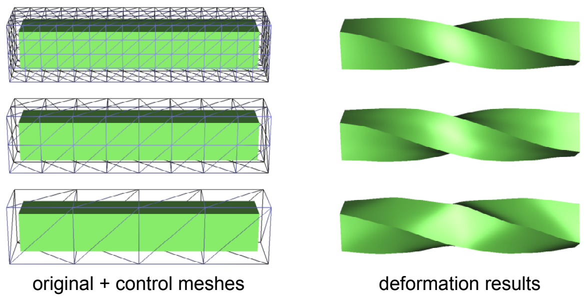


Figure 14: *The effects of control mesh density on the quality of deformation results. Notice the smooth deformation on the top two cases, and the grid artifacts in the bottom case.*

models as demonstrated in [Yu et al. 2004; Zhou et al. 2005]. For example, the Stanford armadillo has 170K vertices, and directly applying our non-linear deformation on it takes 4700 ms (microsecond). Even though this can be accelerated to 775 ms by our subspace solver performed on a coarse mesh with 220 vertices, it is still too slow for typical user interactions.

We perform further acceleration via [Guskov et al. 1999] as follows. First, we build a fine mesh with 30K vertices. We then perform our subspace solver over this fine mesh (together with the 220-vertex coarse mesh) and then add details back via [Guskov et al. 1999] to the original 170K-vertex mesh. This process takes only 200 ms (with 110 ms on our subspace solver + 90 ms for adding details back to original mesh), which is three times faster than our subspace solver directly applied over the original mesh.

7 Conclusions

We present a general framework for constrained deformation tasks using gradient domain techniques. We show how to formulate several widely-used deformation constraints for gradient domain techniques. We also develop a subspace deformation technique that works well with a variety of linear and nonlinear constraints. We demonstrate these deformation constraints and our subspace technique with an interactive deformation system that is user friendly and powerful enough for maintaining surface details as well as geometric properties such as volume, length, and straightness.

A number of related topics merit further investigation. Our subspace approach takes advantage of a coarse control mesh to restrict the deformation within a lower dimensional space. Despite the advantages, combining some other subspace ideas, such as the example-based subspace in [Sumner et al. 2005; Barbic and James 2005], may yield improvements. It is also worthwhile to investigate the possibility of a hierarchical control mesh, so that it can be locally refined to adapt to large deformations.

Mathematically, it has been challenging to provide a precise analysis on the impact of the control mesh and the interpolation methods. One can apply a backward error analysis to establish a bound relating the quality of subspace approximation with the following two parameters: (1) the distances between the surfaces defined by the control mesh and by the original mesh, and (2) the "condition number" of the interpolation, i.e., $\kappa(W^t W)$ using mean value coordinates. The basic idea of this backward analysis is to show that for a continuous path of deformation, there is a solu-

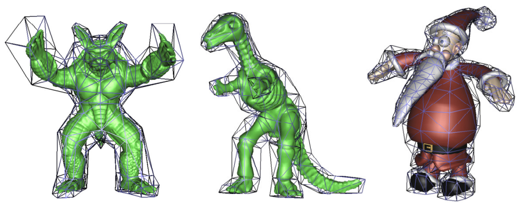


Figure 15: *Coarse control meshes around the original fine meshes.*

model	# vertices (original mesh)	# vertices (coarse mesh)	full space	subspace
Armadillo	30,002	220	2.8	9.1
Horse	14,285	427	6.9	8.2
Tweety	10,240	286	12	23.8
Dinosaur	10,002	159	9.5	34.5
DNA	19,184	194	NA*	16.7
Santa	25,777	448	NA*	5.3

Table 2: *Demo scene statistics, including the performance comparison of full-space and our sub-space solver in frames-per-second (fps). *The DNA and Santa models contain multiple components, so they cannot be deformed in full space.*

tion of the control variables whose inverse projection of the original mesh variables are close to the deformation. However, this type of mathematical analyzes usually struggle to provide mathematical bounds that are close enough to what have been observed in practice. The gap between mathematical analysis and practical observation might be the consequence of the fact that one has to consider the worst-case configurations in the analysis, while the practical deformation in general has better geometric properties (that might be relatively hard to fully capture in theory). For example, we have shown through our experiments that aggressive subspace reductions as shown in Table 2 can be used to obtain high quality deformation.

Thus, it remains an interesting theoretical question to develop a rigorous analysis, using some practically acceptable conditions, such as the short ranges of deformation in interactive graphics and the well-shapedness of the meshes, of our control mesh based subspace deformation technique.

Acknowledgement

We would like to thank the reviewers for their valuable comments that helped to make this paper better, and Michael S. Brown and Becky Sundling for their help during video production. Hujun Bao was partially supported by NSFC (No. 60021201), 973 Program of China (No. 2002CB312101) and the Cultivation Fund of the Key Scientific and Technical Innovation Project, Ministry of Education of China (No.705027).

References

ALEXA, M. 2003. Differential coordinates for local mesh morphing and deformation. *The Visual Computer 19*, 2, 105–114.

AU, O. K.-C., TAI, C.-L., LIU, L., AND FU, H. 2005. Mesh editing with curvature flow laplacian operator. Tech. rep., Computer Science Technical Report, HKUST-CS05-10.

BARBIC, J., AND JAMES, D. 2005. Real-time subspace integration for st. venant-kirchhoff deformable models. *ACM Trans. Graph. 24*, 3, 982–990.

BOTSCH, M., AND KOBBELT, L. 2003. Multiresolution surface representation based on displacement volumes. *Computer Graphics Forum 22*, 3, 483–492.

COQUILLART, S. 1990. Extended free-form deformation: a sculpturing tool for 3d geometric modeling. In *SIGGRAPH 90*, 187–196.

DESBRUN, M., MEYER, M., SCHRODER, P., AND BARR, A. H. 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH 99*, 317–324.

FLOATER, M. S., KOS, G., AND REIMERS, M. 2005. Mean value coordinates in 3d. *CAGD 22*, 623–631.

GUSKOV, I., SWELDENS, W., AND SCHRODER, P. 1999. Multiresolution signal processing for meshes. In *SIGGRAPH 99*, 325–334.

HIROTA, G., MAHESHWARI, R., AND LIN, M. C. 1999. Fast volume-preserving free form deformation using multi-level optimization. In *Proceedings of the fifth ACM symposium on Solid modeling and applications*, 234–245.

HSU, W. M., HUGHES, J. F., AND KAUFMAN, H. 1992. Direct manipulation of free-form deformations. In *SIGGRAPH 92*, 177–184.

JU, T., SCHAEFER, S., AND WARREN, J. 2005. Mean value coordinates for closed triangular meshes. *ACM Trans. Graph. 24*, 3, 561–566.

KAPORIN, I., AND AXELSSON, O. 1994. On a class of nonlinear equation solvers based on the residual norm reduction over a sequence of affine subspaces. *SIAM J. Scientific Computing 16*, 1, 228–249.

KAVAN, L., AND ZARA, J. 2005. Spherical blend skinning: a real-time deformation of articulated models. In *Proceedings of the symposium on Interactive 3D graphics and games*, 9–16.

KOBBELT, L., CAMPAGNA, S., VORSATZ, J., AND SEIDEL, H.-P. 1998. Interactive multi-resolution modeling on arbitrary meshes. In *SIGGRAPH 98*, 105–114.

KRY, P. G., JAMES, D. L., AND PAI, D. K. 2002. Eigenskin: real time large deformation character skinning in hardware. In *Proceedings of the symposium on Computer animation*, 153–159.

LANDER, J. 1998. Skin them bones: Game programming for the web generation. In *Game Developer Magazine*.

LEWIS, J. P., CORDNER, M., AND FONG, N. 2000. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *SIGGRAPH 2000*, 165–172.

LIPMAN, Y., SORKINE, O., COHEN-OR, D., LEVIN, D., RÖSSL, C., AND SEIDEL, H.-P. 2004. Differential coordinates for interactive mesh editing. In *Proceedings of Shape Modeling International*, 181–190.

LIPMAN, Y., SORKINE, O., LEVIN, D., AND COHEN-OR, D. 2005. Linear rotation-invariant coordinates for meshes. *ACM Trans. Graph. 24*, 3, 479–487.

MACCRACKEN, R., AND JOY, K. I. 1996. Free-form deformations with lattices of arbitrary topology. In *SIGGRAPH 96*, 181–188.

MADSEN, K., NIELSEN, H., AND TINGLEFF, O. 2004. Optimization with constraints. Tech. rep., Informatics and Mathematical Modelling, Technical University of Denmark.

MILLIRON, T., JENSEN, R. J., BARZEL, R., AND FINKELSTEIN, A. 2002. A framework for geometric warps and deformations. *ACM Trans. Graph. 21*, 1, 20–51.

NEALEN, A., SORKINE, O., ALEXA, M., AND COHEN-OR, D. 2005. A sketch-based interface for detail-preserving mesh editing. *ACM Trans. Graph. 24*, 3, 1142–1147.

RAPPAPORT, A., SHEFFER, A., AND BERCOVIER, M. 1996. Volume-preserving free-form solids. *IEEE Transactions on Visualization and Computer Graphics 2*, 1 (Mar.), 19–27.

SANDER, P. V., GU, X., GORTLER, S. J., HOPPE, H., AND SNYDER, J. 2000. Silhouette clipping. In *SIGGRAPH 2000*, 327–334.

SEDERBERG, T. W., AND PARRY, S. R. 1986. Free-form deformation of solid geometric models. In *SIGGRAPH 86*, 151–160.

SHEFFER, A., AND KRAEVOY, V. 2004. Pyramid coordinates for morphing and deformation. In *Proceedings of 3DPVT '04*, 68–75.

SHEWCHUK, J. R. 2002. What is a good linear element? interpolation, conditioning, and quality measures. In *11th International Meshing Roundtable*, 115–126.

SINGH, K., AND FIUME, E. 1998. Wires: a geometric deformation technique. In *SIGGRAPH 98*, 405–414.

SORKINE, O., COHEN-OR, D., LIPMAN, Y., ALEXA, M., RÖSSL, C., AND SEIDEL, H.-P. 2004. Laplacian surface editing. In *Proceedings of the symposium on Geometry processing*, 175–184.

STEIHAUG, T. 1995. An inexact gauss-newton approach to mildly nonlinear problems. Tech. rep., Dept. of Mathematics, University of Linkoping.

SUMNER, R. W., ZWICKER, M., GOTSMAN, C., AND POPOVIC, J. 2005. Mesh-based inverse kinematics. *ACM Trans. Graph. 24*, 3, 488–495.

WILHELMS, J., AND GELDER, A. V. 1997. Anatomically based modeling. In *SIGGRAPH 97*, 173–180.

YU, Y., ZHOU, K., XU, D., SHI, X., BAO, H., GUO, B., AND SHUM, H.-Y. 2004. Mesh editing with poisson-based gradient field manipulation. *ACM Trans. Graph. 23*, 3, 644–651.

ZHOU, K., HUANG, J., SNYDER, J., LIU, X., BAO, H., GUO, B., AND SHUM, H.-Y. 2005. Large mesh deformation using the volumetric graph laplacian. *ACM Trans. Graph. 24*, 3, 496–503.

ZORIN, D., SCHRODER, P., AND SWELDENS, W. 1997. Interactive multiresolution mesh editing. In *SIGGRAPH 97*, 259–268.