Hierarchical Deformation of Locally Rigid Meshes

Josiah Manson and Scott Schaefer

Texas A&M University

Abstract

We propose a method for calculating deformations of models by deforming a low-resolution mesh and adding details while ensuring that the details we add satisfy a set of constraints. Our method builds a low-resolution representation of a mesh by using edge collapses and performs an as-rigid-as-possible deformation on the simplified mesh. We then add back details by reversing edge-collapses so that the shape of the mesh is locally preserved. While adding details, we deform the mesh to match the predicted positions of constraints so that constraints on the full-resolution mesh are met. Our method operates on meshes with arbitrary triangulations, satisfies constraints over the full-resolution mesh, and converges quickly.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Boundary representationsI.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Hierarchy and Geometric Transformations

1. Introduction

Modeling tools have improved to the point that it is common for artists to design characters with intricate details directly encoded in the geometry rather than simulated through textures. These character models may have hundreds of thousands to millions of vertices. To animate these shapes, each vertex must be positioned for each frame of animation. Unfortunately, directly positioning individual vertices is not feasible due to the size of these models, so deformation methods have been developed to reduce the degrees of freedom while providing artists with simple, intuitive controls. Many methods provide a separate structure to control the deformation, such as a cage [JSW05] or skeleton [Zel82], but it is difficult to add degrees of freedom to these structures when finer control is needed; sometimes an entirely new control structure needs to be created. Additionally, the deformation tool needs to provide immediate feedback to the artist, so deformations must be computed in real-time.

An ideal deformation method requires no auxiliary control structure and allows an artist to add and remove degrees of freedom at will. Controls should also manipulate the mesh directly so that animated characters can be constrained to touch specific objects, like holding a cup. We believe that the most effective control scheme is to directly constrain the positions of vertices of the mesh. Furthermore, a deformed mesh should retain its shape under deformation. The exact meaning of shape-preserving is ambiguous, but has been defined by previous researchers as locally approximating rigid deformations [SA07]. However, computing a nearly rigid deformation of a mesh requires minimizing a global system of equations and becomes very slow for high-resolutions.

We approach the problem by allowing an artist to specify constraints on a high-resolution mesh, but calculate the large-scale features of the resulting deformation at a lower resolution. Calculating deformations at a lower resolution not only increases the speed at which we calculate deformations, but also increases the quality of deformations. Because constraints on the detailed mesh do not directly map to vertices in the low-resolution mesh, we developed a method for constraining low-resolution deformations with high-resolution constraints. Our key contribution, however, is to introduce a new method of adding details back to the deformed mesh while conforming to constraints. We add the details in a way that depends only on local neighborhoods of vertices and treats all vertices symmetrically.

1.1. Related Work

Surface deformation is a well-studied topic, and we refer the reader to a recent survey by Botsch and Sorkine [BS08]. The classic approach for reducing the dimensionality of surface deformations is to project the space of all possible deformations onto a smaller, user-defined subspace. The most popular projection is skeletal deformation, which represents the position of each vertex as a weighted combination of rigid transformations controlled by "bones" [Zel82]. Although skeletal deformation has been applied successfully to animation, it is poorly suited to representing plastic deformations that occur in soft materials. Additionally, it can be difficult to correctly assign weights to vertices in flexible regions between bones, such as at the shoulders and elbows.

As an alternative to bones, free-form deformations have been used to represent deformations of organic objects by warping space. In these methods, a low-resolution control structure is built to enclose a space that encompasses the object. Sederberg et al. [SP86] use a simple uniform grid that smoothly deforms space when vertices of the grid are moved. Other methods [JSW05, JMD*07] enclose objects in a tight-fitting, low-resolution cage to provide more intuitive and direct control of the shape. Unfortunately, these cages do not provide fine control of a mesh and provide no mechanism for adding more degrees of freedom to control the deformation at finer resolutions.

Differential surface editing methods express vertex positions relative to neighboring vertices. Laplacian surface editing [SCOL*04] encodes vertex positions as offsets relative to neighboring vertices and can introduce undesirable shearing in the deformation because local differences do not rotate with the surface. Yu et al. [YZX*04] define positions and orientations for the triangles of a mesh and then stitch the triangles back together by solving a Poisson equation over the gradient of vertex coordinates. Lipman et al. [LSLCO05] build rotationally invariant coordinates at the cost of solving two global equations: one for local coordinate frames at each vertex and another for the vertex locations themselves.

Some methods enforce the one-ring of each vertex to locally deform as rigidly as possible. Most of these methods, such as [SA07], iteratively solve a least-squares problem over the entire mesh. The time required to perform this minimization grows quickly with the number of vertices in a mesh and is expensive to perform over large meshes. PriMo [BPGK06] minimizes the elastic energy between prisms connected over the surface of a mesh to produce robust, rigid deformations. A more recent method [SSP07] reduces the computational cost of rigid deformations by deforming a coarse approximation of the mesh. Detail vertices are stored relative to nearby nodes and move with the local frames of vertices in the control mesh. In this method, the user manipulates the simplified mesh rather than the full-resolution mesh, whereas, in our method, constraints are placed on the full-resolution mesh and are enforced during expansion of the collapse hierarchy.

Other methods use the hierarchical structure of meshes to accelerate global optimization. Shi et al. [SYBF06] use multi-grid optimization of a discretized Poisson equation to calculate mesh deformations. Mesh Puppetry [SZT*07] employs an alternate approach of using inverse kinematics on a skeleton to first approximate the deformed mesh, which is then refined by minimizing a global surface energy to preserve local shape. This global optimization is accelerated through a cascading optimization that forwards partially computed results between multiple threads. Tools like ZBrush allow the artist to manipulate an object at multiple resolutions, but these tools require the surfaces to have subdivision connectivity. In the same line of thought, subdivision surfaces with displacement maps have been used in surface deformation where deformations are applied to the control mesh [LMH00, ZHX*07].

Early work for unstructured meshes [KCVS98] uses only a few levels of resolution and applies global smoothing to each resolution using a Gauss-Seidel solver. Guskov et al. [GSS99] add details at each level of a multi-resolution hierarchy formed by edge collapse operations. However, this method operates on a large neighborhood (3-ring), is not symmetric, allows stretching and skewing, and provides only indirect control of the final surface, because high-resolution vertices cannot be directly positioned. On the other hand, our method operates on small regions (1-ring) that we treat symmetrically and estimates rigid transformations at each step to predict and optimize surface positions to meet constraints.

Kilian et al. [KMP07] developed a multi-resolution method for interpolating between poses. Their method defines meshes as points in shape space and interpolates between them by finding a shortest path in this space, where distance is measured by how close the shapes are to being isometric. Finding a minimal path is a costly optimization that they accelerated by finding a path for a simplified mesh and refining the path both by increasing the number of vertices and by increasing the number of interpolatory meshes.

2. Calculating Deformations

Our method uses a hierarchy of mesh resolutions, and we refer to the resolution by a superscript. If there are $N+1$ resolutions generated through N edge collapses, we refer to the input mesh as P^N and the fully simplified (base) mesh as P^0. Our method operates on both an undeformed mesh P and a deformed mesh Q simultaneously, where Q has the same topology as P, but different geometry.

We define deformations of Q^N by specifying the positions of a few vertices in Q^N. We then deform Q^0 as rigidly as possible and add back details in a way that approximates an as-rigid-as-possible deformation calculated directly over Q^N. Constraints on vertices of Q^N do not directly correspond to vertices in Q^0, so we develop a method to predict the positions of mesh vertices in Q^N from Q^0.

The details that we add back to Q^0 must satisfy two properties: they must meet constraints on vertices in Q^N and must

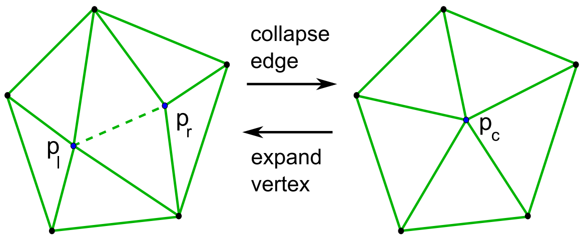

Figure 1: *Edge collapses to a vertex during simplification are reversible. To add back details, vertices are expanded to form edges.*

deform as rigidly as possible. We achieve both objectives through an inverted edge collapse operation. Figure 1 shows how collapsing an edge into a vertex can be reversed to expand a vertex into an edge. Each vertex expansion adds two triangles and one vertex.

We choose the positions of the deformed vertices affected by an edge expansion by expanding the undeformed (rest) mesh P in parallel with the deformed mesh Q. Figure 2 shows the steps of an expansion operation. Our goal is to determine the deformed positions of the edge vertices q_l and q_r that are added during the expansion. We determine their positions by calculating the best-fit, rigid deformation, M_c, of the local neighborhood around the vertex p_c before expanding p_c into an edge. Using the assumption that p_l and p_r undergo the same transformation as the vertices in their neighborhood, we find that $q_l = M_c p_l$ and $q_r = M_c p_r$. We finish by updating the mesh topology to add two new triangles. If constrained vertices are in the neighborhood of q_c, we use the additional steps in the boxed region of Figure 2. We move q_c so that the positions of the constrained vertices in the undeformed mesh c_i match the constraints in the deformed mesh d_i under the transformation M_c.

2.1. Edge Collapse

As a one-time preprocessing step, we calculate P^0 from P^N through a series of edge collapses. While simplifying, we store the order of edge collapses so that we can later reverse the collapses to add details back to the simplified mesh. There are several metrics that choose which edge to collapse and decide the location of the vertex that replaces the edge. One popular metric [GH97] minimizes the distance to the planes formed by triangles in the high-resolution mesh, but depends on Gaussian curvature and can generate triangles with poor aspect ratios in cylindrical features. Although these triangles approximate the undeformed shape well, they do not allow the degrees of freedom required to represent the shape when it bends. For example, Figure 3 (left) bends sharply enough that sliver triangles protrude through the opposite side of the bent cylinder. Instead, we choose a metric that favors uniform triangulations [ACSE05] at all levels

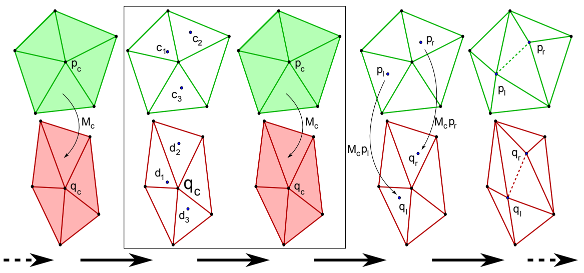

Figure 2: *An expansion operation consists of the three steps that are not boxed. First we calculate the best transformation M_c from the one-ring of p_c in the rest mesh to the deformed mesh. Second, we apply M_c to p_l, p_r to calculate the deformed positions q_l, q_r. Finally, we update the topology of the surface. When constraints exist, we use the steps in the boxed region to modify the position of q_c such that applying M_c to c_i matches d_i.*

of the hierarchy, and allows a more natural bending of the shape, as shown in Figure 3 (right).

We define the error function for a vertex k as the summed, squared distance to all points on the triangles that touch that vertex. We use the notation that $\mathcal{N}(k)$ is the set of triangles that touch vertex k. The error at a point x is then equal to

$$\begin{aligned} E_s(x) &= \sum_{i \in \mathcal{N}(k)} \int_s \int_t |p_i(s,t) - x|^2 \, dt \, ds \\ &= \sum_{i \in \mathcal{N}(k)} \left(|x - \bar{p}_i|^2 + \frac{1}{12} \sum_{j=1}^{3} |p_{i,j} - \bar{p}_i|^2 \right) \Delta_i, \end{aligned}$$

where $p_i(s,t)$ is a point on the i^{th} triangle that is parameterized by s and t, $p_{i,j}$ is the j^{th} vertex of triangle i, Δ_i is the area of the i^{th} triangle, and $\bar{p}_i$ is its centroid. The minimum of this error function is given by

$$x_{\min} = \frac{\sum_i \bar{p}_i \Delta_i}{\sum_i \Delta_i},$$

and we define the error associated with this vertex as $E_s(x_{\min})$. Suppose that an edge has the end points p_l, p_r and will be replaced by the vertex p_c. We define the error function of the edge as the sum of the error functions of p_l and p_r. We then place the vertex p_c that will represent the collapsed edge at the minimum of the edge error function in a greedy fashion, collapsing edges with the smallest error first. We apply this process until we reach the target number of vertices.

An important question is then, "How many vertices should one use in the simplified mesh?" Lower numbers of vertices make the model deform more like a solid and increase the speed of convergence, whereas higher numbers of vertices represent fine features, such as fingers, better. Several factors influence the choice and, in the end, a human must decide how many simplified vertices to use on a case-by-case basis. For most of our examples, we have found that 200 vertices represent movements well. We discuss the effect of the resolution of P^0 on the resulting deformation in Section 3.

2.2. Global Optimization

At the base resolution, we solve for the best rigid deformation of P^0 using As-rigid-as-possible Surface Modeling (ARAPSM) [SA07]. The only difference between our optimization and ARAPSM is that ARAPSM uses a single resolution, whereas we constrain vertices of Q^N but optimize Q^0. This complicates the optimization because there is no direct mapping between vertices in Q^N and Q^0. Lack of a direct mapping means that a simplified vertex may be influenced by multiple constrained vertices in Q^N, which means that we cannot use hard constraints.

Instead, we add a constraint energy E_c to the rigid deformation energy E_d of ARAPSM. The constraint energy E_c measures the distance between the constrained vertices in the deformed mesh $d_k \in Q^N$ and their predicted positions. The position of d_k relative to $q_i \in Q^0$ is predicted by the position of its dual, constrained vertex $c_k \in P^N$ relative to $p_i \in P^0$ under the rotation R_i, where R_i is the best-fit rotation around the vertex p_i between the rest and the deformed meshes. Because the relative positions between points is invariant under translation, E_c can be written as

$$E_c = \sum_i \sum_k |(d_k - q_i) - R_i(c_k - p_i)|^2.$$

The variable i indexes the vertices in Q^0, and k indexes the constrained vertices in Q^N that are under the influence of q_i. The collapse hierarchy forms a binary tree and we consider a constrained vertex c_k to be under the influence of p_c if p_c is an ancestor of c_k. Note that many vertices of P^0 may influence the position of c_k while we use only the direct ancestor. Because our collapse metric favors uniform triangulations, the influence of a vertex tends to fall off smoothly, and we have found that a fast approximation of influence by ancestry works well. We find the best deformation under these constraints by minimizing $E_d + wE_c$, where w is a weight that we set to 100.

We use the same alternating minimization described in ARAPSM to minimize the deformation and constraint energies. We alternately solve for rotations, then hold the rotations constant and solve for vertex positions that minimize the energy.

2.3. Expansion

After deforming the simplified mesh from Section 2.2, we expand the surface to full-resolution, while preserving the nearly-rigid properties of the deformed shape. We estimate that local details near a vertex $p_c \in P^\ell$ move rigidly with the corresponding local neighborhood around $q_c \in Q^\ell$ in the deformed mesh because as-rigid-as-possible deformations are almost rigid at small scales.

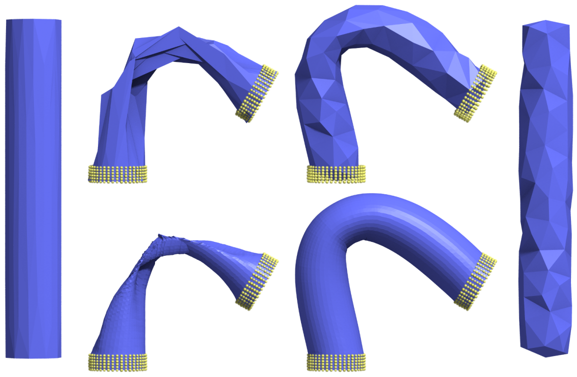
Figure 3: *The deformation of a cylinder collapsed to 100 vertices using a plane distance metric [GH97] (left), and using a point distance metric [ACSE05] (right). The top shows the simplified meshes, and the bottom shows the result after adding details.*

We use the local transformation M_c to calculate the vertices of the newly expanded edge as $q_r = M_c p_r$ and $q_l = M_c p_l$ (see Figure 2). The rigid transformation M_c consists of a rotation R_c and a translation T_c, such that $M_c x = R_c x + T_c$, where x and T_c are column vectors. We find T_c in terms of R_c by subtracting the centroid of the neighboring triangles [SMW06]. The translation component is then

$$T_c = \bar{q} - R_c \bar{p},$$

where

$$\bar{p} = \frac{\sum_{j\in\mathcal{N}(c)} \int_s \int_t p_j(s,t)dtds}{\sum_{j\in\mathcal{N}(c)} \int_s \int_t dtds} = \frac{\sum_{j=1}^{n} (p_c + p_j + p_{j+1})\Delta_j}{\sum_{j=1}^{n} \Delta_j}.$$

In the right-hand formula for $\bar{p}$, we make the simplifying assumption that vertices in the one-ring are ordered circularly from 1 to n and that Δ_j is the area of the j^{th} triangle. The formula for $\bar{q}$ has the same form as $\bar{p}$, but in the deformed mesh.

We calculate R_c as the minimizer of

$$\min_{R_c R_c^T = I} \sum_{j\in\mathcal{N}(c)} \int_s \int_t |(q_j(s,t) - \bar{q}) - R_c(p_j(s,t) - \bar{p})|^2 dtds.$$

The best-fit rotation is found through an SVD decomposition of the matrix

$$B_c = \sum_{j\in\mathcal{N}(c)} \int_s \int_t (p_j(s,t) - \bar{p})(q_j(s,t) - \bar{q})^T dtds.$$

This matrix can be written in closed form, using the notation that P_{j1}, P_{j2}, and P_{j3} are the three vertices of triangle j in the rest mesh, and $\hat{P}_{jk} = P_{jk} - \bar{p}$. We use a similar notation for the vertices of the deformed mesh, Q, so that

$$B_c = \sum_{j\in\mathcal{N}(c)} \frac{\Delta_j}{24} \begin{pmatrix} \hat{P}_{j1} & \hat{P}_{j2} & \hat{P}_{j3} \end{pmatrix} \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix} \begin{pmatrix} \hat{Q}_{j1} & \hat{Q}_{j2} & \hat{Q}_{j3} \end{pmatrix}^T.$$

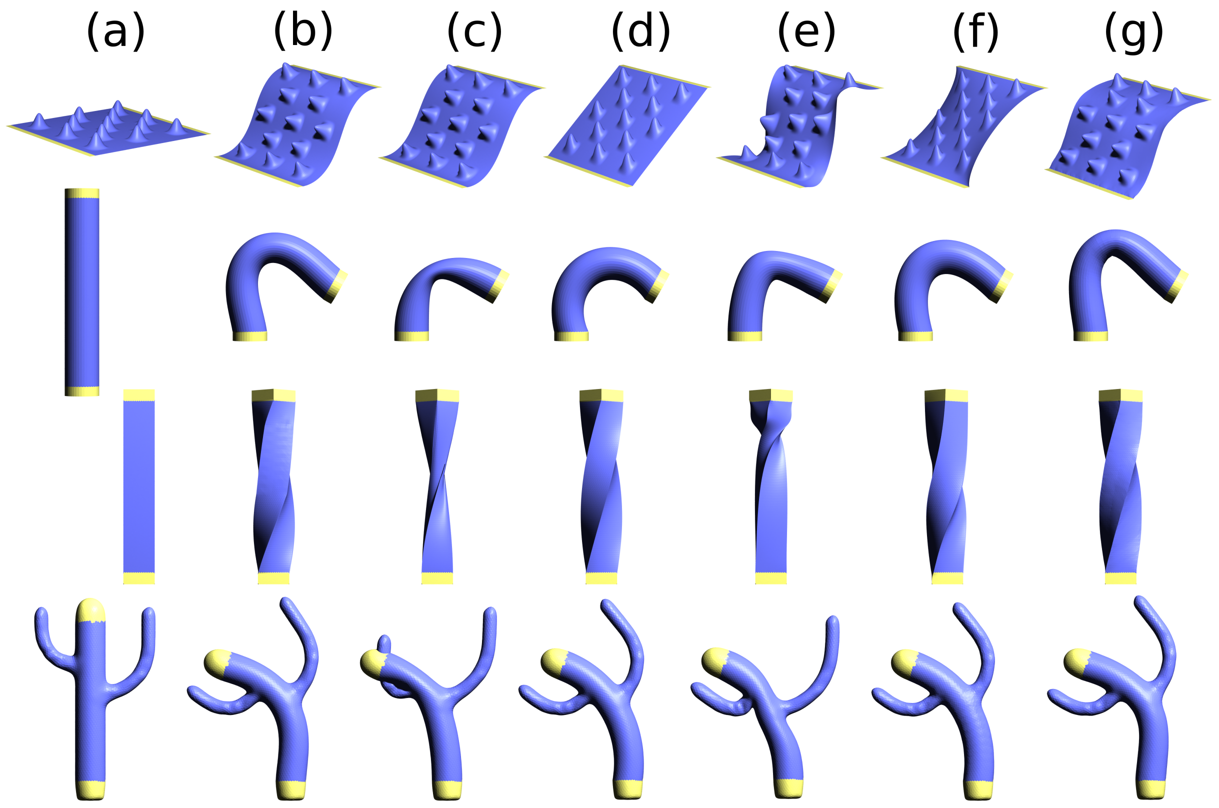

Figure 4: *This grid of images compares deformations produced by a variety of methods in challenging configurations. The constraints in the first row are moved purely by a translation, while the remaining rows contain a bend or twist. The columns from left to right are the resulting meshes after performing (a) no deformation, (b) PriMo [BPGK06], (c) thin shells with deformation transfer [BSPG06], (d) gradient based editing [ZRKS05], (e) Laplacian-based editing with implicit optimization [SCOL*04], (f) rotation invariant coordinates [LSLCO05], (g) our method.*

We compute R_c using the singular value decomposition of $B_c = U_c \Sigma_c V_c^T$ such that $R_c = U_c V_c^T$ [AHB87]. When $det(R_c) < 0$, R_c is a reflection, so we negate the last column in U_c.

While adding details, we also need to ensure that constraints on the positions of detail vertices are satisfied. We do this by optimizing local neighborhoods so that we modify low-resolutions first. We show the full expansion operation, including the constraint matching steps, in Figure 2. If there are any constrained vertices under the influence of q_c, we modify the position of q_c before expanding q_c into an edge. We set the position of q_c to minimize the distance between the predicted positions $M_c c_k$ of the constrained vertices and their target positions d_k. This energy is given by

$$E_x = \sum_k |(d_k - \bar{q}) - R_c(c_k - \bar{p})|^2.$$

The formula for E_x is nearly the same as the formula for E_c, except that rather than summing the error over all vertices in the mesh, we only add the contribution from p_c. Another difference is that E_x is calculated relative to the centroid $\bar{p}$ rather than p_i. Using $\bar{p}$ instead of p_i in E_x produces smoother deformations during the expansion, because the translational component is computed over the entire one-ring rather than simply using the central vertex. After substituting the formulas for $\bar{p}$ and $\bar{q}$ into E_x, it is straightforward to differentiate E_x and solve for the q_c that minimizes E_x. The minimizer of E_x, with m constraints and n neighbor vertices, is

$$q_c = \frac{3}{m}\sum_{k=1}^{m} d_k - \frac{\sum_{j=1}^{n}(q_j + q_{j+1})\Delta_j}{\sum_{j=1}^{n}\Delta_j} + R_c(\bar{p} - \frac{3}{m}\sum_{k=1}^{m} c_k).$$

Once we have calculated the position of q_c, we update R_c by recalculating B_c and performing an SVD decomposition of B_c again. We can iterate this process of solving for q_c by alternately solving for q_c and R_c, but have found that using a single iteration is sufficient. We then determine the positions of q_l and q_r as $q_l = R_c p_l + T_c$ and $q_r = R_c p_r + T_c$.

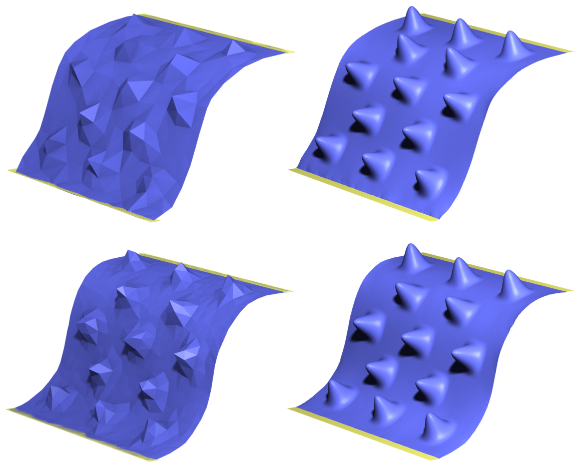

Figure 5: *A comparison of the bumpy plane example using our method at 200 (top) and 500 (bottom) vertices in Q^0. The low-resolution meshes are shown on the left, with the full-resolution meshes on the right.*

3. Results

Our method produces deformations that look nearly as-rigid-as possible by calculating an as-rigid-as-possible deformation over a simplified model and adding approximately rigidly deformed details. This approach greatly reduces the complexity of calculating a deformation while producing high-quality results. Unfortunately, the perceived quality of a deformation is subjective. We provide a qualitative comparison between some deformation methods in a series of examples shown in Figure 4 that have been used in a survey of deformation methods [BS08] to highlight difficult situations where methods often fail to produce reasonable deformations.

In these standard examples, our method produces reasonable deformations with simplified meshes containing 200 vertices. For the cylinder, PriMo is able to prevent compression of the cylinder by bending the shape outward, as does our method. One may notice that the bumpy plane does not match the orientation of the bottom constraints with our method in the comparison diagram. This is an artifact of the constrained region being too thin for the resolution of our mesh with 200 vertices. Figure 5 shows the low-resolution meshes at 200 and 500 vertices for the bumpy plane and the resulting deformations. With 500 vertices, the mesh resolution is high enough that the constrained region is almost a triangle in thickness, and can constrain the orientation of the plane rather than just its position. The bumpy plane, twisted bar, and cactus behave in a plausible fashion for our method, while the methods other than PriMo show obvious artifacts.

Our method also produces nice deformations for complex shapes. Figure 6 shows the complex surface of a dragon that

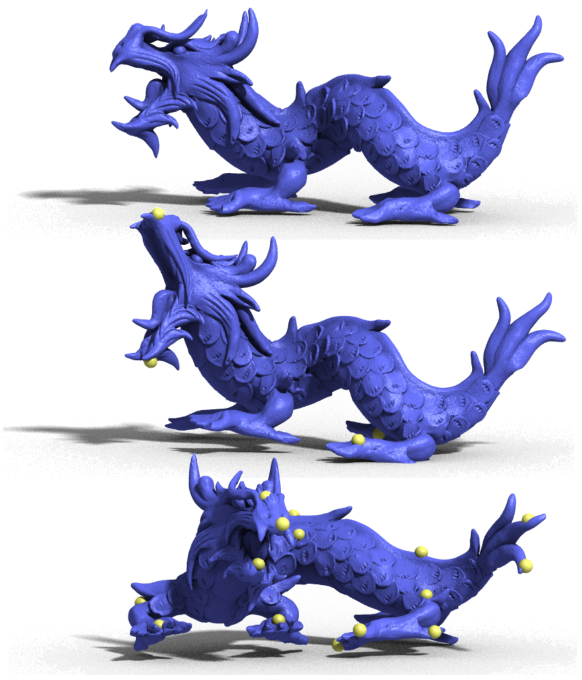

Figure 6: *Deformations of a dragon (bottom, middle) from the rest pose (top) are shown with constraints shown as yellow spheres.*

we deformed using our method. As can be seen from the control points that are displayed as yellow dots, few constraints are required to produce a natural pose.

The global, alternating minimization for the deformation of the base mesh is much slower for high-resolution meshes than it is for low-resolution meshes. If the base mesh is the same as the input mesh, our method is the same as ARAPSM [SA07], because we use ARAPSM to optimize the base mesh. The difference between ARAPSM and our method is that ARAPSM operates only on high-resolution meshes, whereas we add details to a base mesh that we optimize quickly. In part, global minimization of a detailed mesh is slow because each iteration takes a long time. The time taken to calculate best-fit rotations and to back-substitute the *LU* factorized global system is proportional to the number of vertices in the simplified mesh. For the dragon model, we measured that the time to calculate an iteration of the global deformation over 100 vertices was 0.30 milliseconds. For 1000 vertices the time was 3.1 milliseconds, the time for 10,000 vertices was 32 milliseconds, and the full mesh of 112,776 vertices took 0.35 seconds for an iteration. Note that the time taken to calculate a deformation of the base mesh is independent of the number of constraints placed on the model.

Another reason for slow convergence of ARAPSM is the large number of iterations required for the solution to con-

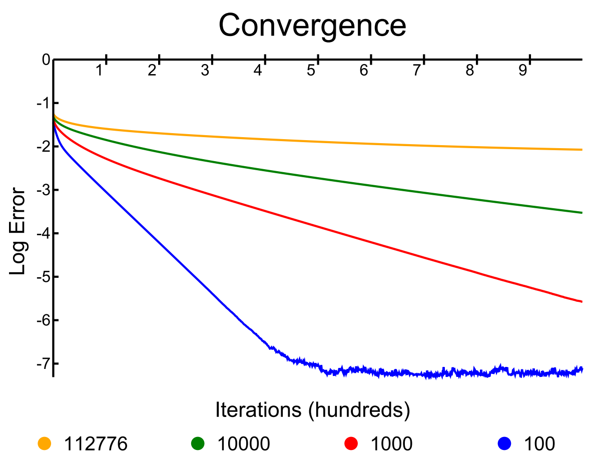

Figure 7: *A single global iteration is allowed per expansion. The* $\log_{10}$ *RMS error is plotted vs. the time to deform the dragon.*

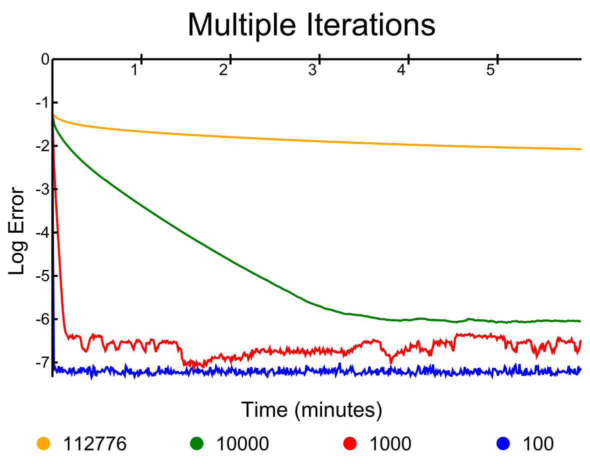

Figure 8: *Multiple global iterations are allowed per expansion, up to how long the expansion takes. The* $\log_{10}$ *RMS error is plotted vs. the time to deform the dragon.*

verge. Figure 7 shows the root mean square (RMS) error of vertex positions in the fully-expanded, deforming mesh compared to vertices in the converged mesh, normalized by the diagonal of the mesh's bounding box. We calculated the time taken to deform the dragon with different resolution base meshes. The plot for the full mesh of 112,776 vertices is shown in yellow, 10,000 vertices is shown in green, 1,000 vertices is shown in red, and 100 vertices is shown in blue. The vertical axis shows the $\log_{10}$ of the RMS error, and 10^{-7} error is at the limit of floating-point precision. After 400 iterations, the mesh with 100 vertices has a thousandth of the error that the mesh with 1000 vertices has and a hundred-thousandth of the error of the full-resolution mesh. The time to converge is the product of the time to perform an iteration and the number of iterations required, which means that using fewer vertices is highly desirable.

For simplified meshes, a single global iteration takes far less time than the expansion of the simplified model to its full resolution, so we can greatly speed up the rate of convergence by calculating multiple global iterations per expansion. We measure the time taken to expand the model, and run as many iterations over the simplified mesh as possible in that period of time. If the time to expand is e and an iteration over the simplified mesh takes s time, the update interval is bounded by $2e+s$.

We plot convergence as a function of time using this technique on the dragon model in Figure 8. Notice that base meshes with few vertices converge very quickly. Our method converges quickly because we are able to run many global iterations in the time it takes to do one full expansion. The time to fully expand the model from 100 vertices with no constraints is 0.33 seconds, and we were unable to measure a difference in time between expanding with no constraints and with 30 constraints distributed evenly over the surface of the dragon. The effects of slow iterations and requiring many iterations compound to make ARAPSM unfeasible to use for high-resolution meshes. Even after several minutes, the complexity of optimizing the full dragon mesh means that ARAPSM is unable to converge, whereas our method can run several iterations very quickly and converge to a reasonable result in less than a second.

Optimizing a low-resolution mesh before adding back details confers another benefit besides speed. As-rigid-as-possible surface deformation methods minimize a thin-shell energy. Under large deformations, this energy allows the surface to fold and pinch in ways that appear unnatural in a solid object. However, when the discretization of the surface is coarse, the edge connectivity is closer to that of a tetrahedralization. By simplifying a model, we approximate the results of a tetrahedralization while performing only surface operations. We then add back details such that we maintain near-rigidity while enforcing vertex constraints.

Figure 9 shows an example of this effect using a base mesh at 200, 500, 2000 vertices, and at full-resolution of 15,002 vertices. The undeformed model is shown on the left, the top row of images shows the low-resolution mesh, and the bottom row shows the resulting mesh after adding back details. The right image shows the result after performing as-rigid-as-possible deformation on the unsimplified mesh, and is equivalent to ARAPSM. Notice that the stomach of the armadillo man collapses in the high-resolution mesh, but that the low-resolution mesh does not collapse and deforms more like a volume. As the resolution of the base mesh increases to the right, the object appears more like a thin-shell and our method approaches ARAPSM.

Although we use multiple mesh resolutions to calculate a deformation, our method is not a multi-grid solver

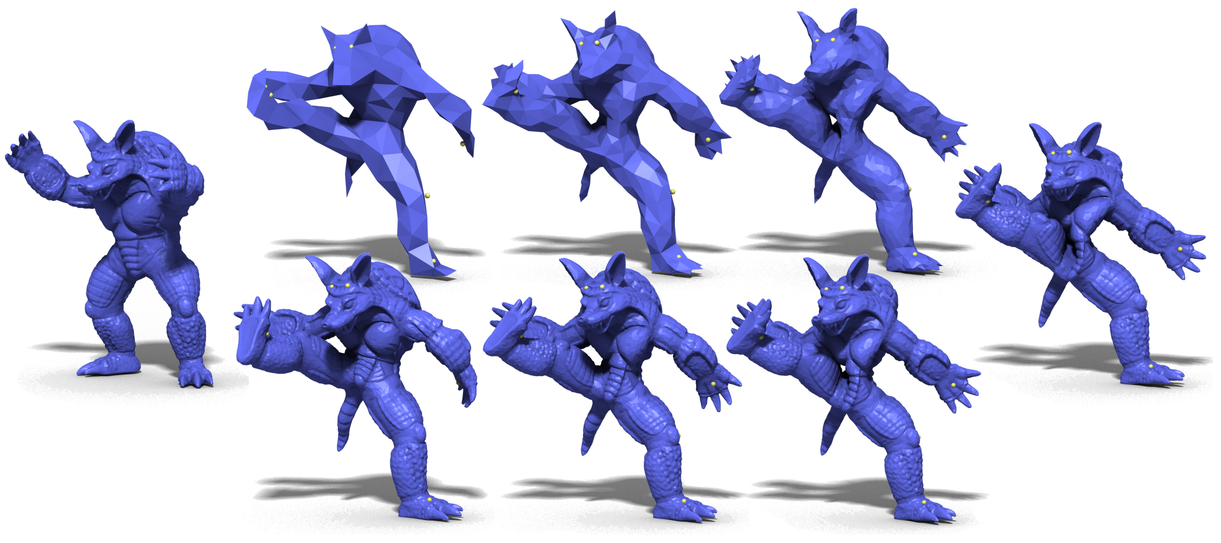

Figure 9: *The rest pose of the armadillo man is shown on the left, followed by deformations with base meshes of 200, 500, 2000 vertices, and unsimplified. Using more vertices makes the mesh deform more like a rubber skin.*

for as-rigid-as-possible deformations. A multi-grid approach would speed up convergence of a deformation, but would converge to the same undesirable thin-shell solution, whereas the deformation calculated by our method makes the object act more like a solid as the number of vertices in the base mesh decreases.

Conceptually, the edge collapses in our hierarchy form a nearly balanced binary tree because our collapse metric favors a uniform triangulation, so the edge collapse tree for n vertices will have an approximate depth of $\log n$. For every level of the tree, any position constraint is included in at most one optimization because there is a unique ancestor for each level. Therefore, the time taken to optimize m constraints during expansion is $O(m \log n)$. Since each expansion takes constant time to update mesh topology, the time taken to apply constrained details to the mesh is $O(n + m \log n)$. Hence, we can deform even massive meshes with our method, because expansion time is nearly linear in vertices, and the time to optimize the simplified mesh is independent of the number of vertices in the detailed model.

4. Conclusions and Future Work

In conclusion, our surface deformation method meets vertex constraints of a high-resolution mesh while preserving local rigidity in the deformed model. Additionally, we calculate deformations quickly enough for artists to interactively deform models with hundreds of thousands of triangles. Artists can add to or remove constraints from models that have already been deformed and can manipulate surfaces with arbitrary triangulations. Although we deform the low-resolution model with an as-rigid-as-possible deformation method, our method of adding details to a low-resolution model does not depend on how the low-resolution model is deformed. For example, it is possible to deform the low-resolution model using skeletal deformation and add back details with our method to reduce artifacts from poor skin weights.

Our method has several benefits over directly optimizing a full-resolution mesh, but has some limitations as well. One problem is that the resulting deformation depends on the resolution of the simplified mesh. Low resolutions can potentially miss important features of the deformation when constraints are close together, but high resolutions will take longer to optimize. We currently do not have an automatic method for choosing a base-resolution, and leave resolution as a user-specified parameter.

Another limitation is that the choice of edge-collapse metric has a strong influence on the resulting deformation, because it affects both the simplified mesh and the order in which we add back details. We chose a metric that favors uniform triangulations so that the mesh will deform uniformly, but when bends should occur at clear joints in the model, the error metric by Garland and Heckbert [GH97] works well. We may be able to take advantage of the effects that different edge collapse metrics have on our method if an artist provides example poses that indicate how joints should bend. When this is the case, we could guide the edge collapses using a deformation-aware collapse metric such as [LS09] to produce better deformations.

There are also a few details of our method that we feel could be improved. A more subtle problem with our collapse metric is that input models that are symmetric can become asymmetric during simplification, which can produce unexpected results. There is also a subtle problem during expansion operations. When we attempt to satisfy constraints during expansion operations, we approximate which vertices are affected by the constraints by using ancestry in the collapse tree. In reality, constrained vertices influence more vertices in the simplified mesh than just their ancestors, and the amount of influence falls-off with distance. It would be interesting if we could approximate the influence of vertices better, but as we calculate the contribution of more constraints per expansion, it may become difficult enforce a low bound on computation time.

Our method may still not be fast enough to be interactive for extremely large models. However, we can take advantage of the multi-resolution nature of our method and perform vertex expansions until a time-limit has expired to maintain a high frame-rate during interaction. The partially expanded mesh approximates the full-resolution mesh and allows editing of truly massive models. It may also be possible to reduce the cost of estimating rigid transformations in our method by exploiting temporal coherence in the SVD calculation and reusing previously computed results as done in FastLSM [RJ07].

Acknowledgements

Funding is provided in part by NSF grant CCF-07024099 and Josiah Manson is supported by an NSF Graduate Research Fellowship.

References

[ACSE05] Attali D., Cohen-Steiner D., Edelsbrunner H.: Extraction and simplification of iso-surfaces in tandem. In *Proceedings of the Symposium on Geometry processing* (2005), Eurographics Association, p. 139. 3, 4

[AHB87] Arun K. S., Huang T. S., Blostein S. D.: Least-squares fitting of two 3-d point sets. *IEEE Trans. Pattern Anal. Mach. Intell. 9*, 5 (1987), 698–700. 5

[BPGK06] Botsch M., Pauly M., Gross M., Kobbelt L.: Primo: coupled prisms for intuitive surface modeling. In *Proceedings of the symposium on Geometry processing* (2006), pp. 11–20. 2, 5

[BS08] Botsch M., Sorkine O.: On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics 14*, 1 (2008), 213–230. 1, 6

[BSPG06] Botsch M., Sumner R., Pauly M., Gross M.: Deformation transfer for detail-preserving surface editing. In *Vision, Modeling, and Visualization (VMV)* (2006), pp. 357–364. 5

[GH97] Garland M., Heckbert P.: Surface simplification using quadric error metrics. In *Proceedings of SIGGRAPH* (1997), pp. 209–216. 3, 4, 8

[GSS99] Guskov I., Sweldens W., Schröder P.: Multiresolution signal processing for meshes. In *Proceedings of SIGGRAPH* (1999), pp. 325–334. 2

[JMD*07] Joshi P., Meyer M., DeRose T., Green B., Sanocki T.: Harmonic coordinates for character articulation. *ACM Trans. Graph. 26*, 3 (2007), 71. 2

[JSW05] Ju T., Schaefer S., Warren J.: Mean value coordinates for closed triangular meshes. In *Proceedings of SIGGRAPH* (2005), pp. 561–566. 1, 2

[KCVS98] Kobbelt L., Campagna S., Vorsatz J., Seidel H.-P.: Interactive multi-resolution modeling on arbitrary meshes. In *Proceedings of SIGGRAPH* (1998), pp. 105–114. 2

[KMP07] Kilian M., Mitra N. J., Pottmann H.: Geometric modeling in shape space. *ACM Transactions on Graphics (SIGGRAPH) 26*, 3 (2007), 64:1–64:8. 2

[LMH00] Lee A., Moreton H., Hoppe H.: Displaced subdivision surfaces. In *Proceedings of Computer Graphics and Interactive Techniques* (2000), pp. 85–94. 2

[LS09] Landreneau E., Schaefer S.: Simplification of articulated meshes. *Comput. Graph. Forum 28*, 2 (2009), 347–353. 8

[LSLCO05] Lipman Y., Sorkine O., Levin D., Cohen-Or D.: Linear rotation-invariant coordinates for meshes. *ACM Trans. Graph. 24*, 3 (2005), 479–487. 2, 5

[RJ07] Rivers A. R., James D. L.: Fastlsm: fast lattice shape matching for robust real-time deformation. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers* (2007), p. 82. 9

[SA07] Sorkine O., Alexa M.: As-rigid-as-possible surface modeling. In *Proceedings of the symposium on Geometry processing* (2007), pp. 109–116. 1, 2, 4, 6

[SCOL*04] Sorkine O., Cohen-Or D., Lipman Y., Alexa M., Rössl C., Seidel H.-P.: Laplacian surface editing. In *Proceedings of the Symposium on Geometry Processing* (2004), pp. 175–184. 2, 5

[SMW06] Schaefer S., McPhail T., Warren J.: Image deformation using moving least squares. *ACM Trans. Graph. 25*, 3 (2006), 533–540. 4

[SP86] Sederberg T. W., Parry S. R.: Free-form deformation of solid geometric models. *SIGGRAPH Comput. Graph. 20*, 4 (1986), 151–160. 2

[SSP07] Sumner R., Schmid J., Pauly M.: Embedded deformation for shape manipulation. *ACM Trans. Graph. 26*, 3 (2007), 80. 2

[SYBF06] Shi L., Yu Y., Bell N., Feng W.-W.: A fast multigrid algorithm for mesh deformation. In *SIGGRAPH* (2006), pp. 1108–1117. 2

[SZT*07] Shi X., Zhou K., Tong Y., Desbrun M., Bao H., Guo B.: Mesh puppetry: cascading optimization of mesh deformation with inverse kinematics. *ACM Trans. Graph. 26*, 3 (2007), 81. 2

[YZX*04] Yu Y., Zhou K., Xu D., Shi X., Bao H., Guo B., Shum H.-Y.: Mesh editing with poisson-based gradient field manipulation. In *SIGGRAPH* (2004), pp. 644–651. 2

[Zel82] Zeltzer D.: Motor control techniques for figure animation. *IEEE Comput. Graph. Appl. 2*, 9 (1982), 53–59. 1, 2

[ZHX*07] Zhou K., Huang X., Xu W., Guo B., Shum H.-Y.: Direct manipulation of subdivision surfaces on gpus. In *SIGGRAPH* (2007). 2

[ZRKS05] Zayer R., Rössl C., Karni Z., Seidel H.-P.: Harmonic guidance for surface deformation. *Computer Graphics Forum 24*, 3 (2005), 601–609. 5