COMPUTER GRAPHICS TECHNICAL REPORTS

CG-2007-3

Simple and Efficient Mesh Editing with Consistent Local Frames

Nikolas Paries, Patrick Degener, Reinhard Klein
Computer Science Dept. II, University of Bonn,
Römerstr. 164, D-53117 Bonn, Germany
{paries, degener, rk}@cs.uni-bonn.de

Institut für Informatik II
Universität Bonn
D-53117 Bonn, Germany

© Universität Bonn 2007
ISSN 1610-8892

Simple and Efficient Mesh Editing with Consistent Local Frames

Nikolas Paries Patrick Degener Reinhard Klein
University of Bonn, Institute of Computer Science II, Germany

Abstract

Mesh editing methods based on differential surface representations like Poisson editing, Laplacian or rotation-invariant coordinates have recently gained much interest as they are both efficient and easy to implement. In order to reconstruct geometry from such representations, rigid local coordinate frames have to be determined for all vertices which is an inherently non-linear problem. Attempts to linearize this problem show at least one of the following problems: First, local frames degenerate for large handle transformations or second, frames can become inconsistent with the reconstructed geometry. Both results in contra-intuitive surface deformation. Existing non-linear approaches do not show these problems. However, they are comparatively slow and considerably more complex.

In this paper we present a differential representation that implicitly enforces orthogonal and geometry consistent frames while at the same time allows for an efficient reconstruction. In particular, it enforces conformal surface deformations so that local texture features are preserved. Tailored to this representation we derive a non-linear reconstruction which is fast as it avoids an iterated matrix factorization and present an efficient implementation, at which most of the calculation is done in parallel on graphics hardware.

The result is a simple and fast mesh editing framework that is robust even for large handle transformations. We demonstrate its efficiency and robustness by several examples.

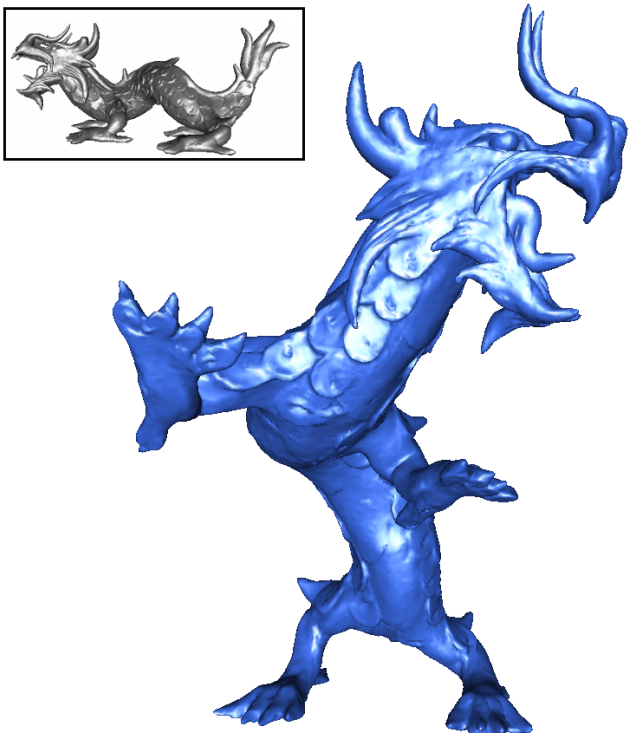

Figure 1. Our non-linear editing method allows for an easy and efficient implementation, that enables intuitive editing of large models at interactive frame rates.

1 Introduction

Due to its numerous applications mesh editing has been an extremely active field ever since the early beginnings of computer graphics and accordingly the variety of approaches is overwhelming reaching from classical splines to multiresolution techniques and space deformations.

Besides these classical approaches, a new class of methods became popular in the last four years that is particularly suited for the manipulation of large triangulated surfaces as supplied by 3D scanning devices. The common characteristic of these approaches is a differential representation of local surfaces to which some authors also refer as "coordinates". For editing, the surface is reconstructed from this differential representation while constraining positions of handle vertices selected by the user.

These methods are very fast while at the same time easy to implement and enable deformation of large meshes at interactive frame rates. In addition, they are formulated directly on triangulated surfaces and thus do not require a prior conversion to volumes or tetrahedra.

Recently, Botsch and Sorkine compared in [7] linear approach based algorithms on differential representations and point out a common central problem: In order to characterize local shape properties independent of its position and orientation, differential representations must be invariant with respect to rigid transformations, e.g. by expressing representations in coordinates of local frames. For mesh editing this leads to a chicken and egg problem as local frames of the deformed surface have to be known to reconstruct it from its differential representation. Solving for both local frames and deformed surface simultaneously, however leads to an inherently non-linear problem.

To circumvent the complexity of non-linear optimization attempts have been made to approximate the problem linearly. While such linear methods are fast, there are two frequent problems: First, local frames are often not constrained to be orthogonal so that they deform in a non-rigid way or even degenerate which causes contra-intuitive surface deformation. Second, some approaches estimate local frames from handle rotations alone so that a pure translation of the handle does not affect the frames. This translation-insensitivity causes frames to become inconsistent with the actual geometry, e.g. the frames normal vector can deviate from the actual normal of the deformed surface, which also leads to contra-intuitive deformations as shown in Figure 2.

These problems of linear methods have motivated to solve the inherent non-linear reconstruction problem directly by applying non-linear optimizations. Although existing non-linear methods yield very plausible deformations even for large handle transformations and pure translations, this comes at the price of comparatively slow and complex reconstruction algorithms.

The contribution of our paper is threefold:

- First, we derive a novel differential representation that is invariant with respect to rigid transformations and implicitly enforces orthogonal frames. Apt to this representation we formulate a non-linear reconstruction problem that includes a frame consistency constraint while allowing for an efficient optimization. Furthermore, we will argue that this formulation enforces conformal deformation that preserves the shape of texture features.

- We propose a fast iterative optimization tailored to our formulation of the reconstruction problem. The key to this is to avoid iterated matrix factorizations as required by standard non-linear solvers. In addition, it does not require sophisticated hierarchical optimization and naturally exhibits frame-to-frame coherence.

- We present a simple and efficient implementation of our algorithm, mainly parallel on graphics hardware, that allows editing of large models at interactive frame rates.

1.1 Previous Work

According to the importance of mesh editing for computer graphics and geometric modeling the amount of previous work in this area is huge prohibiting us a comprehensive discussion. As our approach falls into the class of methods based on differential representations we will therefore concentrate on these approaches and refer to [6] for a survey.

Although in computer graphics the preservation of differential properties has already been applied in modeling of thin shells [11, 20], differential shape representations in context of mesh editing have been introduced by the works of Alexa [1], Lipman et al. [15] and Yu et al. [21]. While the former encode local shape by the Laplacian operator which leads to Laplacian or differential coordinates, Yu et al. use the gradient of coordinate functions as differential representation. Reconstruction from these gradients is equivalent to solving a Poisson equation. A detailed comparison of these approaches and their linear derivatives can be found in a recent survey [7] by Botsch and Sorkine. In the following we briefly comment on both linear and non-linear approaches based on differential representations:

In their original work [21] Yu et al. estimate local frames by applying an interpolated handle transformation to the frames of the undeformed surface. Zayer et al. [22] achieve more intuitive deformations by using harmonic functions for interpolation. Their approach was generalized by Popa et al. [17] to non-uniform materials. All three methods share the problem of translation-insensitivity described in the introduction.

In the Laplace framework of [15] frames are estimated from an underlying smooth base surface. This works well as long as the surface is relatively smooth but produces undesired deformations if it cannot be described as a height field over its base surface. Similarly, Botsch et al. extracted in [8] a base surface that is deformed using a variational minimization approach and estimated local frames from it. Their method can also handle cases where the original surface is no longer a height field over the base surface but yields unintuitive results for large handle deformations.

Sorkine et al. [19] came up with the idea to solve for both geometry and rigid frame transformations simultaneously. For an efficient reconstruction they linearized rotations which causes artifacts for larger rotations as shown in [7]. Following the same basic idea Fu et al. [10] also solve for local transformations but without any rigidity constraint. In a second step, they factor out rotations from these transformations to estimate local frames. While their method better copes with rotations their results seems to be not as good as those obtained by transformation interpolation [22]. Furthermore, vertices in flattish areas need special treatment which complicates their algorithm.

In [16] Lipman et al. proposed another approach to estimate local frames: Borrowing from differential geometry they encode local frame changes by discretizing first and second fundamental forms. This leads to a representation invariant to rigid transformations that they term rotation-invariant coordinates. Reconstruction is done in two stages: First local frames are reconstructed using the discretized forms and then geometry is obtained by solving the Poisson equation. Their method yields good results for large rotations. However, it shows both of the problems mentioned in the introduction. Their approach has been recently improved in [14] to handle rotations larger than 2π and to improve volume preservation.

As a consequence of the above described problems of linear methods, recently attempts have been made to solve the inherent non-linear reconstruction problem directly by applying non-linear optimizations: Pyramid coordinates [18] are like the rotation-invariant coordinates a differential surface representation invariant with respect to rigid transformations. The corresponding non-linear reconstruction problem is solved using an iterative optimization. At the price of a rather slow reconstruction the method is robust even for large deformations and translation-sensitive.

Although physically motivated, the approach of Botsch et al. [5] preserves angles between adjacent triangle which corresponds to an implicit preservation of mean curvature as shown in [11]. In this regard it is related to the approaches here although it does not explicitly extract a differential representation. For mesh editing at interactive frame rates a high dimensional non-linear optimization problem is solved using a highly sophisticated hierarchical solver. The method yields very plausible deformations even for large handle transformations. Huang et al. [13] rewrite the surface Laplacian as linear combination of adjacent triangle normals to obtain a rotation-invariant representation. For reconstruction they solve a non-linear least-squares problem by Gauss-Newton iterations on a subspace domain. The deformation of the subspace is transfered to the mesh using mean value coordinates. Similar to the method of Botsch et al. [5] their method is by far more complex as the approach presented here.

In [2] Au et al. suggest a representation based on the norm of the surface Laplacian as well as its cotangent weights. They also propose a simple and fast iterative reconstruction. However, their method cannot handle surfaces with free boundaries and does not allow for handle scaling.

1.2 General Setup

Denoting the undeformed triangular mesh as $\mathcal{M} = (V, \mathcal{T})$ with a sets of vertices V and triangles $\mathcal{T}$, we adopt the modeling metaphor described in [4]: The user starts editing by specifying a subset $R \subset V$ as region of influence (ROI) to that all further editing is restricted. Then she chooses two further subsets of fixed and handle vertices. During interactive mesh editing fixed vertices are constrained to their original positions and orientation while handle vertices are transformed according to a user-specified affine transformation. In the following we denote the set of both handle and fixed vertices by H. The editing problem consists then in finding a deformed mesh $\mathcal{M}'$ whose handle regions are transformed as specified while the local shape is preserved as much as possible.

2 Formulation

The characterization of local shape and its preservation is at the heart of every mesh editing approach. The basic question that has to be answered is the following: What comprises the local shape of a surface? While many different definitions of local shape have been proposed for triangle meshes over the last years, differential geometry provides us with a clear and simple answer to the question in case of smooth surfaces. It expresses the local shape of a surface in terms of the first and second fundamental form which are defined on regular surfaces. Given a regular surface x defined by a map $x : U \to \mathbb{R}^3$ we can fix a local frame (t^1, t^2, N) in each point $x(u_1, u_2)$ canonically by choosing $t^i = \partial x / \partial u_i$ and N as the surface normal. The first and second fundamental form can then be written as:

$$g_{ij} = \left\langle t^i | t^j \right\rangle \quad h_{ij} = \left\langle \frac{\partial t^i}{\partial u_j} | N \right\rangle \tag{1}$$

From this definition it becomes immediately clear, that both forms and thus all local shape properties are invariant with respect to rigid transformations. Therefore any characterization of local shape should also be invariant with respect to such transformations. Another observation is that the first fundamental form g_{ij} encodes the length and angle of the frames tangential components while the second fundamental form encodes frame differences between neighboring frames.

In [16] Lipman et al. introduce an differential representation based on discrete versions of the first and second fundamental form that encodes differences of adjacent local frames.

However, as described in Section 1.1 their approach shows two problems: First, in the linear least squares solution local frames can deform in a non-rigid way or even degenerate. Second, local frames might not be consistent with the reconstructed geometry (see Figure 2).

Inspired by the approach of Lipman et al. we now propose a novel discrete differential mesh representation that avoids the above mentioned problems. It implicitly preserves the first fundamental form by enforcing orthogonal

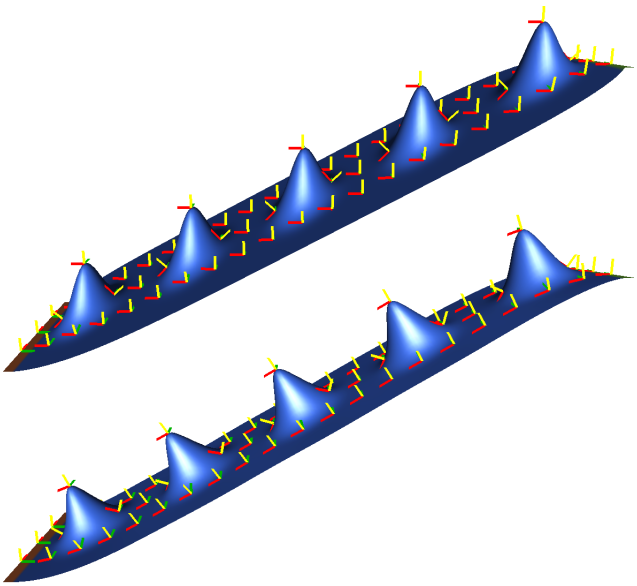

Figure 2. A pure translation is applied to the handle. Top: If local frames are not consistent with the actual geometry, features on the surface are reconstructed in a contra-intuitive and undesired way. Bottom: Enforcing consistent frames leads to a correct reconstruction.

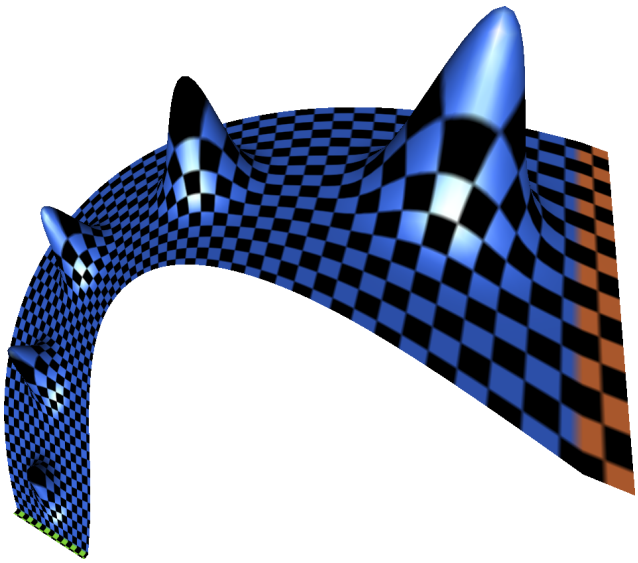

Figure 3. By enforcing conformal deformations local shapes in the texture are preserved even under extreme deformations.

frames. Similar to the second fundamental form this representation encodes differences of neighboring frames in a way that is invariant to rigid transformations.

First, for all vertices in the region of interest $x_i \in R$ and associated normals N_i we fix an arbitrary orthonormal local frame $F_i = (t_i^1, t_i^2, N_i)$ with right hand orientation. In analogy to the second fundamental form we strive to capture differences of adjacent frames. We achieve this by expressing neighboring frames in coordinates of the frame F_i, i.e. for a pair of adjacent vertices (i, j) in $\mathcal{M}$ we compute the matrix

$$F_i^j := F_i^{-1} \cdot F_j. \tag{2}$$

Now, if a rigid transformation $Rx + t$ with a rotation R and $t \in \mathbb{R}^3$ is applied to the mesh local frames F_i will rotate to $R \cdot F_i$ and we have

$$(R \cdot F_i)^{-1} \cdot R \cdot F_j = F_i^{-1} \cdot F_j = F_i^j.$$

Therefore, the definition of F_i^j is invariant with respect to rigid transformations as required.

Since all frames F_i are orthonormal with right handed orientation so are the differences F_i^j and both kind of matrices correspond to rotations. We can therefore rewrite Equation (2) using unit quaternions and the quaternion multiplication as

$$q_i \cdot q_i^j = q_j \tag{3}$$

where q_i, q_j and q_i^j correspond to the rotations F_i, F_j and F_i^j respectively.

One observes that Equation (3) is a linear system in the local frames q_i. To see that it has a unique solution up to a rotation, it is sufficient to fix a unit quaternion q_{i_0} for an arbitrary vertex i_0. Setting $i = i_0$ in Equation (3) the quaternions of adjacent vertices can be computed directly. The quaternions of remaining vertices are obtained analogously by traversing $\mathcal{M}$ in a breadth-first manner. The set of frame differences q_i^j therefore constitutes a rotation invariant representation of the local frames.

The chosen representation of frames as unit quaternions implicitly enforces orthonormal frames. This property ensures that frames do not degenerate even if we add handle constraints to incorporate the user specified handle transformations in the reconstruction. By keeping all frames orthonormal mesh editing is essentially restricted to isometric deformations, i.e. deformations that preserve the first fundamental form and thus angle, length and area of the parameterization. However, enforcing orthonormal frames is usually too restrictive for mesh editing applications. For example the user might want to scale certain parts of a surface which certainly scales length and area of the parameterization. To enable such deformations it is common practice to allow a local isotropic scaling [19, 22].

In our case, this means that we only restrict frames to

be orthogonal not orthonormal. In terms of the parameterization, this allows for deformations that are no longer isometric but conformal. Conformal parameterizations are well known to preserve local shapes, i.e. they preserve angles while allowing length and area to change. If the mesh is textured local features in the texture will therefore also keep their shape during editing as shown in Figure 2.

Fortunately, rotations with isotropic scales can be represented by non-unit quaternions. More precisely, if the rotation R is represented by the unit-quaternion q_r, the combined isotropic scale and rotation sR is given by the quaternion $\sqrt{s}q_r$. This follows immediately from the quaternion-based representation of rotations.

To incorporate isotropic scales in the above described frame reconstruction it is therefore sufficient, to drop the unit length requirement. This is advantageous since constraining Equation (3) to unit-length quaternions would result in a non-linear system.

By now our representation is only based on local frames and does not include the actual geometry. However, as mentioned above frames should be consistent to the geometry to avoid the problem shown in Figure 2, e.g. the normal vector N of each frame should be perpendicular to the reconstructed surface, and the relative position of frame vectors with respect to the 1-ring edges should not change.

To formalize this constraint we require that the coordinates of 1-ring edges in the local frames F_i should not change in the reconstruction. More precisely, for each directed edge (i, j) from vertex x_i to x_j these local coordinates are given by

$$c_i^j := F_i^{-1}(x_j - x_i),$$

which we compute from the original undeformed mesh. In terms of quaternions, this consistency constraint of the frames q_i can then be formulated as:

$$q_i(1, c_i^j)^t \bar{q}_i = (1, x_j - x_i)^t \tag{4}$$

Now we are ready to formulate the constrained reconstruction problem: Given frame differences q_i^j and local coordinates c_i^j for all directed edges (i, j) as well as frame constraints q_k^{const} and positional constraints x_k^{const} for a set of handle vertices H find a set of local frames q_l and absolute vertex coordinates x_l for all vertices within the region of interest R that satisfy

$$\begin{aligned} q_i \cdot q_i^j &= q_j, \\ q_i(1, c_i^j)^t \bar{q}_i &= (1, x_j - x_i)^t \quad \forall i = 1 \ldots n, j \in \text{1-ring}(i) \\ q_k &= q_k^{const}, \\ x_k &= x_k^{const} \quad \forall k \in H \end{aligned}$$

Since this system of equations is overconstrained we solve it in least squares sense to reconstruct a deformed mesh according to the prescribed handle transformations.

Please note that, assuming local frames are known, the geometry can be reconstructed using Equation (4) only. The corresponding normal equation matrix becomes a uniformly weighted Laplacian matrix in this case and solving for the coordinates x_i comes down to reconstruction from Laplacian coordinates. In this respect there is a connection of our approach to linear methods like [15, 16, 21, 22] that try to estimate local frames independent of the geometry.

In contrast to these, in our approach the above equation system is solved for frames and geometry simultaneously. Therefore, the frames q_i are forced to be consistent with the actual geometry x_i. As a consequence, a translation of handle vertices although only specified as constraint to x_k will not only influence the geometry itself, but will indirectly influence the frame field. Therefore, our approach is sensitive to translations as shown in Figure 2.

Although the coupling of frames and geometry turns our formulation in a non-linear problem, we will see in the next section, that this formulation allows for a particularly efficient optimization.

3 An efficient non-linear Optimization scheme

The reconstruction problem formulated in the last section falls into the category of non-linear least squares problems. Such problems are usually solved by Gauss-Newton or Levenberg-Marquardt methods that proceed in an iterative manner requiring the solutions of a sparse linear system in each step. To solve these systems fast sparse direct solvers are commonly applied since they benefit from the fact, that the structure of system matrices does not change.

However, although the matrix structure is constant matrix values do change requiring a refactorization in each step. Unfortunately, iterated refactorization becomes too costly to allow for interactive editing of medium-sized meshes.

To enable interactive mesh editing of large models we therefore strive for an iterative optimization that avoids matrix refactorization by keeping system matrices constant. Our idea is to loosen the above mentioned coupling of frames and geometry during reconstruction allowing them to become temporarily inconsistent. In order to control this deviation we first introduce a set of auxiliary variables: For each vertex in the region of interest we add a quaternion $q_i^{consist}$ of which we think as a frame that is consistent with the actual geometry.

Certainly, we wish that in the final solution q_i and $q_i^{consist}$ should coincide and therefore require them to be close to each other.

If we assume for a moment that a set of $q_i^{consist}$ is known and neglect its dependency on the geometry we can find the frames q_i independent of x_i by solving the sparse linear

system

$$
\begin{aligned}
q_i \cdot q_i^j &= q_j \quad \forall i = 1 \ldots n, j \in \text{1-ring}(i) \\
q_k &= q_k^{const} \quad \forall k \in H \\
q_l &= q_l^{consist} \quad \forall l \in R
\end{aligned} \tag{5}
$$

in least squares sense.

Given a set of q_i as solution of this system we can then compute the left hand side of Equation (4) and reconstruct absolute vertex coordinates x_i by solving a sparse constrained Laplacian system as described in the last section.

Finally, the consistent frames $q_i^{consist}$ have to be updated from the reconstructed geometry x_i. Given an updated set of consistent frames, local frames q_i and geometry x_i can again be computed as just described. We can now formulate our iterative reconstruction as shown in Algorithm 1: Given an initial solution, e.g. the local frames and geometry of the undeformed mesh, it iteratively performs these three steps to refine this solution until it eventually converges.

Algorithm 1 The iterative reconstruction algorithm

Input:
diff. representation (q_i^j, c_i^j)
initial values q_i, x_i
constraints q_i^{const}, x_i^{const}
repeat
  step 1: compute $q_i^{consist}$ from x_i;
  step 2: solve linear system (5) to update q_i;
  step 3: compute left hand side of (4) and solve for updated x_i;
until convergence

Since reconstruction is formulated in least squares sense, a perfectly consistent frame usually does not exist. We therefore actually compute $q_i^{consist}$ as the optimal solution to Equation (4) in least squares sense for a fixed geometry x_i. Although this amounts to a non-linear optimization, it can now be solved for local frames at each vertex individually and constitutes thus only a low-dimensional problem. An efficient solution to this problem is given in [12] that essentially only requires to extract the eigenvector to the largest eigenvalue from a four by four matrix.

Please note that the matrix corresponding to the system of linear Equation (4) only depends on the frame differences q_i^j. The matrix of the constrained Laplacian system is even only dependent on the connectivity of the mesh. Since both do not change as soon as a region of interest has been fixed, these two are constant matrices as desired and can be prefactored in advance using a sparse direct solver.

Although it is difficult to formally proof convergence of the above proposed algorithm, we observed in our experiments that the reconstruction converges after only a few iterations even for large handle transformations.

Furthermore, in a typical mesh editing scenario handle transformations usually change only little between two consecutive frames provided that the system runs at interactive frame rates. The simple iterative nature of our algorithm allows it to fully exploit this frame-to-frame coherence by reusing local frames and geometry of the last frame as initial values for the next. This is in contrast to multigrid methods like [5] that need to optimize over a mesh hierarchy in each frame.

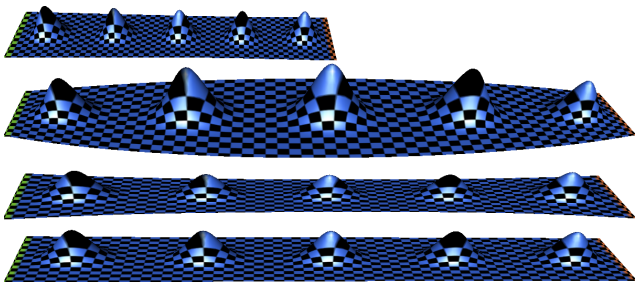

Figure 4. Deformation behavior for different values of α. From top to bottom: Undeformed state, deformation for $\alpha = 1$ (a), $\alpha = -1$ (b) and $\alpha = 0$ (c)

3.1 Parameterizing Deformation

By construction the above described framework produces conformal deformation. In Figure 4a the two handles on both sides of the plate were pushed apart by a translation along the x-axis. Interestingly, the mesh also stretches in the perpendicular direction to preserve the local shape of each 1-ring and thus produces a conformal deformation.

In reality only few materials (so called auxetic materials) show a behavior as depicted in 4a. Linear elasticity characterizes these materials by a negative Poisson's ratio which means that these materials when stretched in one direction also stretch in perpendicular directions. In contrast, most existing materials like metals, rubber or textiles have a positive Possion's ration and correspondingly shrink along unconstrained directions when anisotropically stretched (see Figure 4b).

We can easily extend the above described method to yield mesh deformations similar to those observed for these materials. To this extent we reconsider the update of $q_i^{consist}$ in step 1 of Algorithm 1: As described in Section 3 for each vertex an optimal rotation R_i and an isotropic scale factor s_i is calculated so that the rotated and scaled original 1-ring fits to the deformed 1-ring in least squares sense. Pushing the handles apart as shown in Figure 4 causes an anisotropic stretch along the x-axis in each 1-ring. Therefore the optimal isotropic scale s_i computed in step 1 of the algorithm will be larger than 1. This in turn causes a stretch in the unconstrained directions and finally leads to the above described behavior resembling auxetic materials shown in 4a.

Since in the example of Figure 4 stretch along the x-axis is fixed by the handle constraints, the isotropic scale of $q_i^{consist}$ determines the stretch along the unconstrained y- and z-axis. To obtain a deformation behavior similar to those of materials with positive Poisson's ratio we therefore replace the actual scale s_i of $q_i^{consist}$ by s_i^{α}. For negative α we then obtain the desired shrinking in unconstrained directions as shown in 4b while $\alpha = 1$ results in the original conformal behavior. Figure 4c shows the result for the special case of $\alpha = 0$ that keeps $q_i^{consist}$ normalized so that stretching along the x-axis does not influence the other directions. Such behavior can e.g. be observed for cork which has a Possion's ration of nearly zero. As shown in the figure the parameter α in our method has therefore a similar interpretation as the Possion's ratio. Although our method is not physically motivated, with the above extension the resulting deformations appear physically plausible for different materials. Therefore, its deformation behavior is very intuitive as long as handle transformation does not include scaling.

For scaling, however, there exists no physical counterpart: While real materials can be pulled or bend they cannot be scaled. As demonstrated in Figure 5a physically plausible methods like Primo [5] react to scaling in a way that is intuitive but might yet be undesired. In the figure the case of a pure scaling of the handle by 0.5 is shown. Both Primo and our approach result in a creasing near the scaled handle while the bumps do not scale in height. In some cases, however, the user might desire that features as the bumps near the handle also scale down.

We therefore added a further editing mode that handles scaling differently. In this mode we give up scale consistency between frames and geometry, i.e. we allow frames to scale independent of the actual geometry while still enforcing consistent orientation. This can easily be achieved by rescaling $q_i^{consist}$ after step 1 in Algorithm 1 so that it has the same length as q_i . As shown in 5c this free-scale mode distributes changes in scale over the mesh evenly. Therefore, features near the scaled handle scale down as intended. Please note, that by keeping frame orientations consistent the algorithm still preserves curvature and tangential derivatives. This can be seen from the slight S-shape of the boundaries in 5c.

Both the parameter α and the editing mode do not affect the system matrices in the reconstruction algorithm and can therefore be changed interactively during editing.

4 Implementation Issues

In this section we describe our GPU-based implementation of the reconstruction algorithm and comment on some technical details.

Although the reconstruction algorithm introduced in Section 3 is generally very easy to implement, a slight complication arises that is due to the fact that unit quaternions represent a double cover of the rotation group $SO(3)$. More precisely, two unit quaternions q_R and $-q_R$ represent the same rotation. This remains true if we consider isotropically scaled rotations represented by non-unit quaternions. To compute the correct distance between a given rotation R and a second R' the minimum of the distances from q_R to q'_R and q_R to $-q'_R$ has to be chosen.

When constructing the linear systems in Equation (5) we therefore take care to select q_i^j from the two quaternions representing F_i^j so that the distance between q_j and $q_i * q_i^j$ is minimal. Similarly we switch $q_i^{consist}$ and q_i^{const} so that the distance to q_i is minimal.

Since the quaternions q_i and $q_i^{consist}$ change during optimization this switching must be updated in each iteration. In case of the $q_i^{consist}$ and q_i^{const} this is unproblematic since it does not appear in the constant system matrices. However, in theory it might become necessary to flip individual q_i^j factors in Equation (5) which would require a matrix refactorization. Fortunately, a flip of q_i^j is only necessary if the rotation angle between q_j and $q_i * q_i^j$ is greater than 180 degree. As this distance is zero in the undeformed state, this corresponds to the extraordinary case that the deformations of two adjacent frames differ by a rotation of more than 180 degree. Please note that sharp features or vertices with high curvature do not increase the probability of such a flip as the relevant distance is measured between q_j and $q_i * q_i^j$ and not between q_j and q_i. Therefore such a flip occurs in practice only for extreme deformations in conjunction with unreasonable low mesh resolution, that would anyway require mesh subdivision in order to adequately represent the deformed surface. Although our implementation detects the necessity of q_i^j flips, we never encountered this case in our experiments.

As described in Section 3 the estimation of consistent frames $q_i^{consist}$ amounts to the solution of a set of independent low-dimensional problems and thus naturally lends itself to a parallel implementation. To this extent we used the method of [12] that essentially requires to setup covariance matrices for vertex neighborhoods and to compute the eigenvector to the largest eigenvalue of a four by four matrix. For an easy implementation on graphics hardware we chose the NVIDIA CUDA Framework and adapted an off-the-shelf eigenvalue decomposition which required only minor modifications. Although we did not optimize this code for parallelism or memory access it is fast enough for our purpose and amounts for less than 8 percent of the overall reconstruction time. Our parallel implementation of the eigenvalue and the right hand side vector calculation runs about 80 times faster compared to an equivalent CPU implementation.

With a fast parallel estimation of consistent frames at hand, we strove for an efficient linear solver that is easy

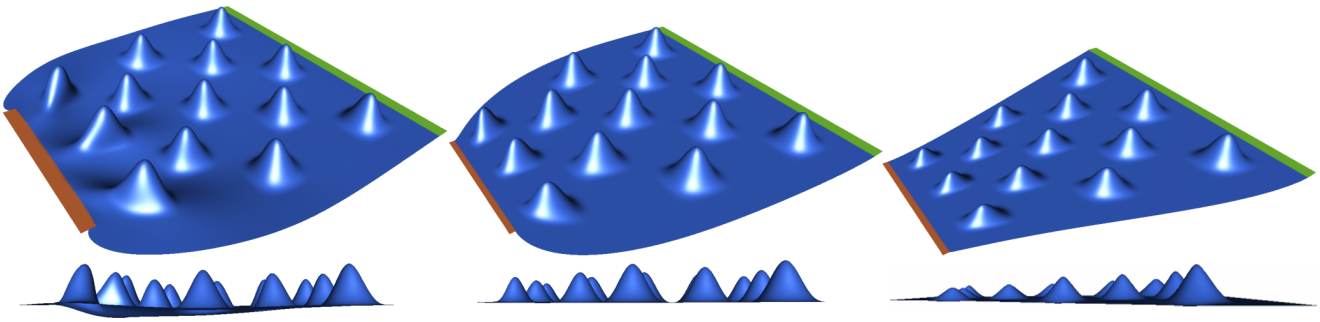

Figure 5. Deformations obtained for pure scaling by 0.5. From left to right: a) Primo, b) our method with $\alpha = 0$ and c) our method in free-scale mode

to implement on graphics hardware. To this extent we tried a conjugate gradient method that naturally lends itself to sparse matrices and parallel execution. However, to our knowledge by now only simple diagonal preconditioners have been implemented on graphics hardware [3] that result in a high number of iterations for our matrices.

Sparse direct solvers are particularly suited for our method since they allow to prefactor the constant system matrices in two sparse triangular factors in the precomputation. During interactive editing they only require fast backsubstitution. Although backsubstitution on a single CPU is very fast, an efficient parallel implementation is challenging and we are currently not aware of an implementation on graphics hardware. Our implementation therefore uses a CPU based backsubstitution from the SuperLU package [9] to solve the linear systems in step 2 and 3 of the reconstruction algorithm. Taken together the backsubstitutions for the two linear systems currently make up nearly 90 percent of the reconstruction time.

As mentioned in Section 3 our algorithm naturally benefits from frame-to-frame coherence during interactive mesh editing. In order to exploit this convergence to its full extend our implementation differs slightly from Algorithm 1. Instead of fixing the handle transformation in the iterative optimization our implementation uses the updated handle transformation in each step. In addition, the mesh is also rendered in each step to provide an immediate feedback. The advantage of this modification is that the user can change or correct handle transformation in each step and does not have to wait until the iteration converges.

5 Results

To demonstrate the quality of the deformations obtained with our method we applied it to the test cases from [7]. The examples in this survey were chosen to reveal limitations of linear approaches, i.e. existing linear methods show gross artifacts on at least one of these examples. Our results are

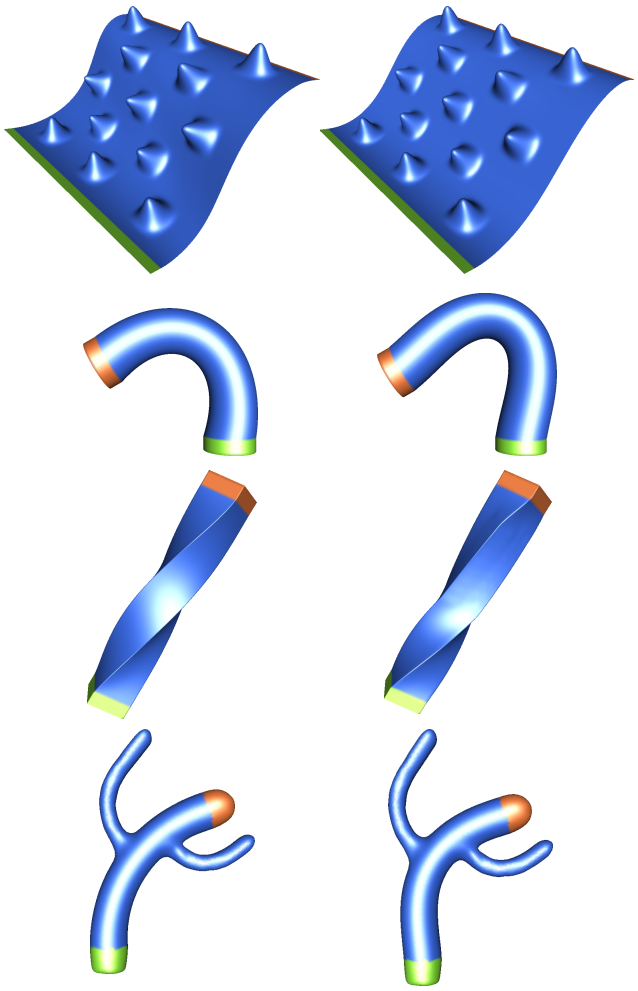

Figure 6. Our results (left) compared to results of Primo [5] (right) for pure translation, a 120° bend, a 135° twist and a 70° bend of different objects

Figure 7. An editing of the Armadillo model in 3 steps: moving and rotating one toe, one claw and the tip of his nose.

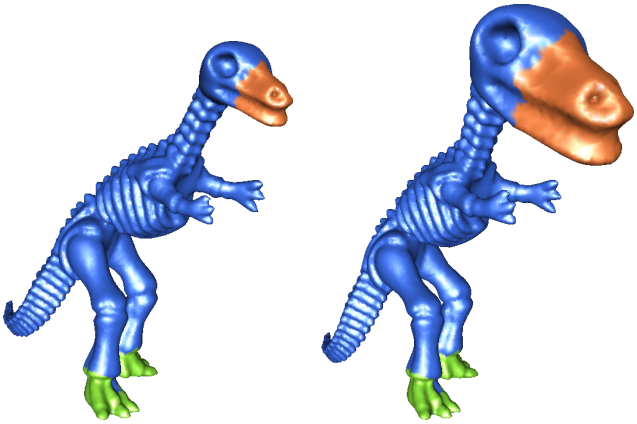

Figure 8. A simple deformation obtained by pure scaling

shown in Figure 6 in direct comparison to the results of the non-linear Primo method [5]. As it can be seen the deformations are physically plausible and comparable in quality to those of Primo. For all test cases we set $\alpha = 0$.

On the cylinder example the two methods show slightly different bending. While for our method the bending is more smooth the Primo result seems to preserve the boundary frames better. For simplicity we used soft constraints to express the frame and positional constraints in the linear systems (4) and (5) in our implementation. Increasing or decreasing these weights yields an additional intuitive way to change deformation behavior. If desired, our method can adapt a similar behavior to Primo by either increasing the constraint weighting in the linear system (5) or by using hard constraints.

Despite its simplicity, our method is fast enough to enable interactive editing of large models: Figure 1 shows the result of a single step editing of the crouching dragon model that took only about 2 minutes of user time. In this editing the region of interest included the whole mesh with 100k triangles. Figure 7 shows three steps in the editing of an even larger model with more than 300K triangles. On both of these models our method runs at interactive frame rates as demonstrated in the accompanying video.

As the major part of the reconstruction time is spent in the backsubstitution the frame rates are inverse proportional to the number of non-zeros in the sparse factors of the system matrices which in turn seems to be nearly proportional to the number of vertices.

In contrast to frame rates, the number of required iterations until convergence is hard to derive. Theoretically, it varies with the shape of handles and the region of interest. In practice, however, we found that it varies only little between typically 3 to 7 iterations which is sufficient for interactive editing as demonstrated in the video.

6 Conclusion

We presented a simple and fast mesh editing method based on a novel differential representation and a reformulation of the reconstruction problem, that enforces consistent and non-degenerated frames. Although our formulation is non-linear, we described an efficient reconstruction scheme that breaks down the high-dimensional non-linear problem into a sequence of linear systems with constant system matrices and a set of independent low-dimensional eigenvector computations.

The resulting algorithm is considerably simpler than previous non-linear methods and yields physically plausible deformations. We also presented a simple modification to emulate deformation behavior of different materials and added an extra editing mode that gracefully handles scaling. Furthermore, our method can be tuned to enforce conformal deformations that preserve texture shapes.

Currently, by far the largest part of the execution time is spent on the CPU based backsubstitution. We therefore

believe that our method has a great potential for speed-up and strive to replace this part of the implementation with a parallel GPU based solver in near future.

7 Acknowledgements

We wish to thank Mario Botsch for the test cases from [7] and the comparison with Primo. We also like to thank the anonymous reviewers for their valuable comments.

References

[1] M. Alexa. Differential coordinates for local mesh morphing and deformation. *The Visual Computer*, 19(2-3):105–114, 2003.

[2] O. K.-C. Au, C.-L. Tai, L. Liu, and H. Fu. Dual laplacian editing for meshes. *IEEE Transactions on Visualization and Computer Graphics*, 12(3):386–395, 2006.

[3] J. Bolz, I. Farmer, E. Grinspun, and P. Schröder. Sparse matrix solvers on the gpu: conjugate gradients and multigrid. *ACM Trans. Graph.*, 22(3):917–924, 2003.

[4] M. Botsch and L. Kobbelt. An intuitive framework for real-time freeform modeling. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 630–634, New York, NY, USA, 2004. ACM Press.

[5] M. Botsch, M. Pauly, M. Gross, and L. Kobbelt. Primo: Coupled prisms for intuitive surface modeling. In *Eurographics Symp. on Geom. Processing*, pages 11–20, June 2006.

[6] M. Botsch, M. Pauly, C. Rossl, S. Bischoff, and L. Kobbelt. Geometric modeling based on triangle meshes. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, page 1, New York, NY, USA, 2006. ACM Press.

[7] M. Botsch and O. Sorkine. On linear variational surface deformation methods. To appear in IEEE Transaction on Visualization on Computer Graphics.

[8] M. Botsch, R. Sumner, M. Pauly, and M. Gross. Deformation transfer for detail-preserving surface editing. In *Vision, Modeling and Visualization*, pages 357–364, 2006.

[9] J. W. Demmel, J. Gilbert, and X. S. Li. SuperLU users' guide. Technical Report CSD-97-944, 8, 1997.

[10] H. Fu, K.-C. O. Au, and C.-L. Tai. Effective derivation of similarity transformations for implicit laplacian mesh editing. *Computer Graphics Forum*, 26(1):34–45, March 2007.

[11] E. Grinspun, A. N. Hirani, M. Desbrun, and P. Schröder. Discrete shells. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 62–67, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[12] B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4:629–642, Apr. 1987.

[13] J. Huang, X. Shi, X. Liu, K. Zhou, L.-Y. Wei, S.-H. Teng, H. Bao, B. Guo, and H.-Y. Shum. Subspace gradient domain mesh deformation. In *SIGGRAPH '06*, pages 1126–1134. ACM Press, 2006.

[14] Y. Lipman, D. Cohen-Or, R. Gal, and D. Levin. Volume and shape preservation via moving frame manipulation. *ACM Trans. Graph.*, 26(1):5, 2007.

[15] Y. Lipman, O. Sorkine, D. Cohen-Or, D. Levin, C. Rössl, and H.-P. Seidel. Differential coordinates for interactive mesh editing. In *Proceedings of Shape Modeling International*, pages 181–190. IEEE Computer Society Press, 2004.

[16] Y. Lipman, O. Sorkine, D. Levin, and D. Cohen-Or. Linear rotation-invariant coordinates for meshes. In *SIGGRAPH '05*, pages 479–487. ACM Press, 2005.

[17] T. Popa, D. Julius, and A. Sheffer. Material-aware mesh deformations. In *SMI '06: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)*, page 22, Washington, DC, USA, 2006. IEEE Computer Society.

[18] A. Sheffer and V. Kraevoy. Pyramid coordinates for morphing and deformation. In *3DPVT '04*, pages 68–75. IEEE Computer Society, 2004.

[19] O. Sorkine, Y. Lipman, D. Cohen-Or, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 179–188. ACM Press, 2004.

[20] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 205–214, New York, NY, USA, 1987. ACM Press.

[21] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.-Y. Shum. Mesh editing with poisson-based gradient field manipulation. In *SIGGRAPH 2004*, pages 644–651, New York, NY, USA, 2004. ACM Press.

[22] R. Zayer, C. Rössl, Z. Karni, and H.-P. Seidel. Harmonic guidance for surface deformation. In M. Alexa and J. Marks, editors, *EUROGRAPHICS 2005*, volume 24 of *Computer Graphics Forum*, pages 601–609, Dublin, Ireland, 2005. Eurographics, Blackwell.