Surface Deformations Driven by Vector-Valued 1-Forms

Gabriel Taubin and Çağatay Demiralp
Brown University, RI 02912, USA.

***Abstract*—We formulate the problem of surface deformations as the integration in the least square sense of a discrete vector-valued 1-forms obtained as the result of applying smooth stretching and rotation fields to the discrete differential of the 0-form defined by the vertex coordinates of a polygon mesh graph. Simple algorithms result from this formulation, which reduces to the solution of sparse linear systems. The method handles large angle rotations in one step and is invariant to rotations, translations, and scaling. We also introduce the integration of 1-forms along spanning trees as a heuristic to speed up the convergence of iterative solvers.**

***Keywords*-triangle meshes; differential forms; gradient domain; algorithms; discrete differential geometry; surface deformation**

I. Introduction

Methods for surface deformation have natural applications in engineering, modeling, and character animation. While many methods have been proposed in recent years, there is still a strong interest for simple and effective algorithms. In the classical theory of surfaces, a surface patch is defined by a smooth 3D-valued parametrization function of two parameters. In the language of differential forms these functions are referred to as 3D-valued differential 0-forms. The two partial derivatives of one of these 0-forms are three dimensional vector fields which define a 3D-valued differential 1-form. A simple approach to surface deformations is to modify this 1-form by locally stretching and rotating its two component vector fields, and then solve for a parametrization function whose partial derivatives match the component vector fields of the modified 1-form.

We formulate the discrete analog of this approach for deformations of graph embeddings, resulting in an extremely simple algorithm. The underlying graphs play the role of the domains of the parametrization functions, and polygon meshes are treated as particular cases of graph embeddings. The integration of a modified discrete vector-valued 1-form is performed in the least-squares sense, and reduces to the solution of a sparse Laplacian system of linear equations. Fast integration of 1-forms along spanning trees results in approximate solutions which speed up the convergence of iterative solvers. The deformations are controlled by smooth stretching and rotation fields associated to vertices and edges of the graph, which can be constructed in various intuitive ways. Large angle rotations are handled in one step, and the method is invariant to rotations, translations, and scaling. Figure 1 illustrates the algorithm.

Contributions: The first contribution of our work is a simpler formulation of mesh deformation/editing using 1-forms which can be used to unify previous results and leads to simpler algorithmic implementations. The second contribution is a spanning tree integration heuristic to speed up the convergence of iterative linear solvers.

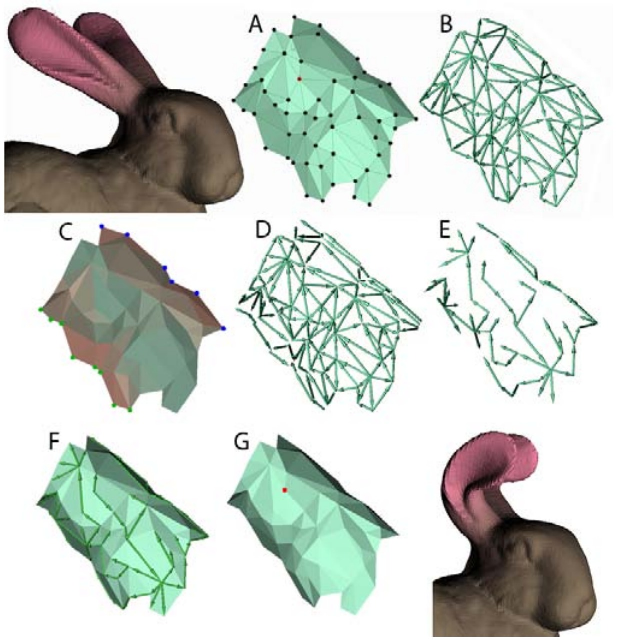

Figure 1: *Algorithm overview. A: Polygon mesh vertex coordinates define a 3D-valued 0-form x on the mesh graph G. B: 1-form v is initialized as the differential dx of x (only one of the two opposite vectors associated to each edge is shown). C: Smooth rotation and stretching fields are created, for example from values attached to handles. D: Rotations and stretching fields are applied to the 1-form v; E: An initial estimate of the deformed 0-form x is computed by integrating v along a spanning tree of G. F: The energy function minimizer x is refined using an iterative solver. G: The result minimizes the quadratic energy function* $\| dx - v \|^2$.

As discussed in an excellent survey [2], linear variational surface deformation methods have been developed for editing detailed high-resolution meshes like those produced by scanning real-world objects. Many of the mesh deformation algorithms mentioned in this survey are based on variational approaches where an energy function which combines approximations of first and second fundamental

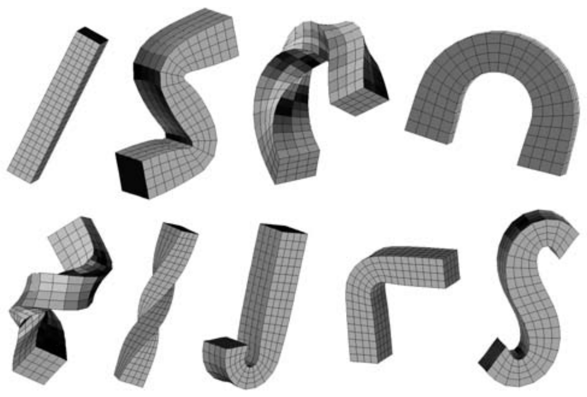

Figure 2: *Various deformations of the bar shown in the upper left corner performed in a single integration step. Arbitrary large rotations and twists do not require multiple steps.*

forms is minimized. Since the first and second fundamental forms are non-linear functions of the surface geometry, accurate approximations of them usually leads to non-linear optimization problems. Therefore, simplifications are made to transform the energy function into a quadratic function of free variables, which is minimized by solving a simple sparse linear system. A common feature of the most recent contributions is to represent the surface with differential coordinates, and through a variational formulation minimize how much these differential coordinates change while the surface undergoes large deformations under the constraints that determine the operation being performed. It turns out that in some of these earlier works the differential coordinates are not rotation-invariant with respect to the global coordinate system. As a result, the details in the deformed mesh are distorted, and the distortion depends on the location and orientation of each detail with respect to the global coordinate system. In [9], [16], [13] it is shown that transforming the differential coordinates with respect to given constraints improves results, as long as deformations are not too large, and the shapes are not too complex. In [10] rotation-invariant differential coordinates are introduced using the discrete analog of Cartan's moving frames [7]. Here a different orthonormal frame is attached to each vertex of the mesh. Solving first for the rotations and then for the positional constraints preserves the rigidity of the local details. The approach is attractive because it only requires the solution of two (large and sparse) linear lest squares problems. However, the geometric meaning of the quantities being minimized is not clear. These problems are partially addressed in [8] through a volume preservation scheme, but the algorithms remain quite complex.

The method described in this paper fits into the linear framework described above, because a deformation is computed by minimizing a quadratic energy function with linear constraints imposed by user, and solving this problem reduces to the solution of a large and sparse Laplacian linear system. However, our energy function does not include approximations of first and second fundamental forms. Also, our conceptually simple approach, which alleviates most shortcomings of recent methods by formulation, explains a number of earlier algorithms, such as Laplacian smoothing [14] and Poisson image editing [12], as special cases. The proposed method can also be regarded as a differential coordinates preservation approach. But neither explicit local frames are constructed and saved, nor local coordinates are computed with respect to local frames for detail transfer.

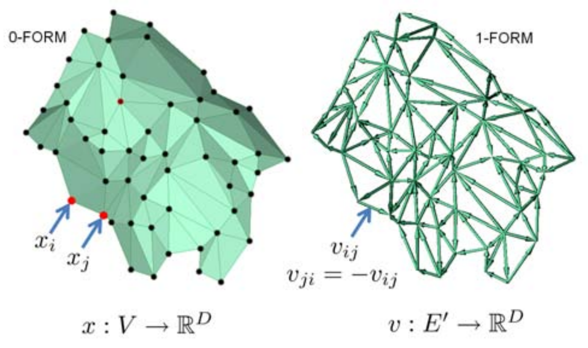

Figure 3: *0-forms and 1-forms on a graph $G = (V, E)$, where V is the set of vertices, E is the set of edges, and E' is the set of oriented edges.*

II. Problem Formulation

To emphasize the simplicity of the proposed approach, and to make the paper self-contained, we present the discrete problem formulation and solution directly in its most abstract form for graph embeddings. More extensive treatment of discrete differential geometry concepts in the geometry processing literature include [11], [6], [4], [15]. Detailed discussion of discrete differential forms can be found in [3].

A. Discrete Vector Valued Forms on Graphs

We consider a finite non-oriented graph $G = (V, E)$ composed of a set of vertices V and a set of edges E, embedded in $\mathbb{R}^D$. We are primarily interested in the case $D = 3$ and graphs obtained from polygon meshes, but the cases $D = 2$ and $D = 1$ are also of practical interest. The embedding is defined by vertex coordinates $x_i \in \mathbb{R}^D$, each one associated with a vertex $i \in V$. We refer to such a mapping $x : V \to \mathbb{R}^D$ as a *discrete D-dimensional 0-form*, and to the pair (G, x) as a *graph embedding*. We refer to a mapping $v : E' \to \mathbb{R}^D$ such that $v_{ij} + v_{ji} = 0$ for each edge $(i, j) \in E$ as a *discrete D-dimensional 1-form*. We denote by E' the set of *oriented edges* of the graph: (i, j) and (j, i) are the same edge in E, but they are regarded as different and *opposite* in E'. Figure 3 illustrates these concepts. From now on all the 0-forms and 1-forms will be

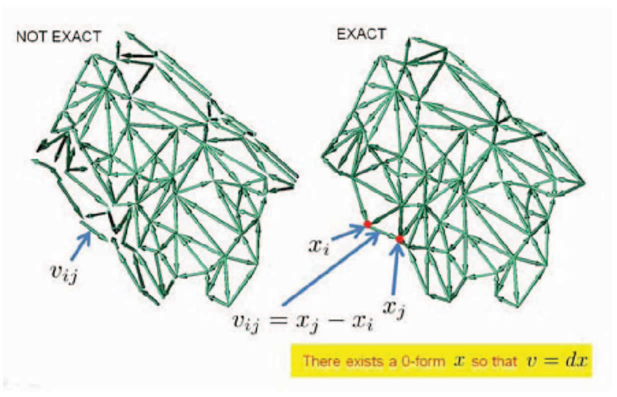

Figure 4: *Exact and not exact 1-forms.*

discrete D-dimensional, and defined on the same graph G. The *differential* of a 0-form x is the 1-form dx defined as $dx_{ij} = x_j - x_i$ for each oriented edge $(i, j) \in E'$. A 1-form v is called *integrable* or *exact* if a 0-form x exists so that $dx = v$, in which case the 0-form x is called *an integral* of v. Figure 4 illustrates these concepts. Note that if v is integrable, the integral is not unique: it is defined up to an additive constant in $\mathbb{R}^D$. If the graph G has cycles, then not every 1-form is exact.

B. Deformations of Graph Embeddings

We apply deformations to a graph embedding (G, x) by first modifying the differential 1-form dx, and then integrating the result. More specifically, the process comprises the following steps:

1) computing the differential 1-form dx;
2) creating a smooth *stretch field* on the graph edges: $\sigma_{ij} = \sigma_{ji} > 0$ for each edge $(i, j) \in E$;
3) creating a smooth *rotation field* on the graph edges: $R_{ij} = R_{ji}$ for each edge $(i, j) \in E$;
4) applying the smooth stretch and rotation fields to the 1-form dx to produce a modified 1-form v:
$$v_{ij} = \sigma_{ij}\, R_{ij}\, dx_{ij} \qquad (i, j) \in E'$$
5) integrating the 1-form v to obtain a deformed 0-form x'.

The deformation process is driven by the smooth stretch and rotation fields, which can be constructed in various ways, including painting interfaces. Our current implementation only supports handle-based construction of these fields using constrained smoothing from values attached to the handles. We use the exponential parametrization of rotations [5], which enables us to define rotations with a single vector per vertex. Once the user defines rotations at the handles we create a smooth rotation field over the mesh by propagating rotations using Laplacian smoothing with hard constraints. Faster direct solvers could be used for this purpose, but our current proof-of-concept implementation does not support them. It is important to note that the same Laplacian systems of linear equations must be solved both for creating the smooth rotation and stretching fields as for integrating the modified 1-forms. The user can also attach stretching factors to handles and similarly create a smooth stretching field over the mesh using Laplacian smoothing.

Linear constraints resulting from geometric constraints, such as specifying some vertex positions, or angles formed by edges, can also be applied. For example, for polygon meshes, specifying face normal vectors results in linear constraints on the vertex positions. However, our current implementation does not support linear constraints.

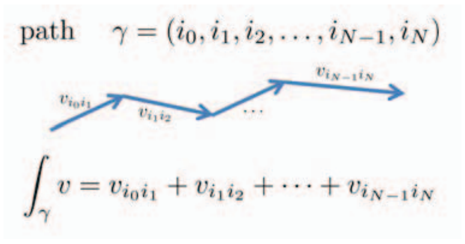
path $\gamma = (i_0, i_1, i_2, \ldots, i_{N-1}, i_N)$
$v_{i_0 i_1}$
$v_{i_1 i_2}$
$\cdots$
$v_{i_{N-1} i_N}$
$\int_\gamma v = v_{i_0 i_1} + v_{i_1 i_2} + \cdots + v_{i_{N-1} i_N}$

Figure 5: *Integration of 1-forms along paths.*

III. Integration of 1-forms

In this section we give a short overview of integration of 1-forms. In Section 3.1 we describe how 1-forms are integrated along paths. Since the modified 1-form v is usually not integrable, in Section 3.2 we discuss how to integrate a 1-form in the least-squares sense, and in Section 3.3 how to integrate 1-forms along spanning trees to speed up the convergence of iterative linear solvers.

A. Integration of 1-forms along Paths

1-forms are entities to be integrated along paths, as figure 5 illustrates. The integral $\int_\gamma v$ of a 1-form v along a path $\gamma = (i_0, i_1, \ldots, i_n)$ in the graph G (a sequence of graph vertices such that each pair of consecutive vertices is an oriented edge), is defined as the vector sum

$$\int_\gamma v = \sum_{h=0}^{n-1} v_{i_h i_{h+1}} = v_{i_0 i_1} + v_{i_1 i_2} + \cdots + v_{i_{n-1} i_n} \ .$$

The value of this integral is a D-dimensional vector. Note that the vertex indices in a path can repeat, and in particular, a path can be closed ($i_0 = i_n$). If two paths γ_1 and γ_2 have the same endpoints, there is no guarantee that the integral of v along the two paths will be the same. However, the integral of an exact 1-form $v = dx$ only depends of the path endpoints:

$$\int_\gamma dx = \sum_{h=0}^{n-1} (x_{i_{h+1}} - x_{i_h}) = x_{i_n} - x_{i_0} \ . \qquad (1)$$

In particular, the integral of an exact 1-form along any closed path is equal to zero.

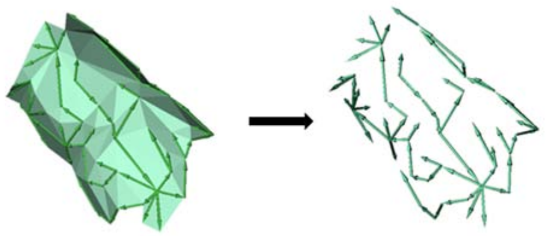

Figure 6: *A spanning tree of a mesh graph.*

B. The Quadratic Energy Function

Since not every 1-form is a differential of a 0-form (i.e., exact) and thus integrable we integrate a 1-form in the least-squares sense by minimizing the following quadratic energy function of a 0-form x

$$\phi(x) = \| dx - v \|^2 = \sum_{(i,j)\in E} \mu_{ij} \| x_j - x_i - v_{ij} \|^2 \quad (2)$$

with the 1-form v regarded as constant. Minimization of this simple energy function drives our algorithm. The norm $\|.\|$ inside the sum is the ℓ_2 norm, and $\{\mu_{ij} > 0 : (i,j) \in E\}$ are positive graph edges weights. The 1-form v is exact if and only if a 0-form x exists that yields zero energy $\phi(x) = 0$. Note that in the absence of additional constraints, $\phi(x)$ does not have a unique minimum: if x and y are 0-forms such that $y_i - x_i$ is a constant in $\mathbb{R}^D$, independent of $i \in V$, then $dx = dy$, and so $\phi(x) = \phi(y)$. However, in general (i.e., for sufficiently complex graphs and 1-forms), this is the only uncertainty in the solution. Specifying the location in $\mathbb{R}^D$ of one vertex x_{i_0}, or of the vertex centroid $\bar{x} = \frac{1}{|V|}\sum_i x_i$ is sufficient to make the solution unique.

The energy function $\phi(x)$ of equation (2) can be written as the sum of D terms, with each term function of one of the D coordinates of the unknown 0-form:

$$\phi(x) = \phi^1(x^1) + \cdots + \phi^D(x^D)$$

where x^h is a $|V|$-dimensional vector composed of all the h-th coordinates of the 0-form values, and

$$\phi^h(x^h) = \sum_{(i,j)\in E} \mu_{ij}\,(x_j^h - x_i^h - v_{ij}^h)^2 \qquad h = 1, \ldots, D$$

Since the constraints do not mix the different coordinates in our case, the problem reduces to solving D linear systems with the same Laplacian matrix [14]. Fast direct solvers based on Cholesky and LU factorizations (i.e., TAUCS, SuperLU, and NAG) for sparse linear systems exist, and they have been used to drive mesh deformation algorithms in the past. A very useful discussion and evaluation of sparse direct solvers for mesh editing can be found in [1]. Our current implementation, however, does not include a fast linear solver yet. Instead, as a proof of concept we implemented two simpler solvers: The Jacobi and conjugate gradient.

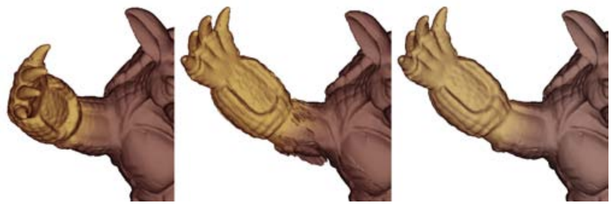

Figure 7: *Armadillo. Left: Undeformed mesh. Center: Result of integrating the deformed 1-form along a spanning tree. Right: Final result after iterative refinement.*

C. Integration of 1-forms along Spanning Trees

Since the quadratic energy function is convex, an iterative solver will converge to the global minimum independently of the initial estimates. For example, the undeformed 0-form x can be used as initial estimate. However, the number of steps to convergence is significantly affected by the initial estimates, particularly if large deformations are being performed, mainly because iterative solvers take very small steps. The magnitude of Δx_i is usually smaller than the average edge length from vertex i to its neighbors. In order to improve the speed of convergence for our iterative minimization, we construct in linear-time a 0-form by integrating the 1-form v along a spanning tree in the connected graph G. Figure 6 shows a spanning tree for a mesh graph. A spanning tree of minimum depth is optimal for this application because it minimizes the accumulation of integration errors along long paths. We obtain good results constructing a maximal spanning tree with respect to edge weights $\mu_{ij} \| x_j - x_i - v_{ij} \|^2$, where x is the undeformed 0-form. Figure 7 illustrates what level of approximation we obtain with this heuristic.

IV. Results

We have developed a prototype implementation of our algorithm in Java in a framework which uses a custom renderer. A typical interaction scenario is for the user to select a region or two on the model surface by painting. Then, based on the intended deformation effect, the user sets the rotation and/or stretch fields for selected regions while smoothing the fields along the surface as desired. Once the user is satisfied with the specified constraints, the program computes the target deformation. At this point the user may decide to modify the constraints, and iterate the process until he is satisfied with the result. We experimented with several mesh models including the *Bunny*, *Bar*, *Armadillo*, *Horse*, and *Hand*. The results shown through Figures 1-5 are promising. Local details are preserved nicely although our formulation do not have any explicit terms to preserve the local frames. In general, we obtain plausible results for arbitrary rotations and stretching in a single integration step.

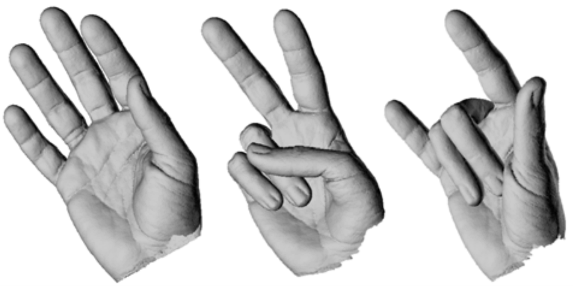
Figure 8: *Hand results. Left: undeformed. Center and Right: results obtained applying only rotations.*

V. Discussion

The contributions of our work are two fold. First, we formulate mesh deformation as an integration of discrete vector-valued 1-forms. This formulation enables a robust and simple algorithm with the ability to provide deformation effects similar to those obtained using more complex existing methods. Our algorithm obtains realistic-looking deformations while preserving local details without requiring construction and/or transfer of local coordinates and frames. Second, we propose the integration along spanning trees of the underlying mesh graph to speed up the convergence of iterative linear solvers in the context of mesh processing. Potential utility of this heuristic goes beyond our current formulation.

In this paper we have focused on demonstrating the main concepts of a new and simple formulation. Therefore, we are well aware that our proof-of-concept implementation is limited, a fact that we intend to remedy in the near future. First, although we experimented only with iterative solvers, our formulation results in a sparse linear system and hence nicely suits for use of high-performance sparse linear solvers. We plan to produce a more efficient implementation based on popular direct and/or iterative sparse linear solvers. Second, we plan to carry out a quantitative analysis of performance gains resulting from the integration along spanning trees heuristic in our algorithm, as well as in other surface deformation algorithms which reduce to the solution of similar equations.

ACKNOWLEDGMENTS

We thank AIM@SHAPE for providing Armadillo and Olivier hand models.

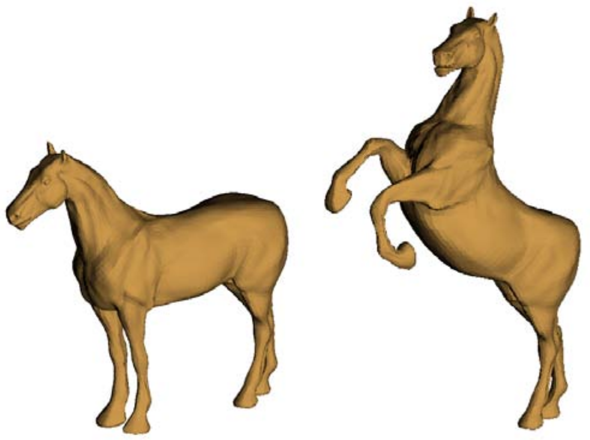
Figure 9: *Horse results. Left: undeformed. Right: result obtained applying only rotations.*

References

[1] M. Botsch, D. Bommes, and L. Kobbelt. Efficient linear system solvers for mesh processing. *IMA Conference on the Mathematics of Surfaces*, pages 62–83, 2005.

[2] M. Botsch and O. Sorkine. On linear variational surface deformation methods. *IEEE TVCG*,14(1):213–230, 2008.

[3] M. Desbrun, E. Kanso, and Y. Tong. Discrete differential forms for computational modeling. In *ACM Siggraph Courses*, pages 39–54, 2006.

[4] S. J. Gortler, C. Gotsman, and D. Thurston. Discrete one-forms on meshes and applications to 3d mesh parameterization. *CAGD*, 23(2):83–112, February 2006.

[5] F. S. Grassia. Practical parameterization of rotations using the exponential map. *J. Graph. Tools*, 3(3):29–48, 1998.

[6] X. Gu and S.-T. Yau. Global conformal surface parameterization. In *Procs. of SGP*, pages 127–137, 2003.

[7] T. Ivey and J. Landsberg. *Cartan for Beginners: Differential Geometry Via Moving Frames and Exterior Differential Systems* AMS, 2003.

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

[9] Y. Lipman, O. Sorkine, D. Cohen-Or, D. Levin, C. Rössl, and H.-P. Seidel. Differential coordinates for interactive mesh editing. In *Procs. of SMI*, pages 181–190, 2004.

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

[11] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. *Visualization and Mathematics III*, chapter Discrete Differential-Geometry Operators for Triangulated 2-Manifolds, pages 35–57. Springer-Verlag, 2003.

[12] P. Perez, M. Gangnet, and A. Blake. Poisson image editing. In *Procs. of ACM Siggraph*, 2003.

[13] O. Sorkine, Y. Lipman, D. Cohen-Or, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Procs. of SGP*, pages 179–188. 2004.

[14] G. Taubin. A signal processing approach to fair surface design. In *Procs. of ACM Siggraph*, pages 351–358, 1995.

[15] G. Tewari, C. Gotsman, and S. J. Gortler. Meshing genus-1 point clouds using discrete one-forms. *C & G*, 30(6), 2006.

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