Local Constraint-Based General Surface Deformation

Richard Pusch
Department of Computer Science
University of Calgary, Canada
rapusch@alumni.ucalgary.ca

Faramarz Samavati
Department of Computer Science
University of Calgary, Canada
samavati@ucalgary.ca

***Abstract*—This paper develops a framework for locally deforming either a parametric surface or hierarchical subdivision surface to match a set of positional and energy minimizing constraints. The positional constraints can be obtained from a wide variety of existing interfaces, and the framework produces a smooth, local and stable deformation through solving a simple local least-squares. We use an indexing scheme to localize optimization to only contributing control points. These points are found and measured by using basis functions or by tracking subdivision mask operations. We demonstrate our framework on B-spline and Loop subdivision surfaces.**

***Keywords*-deformation; parametric surfaces; meshes; subdivision; least-squares;**

I. Introduction

There is an ever present need for modelers to make changes to surfaces during design and creation phases. This task comprises so much of the modeling pipeline that it is important to streamline it as much as possible. The traditional modeling technique of changing many surface "control points", however, is a time-consuming process which requires understanding of the surface representation and may not be the best match the modeler's intention.

A paradigm change to *constraints* is one method for overcoming the drawbacks of control points as a deformation model. Rather than directly manipulating many surface-defining elements, a constraint model specifies how the final surface ought to ideally behave (for example, its final position or smoothness) and then finds the surface representing the best match. This is preferable to adjusting control points for many reasons. For instance, it may be difficult for the modeler to even accurately determine which control points need to be moved to achieve a desired change, and the traditional method often requires several non-trivial, time-consuming adjustments until the deformation is visually acceptable.

In this paper, we develop a robust constraint-based framework for the smooth, local deformation of a surface represented as a parametric patch or with a subdivision hierarchy. The generality of our approach, plus the realization of a fast, local and smooth deformation, are our novel contributions to the field; we develop a tunable paradigm which works with two important surface representations and many existing interfaces, and we achieve fast deformations on local areas regardless of the overall size of the surface. Our method uses both positional 3D constraints, as well as energy minimization constraints. The method for obtaining the positional constraints is not described in depth, as our framework is not dependent on any particular constraint generation technique. They may result from a 3D scan of an object, or they can be generated from a wide variety of existing interfaces [1].

Finding a deformation which "nicely" matches a set of positional constraints is a difficult problem. The deformation must be smooth and stable with a large number of constraints, should only affect a local area of the surface, and the framework still must operate on the surface's inherent representation. The framework is responsible for deciding which control points should be *involved* in the deformation; choosing points which are "near" to the constraints is ideal, but choosing the wrong set can lead to instabilities and loss of locality (Section IV-A).

Given a set of involving control points, the process is then to move these points to best match the given positional constraints. To control the shape and ensure smooth deformations, we place additional *energy constraints* on the system and solve a combined local least-squares system at interactive rates.

II. Related Work

Techniques to deform surfaces that are not based on constraints have been developed in the literature for many surface representations (e.g. [2], [3], [4]). These approaches are often tied directly to a particular surface representation.

The use of constraints to perform deformations has been well-researched in both the parametric and mesh domains, though in the past it has been difficult to develop a constraint-based system in the subdivision context [5]. Hu et al. [6] provide a method for moving a rectangular grid of control points on a parametric surface to match a set of constraints with specified normals. Cheutet et al. [7] encounter a similar energy problem as described in Section IV-B and minimize its effects by removing boundary constraints from the system. Forsey and Bartels [8] fit hierarchical splines with arbitrary basis functions to constraints that conform to a certain "gridded" structure.

In the mesh context, Sorkine and Cohen-Or's least-squares meshes [9] has similarities to our work, though they operate on a mesh with only connectivity information. Certain key vertices near important features are marked as soft constraints, and then the position of every vertex in the mesh is solved through a least-squares system. Each vertex matches to exactly one constraint. LS-meshes can be used

in the context of mesh editing, though not with subdivision surfaces. Nealen et al. [10] improve on LS-meshes in a few ways. Constraints are now expressed through barycentric coordinates and they make use of the cotangent-weighted Laplacians, arguing that it helps create smooth tangents in large deformed areas.

Fibermesh [11] builds models from scratch through the sketching of "defining" curves that are used as constraints. They have chosen to use non-linear optimization of a curvature-based energy functional, which is slower than least-squares solutions. At any time, constraints can be added or modified, but when this is done the entire system is resolved and a new mesh is created, which may be quite different from the old mesh in areas not near to the new constraints. This is one reason a local least-squares system is attractive.

Litke et al. [12] discuss fitting a subdivision surface to a given shape using a quasi-interpolation approach which avoids least-squares calculations. Their approach modifies traditional subdivision surfaces, while our method works on standard surfaces and can be directly ported into existing modeling packages. We also handle irregular meshes without special heuristics and gain more control over the deformed shape with an energy term.

Marinov and Kobbelt [13] develop a technique for fitting a global subdivision surface to a cloud of data points. Their technique iteratively minimizes two norms which prevents real-time results and they heavily change the topology of the mesh which does not suit our application. It is also a requirement for their initial mesh to "be close" to the data points in order to have fast, accurate fitting.

For further reading, see the surveys produced by Botsch and Sorkine [5], Sorkine [14], and Pusch [1].

III. Deforming Parametric Curves

A. Global Deformation

While our final goal is to deform surfaces locally to match constraints, we briefly outline the global 2D curve example using least-squares to ease the notational burden and provide a clear base for the extension to surfaces. For a more detailed look at this technique, see [15] and [16].

Given a set of positional constraints $\mathbf{X} = \{X_0, \ldots, X_n\}$ and a B-spline curve $Q(u) = \sum_{i=0}^{m} B_i(u)P_i$ with initial control vertices $\mathbf{P} = \{P_0, \ldots, P_m\}$, the goal is to adjust each point in $\mathbf{P}$ such that the resulting curve very closely matches $\mathbf{X}$, with $m << n$. The set $\mathbf{D} = \{\Delta_0, \ldots, \Delta_m\}$ contains unknown *deformation vectors* which are added to the corresponding point in $\mathbf{P}$ to achieve this. Denote the set $\widetilde{\mathbf{P}}$ to be the set of deformed control points, where $\widetilde{P}_i = P_i + \Delta_i$, and denote $\widetilde{Q}(u)$ to be the curve defined by $\widetilde{\mathbf{P}}$. $\mathbf{U} = \{u_0, \ldots, u_n\}$ are parameter values that correspond to $\mathbf{X}$; we use arclength parameterization.

Perfectly matching $\widetilde{Q}(u)$ to $\mathbf{X}$ is an overdetermined configuration. Therefore, we use least-squares minimization on the error function:

$$E(\Delta_0, \ldots, \Delta_m) \quad = \quad \sum_{i=0}^{n} \left(\widetilde{Q}(u_i) - X_i\right)^2.$$

If $\mathbf{b}$ is an $n+1$-length vector with $b_i = X_i - Q(u_i)$, we can redefine this error function using vector and matrix operations:

$$E(\mathbf{D}) = (\mathbf{BD} - \mathbf{b})^T(\mathbf{BD} - \mathbf{b}) = ||\mathbf{BD} - \mathbf{b}||_2^2. \quad (1)$$

The minimizer of this function can be found by solving the normal equation [17]:

$$\mathbf{B}^T\mathbf{BD} = \mathbf{B}^T\mathbf{b}. \quad (2)$$

$\mathbf{B}^T\mathbf{B}$ is an $m+1$-square symmetric and semi-positive definite matrix [16], where $\mathbf{B}$ contains each basis function evaluated at each parameter $u_i \in U$; $\mathbf{B}_{i,j} = B_j(u_i)$. Solving for $\mathbf{D}$ produces a curve $\widetilde{Q}(u)$ that matches $\mathbf{X}$ with the least error.

B. Local Deformation

To create local deformations, we introduce *fixed control points* so only a part of the curve is allowed to move. Define a set of integers $S = \{s_0, \ldots, s_k\} \subseteq \{i \mid 0 \leq i \leq m\}$ to be the indices of those control points selected for movement. Methods for choosing S vary and will be outlined in Section IV-A.

The final system of equations remains the same as Equation (2), though $\mathbf{B}$ is modified to contain only those basis functions which belong to points in S and $\mathbf{B}^T\mathbf{B}$ is now $k+1$-square.

IV. Deforming Parametric Surfaces

Moving from curves to surfaces requires special attention in a number of areas. For example, indexing into $Q(u, v)$, a tensor product surface, traditionally requires two indices, which causes trouble for our previous derivations. This can be avoided by imposing a 1D indexing $I : \mathbb{R}^2 \rightarrow \mathbb{R}$ on the surface, such as row-major order. If $I(s, t) = i$, then defining the 2D basis function $B_i(u, v) = B_s(u)B_t(v)$ allows us to rewrite the definition of $Q(u, v)$ with 1D indexing: $Q(u, v) = \sum_i B_i(u, v)P_i$.

This allows the system derivation to follow as per Section III-B, with each 3D constraint X_i having a parameterization $(u_i, v_i) \in U$ obtainable through projection onto $Q(u, v)$. The final system from Equation (2) still holds, where $\mathbf{B}$ is further modified to contain the 2D basis functions. The fact that each B_i is a basis function of the associated spline space ensures that $\mathbf{B}$ is full rank. $\mathbf{B}^T\mathbf{B}$ is still symmetric and now strictly positive definite due to S preventing columns of zeros (Section IV-A), but no longer banded due to the nature of I. Software solutions such as UMFPACK [18] may be used to solve the system quickly.

A. Selecting the Set S

Proper selection of S in the surface context is crucial to the stability of the system. In order to have a solution, $\mathbf{B}^T\mathbf{B}$ must not contain a row or column of all zeros. This implies we cannot choose $i \in S$ such that $B_i(u_k, v_k) = 0$ for all $0 \leq k \leq n$. Conceptually, we must select *only* control points for movement that are "near" the constraints' parameterizations, or find the control points that contribute

to the surface around the constraints. Virtually all choices for S in the curve examples did not experience this singularity due to the nature of arclength parameterization, but choosing S in the surface context is not a task that can fall to the modeler.

We choose S by moving through each parameter $(u, v) \in U$ and choosing the basis function which has *maximal support* in the u- and v-directions. The movable set surrounding the constraints is small (Figure 1) yet still sufficient to match the constraints, and the system is much more stable and local than choosing all non-zero basis functions. By selecting control points through the evaluation of the basis functions, choosing S relies less on visual heuristics and is independent of which basis functions the surface representation uses.

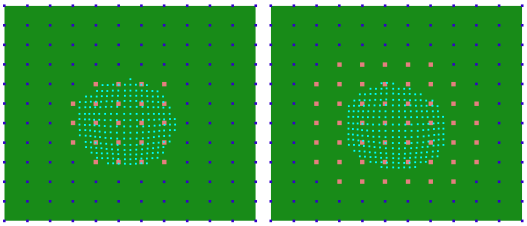

Figure 1. The same set of blue constraints are given, and the selected set S is shown in red. Left: Choosing only the maximal basis functions. Right: Choosing all non-zero basis functions.

B. Energy Minimization

Further control over the shape of the deformation is needed to eliminate areas of high curvature which result from non-regularized least-squares solutions. We introduce an *energy term* into the error function whose goal is to minimize the curvature of the deformed area.

Using complex, non-linear means to evaluate the curvature of a parametric surface [19], [20] is not appropriate when *linear* combinations of control points must be isolated. Therefore, we approximate the curvature using the linear *discrete Laplacian* operator [14]. A grid of surface values M can be precomputed at given resolutions to allow efficient calculation of the Laplacian at a variety of discrete surface points.

The discrete Laplacians of the region to be deformed can be used to remove areas of high energy. Those values in M which contain non-zero contributions from a control point in S are found, and their parameters are placed in a set $T = \{(u_0, v_0), (u_1, v_1), \ldots, (u_{\bar{n}}, v_{\bar{n}})\}$. T is similar to S in that only points in T will be involved in the energy minimizing system. T contains only surface values which will change after deformation.

Derivation of the energy term contains familiar elements to Section III-A. Define a curvature term C, which will be minimized using least-squares:

$$C(\Delta_s \mid s \in S) \quad = \quad \sum_{t \in T} \big(L(u_t, v_t)\big)^2$$

where L is the discrete Laplacian operator. Define a function $Y(t, j)$, $t \in T$, that mimics L but works on the j^{th} B-spline basis function instead of 3D points, a matrix $\mathbf{Y}_{i,j} = Y(i, j)$, and a $\bar{n} + 1$-length vector $\mathbf{y}$ with $\mathbf{y}_i = -L(u_i, v_i)$. The final system which minimizes curvature is:

$$\mathbf{Y}^T\mathbf{Y}\mathbf{D} \quad = \quad \mathbf{Y}^T\mathbf{y}.$$

C. The Complete System

The final system needs to minimize E and C simultaneously. Define a combined error function as follows:

$$T(\Delta_s \mid s \in S) \quad = \quad (1 - \mu)E(\Delta_s) + \mu C(\Delta_s).$$

The constant $\mu \in [0, 1]$ represents the *contribution* of the energy to the system. Small μ is biased towards matching the constraints, and large μ values smoothness.

$T(\Delta_s)$ can be minimized by using μ to combine the minimizers of $E(\Delta_s)$ and $C(\Delta_s)$. Our final system is:

$$\big((1 - \mu)\mathbf{B}^T\mathbf{B} + \mu\mathbf{Y}^T\mathbf{Y}\big)\mathbf{D} = (1 - \mu)\mathbf{B}^T\mathbf{b} + \mu\mathbf{Y}^T\mathbf{y}.$$

Note that because of the properties of the individual components, $(1 - \mu)\mathbf{B}^T\mathbf{B} + \mu\mathbf{Y}^T\mathbf{Y}$ is still a symmetric positive definite matrix. Figure 2 shows the impact μ has on the system.

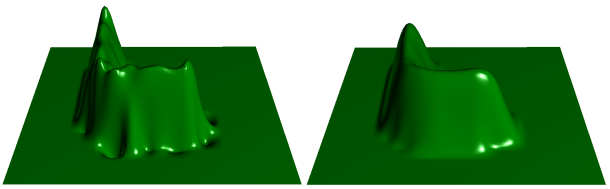

Figure 2. Positional constraints are placed in an "L"-like shape. Left: $\mu = 0$. Right: $\mu = 0.9$.

V. Deforming Subdivision Surfaces

With a solid foundation laid in the parametric domain, we can examine approaches for deforming subdivision surfaces. The biggest difficulty when dealing with discrete surface representations is that they are not directly expressed in terms of basis functions.

A. Basis Functions for Subdivided Objects

Let $\mathbf{P}^0 = \{P_0^0, P_1^0, \ldots, P_m^0\}$ be a base set of "control points" and $\mathbf{P}^i = \{P_0^i, P_1^i, \ldots, P_{\bar{m}_i}^i\}$ be the set after i levels of subdivision. It is necessary to find a method of expressing P_j^i as a linear combination of the points in $\mathbf{P}^0$; identifying these discretized basis functions allows us to model subdivision least-squares systems in a familiar way.

Our approach is to *track the contribution* of the points on each subdivided set of vertices as the subdivision occurs. When subdivision is performed, we can also maintain and store *contribution vectors*. Each B_i^l, the i^{th} contribution vector at the l^{th} subdivision level, will be a vector of $m+1$ values, and the k^{th} element in this vector will be the contribution to P_i^l from the k^{th} point in $\mathbf{P}^0$. To emphasize the meaning of B_i^l, the identity $P_j^i = B_j^i \cdot \mathbf{P}^0$ holds.

B_i^0 is initialized to contain 1 at the i^{th} element, and zeros elsewhere. When subdivision is performed, we apply

the scheme's appropriate masks on both the control points as normal, and **also** on these contribution vectors using standard vector arithmetic. This generates vector-based discrete samples of the underlying basis functions for any arbitrary subdivision scheme.

This technique is similar to a method discussed in Marinov and Kobbelt [13], where they use Stam's evaluation algorithm [21] to directly find the subdivision basis functions on the limit surface. We have chosen to keep discrete vectors representing the basis functions at a given subdivision level for two reasons. Firstly, it is very cheap to obtain these vectors during subdivision, and on most meshes, the minor accuracy improvement Stam's algorithm offers is not worth the cost. Secondly, we prefer to operate on distinct levels of the subdivision hierarchy, rather than the limit surface, as it better fits with practical multiresolution editing.

It is worth emphasizing that the techniques developed throughout this section can still apply directly to a mesh with no subdivision information by simply setting $h = 0$. As such, this research is an important generalization of many standard mesh techniques, as it can also apply to a subdivision hierarchy by increasing h as needed.

B. Deriving the Surface System

A base mesh with vertices $\mathbf{P}^0 = \{P_0, P_1, \ldots, P_m\}$ is loaded, and is subdivided $h \geq 0$ times to obtain $\mathbf{P}^h$ and its corresponding contribution vectors B^h; each point P_i^h has an associated $m+1$-length vector B_i^h such that $P_i^h = B_i^h \cdot \mathbf{P}^0$. The 3D constraints $\mathbf{X} = \{X_0, X_1, \ldots, X_n\}$ have matching parameterizations $U = \{u_0, u_1, \ldots, u_n\}$, where each u_i is a triple containing the barycentric coordinates where the projection of X_i landed on $\mathbf{P}^h$; specifically, $u_i = (u_i^{a_i}, u_i^{b_i}, u_i^{c_i})$, where a_i, b_i, and c_i are the intersecting triangle's vertex indices into $\mathbf{P}^h$. We can parameterize both the surface and the contribution vectors using these barycentric coordinates:

$$\begin{aligned} \mathbf{P}^h(u_i) &= u_i^{a_i} P_{a_i}^h + u_i^{b_i} P_{b_i}^h + u_i^{c_i} P_{c_i}^h \\ B^h(u_i) &= u_i^{a_i} B_{a_i}^h + u_i^{b_i} B_{b_i}^h + u_i^{c_i} B_{c_i}^h \end{aligned}$$

such that the identity $P^i(u) = B^i(u) \cdot \mathbf{P}^0$ still holds.

S and $\mathbf{D}$ are the selected control points in $\mathbf{P}^0$ and the resulting deformation vectors, respectively. Choosing S still entails selecting points "near" the constraints. Each constraint X_i intersects a face on $\mathbf{P}^h$, and this face was subdivided from a *parent face* on $\mathbf{P}^0$. Each vertex on this parent face is tagged for movement and placed in S. This ensures that each selected vertex contributes to some parameter.

We minimize the energy function

$$E(\Delta_s \mid s \in S) = \sum_{i=0}^{n} \left(\mathbf{P}^h(u_i) - X_i\right)^2$$

by solving $\mathbf{B}^T\mathbf{B}\mathbf{D} = \mathbf{B}^T\mathbf{b}$ as before, where $b_i = X_i - \mathbf{P}^h(u_i)$. The entry $\mathbf{B}_{i,j} = B^h(u_i)[s_j]$ is the linear contribution of the j^{th} marked control point to the i^{th} constraint's parameterization.

The energy term must be brought into the subdivision surface formulation, and the discrete Laplacian can once again be used. Given the surface-specific notation, the derivation can follow as per Section IV-B. The set T is comprised of those vertices in $\mathbf{P}^h$ whose parent face in $\mathbf{P}^0$ contains a vertex in S.

The energy term also has a regularizing effect on the shape of the triangles, which has been seen in other works [9], [22]. Our approach nicely extends such techniques to work in the context of a subdivision hierarchy.

VI. Results

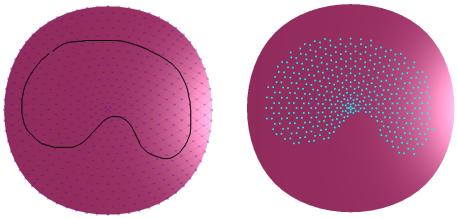

Figure 3. Constraints are added to fill the space on the surface as indicated by a closed sketch.

To input positional constraints, we use a direct sketching tool as demonstrated in Figure 3 and adjust the constraint heights in a hill-like fashion using the mouse wheel. All deformations in this section were computed and adjusted at interactive rates. Figure 4 shows some simple deformations on a flat surface. Figure 5 shows how our framework can be used to create details on existing surfaces and construct models from simple shapes.

Figure 4. Creating a leaf and writing some letters onto a subdivision surface. The constraints for the veins of the leaf and the letters were generated using an etch tool, which placed constraints on the surface to match a user's mouse movement.

VII. Conclusions and Future Work

This paper presents a framework for performing fast, local deformations using both positional and energy constraints on parametric and subdivision surfaces. Finding contribution vectors as a type of discrete basis for subdivision surfaces allows us to elegantly transform the problem from the parametric domain to the subdivision domain. The problem of high-energy deformations is solved by introducing an energy minimization term to the least-squares system. The work nicely combines deforming parametric and subdivision surfaces under one tunable paradigm and can be adapted to work with a variety of existing interfaces with which modelers are familiar. The

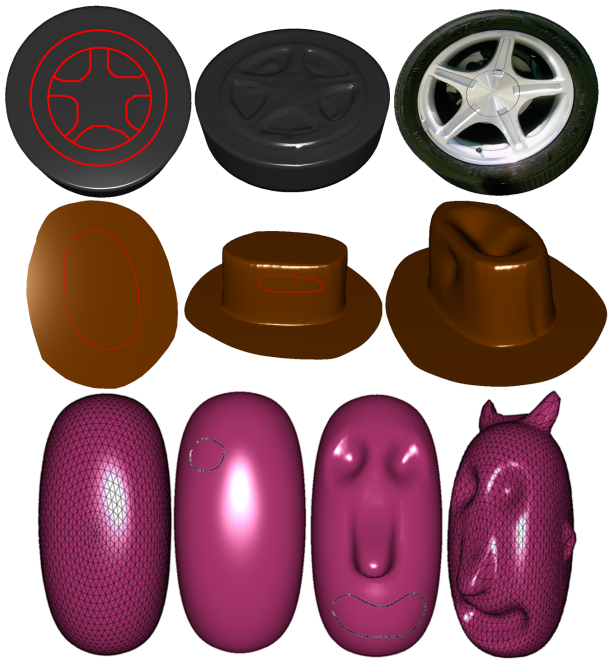

Figure 5. Top: Starting with a cylinder, we create the rim of a tire. Middle: A flat mesh becomes a cowboy hat. Bottom: A face is created by using both standard (eyes, mouth) and non-trivial (nose, horns) constraint height functions.

local nature of our approach ensures fast deformations regardless of the size of the surface.

There are several branches for future work. One example is support for sharp features. The energy term produces a smoothing effect that may not always be desired on particular areas of the surface, and how to introduce sharp features on non-trivial iso-curves of B-spline surfaces is not immediately clear, even with knot insertion techniques. It is worth investigating the results when applying different types of constraints other than soft, such as hard or weighted constraints or constraints on the multiresolution details. Examining this framework's role in a fully-functional multiresolution modeling package is also desirable.

References

[1] R. Pusch, "Constraint-based surface deformation," Master's thesis, University of Calgary, 2009.

[2] M. Botsch and L. Kobbelt, "An intuitive framework for real-time freeform modeling," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 630–634, 2004.

[3] J. J. Cherlin, F. Samavati, M. C. Sousa, and J. A. Jorge, "Sketch-based modeling with few strokes," in *SCCG '05: Proceedings of the 21st spring conference on Computer graphics*. New York, NY, USA: ACM, 2005, pp. 137–145.

[4] L. Olsen, F. Samavati, M. Sousa, and J. Jorge, "Sketch-based mesh augmentation," in *Proc. of 2nd Eurographics Workshop on Sketch-Based Interfaces and Modeling (SBIM'05)*, 2005.

[5] M. Botsch and O. Sorkine, "On linear variational surface deformation methods," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 1, pp. 213–230, 2008.

[6] S.-M. Hu, Y.-F. Li, T. Ju, and X. Zhu, "Modifying the shape of NURBS surfaces with geometric constraints," *Computer-Aided Design*, vol. 33, no. 12, pp. 903–912, 2001.

[7] V. Cheutet, C. Catalano, J. Pernot, B. Falcidieno, F. Giannini, and J. Léon, "3D sketching for aesthetic design using fully free-form deformation features," *Computers & Graphics*, vol. 29, no. 6, pp. 916–930, 2005.

[8] D. Forsey and R. Bartels, "Surface fitting with hierarchical splines," *ACM Trans. Graph.*, vol. 14, no. 2, pp. 134–161, 1995.

[9] O. Sorkine and D. Cohen-Or, "Least-squares meshes," in *Proceedings of Shape Modeling International*. IEEE Computer Society Press, 2004, pp. 191–199.

[10] A. Nealen, O. Sorkine, M. Alexa, and D. Cohen-Or, "A sketch-based interface for detail-preserving mesh editing," *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, vol. 24, no. 3, pp. 1142–1147, 2005.

[11] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa, "Fibermesh: designing freeform surfaces with 3D curves," *ACM Trans. Graph.*, vol. 26, no. 3, p. 41, 2007.

[12] N. Litke, A. Levin, and P. Schröder, "Fitting subdivision surfaces," in *VIS '01: Proceedings of the conference on Visualization '01*. Washington, DC, USA: IEEE Computer Society, 2001, pp. 319–324.

[13] M. Marinov and L. Kobbelt, "Optimization methods for scattered data approximation with subdivision surfaces," *Graphics Models*, vol. 67, no. 5, pp. 452–473, 2005.

[14] O. Sorkine, "Differential representations for mesh processing," *Computer Graphics Forum*, vol. 25, no. 4, pp. 789–807, 2006.

[15] R. Bartels, J. Beatty, and B. Barsky, *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann, 1995.

[16] L. Piegl and W. Tiller, *The NURBS Book*, 2nd ed. Springer, 1996.

[17] Å. Björck, *Numerical Methods for Least Squares Problems*. SIAM, 1996.

[18] T. A. Davis, "Algorithm 832: UMFPACK V4.3—an unsymmetric-pattern multifrontal method," *ACM Trans. Math. Softw.*, vol. 30, no. 2, pp. 196–199, 2004.

[19] M. Do Carmo, *Differential Geometry of Curves and Surfaces*. Prentice Hall, 1976.

[20] G. Farin, *Curves and Surfaces for CAGD: A Practical Guide*, 5th ed. Academic Press, 2002.

[21] J. Stam, "Exact evaluation of catmull-clark subdivision surfaces at arbitrary parameter values," in *SIGGRAPH '98*, 1998, pp. 395–404.

[22] M. Isenburg, S. Gumhold, and C. Gotsman, "Connectivity shapes," in *Proceedings of IEEE Visualization 2001*, 2001, pp. 135–142.