Efficient Nonlinear Optimization via Multiscale Gradient Filtering

Tobias Martin, Pushkar Joshi, Miklós Bergou, Nathan Carr

UUCS-12-001

School of Computing
University of Utah
Salt Lake City, UT 84112 USA

January 17, 2012

Abstract

We present a method for accelerating the convergence of gradient-based nonlinear optimization algorithms. We start with the theory of the Sobolev gradient, which we analyze from a signal processing viewpoint. By varying the order of the Laplacian used in defining the Sobolev gradient, we can effectively filter the gradient and retain only components at certain scales. We use this idea to adaptively change the scale of features being optimized in order to arrive at a solution that is optimal across multiple scales. This is in contrast to traditional descent-based methods, for which the rate of convergence often stalls early once the high frequency components have been optimized. Our method is conceptually similar to multigrid in that it can be used to smooth errors at multiple scales in a problem, but we do not require a hierarchy of representations during the optimization process. We demonstrate how to integrate our method into multiple nonlinear optimization algorithms, and we show a variety of optimization results in variational shape modeling, parameterization, and physical simulation.

Efficient Nonlinear Optimization via Multiscale Gradient Filtering

Tobias Martin[1], Pushkar Joshi[2], Miklós Bergou[3], Nathan Carr[3]
[1]University of Utah, [2]Motorola Mobility, [3]Adobe Systems

Abstract

We present a method for accelerating the convergence of gradient-based nonlinear optimization algorithms. We start with the theory of the Sobolev gradient, which we analyze from a signal processing viewpoint. By varying the order of the Laplacian used in defining the Sobolev gradient, we can effectively filter the gradient and retain only components at certain scales. We use this idea to adaptively change the scale of features being optimized in order to arrive at a solution that is optimal across multiple scales. This is in contrast to traditional descent-based methods, for which the rate of convergence often stalls early once the high frequency components have been optimized. Our method is conceptually similar to multigrid in that it can be used to smooth errors at multiple scales in a problem, but we do not require a hierarchy of representations during the optimization process. We demonstrate how to integrate our method into multiple nonlinear optimization algorithms, and we show a variety of optimization results in variational shape modeling, parameterization, and physical simulation.

1 Introduction

Shape optimization is a problem of fundamental importance to computer graphics. Several graphics tasks can be posed as shape optimization problems, where an input shape is optimized by iteratively modifying its parameters so that an energy associated with the shape is minimized. The choice of the energy depends on the application at hand. Frequently occurring examples include the Willmore energy for variational shape design [35], an angle- and area-preserving energy for surface parameterization [28], and a strain-based energy for statics problems [34]. In all these cases, the shape optimization problem boils down to a numerical optimization of the appropriate energy, subject to proper boundary conditions.

In many cases of practical importance, including the above examples, finding an optimal solution amounts to solving a set of nonlinear equations. Linear approximations (e.g., [3]) do not always produce the expected result [20], and the minimization of nonlinear energies is often necessary. The numerical optimization of such nonlinear energies can be a lengthy process, usually performed iteratively and off-line for starting shapes that are not already close to optimal. Typically, after the first few iterations, the rate of energy decrease is significantly reduced, and the progress towards the minimum is slow.

The reduction in the rate of energy decrease during the optimization process is not necessarily a sign that the shape is close to optimal. More often, this is an indication that the energy being minimized is ill-conditioned. While minimizing an ill-conditioned energy, the rapid initial decrease is due to the optimization of shape features of high spatial frequency, which requires just a few iterations and results in a large reduction of the energy. In other words, the shape becomes *locally* close to optimal. The subsequent progress towards the desired minimum is slow because features of low spatial frequency must be optimized, which require more *global* changes to the shape. This phenomenon is commonly observed in denoising applications, where the high frequency noise can be rapidly removed in just a few iterations, while the low frequency features take a long time to smooth out [33]. The bottom row of Figure 1 shows that after the high frequency details of the dragon model quickly become smooth, the progress towards the global minimum

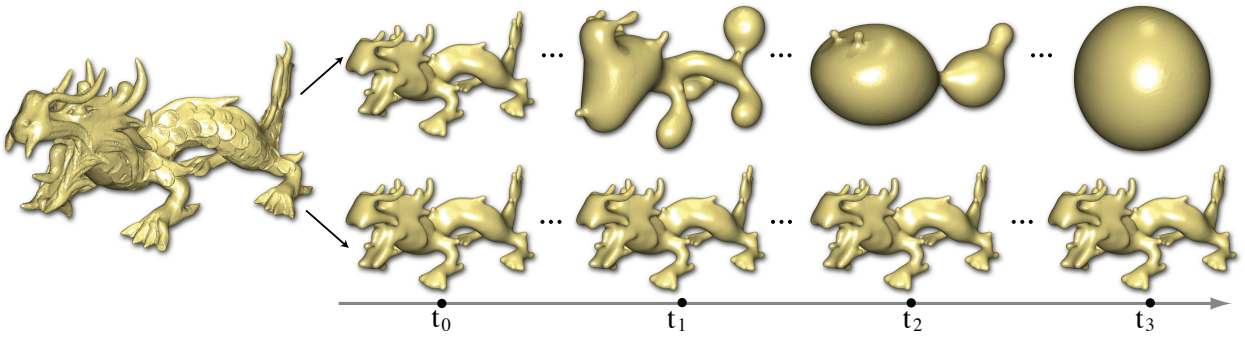

Figure 1: A nonlinear, hinge-based bending energy [9] is optimized using gradient descent for a genus-0 triangle mesh. Depending on the minimization state, our algorithm chooses what feature size to minimize. High frequency features are rapidly removed in the initial optimization steps, and lower frequency features are removed in the later optimization steps. By contrast, a non-preconditioned optimization (shown in the lower row) produces very little shape change as evident in snapshots taken at similar computational times.

becomes negligible.

Contributions Ill-conditioned optimization processes can be sped up by preconditioning, one form of which is to modify the search direction at each step of the optimization process in order to find a direction that allows larger steps towards the minimum. In this paper, we study the Sobolev preconditioner [13]. In particular, we present

- an analysis that shows the connection between the Sobolev gradient and filter design, leading to an intuitive explanation of the behavior of the Sobolev preconditioner in the context of optimization;
- a novel, multi-scale optimization algorithm that uses the Sobolev preconditioner in a general nonlinear optimization pipeline suitable for graphics tasks;
- a novel algorithm for filtering the forces in a physical simulation, which allows larger and more stable timesteps; and
- a comparison that shows the performance benefits of incorporating the Sobolev preconditioner in standard optimization algorithms, such as gradient descent, the nonlinear conjugate gradient method, and the limited memory BFGS (L-BFGS) algorithm [21].

Overview Consider the gradient descent optimization algorithm, in which the gradient of the energy is used as the search direction along which to move towards the minimum. At a high level, the Sobolev preconditioner smooths the standard gradient by removing high frequency components, which allows the optimizer to take steps that change the lower frequency, or more global, components of the shape. We extend this to a multi-scale setting, tuning the frequency response of the preconditioner at each step of the optimization in order to accelerate the overall convergence of the algorithm. This preconditioner can be easily incorporated into an existing implementation of an optimization algorithm by simply applying the Sobolev preconditioner to the search direction at each iteration and leaving the rest of the optimizer unchanged.

We use the Sobolev preconditioner in the contexts of variational shape modeling, parameterization, and elasticity simulation. The results indicate that even for cases where traditional methods tend to stall and do not produce optimal results, we are able to achieve far more optimal shapes in reasonable runtimes (see

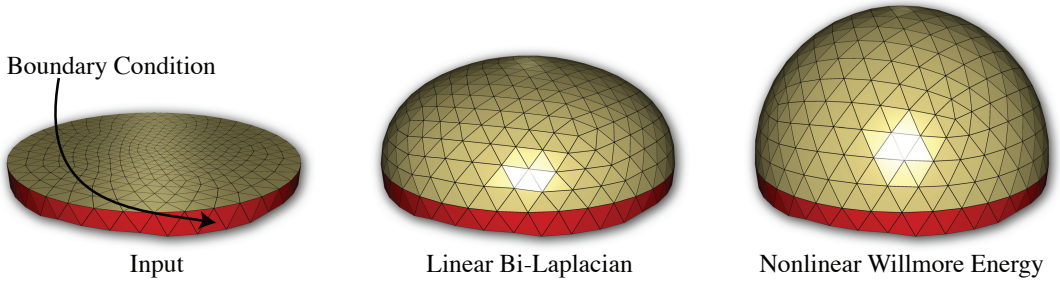

Figure 2: The user-specified tangent constraints (red triangles) are intended to force the input shape (left) to inflate into a hemisphere. A linear method, such as solving the bi-Laplacian in x, y and z, is not able to reconstruct the hemisphere (middle). With our framework, the nonlinear Willmore energy is minimized in a few steps to create the desired shape (right).

Table 1).

2 Related Work

Sobolev preconditioning for elliptic partial differential equations is a well-known approach with a substantial amount of study [13]. In this section we will focus on previous works that use the Sobolev gradient for scientific computing tasks commonly found in computer graphics.

Renka and Neuberger proposed the use of the Sobolev gradient for curve and surface smoothing [25, 26]. Charpiat et al. [5] and Sundaramoorthi et al. [32] used the Sobolev gradient for speeding up curve flows used for active contours in a more general framework that also considered other inner products. Eckstein et al. [7] solved the surface flow equivalent of the previous two papers. These authors all describe how they get significant performance benefits in their flows by using the H^1 Sobolev gradient instead of the standard L^2 gradient. Inspired by these works, we extend this idea to show how higher-order Sobolev gradients can be used as components in preconditioning filters.

The multigrid method [4] is a conceptually closely related, and commonly used, technique for solving ill-conditioned optimization problems. It has successfully been used in a wide variety of contexts, such as finite element methods [27], parameterization [1], and mesh deformation [30]. The multigrid method requires a hierarchical representation for the shape. At each level of the hierarchy, an iterative smoother is used to remove the highest frequency errors. Since the scale of features removed depends on what frequencies can be represented by the discretization, one can use a hierarchy that spans from coarse to fine discretizations to reduce errors across multiple scales. While multi-resolution preconditioners are effective, representing the input shape hierarchically for general representations can be challenging [10].

In Section 5, we demonstrate several applications of our method. We include a discussion of work related to the applications in that section.

3 Mathematical Framework

In this section, we formally introduce the relationship between the gradient of an energy and the inner product on the space of shape deformations. We start with the L^2 gradient, and we show how other gradients can be constructed. For one particular form of this construction, we provide an interpretation of this operation as a filter applied to the L^2 gradient. In Section 4, we exploit this filtering viewpoint to create a shape optimization algorithm in the spirit of a multigrid approach.

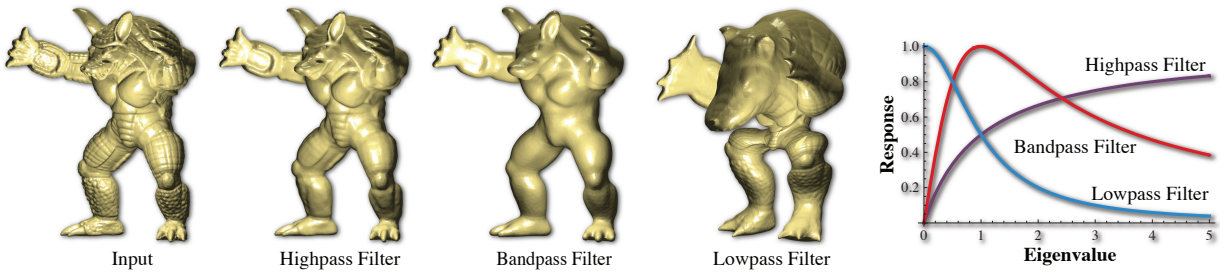

Figure 3: Each of the filters shown on the right can be used to select which eigenmodes of the Laplacian to keep in the gradient during the optimization process. The figures on the left, in which we applied the various filters to the L^2 gradient of the Willmore energy, show that the eigenmodes correspond to surface features at different scales.

Shape and Tangent Space Let Ω be the space of possible shapes for the problem of interest, and let $\mathbf{x} \in \Omega$ be a point in this space. The tangent space, $T_{\mathbf{x}}\Omega$, contains all tangent vectors to Ω at the point $\mathbf{x}$. Given a sufficiently smooth path, $\boldsymbol{\gamma} : \mathbb{R} \to \Omega$, such that $\boldsymbol{\gamma}(t) = \mathbf{x}$, the tangent space represents the space of all possible velocities $\boldsymbol{\gamma}'(t)$ that the path could have at $\mathbf{x}$.

We discretize Ω using a piecewise polynomial basis. In the discrete setting, a point in shape space is given by $\mathbf{x} = \sum_{i=1}^{n} x_i \phi_i$, where $\{\phi_i\}_{i=0}^{n}$ is the set of basis functions and $\{x_i\}_{i=0}^{n}$ is the set of coefficients that defines the shape. For example, one possible discretization of a surface embedded in $\mathbb{R}^3$ is a triangle mesh, for which $\{x_i\}_{i=0}^{n}$ is the set of vertex positions. We can write any path in the discrete setting as $\boldsymbol{\gamma}(t) = \sum_{i=1}^{n} x_i(t)\phi_i$ with corresponding tangent vector given by $\boldsymbol{\gamma}'(t) = \sum_{i=1}^{n} x_i'(t)\phi_i$. As in the continuous setting, the tangent space at a point is the space that contains all possible velocities that a path could have at that point.

Inner Product on Tangent Space We can equip the tangent space with an inner product, $\langle \mathbf{u}, \mathbf{v} \rangle$ for $\mathbf{u}, \mathbf{v} \in T_{\mathbf{x}}\Omega$. The canonical L^2 inner product is given by $\langle \mathbf{u}, \mathbf{v} \rangle_{L^2} = \int \mathbf{u} \cdot \mathbf{v}$.

In the discrete setting, the L^2 inner product for two tangent vectors $\mathbf{u}$ and $\mathbf{v}$ can be written in matrix form as

$$\begin{aligned}\langle \mathbf{u}, \mathbf{v} \rangle_{L^2} &= \langle \sum_{i=1}^{n} u_i \phi_i, \sum_{j=1}^{n} v_i \phi_i \rangle_{L^2} = \sum_{i=1}^{n}\sum_{j=1}^{n} u_i v_j \langle \phi_i, \phi_j \rangle_{L^2} \\ &= \mathbf{u}^T \mathbf{M} \mathbf{v} \ ,\end{aligned}$$

where in the second line, $\mathbf{u}$ and $\mathbf{v}$ are the vectors of coefficients $\{v_i\}_{i=1}^{n}$ and $\{u_i\}_{i=1}^{n}$, and $\mathbf{M}$ is commonly called the "mass matrix" whose entries are defined by $\mathbf{M}_{ij} = \langle \phi_i, \phi_j \rangle_{L^2}$.

As noted by Eckstein et al. [7], we can define alternate inner products using

$$\langle \mathbf{u}, \mathbf{v} \rangle_{\mathcal{A}} = \langle \mathcal{A}\mathbf{u}, \mathbf{v} \rangle_{L^2} = \langle \mathbf{u}, \mathcal{A}\mathbf{v} \rangle_{L^2} \ , \qquad (1)$$

where $\mathcal{A} : T_{\mathbf{x}}\Omega \to T_{\mathbf{x}}\Omega$ is a self-adjoint, positive definite linear operator.

The choice of inner product is key and, as we will show, has a large impact on the performance of descent-based algorithms. Our goal in Section 4 will be to tailor $\mathcal{A}$ to the task of optimizing the performance of these algorithms.

Gradient Vector Field Given a smooth energy, $E : \Omega \to \mathbb{R}$, that assigns a real value to a given shape, the gradient of E at $\mathbf{x}$ is defined as the vector field, $\mathbf{g}(\mathbf{x}) \in T_{\mathbf{x}}\Omega$, that satisfies

$$\langle \mathbf{g}(\mathbf{x}), \mathbf{v} \rangle = \lim_{\epsilon \to 0} \frac{E(\mathbf{x} + \epsilon \mathbf{v}) - E(\mathbf{x})}{\epsilon}$$

for all $\mathbf{v} \in T_{\mathbf{x}}\Omega$ (see, e.g., [6]). In other words, the gradient is the vector field whose inner product with

$\mathbf{v}$ yields the rate of change of the energy along $\mathbf{v}$. Since the gradient is defined with respect to an inner product, each inner product leads to a different vector field corresponding to the gradient. We can relate the $\mathcal{A}$ gradient to the L^2 gradient by combining $\langle \mathbf{g}_{\mathcal{A}}(\mathbf{x}), \mathbf{v} \rangle_{\mathcal{A}} = \langle \mathcal{A}\mathcal{A}^{-1}\mathbf{g}_{L^2}(\mathbf{x}), \mathbf{v} \rangle_{L^2}$ with (1) to obtain

$$\mathbf{g}_{\mathcal{A}}(\mathbf{x}) = \mathcal{A}^{-1}\mathbf{g}_{L^2}(\mathbf{x}) \ . \tag{2}$$

Therefore, if we have an expression for the L^2 gradient of the energy, we can solve for the $\mathcal{A}$ gradient using this equation.

In the discrete setting, the energy E depends on the coefficients $\{x_i\}_{i=1}^n$, which we write using an abuse of notation as $E(\mathbf{x})$. We can evaluate the rate of change of E at $\mathbf{x}$ in the direction of $\mathbf{v}$ as

$$\begin{aligned} \sum_{i=1}^{n} \frac{\partial E(\mathbf{x})}{\partial x_i} v_i &= \frac{\partial E(\mathbf{x})}{\partial \mathbf{x}} \cdot \mathbf{v} = \langle \mathbf{g}_{L^2}(\mathbf{x}), \mathbf{v} \rangle_{L^2} \\ &= \left(\mathbf{g}_{L^2}(\mathbf{x})\right)^T \mathbf{M}\mathbf{v}, \end{aligned}$$

which, using the symmetry of $\mathbf{M}$ and the fact that this equation must hold for arbitrary $\mathbf{v}$, leads to

$$\mathbf{g}_{L^2}(\mathbf{x}) = \mathbf{M}^{-1}\frac{\partial E(\mathbf{x})}{\partial \mathbf{x}} \ .$$

From (2), we obtain that for a linear operator $\mathcal{A}$,

$$\mathbf{g}_{\mathcal{A}}(\mathbf{x}) = \mathcal{A}^{-1}\mathbf{M}^{-1}\frac{\partial E(\mathbf{x})}{\partial \mathbf{x}} \tag{3}$$

Frequency Response of Gradient Filter Motivated by the work of Kim and Rossignac [14] on constructing filters for mesh processing, we choose $\mathcal{A} = \mathbf{B}^{-1}\mathbf{A}$, where $\mathbf{B}$ and $\mathbf{A}$ are both linear combinations of powers of the Laplace-Beltrami operator, Δ, given by $\mathbf{A} = \sum_{i=0}^{p}(-1)^i a_i \Delta^i$ and $\mathbf{B} = \sum_{i=0}^{q}(-1)^i b_i \Delta^i$. Here, $\{a_i\}_{i=0}^p$ and $\{b_i\}_{i=0}^q$ are the sets of scalar coefficients that define each linear combination, and $\Delta^0 = \mathbf{I}$ is the identity operator. Note that using this definition for $\mathcal{A}$, the resulting inner product is no longer symmetric. However, for the purposes of optimization, this is a valid choice because the resulting gradient can still be used as a descent direction.

In order to construct the $\mathcal{A}$ gradient in the discrete setting, we require a discrete version of the Laplace-Beltrami operator, denoted by $-\mathbf{L}$. For example, we use the well-known cotan operator [23] for optimization problems involving triangle meshes. In general, we define the discrete Laplace-Beltrami operator as $-\mathbf{L} = -\mathbf{M}^{-1}\mathbf{K}$, where $\mathbf{M}$ is the mass matrix defined as before by $\mathbf{M}_{ij} = \langle \phi_i, \phi_j \rangle$ and $\mathbf{K}_{ij} = \langle \nabla\phi_i, \nabla\phi_j \rangle_{L^2}$ is the stiffness matrix.

Given a discrete Laplace-Beltrami operator, we can write

$$\mathbf{A} = \sum_{i=0}^{p} a_i \mathbf{L}^i \qquad \text{and} \qquad \mathbf{B} = \sum_{i=0}^{q} b_i \mathbf{L}^i \tag{4}$$

and use (3) to obtain

$$\mathbf{A}\mathbf{g}_{\mathcal{A}} = \mathbf{B}\mathbf{M}^{-1}\frac{\partial E(\mathbf{x})}{\partial \mathbf{x}} \ . \tag{5}$$

Kim and Rossignac [14] show that, under some reasonable assumptions, the eigendecompositon of $\mathbf{L}$ can be written as

$$\mathbf{L} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1} \ ,$$

where $\mathbf{\Lambda}$ is a diagonal matrix whose diagonal elements, $\lambda_i = \mathbf{\Lambda}_{ii}$, are the eigenvalues of $\mathbf{L}$, and $\mathbf{Q}$ is a matrix whose columns are the eigenvectors of $\mathbf{L}$. Using this decomposition, we can write (5) as

$$\left(\sum_{i=0}^{p} a_i \mathbf{\Lambda}^i\right)\mathbf{Q}^{-1}\mathbf{g}_{\mathcal{A}} = \left(\sum_{i=0}^{q} b_i \mathbf{\Lambda}^i\right)\mathbf{Q}^{-1}\mathbf{g}_{L^2} \ .$$

Each row of this equation corresponds to an eigenmode of $\mathbf{L}$, which we can use to examine what happens to the corresponding eigenmode of $\mathbf{L}$ that is present in $\mathbf{g}_{L^2}$ when we compute $\mathbf{g}_{\mathcal{A}}$. We rewrite row j as

$$\left(\mathbf{Q}^{-1}\mathbf{g}_{\mathcal{A}}\right)_j = h(\lambda_j)\left(\mathbf{Q}^{-1}\mathbf{g}_{L^2}\right)_j \ ,$$

where $h(\lambda)$ is the transfer function given by

$$h(\lambda) = \frac{\sum_{i=0}^{q} b_i \lambda^i}{\sum_{i=0}^{p} a_i \lambda^i} \ . \tag{6}$$

This transfer function encodes the amount to which the eigenmode with eigenvalue λ is amplified or attenuated when computing $\mathbf{g}_{\mathcal{A}}$ from $\mathbf{g}_{L^2}$. Thus, by varying the coefficients a_i and b_i, we can control the frequency response of $\mathcal{A}^{-1}$, which we exploit in the following section when designing our optimization procedure.

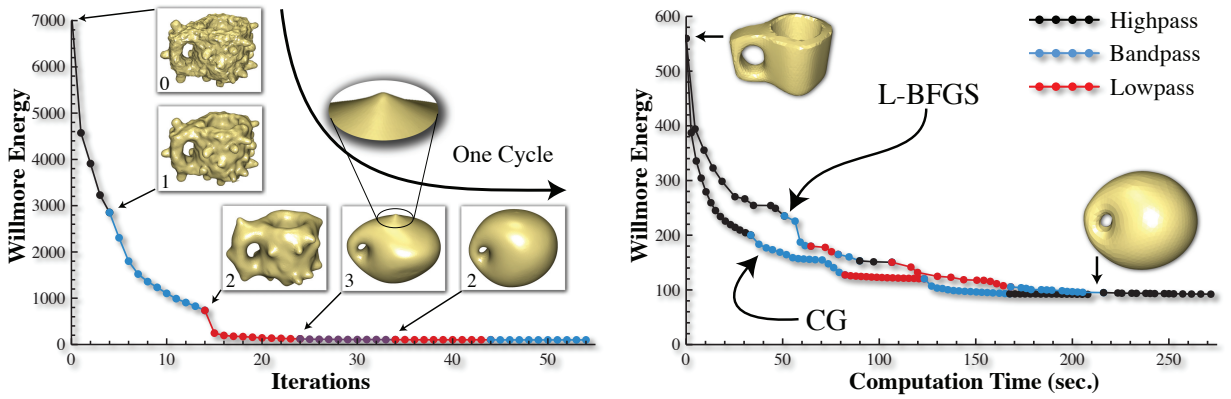

Figure 4: Left: One cycle ($k = 0, 1, 2, 3, 2, 1$), where k is the filter index, to minimize bending energy on the input mesh (top left). The index is increased when the magnitude of changes to the shape falls below a given threshold in order to minimize more global features and then decreased to remove small-scale features introduced by the lower pass filters. Right: Multiscale minimizers based on our implementation of the nonlinear conjugate gradient method and the L-BFGS method exhibit similar convergence rates. While L-BFGS requires fewer iterations, it requires more time per iteration.

4 Algorithm

The goal of shape optimization is to find a path from a given initial shape $\mathbf{x}_0 = \boldsymbol{\gamma}(0)$ to $\mathbf{x}_{\min} = \lim_{t\to\infty} \boldsymbol{\gamma}(t)$ such that $E(\mathbf{x}_{\min})$ is a minimum. Perhaps the simplest such algorithm is gradient descent, which chooses the velocity of the path to be aligned with the gradient of the energy, so that $\boldsymbol{\gamma}'(t) = -\mathbf{g}(\mathbf{x})$. A variant of this is the nonlinear conjugate gradient method, which uses information from previous search directions to guide the optimization towards a solution. The Newton-type methods, such as L-BFGS [21], compute a second-order approximation to the energy at each step to perform the optimization. Regardless of the particulars of each algorithm, most descent-based methods can be described as iteratively computing a search direction and performing a line search that approximately minimizes the energy along this direction. We apply our filtering to this descent direction.

Given a tuple $(\mathbf{A}, \mathbf{B})$, where $\mathbf{A}$ and $\mathbf{B}$ are defined as in the previous section, Algorithm 1 shows how we modify the nonlinear conjugate gradient method to minimize $E(\mathbf{x})$. Traditionally, $\mathbf{A} = \mathbf{B} = \mathbf{I}$ so that $\mathbf{g}_0$ and $\mathbf{g}_1$ correspond to the L^2 gradient. In this case, the algorithm very quickly minimizes the high frequencies features in the shape. However, lower frequency features are minimized only very slowly, and the algorithm tends to stall as the convergence rate slows down.

Let $\mathbb{F} = \{(\mathbf{A}_k, \mathbf{B}_k)\}_{k=1}^m$ be a set consisting of m tuples, with each element of $\mathbb{F}$ corresponding to a different choice for the coefficients $\{a_i\}_{i=0}^p$ and $\{b_i\}_{i=0}^q$ in (4). Each tuple in $\mathbb{F}$ also corresponds to a different transfer function $h_k(\lambda)$. We set $\mathbf{A}_0 = \mathbf{B}_0 = \mathbf{I}$. Our algorithm is designed such that each one of these transfer functions attenuates a different frequency range. For the transfer function in (6), Kim and Rossignac [14] provide a method for computing coefficients based on an intuitive set of parameters available to the user. In our case, we have found that the algorithm works well with only three filters ($m = 3$), shown in Figure 3, which can be viewed as a highpass, bandpass, and lowpass filter. The coefficients are given by $\{a_0 = 1, a_1 = 1, b_1 = 1\}$, $\{a_0 = 1, a_2 = 1, b_1 = 2\}$, and $\{a_0 = 1, a_2 = 1, b_0 = 1\}$ for the highpass, bandpass, and lowpass filters, with all other coefficients equal to zero. The form of the transfer function in (6) is general enough so that more sophisticated filters could be designed if more levels are required for the optimization.

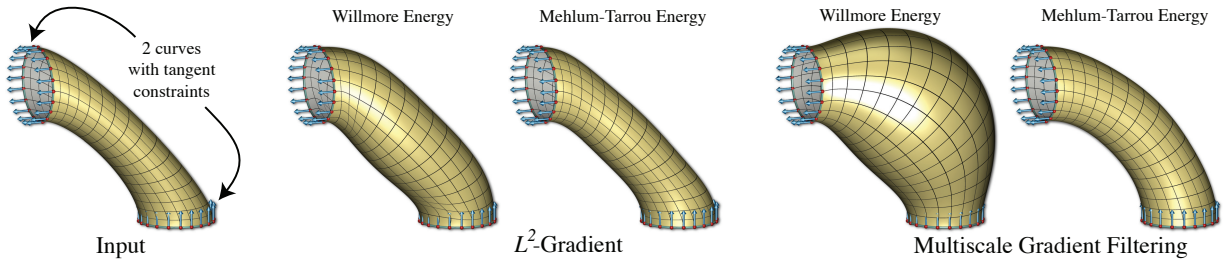

Figure 5: Starting from a C^2-continuous B-spline surface generated from two input curves (left), we minimize the second-order Willmore energy and the third-order Mehlum-Tarrou energy subject to positional and tangential constraints. A traditional descent-based algorithm using the L^2 gradient stalls after a few iterations (middle), whereas the Multiscale Gradient Filtering approach converges to a more optimal solution for both energies (right).

Algorithm 2 summarizes the procedure we use to iteratively optimize the shape at multiple scales. We start by optimizing the high frequency components with $k = 0$, and then optimize lower and lower frequency components by increasing k until the maximum is reached. While increasing k will optimize more global features, it may introduce higher frequency features during the minimization process (see inset in Figure 4–left). Thus, one cycle of Algorithm 2 involves first increasing k and then decreasing k as we move up and down the hierarchy of frequencies during optimization to ensure that the shape is optimized at all scales, similar to the multigrid V-cycle. The algorithm terminates when the magnitude of changes to the shape is below a given threshold or when the maximum number of cycles has been reached.

Figure 4–left shows the convergence plot for minimizing the Willmore energy of a genus-1 object. Interestingly, the minimum of the energy is not unique, and the resulting shape is not necessarily a symmetric torus. Note that as we increase k, even though the energy drops only very slowly, the shape still undergoes large, global deformations.

The call to Algorithm 1 in Algorithm 2 can be replaced with other nonlinear optimization algorithms such as gradient descent or L-BFGS. Figure 4–right shows a comparison between our implementation of the nonlinear conjugate gradient method, shown in Algorithm 1, and the L-BFGS algorithm, for which we use a library that can be found online [22]. While L-BFGS requires fewer steps to reach the minimum, each step requires more computation. The result for gradient descent looks very similar to these plots and has been omitted for clarity.

Our implementation of the nonlinear conjugate gradient method uses Brent's linesearch method [24], which requires an upper bound, ϵ, on the stepsize. If Brent's method uses a stepsize close to ϵ, we increase the stepsize to $2\,\epsilon$. Conversely, if the maximum number of iterations is reached, it is called again with an upper bound of $\epsilon/2$. A starting value of $\epsilon = 0.1$ is used for all examples in this paper. The sparse matrix implementation of the Boost library is used to store $\{\mathbf{A}_k, \mathbf{B}_k\}_{k=1}^{m}$. We use PARDISO [29] to solve the linear system in (4) that defines the preconditioned gradient. Note that we could use an iterative solver, such as the conjugate gradient method, instead of a direct solver, which would avoid having to compute and store higher-order Laplacians.

5 Applications

5.1 Variational Modeling

The variational modeling of surfaces is the process of constructing aesthetically pleasing smooth shapes by optimizing some curvature-based energy. The inputs are generally user-specified positional and tangential constraints.

A commonly used surface energy is the second-order bending energy that computes the area integral of the square of the mean curvature of the surface, $E = \int H^2 dA$, also known as the Willmore energy [35], where H is the mean curvature. Given appropriate boundary conditions, variants of this formulation (e.g., $\int ({\kappa_1}^2 + {\kappa_2}^2) dA$, where κ_i are the principal curvatures, and $\int H^2 dA + \int K dA$ where K is the Gaussian curvature) differ only by constants that depend on topological type and are equivalent in terms of optimization. In this paper, we use the dihedral-angle-based discrete operator [9] for approximating the Willmore energy for triangle meshes (Figure 1).

Minimizing the Willmore energy has the effect that during the minimization process, the surface gets more and more spherical (see Figures 1, 2, and 4). Often however, this behavior is undesirable when the user does not want the shape to bulge out (see Figure 5).

If the bulgy results using the Willmore energy are undesirable, a different energy has to be defined to prevent this behavior. Mehlum and Tarrou [17] proposed to minimize the squared variation in normal curvature integrated over all directions in the tangent plane:

$$\frac{1}{\pi} \int \left(\int_0^{\pi} \kappa_n'(\theta)^2 d\theta \right) dA \tag{7}$$

This is a third-order energy since it requires taking the derivative of the normal curvature, where the resulting expression contains third-order partial derivatives. Third-order energies tend to be even more ill-conditioned than second-order energies such as the Willmore energy, which can make their optimization prohibitively expensive using traditional nonlinear minimization algorithms. Other examples where higher-order energies are desired are discussed in [12].

Figure 5 shows a comparison between the Willmore energy and the Mehlum-Tarrou energy as defined in (7) using our approach. The input to the optimization is a C^2-continuous non-uniform bi-cubic B-spline that is fit through two B-spline curves with tangential constraints. The energy and gradient for this surface representation is straightforward to compute using the equations in [17]. In order to achieve the optimal configuration, the shape needs to move globally into a curved cylinder. The figure also shows a comparison to the state of the surfaces minimized without preconditioning the gradients, demonstrating that our method yields a more optimal shape than the traditional approach in the same computational time.

5.2 Parameterization

Computing high quality surface parameterizations for use in applications such as texture mapping is a well studied topic (see, e.g., [8]). The classic aim is to find a suitable embedding from $\mathbb{R}^3 \rightarrow \mathbb{R}^2$ that minimizes some form of measured distortion. Numerous proposals have been made for defining distortion metrics which produce pleasing results. Perhaps the most popular has been conformal mapping, which attempts to preserve angular distortion. Such conformal solutions are efficient to compute, requiring a linear solve, but

Algorithm 1 : MinimizeCG$((\mathbf{A}, \mathbf{B}), \mathbf{x})$

$\mathbf{g}_0 \leftarrow \mathbf{A}^{-1}\mathbf{B}\mathbf{M}^{-1}\frac{\partial E(\mathbf{x})}{\partial \mathbf{x}}$
$\mathbf{s}_0 \leftarrow -\mathbf{g}_0$
for $i = 1 \rightarrow$ max iterations **do**
  $\alpha \leftarrow LineSearch(\mathbf{x}, \mathbf{s}_0)$
  $\mathbf{x} \leftarrow \mathbf{x} + \alpha\,\mathbf{s}_0$
  $\mathbf{g}_1 \leftarrow \mathbf{A}^{-1}\mathbf{B}\mathbf{M}^{-1}\frac{\partial E(\mathbf{x})}{\partial \mathbf{x}}$
  $\beta \leftarrow \langle \mathbf{g}_1, \mathbf{g}_1 \rangle / \langle \mathbf{g}_0, \mathbf{g}_0 \rangle$
  $\mathbf{s}_0 \leftarrow \beta\,\mathbf{s}_0 - \mathbf{g}_1$
  $\mathbf{g}_0 \leftarrow \mathbf{g}_1$
end for
return $\mathbf{x}$

Algorithm 2 : MinimizeMultiScale$(\mathbf{x})$

for $j = 1 \rightarrow$ number of cycles **do**
  for $k = 0 \rightarrow m$ **do**
    $\mathbf{x} \leftarrow$ **MinimizeCG**$\big((\mathbf{A}_k, \mathbf{B}_k), \mathbf{x}\big)$
  end for
  for $k = m - 1 \rightarrow 0$ **do**
    $\mathbf{x} \leftarrow$ **MinimizeCG**$\big((\mathbf{A}_k, \mathbf{B}_k), \mathbf{x}\big)$
  end for
end for
return $\mathbf{x}$

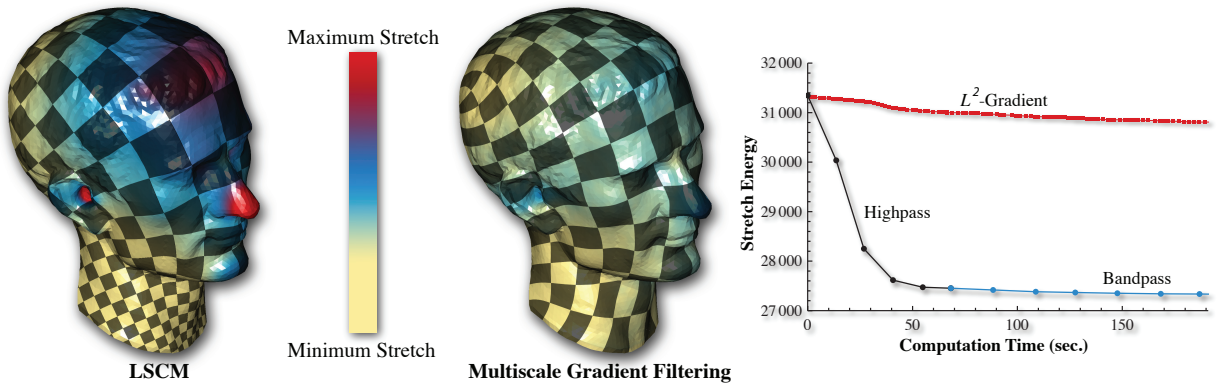

Figure 6: Starting from the LSCM parameterization, the Multiscale Gradient Filtering approach is used to minimize texture stretch. By varying the filter, a minimum is achieved much faster than by just minimizing the energy using the L^2 gradient.

unfortunately do a poor job of preserving area (see Figure 6 for a comparison).

When considering preserving both area and angle during flattening one must turn to nonlinear metrics. A family of such metrics have been proposed based on the singular values of the 3×2 Jacobian matrix J_i that maps each triangle T_i from 3D into the 2D parametric domain [28, 31, 36]. A perfect isometric mapping will have singular values equal to 1. Large and small singular values imply stretching and shrinking respectively.

For our experiments we use the L^2 *texture stretch* metric from [28]. Given a triangle T_i from mesh M, its root-mean-square stretch error can be defined as follows: $E(T_i) = \sqrt{(\tau + \gamma)/2}$, where τ and γ are the singular values of the triangle's Jacobian matrix J_i. The entire parametrization can be computed by minimizing the following expression over the mesh:

$$\sqrt{\sum_{T_i \in M} E(T_i)^2 A(T_i) / \sum_{T_i \in M} A(T_i)} \qquad (8)$$

where $A(T_i)$ is the surface area of the triangle in 3D. When a triangle's area in 2D approaches zero (e.g., becomes degenerate), its stretch error $E(T_i)$ goes to infinity. Thus, relatively small perturbations of the parameterization can result in large changes in expression (8). For this reason the nonlinear system is quite stiff, producing a set of gradients during optimization that may greatly vary in magnitude. The effect of this is a system that is challenging to solve efficiently. We applied our gradient preconditioning method to solve this problem. Figure 6 shows the results of our method for minimizing texture stretch. We start with an initial parametrization using a Least Squares Conformal Map [15], shown on the left. We then run our optimization procedure on the nonlinear energy to obtain the results shown in the middle, demonstrating that we obtain a solution with much less distortion. On the right, we compare the energy behavior to no preconditioning, demonstrating that we obtain the optimal shape with far less computation.

5.3 Nonlinear Elasticity

To this point, we have considered typical shape optimization problems, where the user inputs a shape that is then optimized with respect to a given energy using the algorithm discussed in Section 4. In these applications, the user is generally only interested in the end state and not in the path taken to get there. However, we demonstrate that our proposed framework can also be applied to the simulation of the dynamics of elastic objects, where the user is indeed interested in the path.

We begin with Hooke's law for continuum mechanics

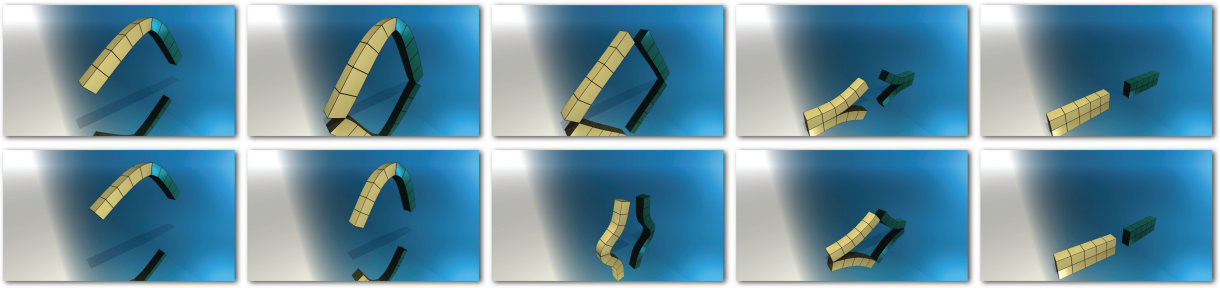

Figure 8: Top row: Animation sequence of an elongated bar undergoing deformations based on the infinitesimal strain tensor. The lack of nonlinear terms causes large distortions and ghost forces in the shape. Bottom row: By filtering high frequencies from the forces based on the finite strain tensor, a larger timestep and nonlinear deformation behavior can be achieved.

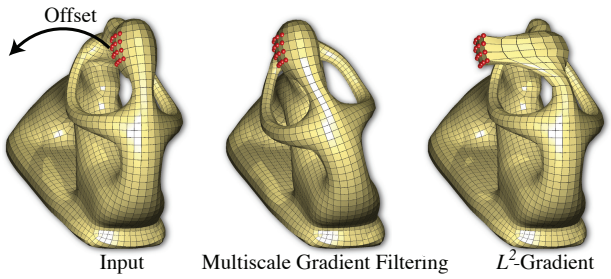

Figure 7: We minimize elastic potential energy based on nonlinear strain on an unstructured hexahedral mesh with a trilinear Bézier basis. Multiscale Gradient Filtering quickly propagates the boundary conditions through the shape, whereas a minimizer based on the L^2 gradient stalls.

(see, e.g., [16]), which states that the stress, $\boldsymbol{\sigma}$, for an isotropic elastic material is linearly related to the strain, $\boldsymbol{\epsilon}$, by

$$\boldsymbol{\sigma} = 2\mu\boldsymbol{\epsilon} + \lambda \mathrm{Tr}\left(\boldsymbol{\epsilon}\right)\mathbf{I}\ ,$$

where λ and μ are the Lamé first and second parameters that determine the elastic response of the material, $\mathrm{Tr}\left(\boldsymbol{\epsilon}\right) = \sum_k \boldsymbol{\epsilon}_{kk}$ is the trace of the strain tensor, and $\mathbf{I}$ is the identity. The finite strain tensor encodes the deformation of the material from its undeformed configuration at $\mathbf{x}_0$ to its deformed configuration $\mathbf{x}$ and is defined as $\boldsymbol{\epsilon} = \frac{1}{2}\left(\mathbf{F}^T\mathbf{F} - \mathbf{I}\right)$, where $\mathbf{F}$ is the deformation gradient, which maps tangent vectors from the undeformed to the deformed configuration. The elastic potential energy, a function of $\mathbf{x}$, is defined as

$$U\left(\mathbf{x}\right) = \frac{1}{2}\int \mathrm{Tr}\left(\boldsymbol{\sigma}\boldsymbol{\epsilon}\right)dA\ , \tag{9}$$

where the integration is over the undeformed configuration. Figure 7 shows an example where $U(\mathbf{x})$ is minimized on an hexahedral mesh using a trilinear Bézier basis. Similar to the previous cases, the preconditioned gradients help to propagate the deformations to the rest of the mesh much faster.

The equations of motion for the elastic material with Rayleigh damping are

$$\mathbf{M}\,\ddot{\mathbf{x}} + \mathbf{C}\,\dot{\mathbf{x}} - \mathbf{f}_{\mathrm{int}}\left(\mathbf{x}\right) = \mathbf{f}_{\mathrm{ext}}\left(\mathbf{x}, \dot{\mathbf{x}}\right), \tag{10}$$

where $\dot{\mathbf{x}}$ and $\ddot{\mathbf{x}}$ are the first and second derivative of $\mathbf{x}$ with respect to time, $\mathbf{M}$ is the physical mass matrix, $\mathbf{C}$ is the Rayleigh damping matrix, $\mathbf{f}_{\mathrm{ext}}$ is a vector of external forces such as gravity, and $\mathbf{f}_{\mathrm{int}}\left(\mathbf{x}\right) = -\,\partial U(\mathbf{x})/\partial\mathbf{x}$ is a vector of internal forces that depends on the elastic potential energy. The equations of motion define a coupled system of ordinary differential equations that must be integrated in time to solve for the path of the object. For stiff materials, implicit integrators are often used because they offer superior stability over explicit methods [2], resulting in a costly system of nonlinear equations that must be solved at each timestep. Furthermore, since the equations are nonlinear, even implicit integrators are

not unconditionally stable, resulting in a restriction on the maximum allowable timestep size [11].

Various strategies exist for reducing the amount of computation needed to integrate the equations of motion. Since many solid materials, such as metal and wood, usually undergo small deformations, the infinitesimal strain tensor, given by $\boldsymbol{\epsilon} = \frac{1}{2}\left(\mathbf{F}^T + \mathbf{F}\right) - \mathbf{I}$, can be used instead of the finite strain tensor in the definition of the elastic potential energy. In the case of large deformation, the naïve use of the infinitesimal strain tensor results in undesirable distortions and ghost forces, so corotational approaches [18, 19], which factor out rotations from the motion on a per-element or per-vertex basis, are often employed. Both for the small displacement case and the corotational approaches, replacing the finite strain tensor with the infinitesimal strain tensor and using an implicit integrator results in having to solve only a linear rather than a nonlinear system per timestep, saving a significant amount of computation. In the case of small deformations, the resulting system is also unconditionally stable, which is very attractive since it allows large timesteps to be used.

We propose an alternative for allowing one to take larger timesteps while integrating the equations of motion based on our framework. We modify Equation 10 to filter the internal forces, so the modified equations are given by

$$\mathbf{M}\,\ddot{\mathbf{x}} + \mathbf{C}\,\dot{\mathbf{x}} - \mathcal{A}^{-1}\mathbf{f}_{\text{int}}(\mathbf{x}) = \mathbf{f}_{\text{ext}}(\mathbf{x}, \dot{\mathbf{x}}) \ , \qquad (11)$$

where $\mathcal{A}^{-1}$ is designed to remove the high frequency components from the internal forces. By filtering the forces, the physics of the system is changed; however, for applications where the timestep size must remain fixed, we can choose $\mathcal{A}$ to filter those frequencies that cannot be resolved, allowing a larger timestep. As shown in Figure 8, this approach does not exhibit any of the distortions apparent when using the small deformation assumption and, like the corotational approach, allows for large timesteps, even if an explicit integrator is used. We believe that this approach can also be used in conjunction with a corotational method, allowing larger timesteps within that context and for implicit integrators in general. In addition, this approach is not limited to the linear constitutive equation given by Hooke's law but can be applied to filter out undesirable frequencies from any forces.

Example	Energy	Rep.	# Elems.	Basis	Time
Fig. 1	W.	T.	153K	$C^{(0)}$	4h
Fig. 2	W.	T.	720	$C^{(0)}$	1m
Fig. 4-left	W.	T.	26K	$C^{(0)}$	6m
Fig. 4-right	W.	T.	6K	$C^{(0)}$	3m
Fig. 5	W.	B.S.	320	$C^{(2)}$	5m
Fig. 5	M.T.	B.S.	320	$C^{(2)}$	30m
Fig. 6	Stretch	T.	16K	$C^{(0)}$	1m
Fig. 7	E.P.	B.V.	3.5K	$C^{(0)}$	1h
Bunny (*)	W.	T.	10K	$C^{(0)}$	3m
Buddha (*)	W.	T.	30K	$C^{(0)}$	6m
Teardrop (*)	W.	T.	3.8K	$C^{(0)}$	5m
Pin (*)	W.	T.	3.3K	$C^{(0)}$	5m

Table 1: Performance of the various optimizations of Willmore (W.), Mehlum-Tarrou (M.T.) and Elastic Potential (E.P.) energy, peformend on triangles (T.), B-spline surfaces (B.S.) and B-spline volumes (B.V.). Timings were taken on a quadcore machine. Datasets denoted with $^{(*)}$ are shown in the supplementary video. In each case, optimization is stopped when shape reaches an acceptable state.

6 Conclusions and Discussion

In this paper, we provide a general framework for nonlinear shape optimization. We show applications of our method for optimizing ill-conditioned, nonlinear energies in a variety of different contexts, in each case demonstrating an improvement in the quality of the results and a reduction in the time required to compute a solution (see Table 1). We also provide an extension of our method to the case of simulating the dynamics of a physical system, which enables the use of larger timesteps during the course of the simulation. We believe that for many cases in which practitioners resort to approximations, sacrificing quality for speed, our framework can be used to obtain superior results at comparable runtimes.

Our method is conceptually similar to a multigrid method in that it can be used to smooth errors at multiple scales. However, there is an important limi-

tation of our method: a coarser level in a multigrid approach usually requires less computation, whereas in our framework, smoothing errors at larger scales increases the requisite computation because of the higher-order Laplacians involved. On the other hand, the implementation of our method does not require a hierarchy of representations during the optimization process and can therefore be easily integrated into existing optimization systems. Therefore, we believe that this framework provides a valuable benchmark that can be used to evaluate the practicality of nonlinear formulations for many problems.

References

[1] Aksoylu, B., Khodakovsky, A., and Schröder, P. Multilevel Solvers for Unstructured Surface Meshes. *SIAM J. Sci. Comput. 26* (April 2005), 1146–1165.

[2] Baraff, D., and Witkin, A. P. Large Steps in Cloth Simulation. In *Proceedings of SIGGRAPH 98* (July 1998), Computer Graphics Proceedings, Annual Conference Series, pp. 43–54.

[3] Botsch, M., and Kobbelt, L. An intuitive framework for real-time freeform modeling. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 630–634.

[4] Briggs, W. L., Henson, V. E., and McCormick, S. F. *A Multigrid Tutorial*, 2nd ed. SIAM, 2000.

[5] Charpiat, G., Keriven, R., Pons, J.-P., and Faugeras, O. Designing Spatially Coherent Minimizing Flows for Variational Problems Based on Active Contours. In *ICCV* (2005), IEEE Computer Society, pp. 1403–1408.

[6] Do Carmo, M. P. *Riemannian Geometry*. Birkhäuser Boston, 1992.

[7] Eckstein, I., Pons, J.-P., Tong, Y., Kuo, C.-C. J., and Desbrun, M. Generalized surface flows for mesh processing. In *SGP* (2007), Eurographics Association, pp. 183–192.

[8] Floater, M. S., and Hormann, K. Surface Parameterization: a Tutorial and Survey. In *Advances in Multiresolution for Geometric Modelling*, Mathematics and Visualization. Springer Berlin Heidelberg, 2005, pp. 157–186.

[9] Grinspun, E., Hirani, A. N., Desbrun, M., and Schröder, P. Discrete shells. In *SCA* (2003), Eurographics Association, pp. 62–67.

[10] Grinspun, E., Krysl, P., and Schröder, P. CHARMS: A Simple Framework for Adaptive Simulation. *ACM Transactions on Graphics 21*, 3 (July 2002), 281–290.

[11] Hauth, M., Etzmuss, O., and Strasser, W. Analysis of numerical methods for the simulation of deformable models. *The Visual Computer 19*, 7-8 (2003), 581–600.

[12] Joshi, P. *Minimizing Curvature Variation for Aesthetic Surface Design*. PhD thesis, EECS Department, University of California, Berkeley, Oct 2008.

[13] Karátson, J., and Faragó, I. Preconditioning operators and Sobolev gradients for nonlinear elliptic problems. *Comput. Math. Appl. 50* (October 2005), 1077–1092.

[14] Kim, B., and Rossignac, J. GeoFilter: Geometric Selection of Mesh Filter Parameters. *Computer Graphics Forum 24*, 3 (Sept. 2005), 295–302.

[15] Lévy, B., Petitjean, S., Ray, N., and Maillot, J. Least Squares Conformal Maps for Automatic Texture Atlas Generation. *ACM Transactions on Graphics 21*, 3 (July 2002), 362–371.

[16] Marsden, J. E., and Hughes, T. J. R. *Mathematical Foundations of Elasticity*. Dover Publications, 1994.

[17] Mehlum, E., and Tarrou, C. Invariant smoothness measures for surfaces. *Advances in Computational Mathematics 8*, 1-2 (1998), 49–63.

[18] Müller, M., Dorsey, J., McMillan, L., Jagnow, R., and Cutler, B. Stable Real-Time Deformations. In *SCA* (2002), pp. 49–54.

[19] Müller, M., and Gross, M. H. Interactive Virtual Materials. In *Graphics Interface 2004* (May 2004), pp. 239–246.

[20] Nealen, A., Igarashi, T., Sorkine, O., and Alexa, M. FiberMesh: Designing Freeform Surfaces with 3D Curves. *ACM Transactions on Graphics 26*, 3 (July 2007), 41:1–41:9.

[21] Nocedal, J. Updating Quasi-Newton Matrices with Limited Storage. *Mathematics of Computation 35*, 151 (1980), 773–782.

[22] Okazaki, N. libLBFGS: a library of Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS), 2010. http://www.chokkan.org/software/liblbfgs/.

[23] Pinkall, U., and Polthier, K. Computing Discrete Minimal Surfaces and Their Conjugates. *Experimental Mathematics 2*, 1 (1993), 15–36.

[24] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. Cambridge University Press, 1992.

[25] Renka, R. J. Constructing fair curves and surfaces with a Sobolev gradient method. *Computer Aided Geometric Design 21*, 2 (2004), 137 – 149.

[26] Renka, R. J., and Neuberger, J. W. Minimal surfaces and Sobolev gradients. *SIAM J. Sci. Comput. 16* (November 1995), 1412–1427.

[27] Rivara, M.-C. Local modification of meshes for adaptive and/or multigrid finite-element methods. *Journal of Computational and Applied Mathematics 36*, 1 (1991), 79–89. Special Issue on Adaptive Methods.

[28] Sander, P. V., Snyder, J., Gortler, S. J., and Hoppe, H. Texture Mapping Progressive Meshes. In *Proceedings of ACM SIGGRAPH 2001* (Aug. 2001), Computer Graphics Proceedings, Annual Conference Series, pp. 409–416.

[29] Schenk, O., and Gärtner, K. Solving unsymmetric sparse systems of linear equations with PARDISO. *Future Gener. Comput. Syst. 20* (April 2004), 475–487.

[30] Shi, L., Yu, Y., Bell, N., and Feng, W.-W. A fast multigrid algorithm for mesh deformation. *ACM Transactions on Graphics 25*, 3 (July 2006), 1108–1117.

[31] Sorkine, O., Cohen-Or, D., Goldenthal, R., and Lischinski, D. Bounded-distortion piecewise mesh parameterization. In *VIS* (2002), IEEE Computer Society, pp. 355–362.

[32] Sundaramoorthi, G., Yezzi, A., and Mennucci, A. C. Sobolev active contours. *International Journal of Computer Vision 73* (2005), 109–120.

[33] Taubin, G. A Signal Processing Approach to Fair Surface Design. In *Proceedings of SIGGRAPH 95* (Aug. 1995), Computer Graphics Proceedings, Annual Conference Series, pp. 351–358.

[34] Terzopoulos, D., Platt, J., Barr, A., and Fleischer, K. Elastically Deformable Models. In *Computer Graphics (Proceedings of SIGGRAPH 87)* (July 1987), pp. 205–214.

[35] Willmore, T. J. Mean Curvature of Riemannian Immersions. *Journal of the London Mathematical Society s2-3*, 2 (1971), 307–310.

[36] Zhang, E., Mischaikow, K., and Turk, G. Feature-based surface parameterization and texture mapping. *ACM Transactions on Graphics 24*, 1 (Jan. 2005), 1–27.