Eurographics Symposium on Geometry Processing 2009
Marc Alexa and Michael Kazhdan
(Guest Editors)

Volume 28 (2009), Number 5

Estimating the Laplace-Beltrami Operator by Restricting 3D Functions

Ming Chuang[1], Linjie Luo[2], Benedict J. Brown[3†], Szymon Rusinkiewicz[2], and Michael Kazhdan[1]

[1]Johns Hopkins University
Baltimore MD, USA

[2]Princeton University
Princeton NJ, USA

[3]Katholieke Universiteit Leuven
Leuven, Belgium

Abstract

We present a novel approach for computing and solving the Poisson equation over the surface of a mesh. As in previous approaches, we define the Laplace-Beltrami operator by considering the derivatives of functions defined on the mesh. However, in this work, we explore a choice of functions that is decoupled from the tessellation. Specifically, we use basis functions (second-order tensor-product B-splines) defined over 3D space, and then restrict them to the surface. We show that in addition to being invariant to mesh topology, this definition of the Laplace-Beltrami operator allows a natural multiresolution structure on the function space that is independent of the mesh structure, enabling the use of a simple multigrid implementation for solving the Poisson equation.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.5]: Boundary Representations—

1. Introduction

Solving the Poisson equation is a fundamental step in numerous image and mesh processing applications. It facilitates the modeling process by fitting a *global* solution to a set of *local* constraints. Specifically, when the system is constrained by prescribing gradients, solving the Poisson equation provides a means for integrating the constraints — returning the function whose gradients best match the desired differences.

An example application is shown in Figure 1. Here, we back-project color onto a model that was reconstructed from 3D scans containing both depth and color information. Due to lighting variation across the scans, setting the color at a vertex to the color of the nearest scan point results in discontinuities at scan transitions (left). Instead, by pulling color *gradients* from the closest scans, we obtain a vector field describing the local change in the texture over the mesh. Solving the Poisson equation for the color field that best fits these gradients produces a seamlessly textured surface (right).

In the context of regular grids, solving the Poisson equation is a straight-forward task. The regularity of the grid provides a domain that is amenable to both Fourier methods and multigrid solvers. For meshes, however, solving the Poisson equation is more challenging: On a mesh, function values are traditionally associated with vertex positions, so the definition of the linear system depends not only on the surface geometry but also on the tessellation. And, the domain of the mesh is not regularly sampled so Fourier and multigrid techniques cannot be used to solve the problem efficiently, necessitating more general-purpose, sparse-matrix solvers.

Figure 1: *Reconstructing colored surfaces from 3D scans: The texture obtained by pulling color* values *from the closest scans is shown on the left, while taking color* gradients *from the closest scans and solving the Poisson equation gives the seamless result on the right.*

† Post-Doctoral Fellow of the Research Foundation - Flanders

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.
Published by Blackwell Publishing, 9600 Garsington Road, Oxford OX4 2DQ, UK and 350 Main Street, Malden, MA 02148, USA.

To address these concerns, we consider a novel finite-elements approach that decouples the definition of the Poisson system from the surface tessellation. The key behind our approach is to define a space of 3D functions (independent of the mesh) and then restrict the functions to the surface. As with previous finite-element approaches, matrix coefficients are defined by integrating elements over the triangles of the mesh, resulting in a system that adapts to the non-uniformity of the tessellation. However, using restrictions of 3D functions to define the elements has several advantages:

Tessellation Independence: Because the initial space of 3D functions is chosen independent of the mesh, and because the system coefficients are defined by integrating over the surface, our method defines a Poisson system that is agnostic to the tessellation. This results in linear systems whose eigenvalues can be more robustly estimated without requiring excessive mesh refinement.

Multiresolution Hierarchy: Because restriction of functions to a sub-domain is a linear operation, nesting 3D functions spaces will remain nested after they are restricted to the surface. As a result function hierarchies used to define 3D multigrid solvers can easily be extended to define multigrid solvers over the mesh, providing simple and efficient methods for solving the Poisson equation.

Regularity: When the initial space of 3D functions is defined over a regular grid, the surface elements inherit the regularity, providing an opportunity for deriving fast implementations of a solver. Though not explicitly discussed in this work, such regularity can be leveraged for parallelization (since red-black-type decompositions in 3D will still satisfy independence after restriction) and for streaming the solver (since a stream order derived for the 3D functions carries over to their restrictions).

We begin our discussion with a brief survey of related work in using the Poisson equation for image and mesh processing (Section 2) and a review of the general finite-elements approach (Section 3). Next, we present our framework for defining and solving the Poisson equation using the restriction of 3D functions to the mesh surface (Section 4). We analyze the utility of our method and present results for several mesh processing applications, including spectral analysis, texture back-projection, and function fitting (Section 5). And we conclude by summarizing our work and discussing directions for future research (Section 6).

2. Related Work

The Poisson equation arises in numerous mesh processing and shape analysis applications. This section briefly reviews some of the more common applications as well as methods used to solve the underlying system of equations.

Processing Images Many image processing techniques operate in the gradient domain. They extract gradient fields from one or more images, process the data to construct a desired gradient field, and solve the Poisson equation for the image whose gradients best fit the constraint field. While these techniques were developed for images, the fundamental challenge in extending them to mesh processing is solving the Poisson equation.

Example applications in image processing have included removing shadow and lighting by zeroing appropriate gradients [Hor74, FHD02] or by selecting the median of gradients from multiple exposures [Wei01]; tone-mapping high dynamic range (HDR) images by adaptively attenuating luminance gradients [FLW02]; seamlessly stitching overlapping images by merging their gradients [PGB03, ADA*04, LZPW04]; and improving photographic tone management using gradient constraints [BPD06].

Editing Meshes Recently, the Poisson equation has also become a key component of mesh editing systems. Using the translation invariance of differential vertex encoding, Alexa [Ale03] proposed a method to transfer detail between models by blending in the differential coordinates and solving the Poisson equation to get back absolute (Euclidean) coordinates. The method was later extended by Sorkine et al. [SCOL*04] and Lipman et al. [LSCO*04] to be invariant to rigid-body transformations by encoding vertex positions relative to a local frame. Using gradient fields to model coordinate functions, mesh editing has also been performed by locally adapting the gradients and solving the Poisson equation for the new coordinate functions [YZX*04].

Defining Shape Invariants The Laplace-Beltrami operator's invariance to isometric deformations has motivated its use in both deformation-invariant shape matching and intrinsic symmetry detection. Using the invariance of its spectrum, Reuter et al. [RWP05] obtain a compact shape descriptor that is fixed under rigid-body transformations. Incorporating the invariance of its eigenvectors as well leads to a deformation-invariant shape representation [Rus07] — canonically embedding a shape in a high-dimensional space by evaluating the eigenfunctions at the mesh vertices. This embedding was later leveraged to detect a shape's intrinsic symmetries [OSG08] using the fact that intrinsic symmetries become Euclidean symmetries in the embedding space.

Solving the Linear System with Multigrid Multigrid methods are well known for their efficiency and scalability in solving large linear systems [Wes04]. They were designed to overcome the limitations of traditional iterative solvers (e.g., those based on Gauss-Seidel), which tend to reduce high-frequency errors more quickly than low-frequency ones and thus exhibit slow convergence for typical (all-frequency) problems. Multigrid methods extend the reduction to all frequencies by computing corrections on grids successively coarsened from the original system [Cha01].

To apply multigrid methods to unstructured meshes,

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.

mesh simplification techniques are usually employed successively to generate different grid levels [KCVS98, RL03, AKS05, NGH04, SYBwF06]. However, as observed in Ni et al. [NGH04], these methods can be sensitive to the initial triangulation of the surface, and a low-quality input mesh may result in lower performance.

In contrast, algebraic multigrid (AMG) methods are "black-box" solvers that rely solely on the algebraic information of the linear system, independent of any geometric information [RS87]. Although AMG proves to be robust and scalable on a large class of problems [CFH*00], especially those resulting from elliptical partial differential equations, its notion of algebraic smoothness is limited in practice by the requirement that the coefficient matrix be an M-matrix [Cha01]. AMG's successors, element-based AMG [BCF*00] and spectral AMGe [CFH*03], extend the concept of algebraic smoothness and broaden its applicability by assuming that the discretization is based on Ritz-type finite-element methods for partial differential equations. Nevertheless, this class of methods remains difficult to apply for certain problems, such as the function-fitting problem considered in Section 5.3.

Leveraging Regularity The facility derived from working over a regular domain has motivated the use of 3D grids for encoding the constraints and linear systems defined on irregular surfaces. In the context of fluid dynamics (e.g. [May84, Pes02, MI05]) the constraint that a fluid not penetrate the boundary of a solid is encoded by modifying a linear system, defined over a regular grid, to include *forcing functions* that reproduce the effect of the boundary. In the context of potential theory [TW03], Tausch et al. show that the orthogonality of piecewise-constant elements can be used to define a multiscale basis derived from a hierarchical decomposition of a 3D bounding-cube.

Similar to Tausch et al., our approach leverages the regularity of a 3D grid to define basis elements for solving a linear systems. However, in this work, we show how the finite-elements setting can be formulated even in the case of higher-order (non-orthogonal) elements, allowing us to extend the approach to the discretization of surface PDEs.

3. A Brief Review of Finite Elements

In this section, we review the finite-elements approach. We show how a choice of elements can be used to define the Laplace-Beltrami operator and how a nesting hierarchy of elements can be used to guide a multigrid approach for solving the Poisson system.

3.1. Defining the Poisson Equation

Given a manifold $\mathscr{M}$ and a function $f:\mathscr{M}\rightarrow\mathbb{R}$, solving the Poisson equation amounts to finding the function $u:\mathscr{M}\rightarrow\mathbb{R}$ whose Laplacian is equal to f:

$$\Delta_{\mathscr{M}} u = f, \tag{1}$$

where $\Delta_{\mathscr{M}}$ is the Laplace-Beltrami operator (the generalization of the Laplacian to the manifold $\mathscr{M}$).

Since the space of functions on $\mathscr{M}$ is infinite-dimensional, the problem is made tractable by constraining the functions f and u to reside within a finite-dimensional subspace $\mathscr{F}$. Additionally, since the Laplacian does not necessarily map $\mathscr{F}$ back into itself, the formulation of the Poisson equation is adapted by replacing the condition "the Laplacian of u equals f" with the condition "*the projection onto* $\mathscr{F}$ *of* the Laplacian of u equals *the projection onto* $\mathscr{F}$ *of* f".

Formally, the condition that the projections be equal requires that the inner product of the Laplacian of u with b equal the inner product of f with b, for any test function $b(p)\in\mathscr{F}$. However, when $\mathscr{F}$ is the span of elements $\{b_1(p),\ldots,b_n(p)\}$, a sufficient condition is that the inner products with the b_i are equal:

$$\int_{\mathscr{M}} \Delta_{\mathscr{M}} u(p)\cdot b_i(p)dp = \int_{\mathscr{M}} f(p)\cdot b_i(p)dp. \tag{2}$$

In this finite-elements setting, solving the Poisson equation amounts to finding the linear combination of elements, $u(p)=\sum \eta_i b_i(p)$, which satisfy Equation 2, and the Poisson equation reduces to the $n\times n$ system:

$$L^{\mathscr{M}}\eta = \phi \tag{3}$$

where $\eta=[\eta_1,\ldots,\eta_n]^T$ are the coefficients of u, and the matrix $L^{\mathscr{M}}$ and vector $\phi=[\phi_1,\ldots\phi_n]^T$ are defined in terms of the dot-products:

$$\begin{aligned} L_{ij}^{\mathscr{M}} &= \int_{\mathscr{M}} \Delta_{\mathscr{M}} b_i(p)\cdot b_j(p)dp \\ \phi_i &= \int_{\mathscr{M}} f(p)\cdot b_i(p)dp. \end{aligned} \tag{4}$$

Evaluating the Matrix Coefficients One challenge to applying Equation 4 in practice is that evaluating the surface Laplacian requires the estimation of mean curvature, a differential property that is not well defined for meshes. When the surface is water-tight, Stokes's Theorem is used to turn a second derivative into two first derivatives giving:

$$L_{ij}^{\mathscr{M}} = -\int_{\mathscr{M}} \left\langle \nabla_{\mathscr{M}} b_i(p), \nabla_{\mathscr{M}} b_j(p)\right\rangle dp. \tag{5}$$

This *weak formulation* only requires first-order derivatives to compute the surface Laplacian and hence can be evaluated without explicitly estimating curvatures.

3.2. The Multigrid Method

The multigrid method is a common technique for solving the Poisson equation, replacing the global system of equations with a multiresolution hierarchy of systems that only require local refinement. It proceeds in two phases:
Restriction: Proceeding from the highest resolution to the lowest, the solution is relaxed and the restricted residual is used as a constraint for the lower resolution problem.

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.

Prolongation: Returning in the opposite direction, the solution is prolonged into each higher resolution, contributing a correction term to the previously estimated solution.

Implementing such a solver in the context of finite-elements requires a nested hierarchy of function spaces, $\mathscr{F}_1 \subset \cdots \subset \mathscr{F}_d = \mathscr{F}$, where each space $\mathscr{F}_l$ is spanned by n_l elements and an $n_{l+1} \times n_l$ prolongation matrix describes how elements at level l are expressed as linear combinations of elements at level $l+1$. Using this structure, the prolongation matrix describes the injection of the solution from the coarser space into the finer, and the transpose of the prolongation operator defines the dual operator that restricts constraints from the finer level into the coarser one.

4. The Restricted 3D Laplace-Beltrami Operator

To implement a multigrid solver, we must choose a space of functions over which to define the linear system and create a nesting hierarchy supporting multigrid.

4.1. Approach

In traditional mesh-processing applications, the elements are defined as tent functions over the mesh vertices. These functions are piecewise linear and are supported within the one-ring of the vertex. Using these in the weak formulation of Equation 5, one obtains the well-known cotangent-weight Laplace-Beltrami operator. Though the elements adapt to the sizes of the triangles, the linear system remains tied to the topology of the mesh representation and does not directly support a multigrid structure.

In this work we propose an alternate approach in which 3D functions are chosen independent of the mesh, then restricted to the surface to define the elements of the system. The advantages of this approach are two-fold. First, the resulting definition of a Laplace-Beltrami operator is agnostic to the surface tessellation and only depends on the geometry. Second, since nesting function spaces remain nested after restriction to a domain, an initial choice of nesting 3D spaces is guaranteed to result in a nesting hierarchy of restricted function spaces that support a multigrid solver.

There are several practical issues to implementing our method. Of course, we must choose appropriate nesting sets of 3D functions. In addition, we must compute the integrals defining the matrix and constraint coefficients of the system, and we need a simple way to index the function spaces to support efficient restriction and prolongation operations.

4.2. Choosing the 3D Elements

To define the 3D function space, we use the span of trivariate, tensor-product B-splines, centered on a 3D grid of resolution $2^d \times 2^d \times 2^d$ [CS97]. In addition to having local support, resulting in a sparse linear system, this choice of 3D functions provides a nested set of function spaces under grid subdivision, allowing for a multigrid solver.

Since the support of B-splines grows with degree, it is natural to consider lower order B-splines. Here, we use second order B-splines. As a result, each B-spline is supported within its voxel's immediate neighbors, and we obtain a prolongation operator expressing the l-th level B-spline as the combination of $4 \times 4 \times 4$ B-splines at level $(l+1)$, where the prolongation stencil is defined as the tensor-product of the 1D stencils, $\frac{1}{4}(1\ 3\ 3\ 1)$.

4.3. Computing the Integrals

Defining the coefficients of the Poisson system in Equation 4 requires computing integrals of products of the restricted B-splines and their derivatives. To compute the coefficients, we can either integrate over the triangle mesh or perform Monte Carlo integration over a uniformly distributed set of samples.

Integrating over the triangles To compute the integrals, we observe that the restriction of a second-order B-spline to the interior of a voxel is polynomial. By splitting the triangles of the mesh so that each is contained within a single voxel, we can reduce the problem to integrating polynomials over the subdivided triangles. This remains true for the surface gradients as they can be obtained by projecting the 3D gradients onto the tangent space:

$$\nabla_{\mathscr{M}} b(p) = \nabla b(p) - \langle \nabla b(p), \vec{N}_{\mathscr{M}}(p) \rangle \vec{N}_{\mathscr{M}}(p) \rangle,$$

where $\vec{N}_{\mathscr{M}}(p)$ is the normal at p. Since the 3D gradient, $\nabla b(p)$, is polynomial and since the normal is constant in the triangle, the surface gradient, $\nabla_{\mathscr{M}} b(p)$, is also polynomial.

Using trivariate B-splines of degree two, the restriction of elements to a triangle are sixth-degree polynomials. Since the system coefficients involve products of elements, the integrands will have degree at most 12. Thus, we can use Taylor's 32-point cubature formula [Tay08] to compute the coefficients of the system efficiently.

Summing over point samples Given a uniformly distributed set of oriented point samples (p_i, n_i) over the mesh $\mathscr{M}$, (i.e. $p_i \in \mathscr{M}$ and $n_i = \vec{N}_{\mathscr{M}}(p_i)$) we can also approximate the integrals with a finite sum, giving:

$$L_{ij}^{\mathscr{M}} \approx \frac{|\mathscr{M}|}{N} \sum_{k=1}^{N} -\left\langle \nabla b_i(p_k) - \langle \nabla b_i(p_k), n_k \rangle n_k, \nabla b_j(p_k) \right\rangle$$

$$\phi_j \approx \frac{|\mathscr{M}|}{N} \sum_{k=1}^{N} f(p_k) \cdot b_j(p_k)$$

Though less accurate than integrating over the triangles, the Monte Carlo approach can be used even when the restrictions of the constraint functions to the triangles cannot be expressed as a low-degree polynomial.

4.4. Indexing the Functions

In the regular 3D multigrid setting, the second-order elements at depth l can be indexed by a $2^l \times 2^l \times 2^l$ voxel grid,

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.

and the multiresolution set of elements can be indexed by a complete octree of depth d.

Although we could set up our Poisson equation using all of the 3D elements, only elements that overlap the model surface contribute to the system, since integrals computed against non-overlapping elements (or their derivatives) are guaranteed to be zero. Thus, to index our system, it suffices to construct a sparse octree $\mathscr{O}^{\mathscr{M}}$ where any node whose immediate neighbors do not overlap the shape $\mathscr{M}$ is pruned from the tree. We then index the elements by the tree nodes, setting $b_o(p)$ to be the element centered at $o \in \mathscr{O}^{\mathscr{M}}$ and scaled by the width of o, setting η_o and ϕ_o to be the coefficients associated with element b_o, and setting $L^{\mathscr{M}}_{o,o'}$ to be the coefficients obtained by integrating:

$$L^{\mathscr{M}}_{o,o'} = -\int_{\mathscr{M}} \langle \nabla_{\mathscr{M}} b_o(p), \nabla_{\mathscr{M}} b_{o'}(p) \rangle \, dp.$$

Restriction and Prolongation Since the nesting of spaces is independent of the surface, computing the restriction and prolongation of an element associated with a node at level l only requires finding the appropriate neighborhood of nodes in levels $l-1$ and $l+1$ and updating their associated function coefficients. For restriction, this requires updating the coefficient associated with the parent of the node and (some of) the coefficients stored in the parent's immediate neighbors. For prolongation, this requires updating the coefficients of the eight children and their immediate neighbors.

Computing the System Coefficients We can also use the octree to efficiently compute the coefficients of the Poisson system. Because elements are supported within the immediate neighbors of their associated nodes we can compute the system matrix and Poisson coefficients at level l by iterating over the triangles/point-samples, finding the node they reside in, and updating the coefficients for all pairs of neighbors (Figure 2). Thus, computing the coefficients for the Poisson equation can be reduced to locating triangles/point-samples within the tree and identifying neighbors, giving rise to an overall set-up time of $O(N \cdot d)$ where N is the number of triangles/samples and d is the depth of the tree.

5. Results

To evaluate our approach, we consider three separate applications. The first focuses on the quality and robustness of the computed Laplace-Beltrami matrix by comparing the spectra of our octree-based system with that of the traditional, cotangent-weights formulation. The other applications, texture back-projection and curvature estimation, demonstrate the need for an effective Poisson solver in geometry processing and compare the performance of our multigrid solver with state-of-the-art algebraic multigrid methods. We conclude with a brief discussion of limitations.

In all our experiments, the dimension of the system defined using our approach is equal to the number of finest-depth nodes in the octree, while the dimension of the cotangent-weight system is equal to the number of vertices. For the analysis of the spectrum, we use an octree of depth five while for the other applications in this section we use an octree of depth eight. For the evaluations of the solver, the images all show the result of a single W-cycle, giving the dimension of the system, the time for defining the lower-resolution systems and running the solver, and the RMS error (when the ground-truth solution is known). Additionally, with the exception of the normal-fitting experiment, all coefficients are computed by using cubature.

LaplaceBeltramiMatrix(*Octree* $\mathscr{O}^{\mathscr{M}}$, *Depth* l, *Mesh* $\mathscr{M}$)

```
Matrix L^M ← 0
for all triangles T ∈ M
    node o = NodeContaining( T , l )
    for all pairs of nodes (o', o'') ∈ Neighbors( o )
        L^M_{o,o'} += ∫_T ⟨∇_M b_{o'}(p), ∇_M b_{o''}(p)⟩ dp
return L^M
```

PoissonConstraints(*Octree* $\mathscr{O}^{\mathscr{M}}$, *Mesh* $\mathscr{M}$, *Function* f)

```
Vector φ ← 0
for all triangles T ∈ M
    node o = LeafNodeContaining( T )
    for all nodes o' ∈ Neighbors( o )
        φ_{o'} -= ∫_T f(p) · b_{o'}(p) dp
return φ
```

Figure 2: *Algorithms for computing the coefficients of the Laplace-Beltrami matrix at different levels of the hierarchy (top) and the coefficients of the constraint vector (bottom).*

5.1. Spectral Analysis

The spectrum of the Laplace-Beltrami operator characterizes the modes of the surface and plays an essential role in a variety of applications, including shape matching [RWP05], mesh editing [Tau95], and signal processing [VL08].

To evaluate the robustness of our approach we compare the spectra obtained from our octree-based Laplace-Beltrami operator with those obtained from the cotangent-weight Laplacian. For both, we compute the generalized eigenvalues λ and eigenfunctions f_λ such that the projection of the Laplacian of f_λ onto $\mathscr{F}$ equals the projection of λf_λ onto $\mathscr{F}$. Formally:

$$L^{\mathscr{M}} \eta = \lambda D^{\mathscr{M}} \eta$$

where $L^{\mathscr{M}}$ is the Laplace-Beltrami operator and $D^{\mathscr{M}}$ is the mass matrix, with $D^{\mathscr{M}}_{ij} = \int_{\mathscr{M}} b_i(p) \cdot b_j(p) dp$.

A comparison of the spectra is shown in Figure 3. The original model is shown on the left, overlaid with the spectra of the Laplace-Beltrami operators computed from different tessellations. We also show a detailed view of the spectra at higher frequencies, zooming in on the results obtained using the cotangent weights (middle) and our restricted finite-

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.

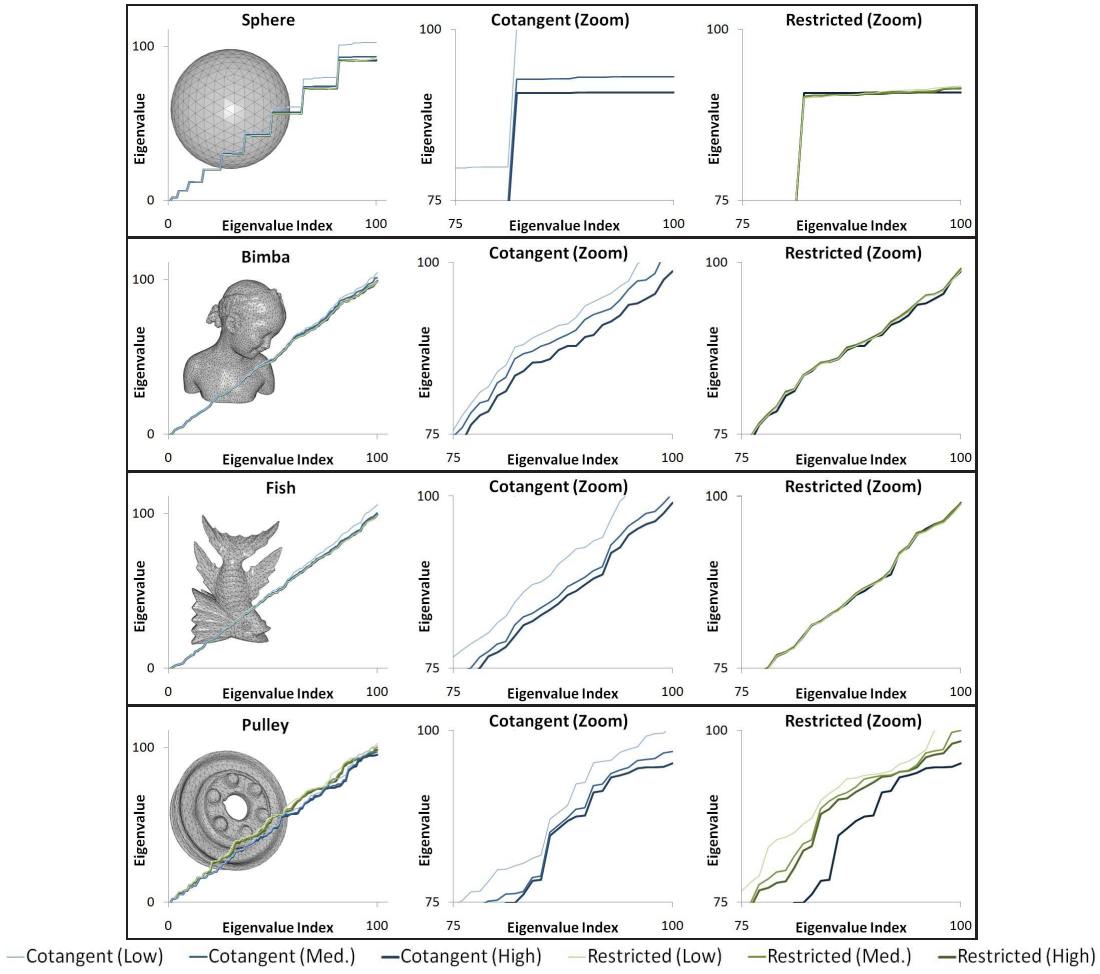

Figure 3: *Using our Laplace-Beltrami operator, we compute a spectrum that only depends on the geometry of the surface. In contrast, using the cotangent formulation results in a spectrum sensitive to the tessellation.*

elements, drawn on top of the spectrum from the finest-resolution cotangent-weight Laplacian (right). The sphere was obtained by recursively subdividing an octahedron. For the bimba, fish and pulley, we started with a high resolution model and used QSlim [GH97] to obtain the coarser, watertight, tessellations.

As the plots indicate, the cotangent Laplacian is sensitive to the tessellation, only converging to the true spectrum at finer triangulations. In contrast, the spectrum defined by our operator remains stable across the different triangulations and is nearly identical to the spectrum of the cotangent Laplacian at the finest tessellation (shown in dark blue in the detailed views on the right). Note that for the sphere, the eigenspaces are known to be multi-dimensional, with the l-th eigenspace consisting of $2l+1$ spherical harmonics with eigenvalues $l \cdot (l+1)$, resulting in the predictable stair-stepping pattern witnessed in the top plots.

The advantage of our approach is further evidenced by considering the dimensions of the Laplace-Beltrami operators shown in Table 1. As the table indicates, using our method to define the operator, we stably compute the eigenvalues using linear systems that are between 5 and 20 times smaller than what would be required for a cotangent-weight Laplacian. Note that the cotangent Laplacian is defined by associating an element with each vertex, so the dimension of the operator grows as the triangulation is refined. In contrast, since the dimension of our system only depends on the

	Cotangent			Restricted		
Model	Low	Med.	High	Low	Med.	High
Sphere	642	2,562	10,242	504	504	504
Bimba	6,100	12,200	74,764	6,071	6,083	6,083
Fish	3,700	14,800	59,200	3,617	3,619	3,616
Pulley	6,459	19,499	45,676	6,160	6,160	6,161

Table 1: *Dimensions of the Laplace-Beltrami operators defined for the different tessellations of the models in Figure 3.*

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.

number of octree-nodes abutting the surface, the dimension of our system remains constant across the triangulations.

The exception to this is the pulley model, for which our octree-based operator fails to define a robust spectrum. We discuss this case in greater detail at the end of the section.

5.2. Texture Back-Projection

In applications where surfaces are reconstructed from a set of registered scans, there is often no one-to-one correspondence between reconstructed surface points and points on the scans. Consequently, it is difficult to re-bind additional scan information, such as color, to the reconstructed surface.

One simple way to assign color to each vertex in the reconstructed model is to use the closest input sample's color. However, due to varying camera exposure and differing lighting conditions, color transitions between scans in overlap regions are not necessarily continuous, as seen in Figure 1 (left). The problems with taking color from nearby scans can be reduced by computing a color correction between different images [BR02], correcting color based on laser return intensities and careful calibration (e.g. [XGRD06]), blending color between different views, or carefully choosing seams between images. Masked Photoblending [CCCS08] combines many of these ideas to blend weighted pixels from different views. Graph cuts are also commonly used in both the image domain (e.g. [Dav98]) and (for texture synthesis) on meshes [ZHW*06], to select boundaries that will not be visible between different images or texture patches. However, blending can smooth out details, while graph cuts cannot always completely eliminate discontinuities.

In gradient-domain image-processing [PGB03, LZPW04, ADA*04], discontinuities are pushed from the texture to its derivative where they are less perceptible. Using our solver, we can extend gradient-domain stitching to meshes. Following the approach used in image processing, we define a gradient field over the mesh and solve the Poisson equation to fit a function to the gradients.

We define a piecewise linear gradient field by first setting the gradients at the vertices of a triangle to the projection of the color gradients from the nearest scan onto the triangle's tangent plane, and then defining the vector field within the triangle to be the vector field that linearly interpolates the vertex gradients. We obtain the coefficients $\eta = [\eta_1, \ldots, \eta_n]^T$ of the function $u(p) = \sum \eta_i b_i(p)$ whose gradients most closely match the constraint field $\vec{v}(p)$ by solving the system $L^{\mathcal{M}} \eta = \phi$, where $\phi = [\phi_1, \ldots, \phi_n]^T$ are the integrated inner products of this vector field with the surface gradients of the finite-elements:

$$\phi_i = -\int_{\mathcal{M}} \langle \vec{v}(p), \nabla_{\mathcal{M}} b_i(p) \rangle \, dp. \qquad (6)$$

An example stitched color field is shown in Figure 1

Figure 4: *An input texture map (left), the solution obtained when the gradients were used as constraints to our Poisson solver (middle), and the solver error (right).*
***Dimension**: 138,288 ; **Times**: 2+4(s) ; **RMS**: 0.0536*

(right). Note that even without any intelligent gradient selection, the final color seamlessly transitions across the scan boundaries without blending artifacts, despite the varying lighting conditions across the different scans.

To validate the quality of our solver, we also extracted the gradient field from a known color map and compared our solution with the original. Figure 4 shows the results of this experiment, with the original color map on the left, our solution in the middle, and the error on the right. Note that the depth of our octree (eight) limits the maximum frequency we can resolve, so our solution exhibits errors near sharp features like the edges of the stripes in the shirt.

5.3. Function Fitting and Curvature Estimation

Though our focus is on the Poisson equation, the same finite-elements structure can be used to find the function in $\mathcal{F}$ that best matches a prescribed scalar field. In particular, given a function $g : \mathcal{M} \to \mathbb{R}$, represented either as a (piecewise polynomial) function over the triangles of the mesh, or as a set of uniformly sampled values, the projection of g onto $\mathcal{F}$ is the function $f \in \mathcal{F}$ satisfying:

$$\int_{\mathcal{M}} f(p) \cdot b_j(p) dp = \int_{\mathcal{M}} g(p) \cdot b_j(p) dp.$$

As in Section 3, finding the coefficients of f requires solving the linear system $D^{\mathcal{M}} \eta = \gamma$, where η is the vector of solution coefficients, $D^{\mathcal{M}}$ is the mass matrix, and γ is the vector of inner products of g with each of the elements:

$$\gamma_j = \int_{\mathcal{M}} g(p) \cdot b_j(p) dp.$$

And, as with the Poisson equation, we can leverage the multiresolution structure on the space of functions $\mathcal{F}$ to efficiently solve the linear system using a multigrid solver.

We briefly describe two ways to use this method to solve for the (mean) curvature of a model.

Differentiating a Normal Field One approach for computing the curvature is to solve for an approximate normal

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.

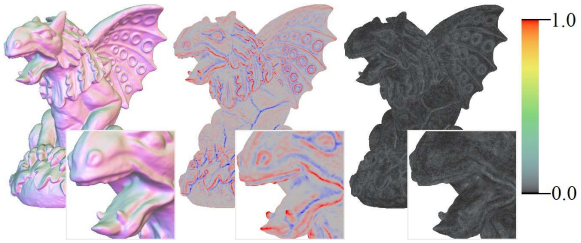

Figure 5: *Fitting a function to the normals, we get a smooth vector field (left) that we can differentiate to get the mean curvatures (middle). The fitting error is shown on the right.* ***Dimension****: 559,839 ;* ***Times****: 11+17(s) ;* ***RMS****: 0.0071*

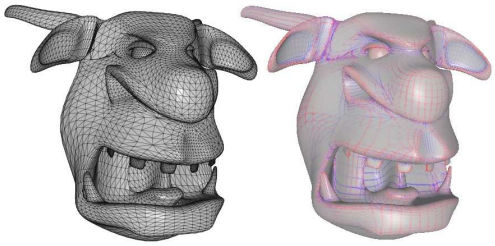

Figure 6: *A piecewise linear surface (left) and the mean curvature (right) obtained by solving for the function that best approximates the Laplacian of the coordinate functions.* ***Dimension****: 612,035 ;* ***Times****: 12+19(s) ;* ***RMS****: (N/A)*

field by finding the linear combination of elements that best fits the sampled normals. Differentiating the normal field along the tangent directions and projecting the gradients back into the tangent space, we obtain a 2×2 matrix that approximates the curvature tensor, and whose trace and determinant approximate the mean and Gaussian curvatures. (Since the original field is defined by interpolating the per-vertex normals and then normalizing, the vector field we fit is not locally polynomial and we approximate the integrals as a sum over uniform samples.)

Figure 5 illustrates this type of curvature estimation. The normal field is shown on the left, with red, green, and blue values at a point set in correspondence to the *x*-, *y*-, and *z*-components of the normals. Mean curvatures are shown in the middle, with blue corresponding to negative curvature values and red corresponding to positive values. Error in the reconstruction of the normal field is shown on the right.

Computing the Laplacian of the Coordinate Function An alternate approach leverages the fact that the Laplacian of the coordinate functions is equal to the mean-curvature vector. Though we cannot compute the coordinate functions' Laplacians (as their derivatives are discontinuous), we can use the weak formulation and compute the inner products of the coordinate functions' gradient with the gradients of the elements. Thus, we can solve for the mean curvature vector by solving the linear system $D^{\mathcal{M}}\eta = \phi$, where $D^{\mathcal{M}}$ is the mass matrix used to project the Laplacian of the coordinate functions onto the space $\mathcal{F}$, η is the vector of solution coefficients, and ϕ is the vector of integrated dot-products:

$$\phi_i = -\int_{\mathcal{M}} \left\langle \nabla_{\mathcal{M}} C(p), \nabla_{\mathcal{M}} b_i^{\mathcal{M}}(p) \right\rangle dp,$$

where $C(p) = p$ is the coordinate function.

Figure 6 shows the resulting mean-curvature values, with the original mesh on the left and the mean curvature on the right. (Gray indicates zero mean curvature, red is positive, and blue is negative.) Since the coordinate functions are piecewise linear, the mean curvature is derived from a piecewise constant normal field, and the resulting values are zero on the interior of the faces. It is only near triangle boundaries that the mean-curvature is non-zero, effectively representing the projection of a "delta-like" function onto the band-limited space of functions spanned by the elements.

5.4. Comparison with Algebraic Multigrid Solvers

To evaluate our solver's efficiency, we compare our performance to two different configurations of the state-of-the-art BoomerAMG solver [HY00]. The first (AMG1) is the classical AMG solver from Ruge and Stueben [RS87], which uses a sequential coloring algorithm to derive coarsened grids. This proves to have good convergence, but the resulting grids have relatively high complexity. We set the strength threshold to 0.25, which is the typical value for Laplacian operators. The second (AMG2) is the best-tuned configuration for BoomerAMG. We adopt the [GMS06] coarsening option and fully tune its parameters to ensure the best performance. To make a fair comparison, we ran these algorithms in a single thread using Gauss-Seidel smoothers.

Table 2 shows the results of the experiment, giving the dimension of the system (defined by the number of nodes at the finest depth of the tree), the time for defining the

Model		AMG1	AMG2	Ours
Rooster	Setup	Out of	Out of	30.7
(1,062,919)	Solve	memory	memory	76.6
Male	Setup	14.1	1.8	3.9
(138,288)	Solve	26.4	13.1	8.5
Cow	Setup	19.7	2.5	5.4
(189,914)	Solve	39.0	17.2	11.4
Cow*	Setup	13.3	2.1	5.4
(189,914)	Solve	37.8	16.7	13.8
Pulley	Setup	Out of	11.1	18.2
(669,975)	Solve	memory	69.1	34.7
Pulley*	Setup	74.9	9.1	18.2
(669,975)	Solve	171.6	65.4	42.1

Table 2: *A comparison of setup and solve time (in seconds) required by AMG and our multigrid solver to reach a relative residual norm of* 5×10^{-3}*. The asterisk denotes experiments in which soft constraints were introduced and the numbers in parentheses give the dimension of the system.*

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.

lower-resolution systems, and the time to solve for a solution whose ratio of ending to starting residual norms is smaller than 5×10^{-3}. Except the rooster model, which was computed from an octree of depth nine, we used a tree of depth eight for all the experiments. As the table indicates, our simple multigrid solver remains competitive with state-of-the-art AMG implementations for small systems, and outperforms both AMG implementations for larger ones.

We do not compare at function fitting as the AMG solvers failed to converge on these experiments. We believe this is because the mass matrix has non-negative values along its diagonal, which (as noted in Ruge and Steuben [RS87]) results in a highly oscillating algebraic error that violates AMG's assumption of an algebraically smooth residual.

5.5. Discussion of Limitations

Though constructing the Laplace-Beltrami operator by restricting 3D functions has several advantages, it also has the property of supplanting geodesic distances with Euclidean ones, at the resolution of the basis functions. This has two important consequences. First, if Euclidean distance is not a good estimate of geodesic distance at some coarse resolution of our octree, the benefit of operating at that resolution within our multigrid solver will be reduced, increasing the time required for the solver to converge. Second, and more important, if the Euclidean distance approximation is poor even at the finest level, the solver may have difficulty converging to the correct result. This is especially the case for models with narrow cross-sections, such as the pulley model, in which the support of individual basis elements is larger than the separating distance of disjoint (and geodesically-distant) patches of the surface. We believe this to be the cause of the instability of the Laplace-Beltrami spectrum in the case of the pulley in Figure 3, bottom.

In practice, we have found that these effects are more pronounced in gradient-fitting applications, where the value at a point is influenced by constraints defined over the entire surface, than they are for value fitting. Figure 7 shows an example in which the Poisson equation was used to fit a vector to the *gradient* of the surface normals (left). In these cases, the thin regions near the lip of the pulley, and the interpenetrating surfaces at the base and end of the cow's tail result in erroneous estimations of the normal field (middle).

Though this is an inherent limitation of our method, we have found that in practice the instability can be addressed by introducing a small regularization term to localize the solution of the system. Specifically, we replace the Poisson equation with the screened Poisson equation ([BCCZ08]), adding a small constraint on the values of the solution:

$$L^{\mathscr{M}}\eta = \phi \quad \Longrightarrow \quad (L^{\mathscr{M}} + \alpha D^{\mathscr{M}})\eta = \phi + \alpha\psi,$$

where ψ is the vector of inner products defining the value constraints. Since the effects of the regularization are predominantly low-frequency, using this approach only requires

Figure 7: *Examples of how the Poisson solver can fail due thin regions in the surface (middle) and how regularization can help alleviate the problem (right).*
***Dimension**: 189,914 ; **Times**: 4+ 6(s) ; **RMS**: 0.3073→0.0962*
***Dimension**: 669,975 ; **Times**: 13+18(s) ; **RMS**: 0.0435→0.0024*

a very coarse estimate of the solution – something readily available in applications such as texture back-projection.

The right side of Figure 7 shows the results of applying this type of regularization, and we see that even using a soft constraint on the values ($\alpha \approx 1/20$) the errors becomes imperceptible in the case of the pulley, and are restricted to the non-manifold regions in the case of the cow.

6. Conclusion

We have presented a novel approach for defining and solving the Poisson equation over the surface of a mesh. By using the restriction of nested subspaces of 3D functions, we have shown how to design a finite-elements setting that easily supports a multigrid solver. We have demonstrated the robustness of our operator by analyzing its spectrum and demonstrated its utility in a number of signal processing applications. Finally, we have also shown that the multigrid solver it defines is competitive with state-of-the-art solvers.

In the future, we would like to continue evaluating the system and solver, investigating questions including the conditioning of the Laplace-Beltrami operator and mass matrix, the sensitivity of the system to spatially adjacent but geodesically distant points, and the implications of the multiresolution hierarchy on challenges such as computing the lower-frequencies of the Laplace-Beltrami spectrum.

References

[ADA*04] AGARWALA A., DONTCHEVA M., AGRAWALA M., DRUCKER S., COLBURN A., CURLESS B., SALESIN D., COHEN M.: Interactive digital photomontage. *ACM Transactions on Graphics (SIGGRAPH '04)* (2004), 294–302.

[AKS05] AKSOYLU B., KHODAKOVSKY A., SCHRÖDER P.: Multilevel solvers for unstructured surface meshes. *SIAM Journal of Scientific Computing 26*, 4 (2005), 1146–1165.

[Ale03] ALEXA M.: Differential coordinates for local mesh morphing and deformation. *The Visual Computer 19* (2003), 105–114.

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.

[BCCZ08] BHAT P., CURLESS B., COHEN M., ZITNICK L.: Fourier analysis of the 2D screened Poisson equation for gradient domain problems. In *European Conference on Computer Vision* (2008), pp. 114–128.

[BCF*00] BREZINA M., CLEARY J., FALGOUT R., HENSON V., JONES J., MANTEUFFEL T., MCCORMICK S., RUGE J.: Algebraic multigrid based on element interpolation (AMGe). *SIAM Journal of Scientific Computing 22*, 5 (2000), 1570–1592.

[BPD06] BAE S., PARIS S., DURAND F.: Two-scale tone management for photographic look. *ACM Transactions on Graphics (SIGGRAPH '06)* (2006), 637–645.

[BR02] BERNARDINI F., RUSHMEIER H.: The 3d model acquisition pipeline. *Computer Graphics Forum 21*, 2 (Jun 2002).

[CCCS08] CALLIERI M., CIGNONI P., CORSINI M., SCOPIGNO R.: Masked photo blending: mapping dense photographic dataset on high-resolution 3d models. *Computer & Graphics 32*, 4 (Aug 2008), 464–473.

[CFH*00] CLEARY A. J., FALGOUT R. D., HENSON V. E., JONES J. E., MANTEUFFEL T. A., MCCORMICK S. F., MIRANDA G. N., RUGE J. W.: Robustness and scalability of algebraic multigrid. *SIAM Journal of Scientific Computing 21*, 5 (2000), 1886–1908.

[CFH*03] CHARTIER T., FALGOUT R. D., HENSON V. E., JONES J., MANTEUFFEL T., MCCORMICK S., RUGE J., VASSILEVSKI P. S.: Spectral AMGe (ρAMGe). *SIAM Journal of Scientific Computing 25*, 1 (2003), 1–26.

[Cha01] CHARTIER T. P.: *Element-based algebraic multigrid (AMGe) and spectral AMGe*. PhD thesis, University of Colorado, 2001.

[CS97] CHRISTARA C., SMITH B.: Multigrid and multilevel methods for quadratic spline collocation. *BIT Numerical Mathematics 37* (1997), 781–803.

[Dav98] DAVIS J.: Mosaics of scenes with moving objects. In *IEEE Computer Vision and Pattern Recognition* (1998), IEEE Computer Society, p. 354.

[FHD02] FINLAYSON G., HORDLEY S., DREW M.: Removing shadows from images. In *European Conference on Computer Vision* (2002), pp. 129–132.

[FLW02] FATTAL R., LISCHINKSI D., WERMAN M.: Gradient domain high dynamic range compression. vol. 21, pp. 249–256.

[GH97] GARLAND M., HECKBERT P.: Surface simplification using quadric error metrics. In *ACM SIGGRAPH* (1997), pp. 209–216.

[GMS06] GRIEBEL M., METSCH B., SCHWEITZER M. A.: Coarse grid classification: a parallel coarsening scheme for algebraic multigrid methods. 193–214.

[Hor74] HORN B.: Determining lightness from an image. *Computer Graphics and Image Processing 3* (1974), 277–299.

[HY00] HENSON V. E., YANG U. M.: Boomeramg: A parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics 41* (2000), 155–177.

[KCVS98] KOBBELT L., CAMPAGNA S., VORSATZ J., SEIDEL H.-P.: Interactive multi-resolution modeling on arbitrary meshes. In *ACM SIGGRAPH* (1998).

[LSCO*04] LIPMAN Y., SORKINE O., COHEN-OR D., LEVIN D., RÖSSL C., SEIDEL H.-P.: Differential coordinates for interactive mesh editing. In *Shape Modeling International* (2004), pp. 181–190.

[LZPW04] LEVIN A., ZOMET A., PELEG S., WEISS Y.: Seamless image stitching in the gradient domain. In *European Conference on Computer Vision* (2004), pp. 377–389.

[May84] MAYO A.: The fast solution of poisson's and the biharmonic equations on irregular regions. *SIAM Journal on Numerical Analysis 21*, 2 (1984), 285–299.

[MI05] MITTAL R., IACCARINO G.: Immersed boundary methods. *Annual Review of Fluid Mechanics 37* (2005), 239–261.

[NGH04] NI X., GARLAND M., HART J.: Fair Morse functions for extracting the topological structure of a surface mesh. *ACM Transactions on Graphics 23*, 2 (2004).

[OSG08] OVSJANIKOV M., SUN J., GUIBAS L.: Global intrinsic symmetries of shapes. *Computer Graphics Forum (SGP '08) 27* (2008), 1341–1348.

[Pes02] PESKIN C.: The immersed boundary method. *Acta Numerica 11* (2002), 479–517.

[PGB03] PÉREZ P., GANGNET M., BLAKE A.: Poisson image editing. *ACM Transactions on Graphics (SIGGRAPH '03)* (2003), 313–318.

[RL03] RAY N., LEVY B.: Hierarchical least squares conformal map. In *Pacific Graphics* (2003), p. 263.

[RS87] RUGE J., STUEBEN K.: Algebraic multigrid. 73–130.

[Rus07] RUSTAMOV R. M.: Laplace-beltrami eigenfunctions for deformation invariant shape representation. In *Symposium on Geometry Processing* (2007), pp. 225–233.

[RWP05] REUTER M., WOLTER F.-E., PEINECKE N.: Laplace-spectra as fingerprints for shape matching. In *Symposium on Solid and Physical Modeling* (2005), pp. 101–106.

[SCOL*04] SORKINE O., COHEN-OR D., LIPMAN Y., ALEXA M., RÖSSL C., SEIDEL H.-P.: Laplacian surface editing. In *Symposium on Geometry Processing* (2004), pp. 179–188.

[SYBwF06] SHI L., YU Y., BELL N., WEN FENG W.: A fast multigrid algorithm for mesh deformation. *ACM Transactions on Graph 25*, 3 (2006), 1108–1117.

[Tau95] TAUBIN G.: A signal processing approach to fair surface design. In *ACM SIGGRAPH* (1995), pp. 351–358.

[Tay08] TAYLOR M.: Asymmetric cubature formulas for polynomial integration in the triangle and square. *Journal of Computational and Applied Mathematics 218* (2008), 184–191.

[TW03] TAUSCH J., WHITE J.: Multiscale bases for the sparse representation of boundary integral operators on complex geometry. *SIAM Journal of Scientific Computing 24*, 5 (2003), 1610–1629.

[VL08] VALLET B., LÉVY B.: Spectral geometry processing with manifold harmonics. *Computer Graphics Forum (Eurographics '08) 2* (2008).

[Wei01] WEISS Y.: Deriving intrinsic images from image sequences. In *International Conference on Computer Vision* (2001), pp. 68–75.

[Wes04] WESSELING P.: *An Introduction to Multigrid Methods*. R. T. Edwards, Inc., 2004.

[XGRD06] XU C., GEORGHIADES A., RUSHMEIER H., DORSEY J.: A system for reconstructing integrated texture maps for large structures. In *3D Data Processing, Visualization, and Transmission* (2006), IEEE Computer Society, pp. 822–829.

[YZX*04] YU Y., ZHOU K., XU D., SHI X., BAO H., GUO B., SHUM H.-Y.: Mesh editing with poisson-based gradient field manipulation. *ACM Transactions on Graphics (SIGGRAPH '04) 23* (2004), 644–651.

[ZHW*06] ZHOU K., HUANG X., WANG X., TONG Y., DESBRUN M., GUO B., SHUM H.-Y.: Mesh quilting for geometric texture synthesis. *ACM Transactions on Graphics (SIGGRAPH '06) 25*, 3 (2006), 690–697.

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.