Eurographics Symposium on Geometry Processing (2006)
Konrad Polthier, Alla Sheffer (Editors)

Poisson Surface Reconstruction

Michael Kazhdan[1], Matthew Bolitho[1] and Hugues Hoppe[2]

[1]Johns Hopkins University, Baltimore MD, USA
[2]Microsoft Research, Redmond WA, USA

Abstract

We show that surface reconstruction from oriented points can be cast as a spatial Poisson problem. This Poisson formulation considers all the points at once, without resorting to heuristic spatial partitioning or blending, and is therefore highly resilient to data noise. Unlike radial basis function schemes, our Poisson approach allows a hierarchy of locally supported basis functions, and therefore the solution reduces to a well conditioned sparse linear system. We describe a spatially adaptive multiscale algorithm whose time and space complexities are proportional to the size of the reconstructed model. Experimenting with publicly available scan data, we demonstrate reconstruction of surfaces with greater detail than previously achievable.

1. Introduction

Reconstructing 3D surfaces from point samples is a well studied problem in computer graphics. It allows fitting of scanned data, filling of surface holes, and remeshing of existing models. We provide a novel approach that expresses surface reconstruction as the solution to a Poisson equation.

Like much previous work (Section 2), we approach the problem of surface reconstruction using an implicit function framework. Specifically, like [Kaz05] we compute a 3D *indicator function* χ (defined as 1 at points inside the model, and 0 at points outside), and then obtain the reconstructed surface by extracting an appropriate isosurface.

Our key insight is that there is an integral relationship between oriented points sampled from the surface of a model and the indicator function of the model. Specifically, the gradient of the indicator function is a vector field that is zero almost everywhere (since the indicator function is constant almost everywhere), except at points near the surface, where it is equal to the inward surface normal. Thus, the oriented point samples can be viewed as samples of the gradient of the model's indicator function (Figure 1).

The problem of computing the indicator function thus reduces to inverting the gradient operator, i.e. finding the scalar function χ whose gradient best approximates a vector field $\vec{V}$ defined by the samples, i.e. $\min_\chi \|\nabla\chi - \vec{V}\|$. If we apply the divergence operator, this variational problem transforms into a standard Poisson problem: compute the scalar function χ whose Laplacian (divergence of gradient) equals the divergence of the vector field $\vec{V}$,

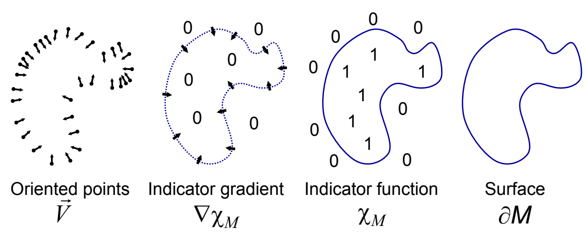

Figure 1: *Intuitive illustration of Poisson reconstruction in 2D.*

$$\Delta\chi \equiv \nabla\cdot\nabla\chi = \nabla\cdot\vec{V}.$$

We will make these definitions precise in Sections 3 and 4.

Formulating surface reconstruction as a Poisson problem offers a number of advantages. Many implicit surface fitting methods segment the data into regions for local fitting, and further combine these local approximations using blending functions. In contrast, Poisson reconstruction is a global solution that considers all the data at once, without resorting to heuristic partitioning or blending. Thus, like radial basis function (RBF) approaches, Poisson reconstruction creates very smooth surfaces that robustly approximate noisy data. But, whereas ideal RBFs are globally supported and nondecaying, the Poisson problem admits a hierarchy of *locally supported* functions, and therefore its solution reduces to a well-conditioned sparse linear system.

Moreover, in many implicit fitting schemes, the value of the implicit function is constrained only near the sample points, and consequently the reconstruction may contain spurious surface sheets away from these samples. Typically this problem is attenuated by introducing auxiliary "off-surface" points (e.g. [CBC*01, OBA*03]). With Poisson surface reconstruction, such surface sheets seldom arise because the gradient of the implicit function is constrained at *all* spatial points. In particular it is constrained to zero away from the samples.

Poisson systems are well known for their resilience in the presence of imperfect data. For instance, "gradient domain" manipulation algorithms (e.g. [FLW02]) intentionally modify the gradient data such that it no longer corresponds to any real potential field, and rely on a Poisson system to recover the globally best-fitting model.

There has been broad interdisciplinary research on solving Poisson problems and many efficient and robust methods have been developed. One particular aspect of our problem instance is that an accurate solution to the Poisson equation is only necessary near the reconstructed surface. This allows us to leverage adaptive Poisson solvers to develop a reconstruction algorithm whose spatial and temporal complexities are proportional to the size of the reconstructed surface.

2. Related Work

Surface reconstruction The reconstruction of surfaces from oriented points has a number of difficulties in practice. The point sampling is often nonuniform. The positions and normals are generally noisy due to sampling inaccuracy and scan misregistration. And, accessibility constraints during scanning may leave some surface regions devoid of data. Given these challenges, reconstruction methods attempt to infer the topology of the unknown surface, accurately fit (but not overfit) the noisy data, and fill holes reasonably.

Several approaches are based on combinatorial structures, such as Delaunay triangulations (e.g. [Boi84, KSO04]), alpha shapes [EM94, BBX95, BMR*99]), or Voronoi diagrams [ABK98, ACK01]. These schemes typically create a triangle mesh that interpolates all or a most of the points. In the presence of noisy data, the resulting surface is often jagged, and is therefore smoothed (e.g. [KSO04]) or refit to the points (e.g. [BBX95]) in subsequent processing.

Other schemes directly reconstruct an approximating surface, typically represented in implicit form. We can broadly classify these as either global or local approaches.

Global fitting methods commonly define the implicit function as the sum of radial basis functions (RBFs) centered at the points (e.g. [Mur91, CBC*01, TO02]). However, the ideal RBFs (polyharmonics) are globally supported and nondecaying, so the solution matrix is dense and ill-conditioned. Practical solutions on large datasets involve adaptive RBF reduction and the fast multipole method [CBC*01].

Local fitting methods consider subsets of nearby points at a time. A simple scheme is to estimate tangent planes and define the implicit function as the signed distance to the tangent plane of the closest point [HDD*92]. Signed distance can also be accumulated into a volumetric grid [CL96]. For function continuity, the influence of several nearby points can be blended together, for instance using moving least squares [ABCO*01,SOS04]. A different approach is to form point neighborhoods by adaptively subdividing space, for example with an adaptive octree. Blending is possible over an octree structure using a multilevel partition of unity, and the type of local implicit patch within each octree node can be selected heuristically [OBA*03].

Our Poisson reconstruction combines benefits of both global and local fitting schemes. It is global and therefore does not involve heuristic decisions for forming local neighborhoods, selecting surface patch types, and choosing blend weights. Yet, the basis functions are associated with the ambient space rather than the data points, are locally supported, and have a simple hierarchical structure that results in a sparse, well-conditioned system.

Our approach of solving an indicator function is similar to the Fourier-based reconstruction scheme of Kazhdan [Kaz05]. In fact, we show in Appendix A that our basic Poisson formulation is mathematically equivalent. Indeed, the Fast Fourier Transform (FFT) is a common technique for solving *dense*, *periodic* Poisson systems. However, the FFT requires $O(r^3 \log r)$ time and $O(r^3)$ space where r is the 3D grid resolution, quickly becoming prohibitive for fine resolutions. In contrast, the Poisson system allows adaptive discretization, and thus yields a scalable solution.

Poisson problems The Poisson equation arises in numerous applications areas. For instance, in computer graphics it is used for tone mapping of high dynamic range images [FLW02], seamless editing of image regions [PGB03], fluid mechanics [LGF04], and mesh editing [YZX*04]. Multigrid Poisson solutions have even been adapted for efficient GPU computation [BFGS03, GWL*03].

The Poisson equation is also used in heat transfer and diffusion problems. Interestingly, Davis et al [DMGL02] use diffusion to fill holes in reconstructed surfaces. Given boundary conditions in the form of a clamped signed distance function d, their diffusion approach essentially solves the homogeneous Poisson equation $\Delta d = 0$ to create an implicit surface spanning the boundaries. They use a local iterative solution rather than a global multiscale Poisson system.

Nehab et al [NRDR05] use a Poisson system to fit a 2.5D height field to sampled positions and normals. Their approach fits a given parametric surface and is well-suited to the reconstruction of surfaces from individual scans. However, in the case that the samples are obtained from the union of multiple scans, their approach cannot be directly applied to obtain a connected, watertight surface.

3. Our Poisson reconstruction approach

The input data S is a set of samples $s \in S$, each consisting of a point $s.p$ and an inward-facing normal $s.\vec{N}$, assumed to lie on or near the surface ∂M of an unknown model M. Our goal is to reconstruct a watertight, triangulated approximation to the surface by approximating the indicator function of the model and extracting the isosurface, as illustrated in Figure 2.

The key challenge is to accurately compute the indicator function from the samples. In this section, we derive a relationship between the gradient of the indicator function and an integral of the surface normal field. We then approximate this surface integral by a summation over the given oriented point samples. Finally, we reconstruct the indicator function from this gradient field as a Poisson problem.

Defining the gradient field Because the indicator function is a piecewise constant function, explicit computation of its gradient field would result in a vector field with unbounded values at the surface boundary. To avoid this, we convolve the indicator function with a smoothing filter and consider the gradient field of the smoothed function. The following lemma formalizes the relationship between the gradient of the smoothed indicator function and the surface normal field.

Lemma: Given a solid M with boundary ∂M, let χ_M denote the indicator function of M, $\vec{N}_{\partial M}(p)$ be the inward surface normal at $p \in \partial M$, $\tilde{F}(q)$ be a smoothing filter, and $\tilde{F}_p(q) = \tilde{F}(q-p)$ its translation to the point p. The gradient of the smoothed indicator function is equal to the vector field obtained by smoothing the surface normal field:

$$\nabla\left(\chi_M * \tilde{F}\right)(q_0) = \int_{\partial M} \tilde{F}_p(q_0)\vec{N}_{\partial M}(p)dp. \tag{1}$$

Proof: To prove this, we show equality for each of the components of the vector field. Computing the partial derivative of the smoothed indicator function with respect to x, we get:

$$\begin{aligned}
\left.\frac{\partial}{\partial x}\right|_{q_0}\left(\chi_M * \tilde{F}\right) &= \left.\frac{\partial}{\partial x}\right|_{q=q_0}\int_M \tilde{F}(q-p)dp \\
&= \int_M \left(-\frac{\partial}{\partial x}\tilde{F}(q_0-p)\right)dp \\
&= -\int_M \nabla\cdot\left(\tilde{F}(q_0-p),0,0\right)dp \\
&= \int_{\partial M}\left\langle\left(\tilde{F}_p(q_0),0,0\right),\vec{N}_{\partial M}(p)\right\rangle dp.
\end{aligned}$$

(The first equality follows from the fact that χ_M is equal to zero outside of M and one inside. The second follows from the fact that $(\partial/\partial q)\tilde{F}(q-p) = -(\partial/\partial p)\tilde{F}(q-p)$. The last follows from the Divergence Theorem.)

A similar argument shows that the y-, and z-components of the two sides are equal, thereby completing the proof. □

Approximating the gradient field Of course, we cannot evaluate the surface integral since we do not yet know the surface geometry. However, the input set of oriented points provides precisely enough information to approximate the integral with a discrete summation. Specifically, using the point set S to partition ∂M into distinct patches $\mathscr{P}_s \subset \partial M$, we can approximate the integral over a patch $\mathscr{P}_s$ by the value at point sample $s.p$, scaled by the area of the patch:

$$\begin{aligned}
\nabla(\chi_M * \tilde{F})(q) &= \sum_{s\in S}\int_{\mathscr{P}_s}\tilde{F}_p(q)\vec{N}_{\partial M}(p)dp \\
&\approx \sum_{s\in S}|\mathscr{P}_s|\,\tilde{F}_{s.p}(q)\,s.\vec{N} \;\equiv\; \vec{V}(q).
\end{aligned} \tag{2}$$

It should be noted that though Equation 1 is true for any smoothing filter $\tilde{F}$, in practice, care must be taken in choosing the filter. In particular, we would like the filter to satisfy two conditions. On the one hand, it should be sufficiently narrow so that we do not over-smooth the data. And on the other hand, it should be wide enough so that the integral over $\mathscr{P}_s$ is well approximated by the value at $s.p$ scaled by the patch area. A good choice of filter that balances these two requirements is a Gaussian whose variance is on the order of the sampling resolution.

Solving the Poisson problem Having formed a vector field $\vec{V}$, we want to solve for the function $\tilde{\chi}$ such that $\nabla\tilde{\chi} = \vec{V}$. However, $\vec{V}$ is generally not integrable (i.e. it is not curl-free), so an exact solution does not generally exist. To find the best least-squares approximate solution, we apply the divergence operator to form the Poisson equation

$$\Delta\tilde{\chi} = \nabla\cdot\vec{V}.$$

In the next section, we describe our implementation of these steps in more detail.

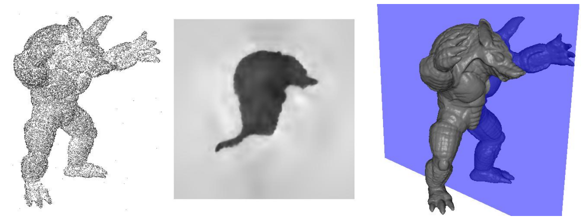

Figure 2: *Points from scans of the "Armadillo Man" model (left), our Poisson surface reconstruction (right), and a visualization of the indicator function (middle) along a plane through the 3D volume.*

4. Implementation

We first present our reconstruction algorithm under the assumption that the point samples are uniformly distributed over the model surface. We define a space of functions with high resolution near the surface of the model and coarser resolution away from it, express the vector field $\vec{V}$ as a linear sum of functions in this space, set up and solve the Poisson equation, and extract an isosurface of the resulting indicator function. We then extend our algorithm to address the case of non-uniformly sampled points.

4.1. Problem Discretization

First, we must choose the space of functions in which to discretize the problem. The most straightforward approach is to start with a regular 3D grid [Kaz05], but such a uniform structure becomes impractical for fine-detail reconstruction, since the dimension of the space is cubic in the resolution while the number of surface triangles grows quadratically.

Fortunately, an accurate representation of the implicit function is only necessary near the reconstructed surface. This motivates the use of an adaptive octree both to represent the implicit function and to solve the Poisson system (e.g. [GKS02, LGF04]). Specifically, we use the positions of the sample points to define an octree $\mathscr{O}$ and associate a function F_o to each node $o \in \mathscr{O}$ of the tree, choosing the tree and the functions so that the following conditions are satisfied:

1. The vector field $\vec{V}$ can be precisely and efficiently represented as the linear sum of the F_o.
2. The matrix representation of the Poisson equation, expressed in terms of the F_o can be solved efficiently.
3. A representation of the indicator function as the sum of the F_o can be precisely and efficiently evaluated near the surface of the model.

Defining the function space Given a set of point samples S and a maximum tree depth D, we define the octree $\mathscr{O}$ to be the minimal octree with the property that every point sample falls into a leaf node at depth D.

Next, we define a space of functions obtained as the span of translates and scales of a fixed, unit-integral, base function $F : \mathbb{R}^3 \to \mathbb{R}$. For every node $o \in \mathscr{O}$, we set F_o to be the unit-integral "node function" centered about the node o and stretched by the size of o:

$$F_o(q) \equiv F\left(\frac{q - o.c}{o.w}\right)\frac{1}{o.w^3}.$$

where $o.c$ and $o.w$ are the center and width of node o.

This space of functions $\mathscr{F}_{\mathscr{O},F} \equiv \mathrm{Span}\{F_o\}$ has a multiresolution structure similar to that of traditional wavelet representations. Finer nodes are associated with higher-frequency functions, and the function representation becomes more precise as we near the surface.

Selecting a base function In selecting a base function F, our goal is to choose a function so that the vector field $\vec{V}$, defined in Equation 2, can be precisely and efficiently represented as the linear sum of the node functions $\{F_o\}$.

If we were to replace the position of each sample with the center of the leaf node containing it, the vector field $\vec{V}$ could be efficiently expressed as the linear sum of $\{F_o\}$ by setting:

$$F(q) = \tilde{F}\left(\frac{q}{2^D}\right).$$

This way, each sample would contribute a single term (the normal vector) to the coefficient corresponding to its leaf's node function. Since the sampling width is 2^{-D} and the samples all fall into leaf nodes of depth D, the error arising from the clamping can never be too big (at most, on the order of half the sampling width). In the next section, we show how the error can be further reduced by using trilinear interpolation to allow for sub-node precision.

Finally, since a maximum tree depth of D corresponds to a sampling width of 2^{-D}, the smoothing filter should approximate a Gaussian with variance on the order of 2^{-D}. Thus, F should approximate a Gaussian with unit-variance.

For efficiency, we approximate the unit-variance Gaussian by a compactly supported function so that (1) the resulting Divergence and Laplacian operators are sparse and (2) the evaluation of a function expressed as the linear sum of F_o at some point q only requires summing over the nodes $o \in \mathscr{O}$ that are close to q. Thus, we set F to be the n-th convolution of a box filter with itself resulting in the base function F:

$$F(x,y,z) \equiv (B(x)B(y)B(z))^{*n} \quad \text{with} \quad B(t) = \begin{cases} 1 & |t| < 0.5 \\ 0 & \text{otherwise} \end{cases}$$

Note that as n is increased, F more closely approximates a Gaussian and its support grows larger; in our implementation we use a piecewise quadratic approximation with $n = 3$. Therefore, the function F is supported on the domain $[-1.5, 1.5]^3$ and, for the basis function of any octree node, there are at most 5^3-1 $= 124$ other nodes at the same depth whose functions overlap with it.

4.2. Vector Field Definition

To allow for sub-node precision, we avoid clamping a sample's position to the center of the containing leaf node and instead use trilinear interpolation to distribute the sample across the eight nearest nodes. Thus, we define our approximation to the gradient field of the indicator function as:

$$\vec{V}(q) \equiv \sum_{s \in S} \sum_{o \in \mathrm{Ngbr}_D(s)} \alpha_{o,s} F_o(q) s.\vec{N} \tag{3}$$

where $\mathrm{Ngbr}_D(s)$ are the eight depth-D nodes closest to $s.p$ and $\{\alpha_{o,s}\}$ are the trilinear interpolation weights. (If the neighbors are not in the tree, we refine it to include them.)

Since the samples are uniform, we can assume that the area of a patch $\mathscr{P}_s$ is constant and $\vec{V}$ is a good approximation, up to a multiplicative constant, of the gradient of the smoothed indicator function. We will show that the choice of multiplicative constant does not affect the reconstruction.

4.3. Poisson Solution

Having defined the vector field $\vec{V}$, we would like to solve for the function $\tilde{\chi} \in \mathscr{F}_{\mathscr{O},F}$ such that the gradient of $\tilde{\chi}$ is closest to $\vec{V}$, i.e. a solution to the Poisson equation $\Delta\tilde{\chi} = \nabla \cdot \vec{V}$.

One challenge of solving for $\tilde{\chi}$ is that though $\tilde{\chi}$ and the coordinate functions of $\vec{V}$ are in the space $\mathscr{F}_{\mathscr{O},F}$ it is not necessarily the case that the functions $\Delta\tilde{\chi}$ and $\nabla\cdot\vec{V}$ are.

To address this issue, we need to solve for the function $\tilde{\chi}$ such that the projection of $\Delta\tilde{\chi}$ onto the space $\mathscr{F}_{\mathscr{O},F}$ is closest to the projection of $\nabla\cdot\vec{V}$. Since, in general, the functions F_o do not form an orthonormal basis, solving this problem directly is expensive. However, we can simplify the problem by solving for the function $\tilde{\chi}$ minimizing:

$$\sum_{o\in\mathscr{O}}\left\|\langle\Delta\tilde{\chi}-\nabla\cdot\vec{V},F_o\rangle\right\|^2=\sum_{o\in\mathscr{O}}\left\|\langle\Delta\tilde{\chi},F_o\rangle-\langle\nabla\cdot\vec{V},F_o\rangle\right\|^2.$$

Thus given the $|\mathscr{O}|$-dimensional vector v whose o-th coordinate is $v_o=\langle\nabla\cdot\vec{V},F_o\rangle$, the goal is to solve for the function $\tilde{\chi}$ such that the vector obtained by projecting the Laplacian of $\tilde{\chi}$ onto each of the F_o is as close to v as possible.

To express this in matrix form, let $\tilde{\chi}=\sum_o x_o F_o$, so that we are solving for the vector $x\in\mathbb{R}^{|\mathscr{O}|}$. Then, let us define the $|\mathscr{O}|\times|\mathscr{O}|$ matrix L such that Lx returns the dot product of the Laplacian with each of the F_o. Specifically, for all $o,o'\in\mathscr{O}$, the (o,o')-th entry of L is set to:

$$L_{o,o'}\equiv\left\langle\frac{\partial^2F_o}{\partial x^2},F_{o'}\right\rangle+\left\langle\frac{\partial^2F_o}{\partial y^2},F_{o'}\right\rangle+\left\langle\frac{\partial^2F_o}{\partial z^2},F_{o'}\right\rangle.$$

Thus, solving for $\tilde{\chi}$ amounts to finding

$$\min_{x\in\mathbb{R}^{|\mathscr{O}|}}\|Lx-v\|^2.$$

Note that the matrix L is sparse and symmetric. (Sparse because the F_o are compactly supported, and symmetric because $\int f''g=-\int f'g'$.) Furthermore, there is an inherent multiresolution structure on $\mathscr{F}_{\mathscr{O},F}$, so we use an approach similar to the multigrid approach in [GKS02], solving the restriction L_d of L to the space spanned by the depth d functions (using a conjugate gradient solver) and projecting the fixed-depth solution back onto $\mathscr{F}_{\mathscr{O},F}$ to update the residual.

Addressing memory concerns In practice, as the depth increases, the matrix L_d becomes larger and it may not be practical to store it in memory. Although the number of entries in a column of L_d is bounded by a constant, the constant value can be large. For example, even using a piecewise quadratic base function F, we end up with as many as 125 non-zero entries in a column, resulting in a memory requirement that is 125 times larger than the size of the octree.

To address this issue, we augment our solver with a block Gauss-Seidel solver. That is, we decompose the d-th dimensional space into overlapping regions and solve the restriction of L_d to these different regions, projecting the local solutions back into the d-dimensional space and updating the residuals. By choosing the number of regions to be a function of the depth d, we ensure that the size of the matrix used by the solver never exceeds a desired memory threshold.

4.4. Isosurface Extraction

In order to obtain a reconstructed surface $\partial\tilde{M}$, it is necessary to first select an isovalue and then extract the corresponding isosurface from the computed indicator function.

We choose the isovalue so that the extracted surface closely approximates the positions of the input samples. We do this by evaluating $\tilde{\chi}$ at the sample positions and use the average of the values for isosurface extraction:

$$\partial\tilde{M}\equiv\{q\in\mathbb{R}^3\mid\tilde{\chi}(q)=\gamma\}\quad\text{with}\quad\gamma=\frac{1}{|S|}\sum_{s\in S}\tilde{\chi}(s.p).$$

This choice of isovalue has the property that scaling $\tilde{\chi}$ does not change the isosurface. Thus, knowing the vector field $\vec{V}$ up to a multiplicative constant provides sufficient information for reconstructing the surface.

To extract the isosurface from the indicator function, we use a method similar to previous adaptations of the Marching Cubes [LC87] to octree representations (e.g. [WG92, SFYC96, WKE99]). However, due to the nonconforming properties of our tree, we modify the reconstruction approach slightly, defining the positions of zero-crossings along an edge in terms of the zero-crossings computed by the finest level nodes adjacent to the edge. In the case that an edge of a leaf node has more than one zero-crossing associated to it, the node is subdivided. As in previous approaches, we avoid cracks arising when coarser nodes share a face with finer ones by projecting the isocurve segments from the faces of finer nodes onto the face of the coarser one.

4.5. Non-uniform Samples

We now extend our method to the case of non-uniformly distributed point samples. As in [Kaz05], our approach is to estimate the local sampling density, and scale the contribution of each point accordingly. However, rather than simply scaling the *magnitude* of a *fixed*-width kernel associated with each point, we additionally adapt the kernel width. This results in a reconstruction that maintains sharp features in areas of dense sampling and provides a smooth fit in sparsely sampled regions.

Estimating local sampling density Following the approach of [Kaz05], we implement the density computation using a kernel density estimator [Par62]. The approach is to estimate the number of points in a neighborhood of a sample by "splatting" the samples into a 3D grid, convolving the "splatting" function with a smoothing filter, and evaluating the convolution at each of the sample points.

We implement the convolution in a manner similar to Equation 3. Given a depth $\hat{D}\le D$ we set the density estimator to be the sum of node functions at depth $\hat{D}$:

$$W_{\hat{D}}(q)\equiv\sum_{s\in S}\sum_{o\in\mathrm{Ngbr}_{\hat{D}}(s)}\alpha_{o,s}F_o(q).$$

Since octree nodes at lower resolution are associated with functions that approximate Gaussians of larger width, the parameter $\hat{D}$ provides away for specifying the locality of the density estimation, with smaller values of $\hat{D}$ giving sampling density estimates over larger regions.

Computing the vector field Using the density estimator, we modify the summation in Equation 3 so that each sample's contribution is proportional to its associated area on the surface. Specifically, using the fact that the area is inversely proportional to sampling density, we set:

$$\vec{V}(q) \equiv \sum_{s\in S} \frac{1}{W_{\hat{D}}(s.p)} \sum_{o\in \mathrm{Ngbr}_D(s)} \alpha_{o,s} F_o(q).$$

However, adapting only the magnitudes of the sample contributions results in poor noise filtering in sparsely sampled regions as demonstrated later in Figure 7. Therefore, we additionally adapt the width of the smoothing filter $\tilde{F}$ to the local sampling density. Adapting the filter width lets us retain fine detail in regions of dense sampling, while smoothing out noise in regions of sparse sampling.

Using the fact that node functions at smaller depths correspond to wider smoothing filters, we define

$$\vec{V}(q) \equiv \sum_{s\in S} \frac{1}{W_{\hat{D}}(s.p)} \sum_{o\in \mathrm{Ngbr}_{\mathrm{Depth}(s.p)}(s)} \alpha_{o,s} F_o(q).$$

In this definition, $\mathrm{Depth}(s.p)$ represents the desired depth of a sample point $s \in S$. It is defined by computing the average sampling density W over all of the samples and setting:

$$\mathrm{Depth}(s.p) \equiv \min\left(D, D + \log_4(W_{\hat{D}}(s.p)/W)\right)$$

so that the width of the smoothing filter with which s contributes to $\vec{V}$ is proportional to the radius of its associated surface patch P_s.

Selecting an isovalue Finally, we modify the surface extraction step by selecting an isovalue which is the weighted average of the values of $\tilde{\chi}$ at the sample positions:

$$\partial\tilde{M} \equiv \{q \in \mathbb{R}^3 \mid \tilde{\chi}(q) = \gamma\} \quad \text{with} \quad \gamma = \frac{\sum \frac{1}{W_{\hat{D}}(s.p)} \tilde{\chi}(s.p)}{\sum \frac{1}{W_{\hat{D}}(s.p)}}.$$

5. Results

To evaluate our method we conducted a series of experiments. Our goal was to address three separate questions: How well does the algorithm reconstruct surfaces? How does it compare to other reconstruction methods? And, what are its performance characteristics?

Much practical motivation for surface reconstruction derives from 3D scanning, so we have focused our experiments on the reconstruction of 3D models from real-world data.

5.1. Resolution

We first consider the effects of the maximum octree depth on the reconstructed surface.

Figure 3 shows our reconstruction results for the "dragon" model at octree depths 6, 8, and 10. (In the context of reconstruction on a regular grid, this would correspond to resolutions of 64^3, 256^3, and 1024^3, respectively.) As the tree depth is increased, higher-resolution functions are used to fit the indicator function, and consequently the reconstructions capture finer detail. For example, the scales of the dragon, which are too fine to be captured at the coarsest resolution begin appearing and become more sharply pronounced as the octree depth is increased.

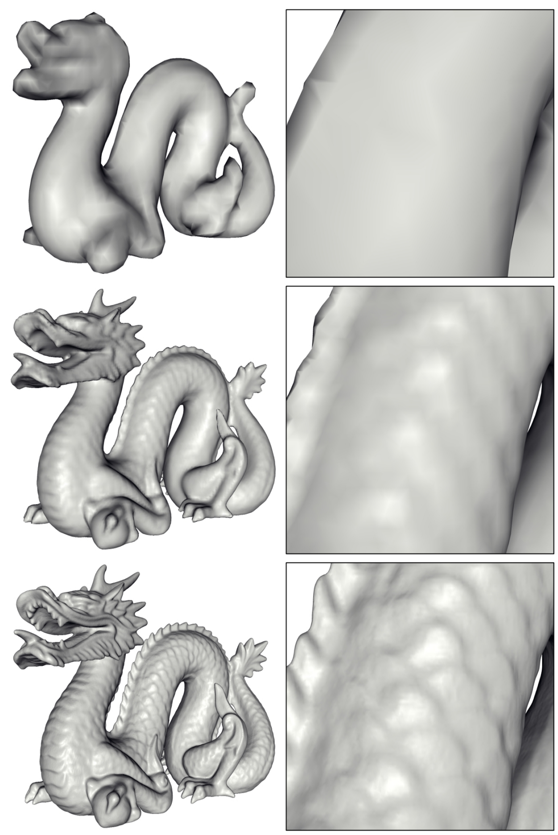

Figure 3: *Reconstructions of the dragon model at octree depths 6 (top), 8 (middle), and 10 (bottom).*

5.2. Comparison to Previous Work

We compare the results of our reconstruction algorithm to the results obtained using Power Crust [ACK01], Robust Cocone [DG04], Fast Radial Basis Functions (FastRBF) [CBC*01], Multi-Level Partition of Unity Implicits (MPU) [OBA*03], Surface Reconstruction from Unorganized Points [HDD*92], Volumetric Range Image Processing (VRIP) [CL96], and the FFT-based method of [Kaz05].

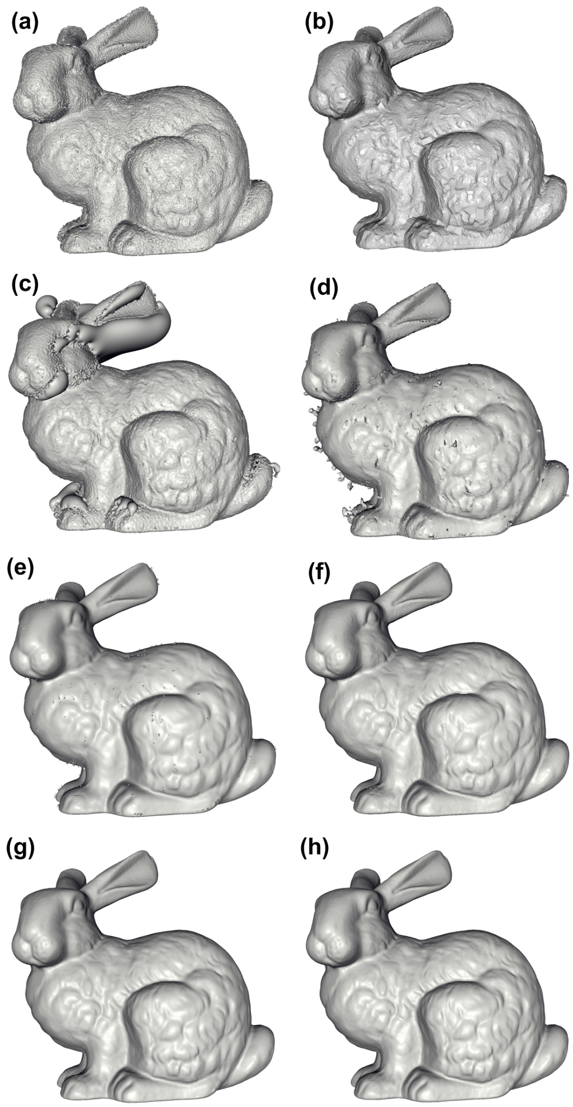

Figure 4: *Reconstructions of the Stanford bunny using Power Crust (a), Robust Cocone (b), Fast RBF (c), MPU (d), Hoppe* et al.*'s reconstruction (e), VRIP (f), FFT-based reconstruction (g), and our Poisson reconstruction (h).*

Our initial test case is the Stanford "bunny" raw dataset of 362,000 points assembled from ten range images. The data was processed to fit the input format of each algorithm. For example, when running our method, we estimated a sample's normal from the positions of the neighbors; Running VRIP, we used the registered scans as input, maintaining the regularity of the sampling, and providing the confidence values.

Figure 4 compares the different reconstructions. Since the scanned data contains noise, interpolatory methods such as Power Crust (a) and Robust Cocone (b) generate surfaces that are themselves noisy. Methods such as Fast RBF (c) and MPU (d), which only constrain the implicit function near the sample points, result in reconstructions with spurious surface sheets. Non-interpolatory methods, such as the approach of [HDD*92] (e), can smooth out the noise, although often at the cost of model detail. VRIP (f), the FFT-based approach (g), and the Poisson approach (h) all accurately reconstruct the surface of the bunny, even in the presence of noise, and we compare these three methods in more detail.

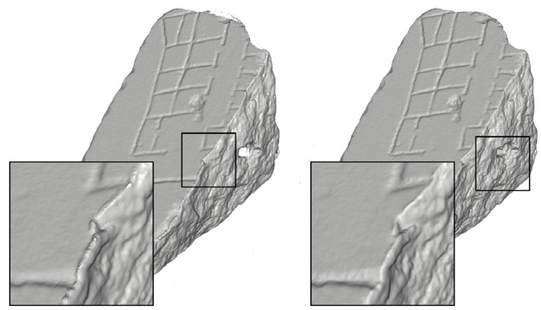

Figure 5: *Reconstructions of a fragment of the Forma Urbis Romae tablet using VRIP (left) and the Poisson solution (right).*

Comparison to VRIP A challenge in surface reconstruction is the recovery of sharp features. We compared our method to VRIP by evaluating the reconstruction of sample points obtained from fragment 661a of the Forma Urbis Romae (30 scans, $2,470,000$ points) and the "Happy Buddha" model (48 scans, $2,468,000$ points), shown in Figures 5 and 6. In both cases, we find that VRIP exhibits a "lipping" phenomenon at sharp creases. This is due to the fact that VRIP's distance function is grown perpendicular to the view direction, not the surface normal. In contrast, our Poisson reconstruction, which is independent of view direction, accurately reconstructs the corner of the fragment and the sharp creases in the Buddha's cloak.

Comparison to the FFT-based approach As Figure 4 demonstrates, our Poisson reconstruction (h) closely matches the one obtained with the FFT-based method (g). Since our method provides an adaptive solution to the same problem, the similarity is a confirmation that in adapting the octree to the data, our method does not discard salient, high-frequency information. We have also confirmed that our Poisson method maintains the high noise resilience already demonstrated in the results of [Kaz05].

Though theoretically equivalent in the context of uniformly sampled data, our use of adaptive-width filters (Section 4.5) gives better reconstructions than the FFT-based method on the non-uniform data commonly encountered in 3D scanning. For example, let us consider the region around the left eye of the "David" model, shown in Figure 7(a). The area above the eyelid (highlighted in red) is sparsely sampled due to the fact that it is in a concave region and is seen only by a few scans. Furthermore, the scans that do sample

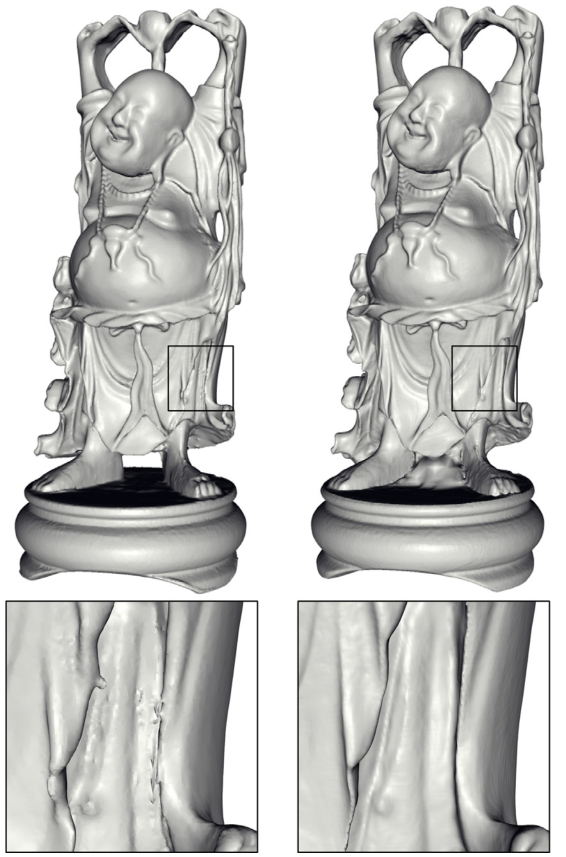

Figure 6: *Reconstructions of the "Happy Buddha" model using VRIP (left) and Poisson reconstruction (right).*

the region tend to sample at near-grazing angles resulting in noisy position and normal estimates. Consequently, fixed-resolution reconstruction schemes such as the FFT-based approach (b) introduce high-frequency noise in these regions. In contrast, our method (c), which adapts both the scale and the variance of the samples' contributions, fits a smoother reconstruction to these regions, without sacrificing fidelity in areas of dense sampling (e.g. the region highlighted in blue).

Limitation of our approach A limitation of our method is that it does not incorporate information associated with the acquisition modality. Figure 6 shows an example of this in the reconstruction at the base of the Buddha. Since there are no samples between the two feet, our method (right) connects the two regions. In contrast, the ability to use secondary information such as line of sight allows VRIP (left) to perform the space carving necessary to disconnect the two feet, resulting in a more accurate reconstruction.

5.3. Performance and Scalability

Table 1 summarizes the temporal and spatial efficiency of our algorithm on the "dragon" model, and indicates that the

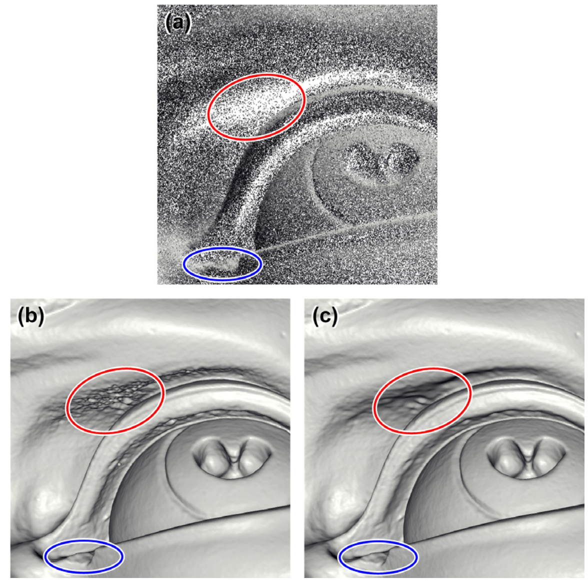

Figure 7: *Reconstruction of samples from the region around the left eye of the David model (a), using the fixed-resolution FFT approach (b), and Poisson reconstruction (c).*

memory and time requirements of our algorithm are roughly quadratic in the resolution. Thus, as we increase the octree depth by one, we find that the running time, the memory overhead, and the number of output triangles increases roughly by a factor of four.

Tree Depth	Time	Peak Memory	# of Tris.
7	6	19	21,000
8	26	75	90,244
9	126	155	374,868
10	633	699	1,516,806

Table 1: *The running time (in seconds), the peak memory usage (in megabytes), and the number of triangles in the reconstructed model for the different depth reconstructions of the dragon model. A kernel depth of 6 was used for density estimation.*

The running time and memory performance of our method in reconstructing the Stanford Bunny at a depth of 9 is compared to the performance of related methods in Table 2. Although in this experiment, our method is neither fastest nor most memory efficient, its quadratic nature makes it scalable to higher resolution reconstructions. As an example, Figure 8 shows a reconstruction of the head of Michelangelo's David at a depth of 11 from a set of 215,613,477 samples. The reconstruction was computed in 1.9 hours and 5.2GB of RAM, generating a 16,328,329 triangle model. Trying to compute an equivalent reconstruction with methods such as the FFT approach would require constructing two voxel grids at a resolution of 2048^3 and would require in excess of 100GB of memory.

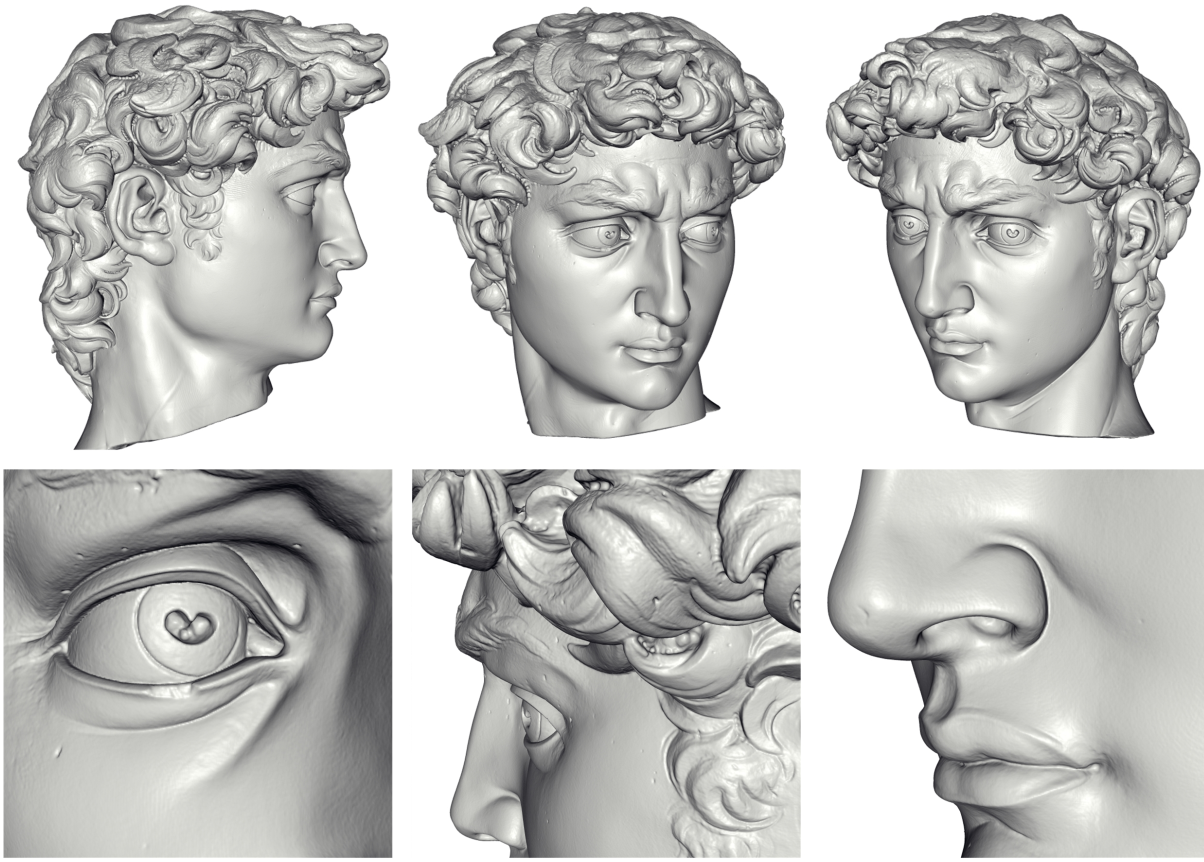

Figure 8: *Several images of the reconstruction of the head of Michelangelo's David, obtained running our algorithm with a maximum tree depth of 11. The ability to reconstruct the head at such a high resolution allows us to make out the fine features in the model such as the inset iris, the drill marks in the hair, the chip on the eyelid, and the creases around the nose and mouth.*

Method	Time	Peak Memory	# of Tris.
Power Crust	380	2653	554,332
Robust Cocone	892	544	272,662
FastRBF	4919	796	1,798,154
MPU	28	260	925,240
Hoppe et al 1992	70	330	950,562
VRIP	86	186	1,038,055
FFT	125	1684	910,320
Poisson	263	310	911,390

Table 2: *The running time (in seconds), the peak memory usage (in megabytes), and the number of triangles in the reconstructed surface of the Stanford Bunny generated by the different methods.*

6. Conclusion

We have shown that surface reconstruction can be expressed as a Poisson problem, which seeks the indicator function that best agrees with a set of noisy, non-uniform observations, and we have demonstrated that this approach can robustly recover fine detail from noisy real-world scans.

There are several avenues for future work:

- Extend the approach to exploit sample confidence values.
- Incorporate line-of-sight information from the scanning process into the solution process.
- Extend the system to allow out-of-core processing for huge datasets.

Acknowledgements

The authors would like to express their thanks to the Stanford 3D Scanning Repository for their generosity in distributing their 3D models. The authors would also like to express particular gratitude to Szymon Rusinkiewicz and Benedict Brown for sharing valuable experiences and ideas, and for providing non-rigid body aligned David data.

References

[ABCO*01] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., LEVIN D., SILVA C.: Point set surfaces. In *Proc. of the Conference on Visualization '01* (2001), 21–28.

[ABK98] AMENTA N., BERN M., KAMVYSSELIS M.: A new Voronoi-based surface reconstruction algorithm. *Computer Graphics (SIGGRAPH '98)* (1998), 415–21.

[ACK01] AMENTA N., CHOI S., KOLLURI R.: The power crust, unions of balls, and the medial axis transform. *Computational Geometry: Theory and Applications 19* (2001), 127–153.

[BBX95] BAJAJ C., BERNARDINI F., XU G.: Automatic reconstruction of surfaces and scalar fields from 3d scans. In *SIGGRAPH* (1995), 109–18.

[BFGS03] BOLZ J., FARMER I., GRINSPUN E., SCHRÖDER P.: Sparse matrix solvers on the GPU: Conjugate gradients and multigrid. *TOG 22* (2003), 917–924.

[BMR*99] BERNARDINI F., MITTLEMAN J., RUSHMEIER H., SILVA C., TAUBIN G.: The ball-pivoting algorithm for surface reconstruction. *IEEE TVCG 5* (1999), 349–359.

[Boi84] BOISSONNAT J.: Geometric structures for three dimensional shape representation. *TOG* (1984), 266–286.

[CBC*01] CARR J., BEATSON R., CHERRIE H., MITCHEL T., FRIGHT W., MCCALLUM B., EVANS T.: Reconstruction and representation of 3D objects with radial basis functions. *SIGGRAPH* (2001), 67–76.

[CL96] CURLESS B., LEVOY M.: A volumetric method for building complex models from range images. *Computer Graphics (SIGGRAPH '96)* (1996), 303–312.

[DG04] DEY T., GOSWAMI S.: Provable surface reconstruction from noisy samples. In *Proc. of the Ann. Symp. Comp. Geom.* (2004), 428–438.

[DMGL02] DAVIS J., MARSCHNER S., GARR M., LEVOY M.: Filling holes in complex surfaces using volumetric diffusion. In *Int. Symp. 3DPVT* (2002), 428–438.

[EM94] EDELSBRUNNER H., MÜCKE E.: Three-dimensional alpha shapes. *TOG* (1994), 43–72.

[FLW02] FATTAL R., LISCHINKSI D., WERMAN M.: Gradient domain high dynamic range compression. In *SIGGRAPH* (2002), 249–256.

[GKS02] GRINSPUN E., KRYSL P., SCHRÖDER P.: Charms: a simple framework for adaptive simulation. In *SIGGRAPH* (2002), 281–290.

[GWL*03] GOODNIGHT N., WOOLLEY C., LEWIN G., LUEBKE D., HUMPHREYS G.: A multigrid solver for boundary value problems using programmable graphics hardware. In *Graphics Hardware* (2003), 102–111.

[HDD*92] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Surface reconstruction from unorganized points. *Computer Graphics 26* (1992), 71–78.

[Kaz05] KAZHDAN M.: Reconstruction of solid models from oriented point sets. *SGP* (2005), 73–82.

[KSO04] KOLLURI R., SHEWCHUK J., O'BRIEN J.: Spectral surface reconstruction from noisy point clouds. In *SGP* (2004), 11–21.

[LC87] LORENSEN W., CLINE H.: Marching cubes: A high resolution 3d surface reconstruction algorithm. *SIGGRAPH* (1987), 163–169.

[LGF04] LOSASSO F., GIBOU F., FEDKIW R.: Simulating water and smoke with an octree data structure. *TOG (SIGGRAPH '04) 23* (2004), 457–462.

[Mur91] MURAKI S.: Volumetric shape description of range data using "blobby model". *Computer Graphics 25* (1991), 227–235.

[NRDR05] NEHAB D., RUSINKIEWICZ S., DAVIS J., RAMAMOORTHI R.: Efficiently combining positions and normals for precise 3D geometry. *TOG (SIGGRAPH '05) 24* (2005).

[OBA*03] OHTAKE Y., BELYAEV A., ALEXA M., TURK G., SEIDEL H.: Multi-level partition of unity implicits. *TOG* (2003), 463–470.

[Par62] PARZEN E.: On estimation of a probability density function and mode. *Ann. Math Stat. 33* (1962), 1065–1076.

[PGB03] PÉREZ P., GANGNET M., BLAKE A.: Poisson image editing. *TOG (SIGGRAPH '03) 22* (2003), 313–318.

[SFYC96] SHEKHAR R., FAYYAD E., YAGEL R., CORNHILL J.: Octree-based decimation of marching cubes surfaces. In *IEEE Visualization* (1996), 335–342.

[SOS04] SHEN C., O'BRIEN J., SHEWCHUK J.: Interpolating and approximating implicit surfaces from polygon soup. *TOG (SIGGRAPH '04) 23* (2004), 896–904.

[TO02] TURK G., O'BRIEN J.: Modelling with implicit surfaces that interpolate. In *TOG* (2002), 855–873.

[WG92] WILHELMS J., GELDER A. V.: Octrees for faster isosurface generation. *TOG 11* (1992), 201–227.

[WKE99] WESTERMANN R., KOBBELT L., ERTL T.: Real-time exploration of regular volume data by adaptive reconstruction of iso-surfaces. *The Visual Computer 15* (1999), 100–111.

[YZX*04] YU Y., ZHOU K., XU D., SHI X., BAO H., GUO B., SHUM H.: Mesh editing with Poisson-based gradient field manipulation. *TOG (SIGGRAPH '04) 23* (2004), 641–648.

Appendix A:

The solution to surface reconstruction described in this paper approaches the problem in a manner similar to the solution of [Kaz05] in that the reconstructed surface is obtained by first computing the indicator function and then extracting the appropriate isosurface.

While the two methods seem to approach the problem of computing the indicator function in different manners ([Kaz05] uses Stokes' Theorem to define the Fourier coefficients of the indicator function while we use the Poisson equation), the two methods are in fact equivalent.

To show this, we use the fact that the Poisson equation $\Delta u = f$ where f is periodic can be solved using the Fourier transform. The Fourier series expansion is $-|\zeta|^2\hat{u}(\zeta) = \hat{f}(\zeta)$, or equivalently $\hat{u}(\zeta) = \frac{-1}{|\zeta|^2}\hat{f}(\zeta)$.

Thus, our Poisson equation $\Delta\chi = \nabla\cdot\vec{V}$ can be solved using $\hat{\chi} = \frac{-1}{|\zeta|^2}\widehat{\nabla\cdot\vec{V}}$. With the well known identity $\hat{f'} = -i\zeta\hat{f}$ and its generalization $\widehat{\nabla\cdot\vec{V}} = -i\zeta\cdot\hat{\vec{V}}$, we get $\hat{\chi} = \frac{i}{|\zeta|^2}\zeta\cdot\hat{\vec{V}}$, which is *identical* to [Kaz05].