DOI:10.1111/cgf.14077
Eurographics Symposium on Geometry Processing 2020
Q. Huang and A. Jacobson
(Guest Editors)

Volume 39 (2020), Number 5

Poisson Surface Reconstruction with Envelope Constraints

Misha Kazhdan[1], Ming Chuang, Szymon Rusinkiewicz[2], and Hugues Hoppe[3]

[1]Johns Hopkins University, [2]Princeton University, [3]Google

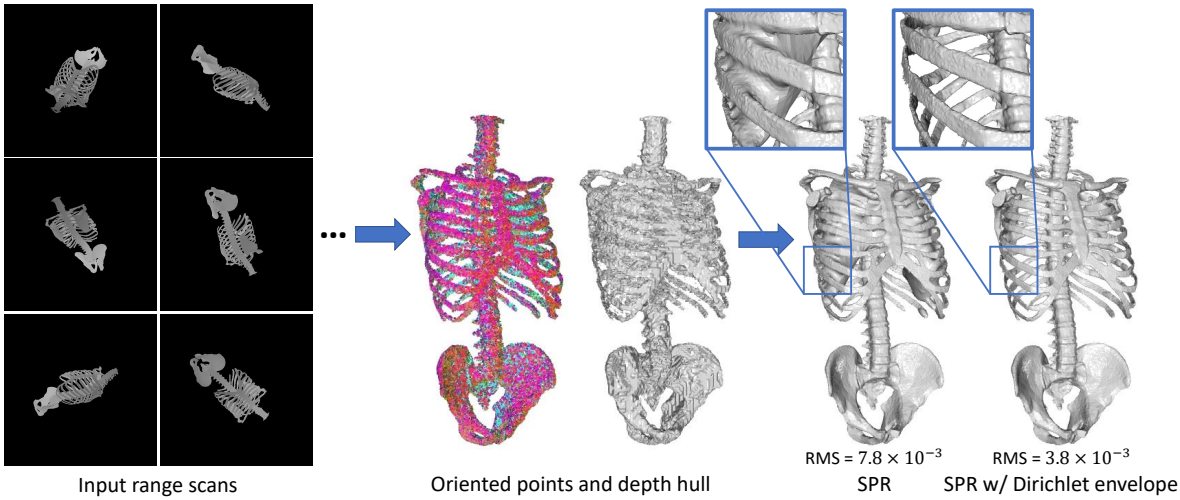

Figure 1: *Given a collection of range scans, one can obtain a set of oriented points and a depth hull volume. (Points are colored by normal orientation.) Applying traditional Screened Poisson Reconstruction (SPR) to the oriented points yields a surface with unwanted artifacts in regions with missing data. By incorporating the depth hull as a Dirichlet constraint within the global Poisson formulation, we prevent the emergence of extraneous surfaces, resulting in a more accurate model.*

Abstract

Reconstructing surfaces from scanned 3D points has been an important research area for several decades. One common approach that has proven efficient and robust to noise is implicit surface reconstruction, i.e. fitting to the points a 3D scalar function (such as an indicator function or signed-distance field) and then extracting an isosurface. Though many techniques fall within this category, existing methods either impose no boundary constraints or impose Dirichlet/Neumann conditions on the surface of a bounding box containing the scanned data.

In this work, we demonstrate the benefit of supporting Dirichlet constraints on a general boundary. To this end, we adapt the Screened Poisson Reconstruction algorithm to input a constraint envelope in addition to the oriented point cloud. We impose Dirichlet boundary conditions, forcing the reconstructed implicit function to be zero outside this constraint surface. Using a visual hull and/or depth hull derived from RGB-D scans to define the constraint envelope, we obtain substantially improved surface reconstructions in regions of missing data.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems

1. Introduction

With the wide proliferation of 3D scanners, the problem of surface reconstruction has received significant attention in the graphics and vision communities. A popular approach is to cast reconstruction as a global optimization problem in which one solves for an implicit function whose level-set fits the input data points. The benefit of combining implicit representation with global optimization is that such techniques automatically produce watertight manifold meshes and tend to be resilient to scanner noise, misalignment, and missing data. However, a limitation is that these techniques are prone to

introducing unwanted surface patches in the vicinity of occluded or unobserved surfaces.

One approach for addressing this issue is to refrain from generating a surface in regions devoid of samples. This can be done in one of two ways. Using compactly supported functions, the implicit surface can be defined in regions near the samples, with no isosurface extracted in regions outside of the support [HDD*92, FG14]. Alternatively, the implicit surface can be trimmed in a post-processing phase by measuring the sampling density of the input point set at the vertices of the output mesh and discarding subsets of the mesh where the sampling density is too low [Kaz13].

While straightforward to implement, these approaches no longer generate watertight surfaces and, as in the case of trimming, require the introduction of additional tuning parameters, which may be difficult to set in a consistent manner. (See Figure 2.)

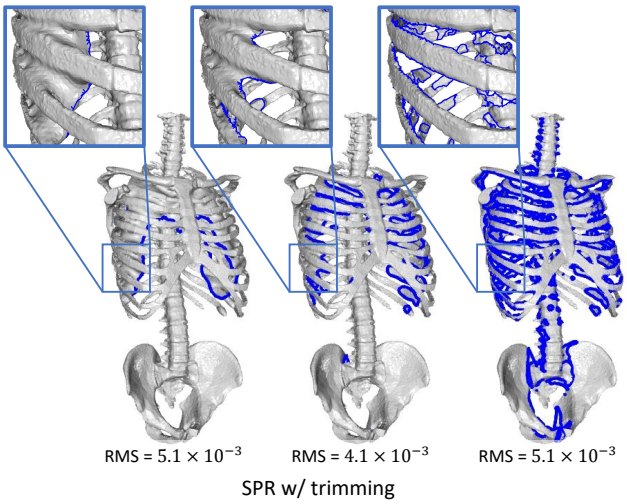

Figure 2: *Results of Screened Poisson Reconstruction after trimming of the surface in regions of low sampling density, for different sampling density thresholds. (Boundary edges of the triangle meshes are drawn in blue.)*

A second approach is to leverage the fact that the input point-set is structured. Specifically, that it is most often obtained using 3D range scanners. Assuming opaque surfaces, this not only provides information about where the surface is, but also where it cannot be: if a point is seen in the range scan, then the reconstructed surface should not intersect the line of sight from the sensor to the point. That is, the reconstructed surface should lie within the object's depth hull [BGM06] (or equivalently, ray hull [ACCS04]).

The presence of line-of-sight information has motivated *carving* techniques which have been integrated within local implicit methods [CL96] and have been used in post-processing to modify the solution of a global system [SSZCO10]. However, there have been no methods to date integrating line-of-sight constraints within a global solver, simultaneously providing the benefits of space carving and the robustness of global methods.

Key Idea Our insight is that when the implicit function representing the reconstructed surface is an *indicator* function (i.e. with value zero outside of the surface and value one inside), line-of-sight constraints can be straightforwardly formulated as Dirichlet constraints on an enclosing envelope. In contrast, the signed-distance function does not admit such constraints as the values of the implicit function within the enclosing envelope depend on the distance from the yet-to-be-determined surface. (In principle, one could impose linear inequality constraints – that the values within the envelope should be greater than zero – but this would require solving a quadratic programming problem and would be significantly less efficient.)

Approach To this end, we adapt the Screened Poisson Reconstruction method [KH13, KH19] to take as additional input a watertight mesh that envelops the space within which the reconstructed surface must lie. We rasterize the envelope mesh into the adapted octree and perform a flood-fill so as to identify octree leaf nodes that are strictly exterior to the surface. We then adapt the finite-element basis such that the basis functions' supports do not overlap exterior nodes, so as to obtain an implicit function that is guaranteed, by construction, to vanish outside the envelope surface.

2. Related Work

Reconstructing surfaces from point samples has been an important problem in the graphics and vision community for more than three decades. While there are numerous approaches to solving this problem, including techniques using surface-fitting [TV91, CM95], spectral segmentation [KSO04], and computational geometry techniques [Boi84, EM94, BBX95, BMR*99, ABK98, ACK01], we focus this review on implicit approaches. These approaches reconstruct the surface by first fitting an implicit function to the input point-set and then extracting an isosurface of the implicit function. (A more general review of surface reconstruction techniques can be found, for example, in the recent survey of Berger *et al.* [BTS*17].)

2.1. Implicit Surface Reconstruction

Implicit surface reconstruction approaches can be broadly decomposed into *local* and *global* techniques.

Local Methods Local methods proceed by fitting a compactly supported implicit function to the point samples and then blending the local implicit functions into a single solution. Most often, these approaches locally fit a signed distance function to the data using the points' normals to define a linear [HDD*92] or higher-order [FG14] function, using line-of-sight information from the scanner [CL96], or using local polynomial fitting [ABCO*01, OBA*03, SOS04].

These methods tend to have the advantage of being remarkably fast and often support discretizations over adapted data-structures. However, in only using local information to define the reconstruction, these methods can be more sensitive to outliers, noise, and missing data.

Global Methods In contrast, global techniques tend to provide more robustness by using *all* of the input data to define the value of the implicit function at a given point. This can be done either by

using a simple pointwise evaluation but using globally supported functions [JKSH13, LSSW18] or by choosing a function basis and representing the implicit function as a solution to a global linear system with respect to this basis [CBC*01, KBH06, MPS08, CT11, KH13, KH19].

Because naïvely computing the implicit functions can result in a reconstruction algorithm that is too slow to be useful, many global methods leverage a hierarchical structure such as the fast multipole method [EMRV92] or multigrid [BHM00] to obtain a solution efficiently and with a low memory footprint.

3. Our Approach

Given an oriented point-set $\mathcal{P} = \{(p_i, n_i)\}$ and an enveloping surface $\mathcal{E}$, our goal is to reconstruct a surface $\mathcal{S}$ that fits the point-set while satisfying the constraint that $\mathcal{S}$ is strictly inside $\mathcal{E}$. We do this by adapting the Screened Poisson Reconstruction algorithm.

Reviewing Screened Poisson Reconstruction

Given the oriented point-set $\mathcal{P}$, Screened Poisson Reconstruction uses $\mathcal{P}$ to define an indicator function $\chi : \mathbb{R}^3 \to \mathbb{R}$ (i.e., with value zero outside the surface and value one inside), and then extracts the reconstructed surface as the level-set $\mathcal{S} = \chi^{-1}(\frac{1}{2})$.

The indicator function is obtained by first transforming the oriented point-set into a compactly supported vector field $\vec{V}$ and then solving for the function χ whose gradient matches $\vec{V}$ *and* whose value at the input samples is close to 0.5. That is, the function χ minimizes the energy:

$$E(\chi) = \int \left\| \vec{V}(p) - \nabla\chi(p) \right\|^2 dp \; + \; \alpha \sum_{p \in \mathcal{P}} (\chi(p) - 0.5)^2,$$

with α the screening weight.

This system is solved by adapting an octree $\mathcal{O}$ to the point-set and using B-splines $\{B_o\}_{o \in \mathcal{O}}$ centered at the octree nodes/corners to discretize the space of functions. Finding the minimizer χ then reduces to solving a sparse, symmetric, positive-definite linear system on the coefficients of the B-splines, which is achieved efficiently using a multigrid solver defined over the hierarchical structure of the octree.

Adapting Screened Poisson Reconstruction

Noting that, by construction, points at which χ is equal to zero are exterior to the surface, we adapt the Screened Poisson Reconstruction to support the constraint that the reconstructed surface $\mathcal{S}$ is inside the envelope $\mathcal{E}$ by enforcing the Dirichlet constraint that χ vanishes outside the envelope. We do this by redefining the space of functions $\{B_o\}$ to consist of functions whose support is wholly contained within the interior of $\mathcal{E}$.

4. Implementation

Implementing Screened Poisson Reconstruction with Dirichlet constraints requires addressing two challenges. We need to identify the leaf nodes of the octree that are wholly exterior to the envelope and we need to adapt the finite-elements discretization so that the basis functions are not supported on the exterior leaf nodes. We begin by describing how the finite-elements discretization is adapted and then describe how to identify the exterior leaf nodes.

4.1. Adapting the finite-elements discretization

Assume that we have identified the exterior leaf nodes. To enforce Dirichlet boundary constraints, we require that the supports of the B-splines, $\{B_o\}$, do not overlap the cube cells of leaf nodes designated *exterior*. This can be done in one of two ways. The simplest approach is simply to remove every basis function B_o whose support overlaps an exterior leaf node. Alternatively, basis functions can be "reshaped" so their support no longer overlaps the exterior nodes [HRW01].

In our implementation we use a hybrid approach. Starting with an octree adapted to the points' sampling density, we fully refine the octree up to depth d_C. We reshape the finite elements within the coarse levels of the tree (at depths less than d_C). And, within the fine levels of the tree (at depth d_C or greater), we remove exterior-overlapping finite elements.

Finer discretization

For the adaptive grid at depth d_C or finer, we remove all basis functions whose support overlaps an exterior node, allowing us to use a simple Laplacian stencil to define the linear system at finer depths.

Coarser discretization

For the regular grid at depths coarser than d_C, we implicitly "reshape" the basis functions by recursively defining coarser basis functions as linear combinations of finer basis functions whose support does not overlap the exterior leaf nodes.

We recall that on a regular grid discretized with B-splines, the Galerkin method relates the coarser ($\mathbf{A}^C$) and finer ($\mathbf{A}^F$) linear systems by:

$$\mathbf{A}^C = \mathbf{P}^\top \cdot \mathbf{A}^F \cdot \mathbf{P},$$

where $\mathbf{P}$ is the prolongation operator expressing coarser B-splines as a linear combination of finer ones.

Although we remove basis functions supported on exterior nodes at depth d_C, we can still use the Galerkin method to obtain a system at depth $d_C - 1$ by zeroing out rows in the prolongation matrix corresponding to the removed B-splines at depth d_C. At coarser levels, basis functions are no longer removed so we can directly apply the Galerkin method.

Figure 3 visualizes the effects of the choice of basis by considering a 1D problem where we would like to reconstruct the indicator function of an interval from the positions and orientations of its two endpoints. For this problem, we use a fully refined tree of depth 5, first order B-splines as our finite elements, and a constraint envelope which is a dilation of the interval by a factor of 1%.

Figure 3 (left) shows the five finite element functions at depth 2. Without envelope constraints (top), we have the standard hat basis

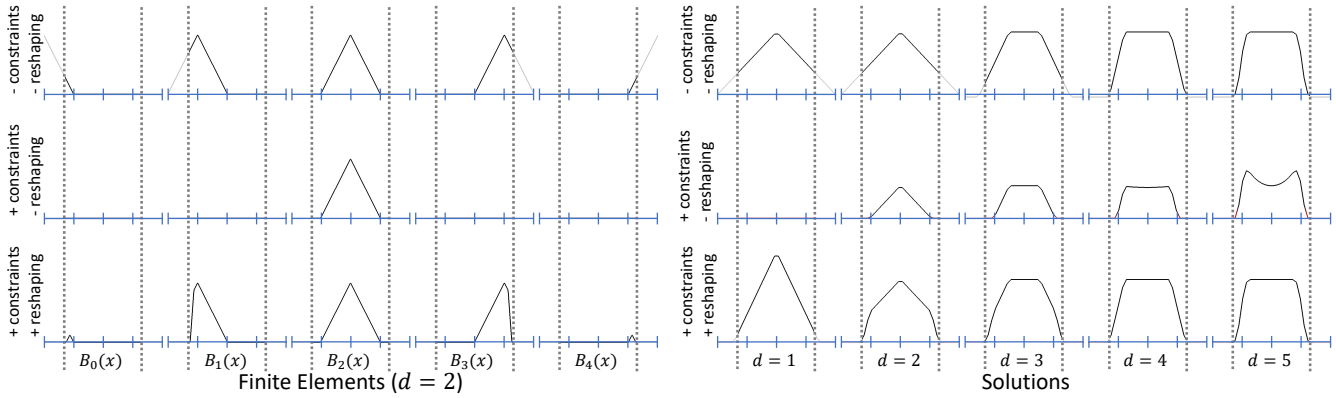

Figure 3: *Comparison of different 1D B-spline finite elements, $B_0, B_1, \ldots, B_4$, at depth $d = 2$ (left) and solutions at progressively finer depths d (right) without envelope constraints (top), with envelope constraints but without reshaping (middle), and with envelope constraints and reshaping (bottom). The two endpoints of the 1D envelope interval are denoted by the dotted lines.*

functions. Introducing envelope constraints but no reshaping (middle), we either have the original hat basis functions (if they are supported inside the envelope) or the zero function. Using envelope constraints and reshaping (bottom), the finite element functions resemble a continuous truncation of the hat basis functions to the interior of the envelope.

Figure 3 (right) shows the approximate solutions obtained after solving (in a coarse-to-fine manner) up to depths 1, 2, 3, 4, and 5, using four Gauss-Seidel relaxations per level. Without envelope constraints (top), we obtain a progressively sharper approximation of the indicator function, though the reconstructed function is non-zero (slightly negative) outside the envelope. With envelope constraints but without reshaping (middle), we obtain a function that is guaranteed to be zero exterior to the envelope. However, the solution obtained at coarser resolutions is a dampened version of the true solution, resulting in an undesirable dip in the values of the finer solution near the middle of the interval. With envelope constraints and reshaping (right), we obtain solutions similar to those obtained without envelope constraints but now constrained to be zero outside the envelope.

4.2. Identifying exterior leaf nodes

Given the envelope $\mathcal{E}$ we identify exterior leaf nodes by rasterizing the envelope into the octree, flood-filling an *exterior* node designation, and then eroding the designation to ensure that the vector field $\vec{V}$ is supported away from *exterior* nodes.

Rasterization of envelope

Given the triangle mesh $\mathcal{E}$ and a target rasterization depth $d_{\mathcal{E}} \geq d_C$ we rasterize each triangle $t \in \mathcal{E}$ into the octree by identifying the set of octree nodes at depth $d_{\mathcal{E}}$ that intersect t (refining the octree as needed to ensure the nodes exist), clipping the triangle to each of the intersecting nodes, and pushing the clipped triangle fragment into a vector stored with the node.

We do this efficiently by first identifying the finest depth at which the three vertices of the triangle are contained within a single node. Then, we split the triangle by the three axis-aligned planes passing through the center of the node and recursively pass each triangle fragment to the appropriate child node for splitting at the next level of the octree hierarchy. (We implement this in parallel over the different triangles of the envelope using a mutex to protect the vector of fragments stored with each node.)

Flood-fill of exterior designator

Given the list of fragments stored within each octree node at depth $d_{\mathcal{E}}$ we flood-fill the octree, designating those leaf nodes that are wholly exterior to the envelope. We do this in three steps.

First, we iterate over all nodes at depth $d_{\mathcal{E}}$ which contain fragments. For each such node we use ray-tracing in conjunction with the fragment normals to designate the center of each face of the cell as either *interior* or *exterior*. To do this, we construct a ray emanating from the center of the face and passing through the center of one of the fragments. We intersect the ray with all triangle fragments inside the cell and compute the first point of intersection. If the normal of the first intersecting fragment points in the same direction as the ray we designate the center of the face as interior, otherwise we designate the center of the face exterior. Then, if the center of the face is exterior and the finest face-adjacent neighbor (at depth $d_{\mathcal{E}}$ or coarser) does not contain triangle fragments, we mark the neighboring node as *exterior* and add it to a queue.

Next, we flood-fill the *exterior* designation by popping an exterior node off of the queue and considering the node's neighbors. For each neighbor, we test if (1) it, or its descendents at depth $d_{\mathcal{E}}$, is empty of fragments and (2) it has not been designated *exterior*. If a neighbor is both empty and un-designated we mark it *exterior* and add it to the queue. We repeat this process until the queue is empty. (Note that if a node does not contain geometry, its faces must either all be interior or all be exterior. Thus if it is adjacent to a node that is entirely exterior, it too must be entirely exterior.)

Finally, as the rasterization depth $d_{\mathcal{E}}$ need not be the depth of the octree and since we may insert nodes at depth coarser than $d_{\mathcal{E}}$ into the queue, we push the *exterior* designators to the leaf nodes. We do this by iterating over the nodes of the tree in a depth-first order, identifying nodes marked as *exterior*, and propagating that designation to the children when the node is not a leaf.

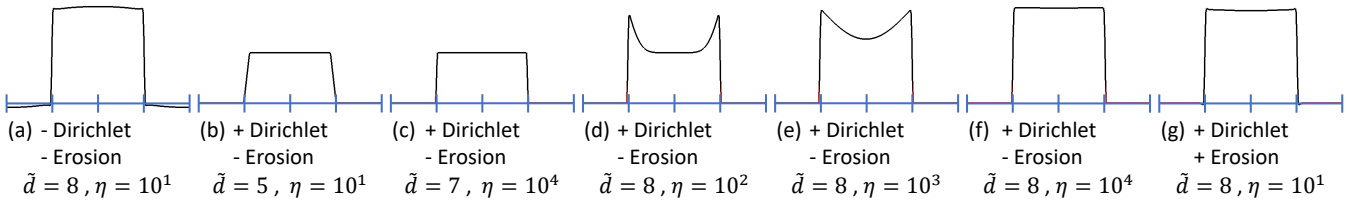

Figure 4: *Implicit functions obtained in the 1D reconstruction from just two sample points at the endpoints of a segment without using an envelope to define Dirichlet constraints (a), using an envelope but without eroding the* exterior *designator to ensure that the target vector field can be represented (b-f), and using an envelope and eroding (g). The implicit function is discretized over a fully refined binary tree of depth $d = 8$ and the images show results solving up to different depths ($\tilde{d}$) and using different numbers of Gauss-Seidel iterations (η).*

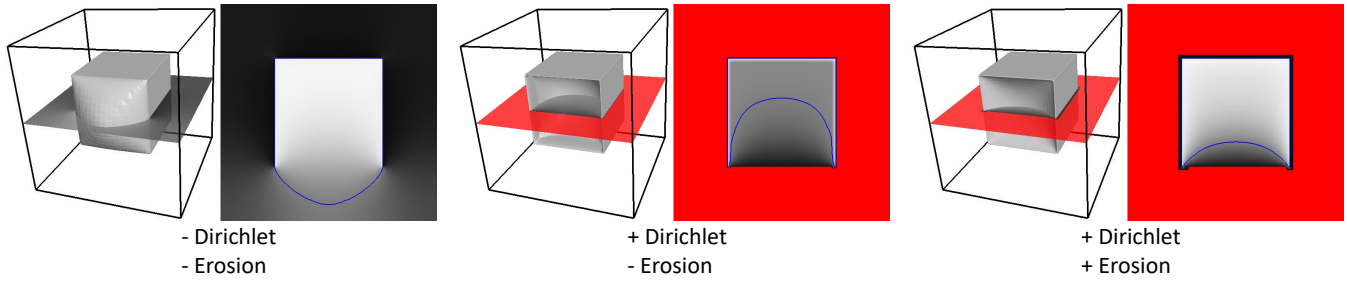

Figure 5: *Visualization of the reconstructions of a point-set obtained by uniformly sampling five of the six faces of a cube without using an envelope to define Dirichlet constraints (left), using an envelope but without eroding the* exterior *designator to ensure that the target vector field can be represented (middle), and using an envelope and eroding (right).*

Limitation We note that our rasterization assumes that the envelope $\mathcal{E}$ is represented by an embedded, manifold, triangle mesh. Furthermore, while we have found that our naïve implementation works well in practice, numerical imprecision could result in the assignment of an incorrect label to a face center, which would then be propagated by the flood-filling.

Alternative designation of exterior nodes Our implementation of the finite-elements discretization (Section 4.1) involves a general functor that takes as input a leaf node and returns a boolean value indicating if the node is *exterior*. This allows for flexibility in representing the envelope. For example, in the case of a set of range scans with associated camera parameters, the functor can project a node point into each range image to determine if it lies in front of a visible surface.

Erosion of exterior

We observe that using an envelope constraint that is too tight, i.e. too close to the sample points, can remove the finite-elements at depths d_C or greater that are needed to represent the target vector field $\vec{V}$. Without these, $\vec{V}$ is effectively cropped and the resulting implicit function has value less than one in the interior. Because this effect is more significant at coarser levels of the hierarchy, where the B-splines supporting $\vec{V}$ have larger support and are more likely to be cropped, the multigrid solver fails to provide a good approximate solution at coarser resolutions, leading to poor convergence.

To address this, we erode the exterior designator so that the envelope contains the support of $\vec{V}$. The constraint vector field, $\vec{V}$ is represented as a linear combination of the B-splines, $\{B_o\}$, with a 3D vector associated to each B-spline. For an octree node $o \in \mathcal{O}$ at depth d, we determine if the associated vector coefficient is non-zero. If it is, we identify the octree nodes at depth d_C contained in the support of B_o and remove the *exterior* designation from these nodes and all their descendants.

Figure 4 demonstrates the need for eroding in the 1D case where the input point-set consists of the two endpoints of a line segment. For these reconstructions we use a fully refined (binary) tree up to depth $d = 8$, but only perform the basis function reshaping up to depth $d_C = 5$. We show results when solving the system up to depth $\tilde{d}$ using η Gauss-Seidel relaxations per level. (a) Without Dirichlet constraints and setting $\tilde{d} = 8$ and $\eta = 10$, the implicit function provides a good approximation to the indicator function, evaluating to a value slightly less than zero outside the segment and a value close to one in the interior. (b) Using Dirichlet constraints and not eroding the *exterior* designator, with $\tilde{d} = 5$ and $\eta = 10$, the implicit function is piecewise constant and vanishes outside the envelope. However, the cropping of $\vec{V}$ at coarser resolutions dampens the function so that it evaluates to a value close to one half in the interior. (c) Again, using Dirichlet constraints and not eroding, but now setting $\tilde{d} = 7$ and $\eta = 10^4$, the vector field $\vec{V}$ remains cropped and the implicit function, though sharpened at the endpoints, continues to evaluate to a value close to one half in the interior. (d-f) Using Dirichlet constraints and not eroding the *exterior* designator, this time with $\tilde{d} = 8$ and increasing numbers of Gauss-Seidel iterations, $\vec{V}$ is well-represented at the finest depth. Combined with the use of a fully refined tree, the result eventually converges to a function resembling the indicator function. However, this takes many iterations as the solution from depth 7 does not provide a good approximation to the indicator function. (g) Using Dirichlet constraints and eroding the *exterior* designator, $\vec{V}$ is well-represented

at all depths and the multigrid solver quickly converges to a good solution.

Figure 5 shows a more challenging 3D case where the point-set is obtained by uniformly sampling five of the six faces of a cube. The system is discretized over an adapted octree of depth $d = 9$ which is fully refined up to depth $d_C = 5$. (The solutions are obtained by solving up to the finest depth and using eight Gauss-Seidel iterations per level.) In addition to the missing data, this is a more difficult reconstruction problem because the adaptivity of the octree implies that if the coarser solve fails to provide a good approximation to the indicator function, there may not be an opportunity to correct the solution at finer resolutions.

For this visualization, we computed reconstructions without using an envelope (left), using a dilation of the original cube by 1% as an envelope but without eroding the *exterior* designator, and using a dilation of the original by 1% followed by an erosion of the *exterior* designator. Each of the results shows a 2D slice through the 3D implicit function as well as the level set $\mathcal{S} = \chi\left(\frac{1}{2}\right)$. Pixels in the slice are colored by clamping the values to the range $[-0.1, 1.1]$ and using gray scale. Pixels in regions marked *exterior* are drawn in red. (Note that the slice is aligned so that the bottom intersects the unsampled face of the cube. This results in an indentation of the Dirichlet region in the rightmost image – since the target vector field $\vec{V}$ vanishes near the bottom, there is no erosion of *exterior* designators.)

As the figure shows, without an envelope we obtain a reconstruction that balloons out in regions of missing data. Incorporating the dilated cube as an envelope but not eroding the *exterior* designator causes much of the target vector field $\vec{V}$ to fall into the *exterior* region and hence be discarded. The resulting surface is guaranteed not to extend into the region marked *exterior* but the implicit function only attains values close to one near the point samples, where the Laplacian constraint is non-zero and Gauss-Seidel relaxation at the finer depths corrects the solution. In the interior, where the coarse solution is too small due to the cropping of the target vector field and Gauss-Seidel relaxation cannot correct the solution (because the number of Gauss-Seidel relaxations is small and the adapted octree is not refined in the interior) the implicit function has values close to one half. Using the envelope and eroding we obtain an implicit function that behaves like a discontinuous indicator function near the five sampled faces and transitions smoothly (harmonically) from zero to one across the missing face.

Erosion extent In eroding the exterior, our goal is to preserve the basis functions required to represent the constraint vector field $\vec{V}$ at coarser depths. In principal, this would require ensuring that any B-Spline at depth d_C or finer whose support overlaps the support of $\vec{V}$ is not removed (since the system constraints are obtained by integrating $\vec{V}$ against the gradients of the B-splines). In practice however, we find that it is sufficient to ensure that nodes at depth d_C overlapping the support of $\vec{V}$ (and their descendants) are not marked *exterior*.

The full implementation of Dirichlet constraints within the screened Poisson reconstruction framework can be found at: https://github.com/mkazhdan/PoissonRecon/.

5. Results and Discussion

We begin by providing a motivation for the design decisions and parameter choices made in incorporating the envelope constraints within the Screened Poisson Reconstruction Algorithm. To support ground-truth comparisons, we use data obtained by virtually scanning 3D meshes. We also consider the real-world application of our approach by using the BigBIRD dataset [SSN*14]. We conclude this section by discussing a difficulty that arises when the envelope is too tight.

5.1. Motivation

In adapting the Screened Poisson Reconstruction algorithm to support envelope constraints, we have made general design decisions as well as particular parameter choices. We begin by motivating these.

Incorporating the envelope information While we have shown that it is possible to realize the envelope constraints as a Dirichlet boundary condition, it is natural to consider whether there are not easier ways of using the envelope. For example, one could run the unconstrained reconstruction algorithm and then either (1) trim the output surface to the envelope, or (2) clamp the implicit function so that voxels outside the envelope are set to zero before isosurfacing. (The latter is similar to Cone Carving [SSZCO10] which first solves for an approximate signed distance function and then merges the approximation with the distance function to the surface defined by the union of visibility cones.)

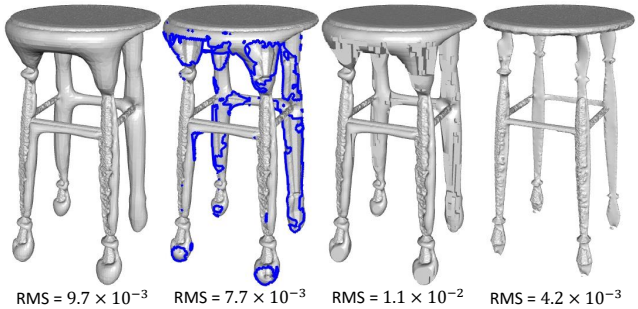

Figure 6: *Reconstruction of the* `stool` *without using the hull information, using the hull information to trim the reconstructed surface, using the hull information to clamp the implicit function, and using the hull information to prescribe envelope constraints.*

Figure 6 compares these alternatives for the `stool`, virtually scanned from three different positions. The unconstrained reconstruction (first column) exhibits the usual "ballooning" in regions of missing data. Trimming to the envelope (second column) and clamping the values of the implicit function outside the envelope to zero (third column) both suffer from the fact that the envelope is not guaranteed to lie close to the surface. Furthermore, trimming produces surfaces with boundaries while clamping causes sharp discontinuities revealing the underlying octree discretization. Our approach, which incorporates the envelope as a Dirichlet constraint

(fourth column) faithfully reproduces the surface in sampled regions and closes off the surface in regions of missing data without introducing unwanted "ballooning".

Choosing regular-grid and rasterization depths Choosing a regular-grid depth, d_C, requires trading-off between accuracy and performance. On the one hand, making the octree regular up to a finer depth results in less erosion of the envelope. This results in tighter constraints on the reconstruction and enables the use of envelope constraints with finer features. On the other hand, increasing d_C also means that the solver needs to store and solve a system on the order of $O(2^{d_C} \times 2^{d_C} \times 2^{d_C}) + O(2^d \times 2^d)$, with d the depth of the tree. In contrast, using a fully adapted octree would give memory usage and run-times on the order of $O(2^d \times 2^d)$.

In principle, choosing a finer rasterization depth, $d_{\mathcal{E}}$, would let us more accurately represent the constraint envelope. However, the erosion of the *exterior* designators required for representing the target vector field effectively removes the finer details of the envelope (particularly in regions where the envelope is close to the point samples) resulting in only negligible improvement with increased rasterization depth.

Figure 7 reconstructions of the `hexi_pot` model [ZJ16], with a point-set and depth hull obtained by virtually scanning the model from eight (regularly sampled) positions on the view sphere. The figure shows the reconstruction results as the regular-grid depth and rasterization depth are increased. As the figure shows, using values of d_C and $d_{\mathcal{E}}$ that are too low results in significant erosion of the envelope, allowing the reconstruction to balloon into regions which the original envelope had identified as exterior. Increasing the values of d_C and $d_{\mathcal{E}}$ mitigates this problem, reducing the erosion and forcing the reconstruction to be more tightly constrained.

Table 1 shows the performance overhead for reconstructing the `hexi_pot` using the envelope constraints, as the regular-grid depth and rasterization depth are increased. (Without envelope constraints, the surface is reconstructed in 143 seconds using 3740 megabytes of RAM.)

For smaller values of regular-grid depth ($d_C \leq 6$), the computation does not bottleneck on the generation and solution of the regular-grid system. Specifically, there is an overhead in terms of both time ($\sim 10\%$) and memory ($\sim 5\%$), but that overhead is roughly constant. However, as the depth of the regular grid is increased ($d_C \geq 7$), the complexity of defining and solving a regular system starts to dominate, resulting in significant increases in both running times and memory usage.

Increasing the rasterization depth, $d_{\mathcal{E}}$, we see a small increase in running time. (This is due to the added cost of managing mutual exclusion when rasterizing into an adapted octree.) We also see an increase in memory usage, particularly at finer rasterization depths. (This is because at finer depths, triangles are larger than the tree nodes, requiring more clipping, thereby increasing the number of geometry fragments generated.)

Quantitative evaluation Reconstructing surfaces from virtual scans let us use the ground-truth original surface to quantitatively evaluate the benefit of incorporating the depth hull as a Dirichlet constraint. To this end, we use the Metro tool [CS98] to compute the symmetric root mean squared (RMS) distance between the ground truth and the reconstructed model. We provide the RMS values, normalized to the size of the bounding box of the model, for the `skeleton` (Figures 1 and 2; input data generated using 12 virtual scans), `stool` (Figure 6; input data generated using 3 virtual scans), and `hexi_pot` models (Table 2; input data generated using 8 virtual scans).

Figure 7: *Comparison of the reconstructions obtained using a Dirichlet envelope with increasing rasterization depth, $d_{\mathcal{E}}$, and increasing regular-grid depth, d_C.*

As the RMS values demonstrate, incorporating the depth hull as a Dirichlet constraint produces higher quality reconstructions, compared to the original Screened Poisson Reconstruction and the alternatives of either trimming the surface or clamping the implicit function. Examining the `hexi_pot` model, we also find that while the quality of the reconstruction initially improves as the rasterization depth ($d_{\mathcal{E}}$) is increased, the improvements become less significant by depth $d_C = d_{\mathcal{E}} = 6$. In fact, the table shows that the quality degrades somewhat when the regular-grid depth and rasterization depth become very high. (See Section 5.3 for a discussion of this phenomenon.)

5.2. Real-world data

We evaluate our approach on models from the BIGBird dataset [SSN*14]. Each model in the dataset was obtained by placing an object on a turntable at 120 different poses and imaging from five RGB-D camera pairs. (The RGB and depth cameras imaged the scene at resolutions 1280×1024 and 640×480 respectively).

	$d_{\mathcal{E}}=4$	$d_{\mathcal{E}}=5$	$d_{\mathcal{E}}=6$	$d_{\mathcal{E}}=7$	$d_{\mathcal{E}}=8$
$d_C=4$	+13(s) +139(MB)	+15(s) +140(MB)	+17(s) +141(MB)	+21(s) +148(MB)	+33(s) +226(MB)
$d_C=5$		+12(s) +140(MB)	+16(s) +139(MB)	+20(s) +146(MB)	+31(s) +265(MB)
$d_C=6$			+13(s) +140(MB)	+19(s) +152(MB)	+31(s) +337(MB)
$d_C=7$				+20(s) +201(MB)	+35(s) +588(MB)
$d_C=8$					+72(s) +2513(MB)

Table 1: *Performance overhead for imposing Dirichlet envelope constraints when reconstructing the* `hexi_pot` *model. The table gives the increase in running time and memory usage when reconstructing the surface at depth $d=10$ using the Dirichlet envelope constraints with different regular-grid depths, d_C and different rasterization depths $d_{\mathcal{E}}$. The input consists of a point-set with 1.3×10^7 samples and a depth hull with 1.6×10^6 triangles. Without the envelope constraints, the surface is reconstructed in 143 seconds, using 3740 megabytes of RAM.*

	$d_{\mathcal{E}}=4$	$d_{\mathcal{E}}=5$	$d_{\mathcal{E}}=6$	$d_{\mathcal{E}}=7$	$d_{\mathcal{E}}=8$
$d_C=4$	8.1×10^{-3}	3.4×10^{-3}	2.6×10^{-3}	2.4×10^{-3}	2.4×10^{-3}
$d_C=5$		3.1×10^{-3}	2.1×10^{-3}	2.0×10^{-3}	2.0×10^{-3}
$d_C=6$			2.0×10^{-3}	2.0×10^{-3}	2.2×10^{-3}
$d_C=7$				2.0×10^{-3}	2.2×10^{-3}
$d_C=8$					2.3×10^{-3}

Table 2: *Reconstruction quality as a function of the regular-grid depth d_C and rasterization depth $d_{\mathcal{E}}$.*

Background-foreground segmentation was also performed on the RGB images, resulting in a binary mask for each RGB image.

As input to our reconstruction algorithm, we computed the input point-set using the depth images. We assigned a normal by finding the best-fit plane to the depth samples within an eight pixel radius and chose the sign so that the normal was front-facing to the camera. We assigned colors to the points from a depth image by back-projecting and sampling the corresponding RGB image. Points were discarded if they did not back-project into an occupied pixel in the segmentation mask.

We computed the depth hull using the depth images and segmentation mask. To do this conservatively, we dilated the segmentation masks and depth images. This ensured that the derived depth hull encompassed the point-set without fitting too tightly and allowed for some imprecision in the depth measurements, the segmentation, and the estimated camera parameters. We dilated the segmentation masks using a radius of eight pixels. For the depth images, we replaced the depth value at a pixel with the minimum depth value within an eight pixel radius. Then, we computed the depth hull by back-projecting each voxel in a $256\times256\times256$ grid into each of the camera pairs and counting the number of camera pairs for which the voxel (1) back-projected to an RGB pixel that was interior to the segmentation mask and (2) had a depth value at least 0.99 times the value of the back-projected depth pixel. Finally, we used marching cubes [LC87] to extract the level-set corresponding to the voxels considered to be exterior by at least 10% of the camera pairs.

Figure 8 compares the results of Screened Poisson Reconstruction at depth 10 with Neumann boundary constraints on the surface of the bounding cube, Dirichlet boundary constraints on the surface of the bounding cube, and Dirichlet constraints on the depth hull envelope, for the `detergent`, `mahatma_rice`, and `paper_cup_holder` models in the dataset.

As the models were placed on a turntable, the bottoms were inaccessible to the scanner, resulting in missing samples. Using traditional Screened Poisson Reconstruction generates a "bulging" reconstruction in these regions. While Dirichlet constraints on the

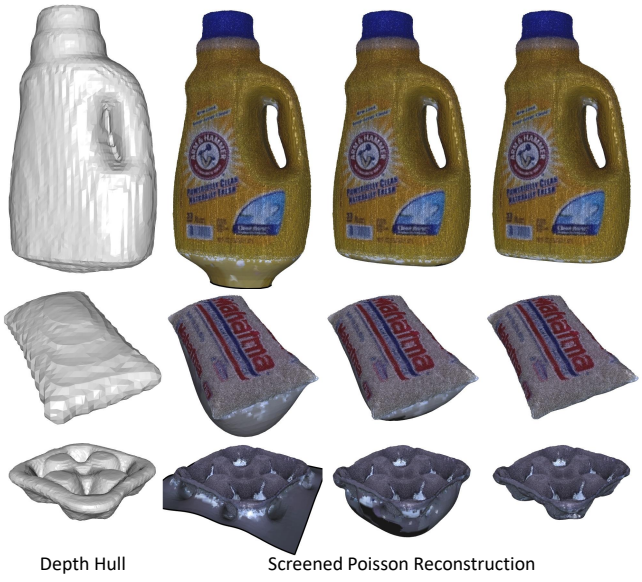

Figure 8: *Comparison of Screened Poisson Reconstructions with different boundary conditions including Neumann constraints on the boundary of the bounding cube, Dirichlet constraints on the boundary of the bounding cube, and Dirichlet constraints on the depth hull envelope.*

surface of the bounding cube guarantee that the reconstruction will close off in the interior of the cube, there is little control over where the closing will happen. The situation is worse with Neumann constraints on the surface of the bounding cube because the reconstructed surface is not even guaranteed to close off in the interior of the bounding cube. In contrast, by imposing Dirichlet constraints on the exterior of the depth hull, we constrain the reconstruction to close the surface within a region known to contain the correct solution. We note that in regions that are well-sampled, the reconstruction with Dirichlet constraints on the exterior of the depth hull is

indistinguishable from the reconstruction without, confirming that the introduction of Dirichlet constraints is compatible with the original Poisson Reconstruction.

5.3. Limitation of a Tight Envelope

Our approach provides a way to force the reconstructed surface to stay within a constraint envelope. However, as Figure 5 (right) shows, this can come at the cost of having the reconstructed surface recede from the true solution. One way to understand this behavior is from the perspective of solving Laplace's equation.

In the case of the cube where only five of the six faces are sampled, we can think of the solution in the interior of the cube as approximating the harmonic function that is equal to one on the five sampled faces and zero on the unsampled one. The function is softly constrained to be equal to one at the five faces through the interpolation constraints and the target vector field $\vec{V}$. It is constrained to equal zero near the other face through the Dirichlet constraints. And, the solution is harmonic because the target vector field is zero in the interior of the cube. As a result, the implicit function transitions smoothly from zero to one near the unsampled face, making the 0.5 isosurface smooth and concave in this region.

The recession of the surface may be mitigated by dilating the constraint envelope, thereby pushing the isosurface further out. In our implementation, this can be done simply by using a lower rasterization depth, $d_{\mathcal{E}}$, since our approach only marks nodes as *exterior* if they do not contain any geometry. However, the choice of a good dilation radius is likely to be problem-specific and we do not have a good general solution at this time.

The effects of dilating the envelope are also evidenced in Table 2. Initially, we see the expected improvement in reconstruction quality as the rasterization depth is increased, since this generates more accurate Dirichlet constraints. However, the RMS error starts to increase again when using a fine rasterization depth of $d_{\mathcal{E}} = 8$.

6. Conclusion and Future Work

This work presents a new approach for incorporating spatial occupancy information like line-of-sight and visual hull to constrain the result of surface reconstruction. In the context of the Screened Poisson Reconstruction algorithm, which solves the reconstruction problem by computing the indicator function, we show that envelope constraints can be expressed as Dirichlet boundary constraints. By adapting the Screened Poisson Reconstruction algorithm to support these types of envelope constraints, we obtain noticeably improved reconstructions at negligible costs in running time and memory usage.

In supporting Dirichlet envelope constraints, we have designed an implementation that either excludes finite elements if their support overlaps the constraint region (at finer resolutions), or reshapes the finite elements (at coarser resolutions where the octree is fully regular). In the future, we would like to explore whether there is a hybrid approach that allows for reshaping the finite-elements without requiring a fully regular grid, allowing us to maintain a run-time complexity that is quadratic in the resolution instead of cubic, while still supporting a finite-elements discretization that better adapts to the constraint region.

In addition to surface reconstruction, the efficient solution of the Poisson equation has applications in a number of other domains. With the ability to support Dirichlet boundary conditions, we would like to explore applications in the simulation of incompressible fluids where the no-slip condition at fluid-solid interfaces is realized using Dirichlet constraints.

Acknowledgements

We thank the anonymous reviewers for their valuable comments and suggestions, and Thingi10K, Aim@Shape, and BIGBird for sharing data. This work was sponsored in part by NSF Awards 1422325, 1617236, and 1815070.

References

[ABCO*01] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., LEVIN D., SILVA C. T.: Point set surfaces. In *Proceedings of the Conference on Visualization '01* (2001), p. 21–28. 2

[ABK98] AMENTA N., BERN M., KAMVYSSELIS M.: A new Voronoi-based surface reconstruction algorithm. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (1998), pp. 415—-421. 2

[ACCS04] AHN H.-K., CHENG S.-W., CHEONG O., SNOEYINK J.: The reflex-free hull. *International Journal of Computational Geometry & Applications 14*, 06 (2004), 453–474. 2

[ACK01] AMENTA N., CHOI S., KOLLURI R. K.: The power crust, unions of balls, and the medial axis transform. *Comput. Geom. Theory Appl. 19*, 2–3 (2001), 127—-153. 2

[BBX95] BAJAJ C. L., BERNARDINI F., XU G.: Automatic reconstruction of surfaces and scalar fields from 3D scans. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques* (1995), pp. 109—-118. 2

[BGM06] BOGOMJAKOV A., GOTSMANN C., MAGNOR M.: Free-viewpoint video from depth cameras. In *Proceedings of the Vision, Modeling and Visualization Workshop (VMV)* (2006), pp. 89–96. 2

[BHM00] BRIGGS W. L., HENSON V. E., MCCORMICK S. F.: *A Multigrid Tutorial (2nd Ed.)*. Society for Industrial and Applied Mathematics, USA, 2000. 3

[BMR*99] BERNARDINI F., MITTLEMAN J., RUSHMEIER H., SILVA C., TAUBIN G.: The ball-pivoting algorithm for surface reconstruction. *IEEE Tran. Vis. Comp. Graph. 5*, 4 (1999), 349–359. 2

[Boi84] BOISSONNAT J.-D.: Geometric structures for three-dimensional shape representation. *ACM Trans. Graph. 3*, 4 (1984), 266—286. 2

[BTS*17] BERGER M., TAGLIASACCHI A., SEVERSKY L. M., ALLIEZ P., GUENNEBAUD G., LEVINE J. A., SHARF A., SILVA C. T.: A survey of surface reconstruction from point clouds. *Computer Graphics Forum* (2017), 301–329. 2

[CBC*01] CARR J. C., BEATSON R. K., CHERRIE J. B., MITCHELL T. J., FRIGHT W. R., MCCALLUM B. C., EVANS T. R.: Reconstruction and representation of 3D objects with radial basis functions. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (2001), p. 67–76. 3

[CL96] CURLESS B., LEVOY M.: A volumetric method for building complex models from range images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (1996), pp. 303–312. 2

[CM95] CHEN Y., MEDIONI G.: Description of complex objects from multiple range images using an inflating balloon model. *Computer Vision and Image Understanding 61*, 3 (1995), 325–334. 2

[CS98] CIGNONI P., SCOPIGNO R.: Metro: Measuring error on simplified surfaces. *Computer Graphics Forum 17*, 2 (1998). 7

[CT11] CALAKLI F., TAUBIN G.: SSD: Smooth signed distance surface reconstruction. *Computer Graphics Forum 30*, 7 (2011), 1993–2002.

[EM94] EDELSBRUNNER H., MÜCKE E. P.: Three-dimensional alpha shapes. *ACM Trans. Graph. 13*, 1 (1994), 43—-72. 2

[EMRV92] ENGHETA N., MURPHY W., ROKHLIN V., VASSILOU M.: The fast multipole method for electromagnetic scattering computation. *IEEE Transactions on Antennas and Propagation 40* (1992), 634–641. 3

[FG14] FUHRMANN S., GOESELE M.: Floating scale surface reconstruction. *ACM Trans. Graph. 33*, 4 (2014), 46:1–46:11. 2

[HDD*92] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Surface reconstruction from unorganized points. *SIGGRAPH Comput. Graph. 26*, 2 (1992), 71–78. 2

[HRW01] HÖLLIG K., REIF U., WIPPER J.: Weighted extended B-spline approximation of Dirichlet problems. *SIAM J. Numer. Anal. 39*, 2 (2001), 442–462. 3

[JKSH13] JACOBSON A., KAVAN L., SORKINE-HORNUNG O.: Robust inside-outside segmentation using generalized winding numbers. *ACM Trans. Graph. 32*, 4 (2013). 3

[Kaz13] KAZHDAN M.: Screened Poisson surface reconstruction. https://github.com/mkazhdan/PoissonRecon, 2013. 2

[KBH06] KAZHDAN M., BOLITHO M., HOPPE H.: Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing* (2006), p. 61–70. 3

[KH13] KAZHDAN M., HOPPE H.: Screened Poisson surface reconstruction. *ACM Trans. Graph. 32* (2013), 29:1–29:13. 2, 3

[KH19] KAZHDAN M., HOPPE H.: An adaptive multigrid solver for applications in computer graphics. *Computer Graphics Forum 38*, 1 (2019), 138–150. 2, 3

[KSO04] KOLLURI R., SHEWCHUK J. R., O'BRIEN J. F.: Spectral surface reconstruction from noisy point clouds. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (2004), pp. 11—-21. 2

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3D surface construction algorithm. *SIGGRAPH Comput. Graph. 21*, 4 (1987), 163–169. 8

[LSSW18] LU W., SHI Z., SUN J., WANG B.: Surface reconstruction based on the modified Gauss formula. *ACM Trans. Graph. 38*, 1 (2018). 3

[MPS08] MANSON J., PETROVA G., SCHAEFER S.: Streaming surface reconstruction using wavelets. In *Proceedings of the Symposium on Geometry Processing* (2008), p. 1411–1420. 3

[OBA*03] OHTAKE Y., BELYAEV A., ALEXA M., TURK G., SEIDEL H.-P.: Multi-level partition of unity implicits. *ACM Trans. Graph. 22*, 3 (2003), 463–470. 2

[SOS04] SHEN C., O'BRIEN J. F., SHEWCHUK J. R.: Interpolating and approximating implicit surfaces from polygon soup. In *ACM SIGGRAPH 2004 Papers* (2004), p. 896–904. 2

[SSN*14] SINGH A., SHA J., NARAYAN K., ACHIM T., ABBEEL P.: Bigbird: A large-scale 3D database of object instances. In *IEEE International Conference on Robotics and Automation* (2014), pp. 509–516. 6, 7

[SSZCO10] SHALOM S., SHAMIR A., ZHANG H., COHEN-OR D.: Cone carving for surface reconstruction. *ACM Trans. Graph. 29* (2010), 150:1–150:10. 2, 6

[TV91] TERZOPOULOS D., VASILESCU M.: Sampling and reconstruction with adaptive meshes. In *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (1991), pp. 70–75. 2

[ZJ16] ZHOU Q., JACOBSON A.: Thingi10K: A dataset of 10,000 3D-printing models. *arXiv preprint arXiv:1605.04797* (2016). 7