Eurographics Symposium on Geometry Processing 2008
Pierre Alliez and Szymon Rusinkiewicz
(Guest Editors)

Volume 27 (2008), Number 5

Streaming Surface Reconstruction Using Wavelets

J. Manson, G. Petrova, S. Schaefer

Texas A&M University

Abstract

We present a streaming method for reconstructing surfaces from large data sets generated by a laser range scanner using wavelets. Wavelets provide a localized, multiresolution representation of functions and this makes them ideal candidates for streaming surface reconstruction algorithms. We show how wavelets can be used to reconstruct the indicator function of a shape from a cloud of points with associated normals. Our method proceeds in several steps. We first compute a low-resolution approximation of the indicator function using an octree followed by a second pass that incrementally adds fine resolution details. The indicator function is then smoothed using a modified octree convolution step and contoured to produce the final surface. Due to the local, multiresolution nature of wavelets, our approach results in an algorithm over 10 times faster than previous methods and can process extremely large data sets in the order of several hundred million points in only an hour.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling

1. Introduction

Creating digital models of real-world objects has a number of applications ranging from industry, where digital models of real objects are used to perform a physical simulation or to visualize a shape in a way not possible in real life, to entertainment where digitized clay models are animated for games or movies, to archeology and art, where these techniques are used to create digital repositories of artistic works (as done in the Digital Michelangelo Project [LPC*00]). Typically, the shape is acquired using a laser range scanning device such as the Cyberware Large Statue Scanner, which provides point samples from the surface of the object. The point samples are then used to build a three dimensional polygon model that approximates the shape of the object being scanned.

Reconstructing surfaces from the data produced by these scanning devices can be inherently difficult. The surface may be oversampled in some regions due to multiple, overlapping scans, typically taken in the attempt to cover the entire shape. On the other hand, cracks and crevices usually cannot be scanned and physical size limitations may prevent the scanner from scanning every portion of the shape, so the data may contain gaps and holes. Furthermore, with the advent of faster processors and cheaper storage, the amount of collected data has grown dramatically. For example, for large statues such as Michelangelo's David standing at 14 feet tall, the number of samples can easily be in the hundreds of millions to billions of points. Such large amounts of data necessitate efficient algorithms, both in terms of time and memory, to process the collected samples in a reasonable amount of time. In addition, real world data always contains noise due to sensor inaccuracies, which calls for robust and reliable techniques for handling this problem.

Recently, implicit methods, such as the level set methods pioneered by Osher [OF02] and Sethian [Set99], have gained popularity as techniques for surface reconstruction. They rely on the idea that it is much easier to work with a shape through its level set function than with the shape directly. This is because one can perform computations on levels sets of the function (i.e. surfaces) on a fixed Cartesian grid without having to parameterize the surfaces. On the other hand, in the last two decades, wavelets and other multiscale representations have had an enormous impact on image and signal processing, mainly due to their hierarchical structure and localization properties. The objective of this paper is to fuse these two techniques and create a fast, robust streaming algorithm for surface reconstruction from point cloud data.

Contributions

We present a fast, simple and efficient method that

combines the advantages of implicit methods and the multiscale structure of the wavelet/subdivision based methods to automatically reconstruct a surface from an unorganized oriented cloud of points. We use point samples from the surface of a solid $M \subset \mathbb{R}^3$ to reconstruct the indicator function χ_M of M, whose appropriate level set is an approximation to the solid M.

Our method is based on selecting the right basis to represent the function χ_M. Instead of using globally supported functions (for example the Fourier basis), we resort to a basis that consists of a hierarchy of compactly supported basis functions. Thus, each sample point only influences a small number of coefficients in the representation of χ_M, leading to a computationally efficient algorithm many times faster than previous techniques. Our method is very general in the sense that we can use any orthogonal or biorthogonal compactly supported wavelet basis. The choice of basis depends on the particular application in mind and is dictated by the user's desire for quality/smoothness of the reconstructed surface versus the speed of reconstruction. In some applications quality of the reconstructed surface is more important than the speed of the reconstruction. In others, such as the navigation of unmanned vehicles, reconstruction speed is more imperative than quality. Our technique provides a general framework for handling any of these applications simply by selecting the appropriate wavelet basis. Smoother wavelets will provide smoother surfaces but require more computation. As the support of the wavelet decreases so does the smoothness of the wavelet and, hence, the smoothness of the reconstructed surface. However, smaller support leads to a more computationally efficient algorithm.

We also utilize the hierarchical structure of the wavelet basis to develop a streaming implementation, where we keep only an octree of some small depth d_m in memory and encode subtrees corresponding to high-resolution details in a streaming fashion. This streaming technique allows the consumer to process massive data sets with complexity exceeding the available computer memory.

2. Previous Work

The problem of surface reconstruction has been well studied in the last few decades and we cannot possibly cover all contributions. Here, we discuss only some of the existing methods and refer the reader to [SS05] for a brief survey on some recent developments in this field or to [KBH06] where an excellent comparison of many reconstruction techniques is provided.

Surface reconstruction techniques fall into two main categories: explicit and implicit methods. Explicit methods are typically triangulation-based techniques. The Power Crust algorithm [ACK01], Robust Cocone [DG06] and Super Cocone [DGH01] are among the well-known examples of such methods. Streaming triangulation algorithms for surface reconstruction [BMR*99, ACA07] have also been developed in the pursuit of handling large data sets. However, since these algorithms interpolate the input data, they do not perform well in the presence of noise. Furthermore, they typically need neighboring information to create the triangulations, which makes them many times slower than most implicit methods and thus unsuitable for reconstructing extremely large data sets.

Implicit algorithms reconstruct a surface using a level-set of a function. In this case parametrization is not necessary and operations such as shape blending, offsets, deformations and others are simple to perform. Also, these methods typically approximate the input data and, hence, are more robust to noise in the input data than many triangulation-based techniques.

One approach utilizing this technique is based on Radial basis functions (RBF) [CBC*01]. However, fitting and evaluation of RBFs on large data sets is quite slow and, therefore, it is difficult to use this technique to reconstruct implicit surfaces from large point sets consisting of more than several thousands of points.

Another well known implicit method is the MPU Implicits [OBA*03]. This technique is an octree subdivision method that locally fits piecewise quadratic functions to the data and uses weighting functions (partitions of unity) to blend these functions together. Similar to the FastRBF method, MPU Implicits can produce noisy surfaces with extraneous parts. However, MPU Implicits is simple and very fast.

Other implicit methods are the method of Hoppe et al. [HDD*92] and VRIP [CL96]. They are more robust with respect to noise than many other algorithms. While slower than MPU Implicits, both methods are still quite fast. Note that, despite run-length encoding tricks used in VRIP, both methods have difficulty processing extremely large data sets due to the requirement that the representation of the implicit function resides in memory.

More recently, an implicit surface reconstruction method based on Fourier series was developed in [Kaz05]. This method has the advantage that the reconstructed surface is smooth and the method robustly handles noise and gaps in the data. However, computing a single Fourier coefficient requires a summation over all input samples since the basis functions are globally supported. The method also requires huge amount of memory due to the use of uniform grid, which limits its application to relatively modest size data sets. A solution to this problem was recently proposed in [SBS07], where the authors suggest combining the approach in [Kaz05] with adaptive subdivision and partition of unity blending techniques in [OBA*03].

The FFT approach, developed in [Kaz05], was later modified in [KBH06]. The modification utilizes an octree and finds the implicit function by solving a Poisson equation. In [BKBH07], the method is further enhanced by using a

Figure 1: *Surface reconstruction of "Barbuto" from laser range scans containing at total of* 329 *million points (7.34GB of data). Our wavelet surface reconstruction method completed the reconstruction in* 112 *minutes with* 329*MB of memory.*

streaming approach, which allows the algorithm to handle truly massive data sets on the order of hundreds of millions of point samples. However, despite its speed, processing can still take days to even weeks for large number of points.

3. Wavelets for Surface Reconstruction

Wavelets provide an alternative to classical Fourier methods for one- and multi-dimensional data analysis and synthesis, and have numerous applications both within mathematics and in areas as diverse as physics, seismology, medical imaging, digital image processing, signal processing, computer graphics and video. The main appeal of wavelets stems from their simultaneous localization in both frequency (wave number) and spatial (position) domains. These properties allow many classes of functions to be approximated by a relatively small number of wavelet basis functions while keeping most of their information content.

In our method, we use the input points p_i on the surface ∂M and their outward normals $\vec{n}_i$ to construct an approximation to the indicator function $\tilde{\chi}_M$ of the solid M with boundary ∂M. The indicator function is defined to be 1 inside M and 0 otherwise,

$$\chi_M(\mathbf{x}) = \begin{cases} 1, & \mathbf{x} \in M, \\ 0, & \text{elsewhere.} \end{cases}$$

We construct the approximation $\tilde{\chi}_M$ by approximating the wavelet coefficients of χ_M. Then, the surface $\partial\tilde{M}$ of a level set $\tilde{M}$ of $\tilde{\chi}_M$ is an approximation to the original surface.

3.1. Wavelet Representation

First, we briefly review some properties of wavelets necessary for our construction. Similar properties hold for biorthogonal wavelets.

Let φ be a compactly supported univariate scaling function with orthogonal shifts which satisfies the two-scale relation

$$\varphi(t) = \sum_{\ell \in \mathbb{Z}} \alpha_\ell \varphi(2t - \ell), \tag{1}$$

where only finite number of coefficients α_ℓ are nonzero. Let ψ be the univariate wavelet function with compact support which is obtained from φ by multiresolution. The formula for ψ is

$$\psi(t) = \sum_{\ell \in \mathbb{Z}} (-1)^\ell \overline{\alpha_{1-\ell}} \varphi(2t - \ell), \tag{2}$$

where $\overline{\alpha_{1-\ell}}$ denotes the complex conjugate of $\alpha_{1-\ell}$. Examples of such wavelets and scaling functions were given by Daubechies [Dau92]. We use standard construction of three dimensional wavelet bases. We shall use the notation $\psi^0 = \varphi$ and $\psi^1 = \psi$. Let E' denote the set of vertices of the cube $[0,1]^3$ and let E denote the set of vertices excluding the origin (i.e. $E = E' \setminus (0,0,0)$). For each $e = (e_1, e_2, e_3) \in E'$, $j \in \mathbb{N}$ and $\mathbf{k} = (k_1, k_2, k_3)$, we define

$$\psi^e_{j,\mathbf{k}}(\mathbf{x}) = 2^{3j/2} \psi^{e_1}(2^j x_1 - k_1) \psi^{e_2}(2^j x_2 - k_2) \psi^{e_3}(2^j x_3 - k_3)$$

where $\mathbf{x} = (x_1, x_2, x_3)$. Each function f that is locally integrable on $\mathbb{R}^3$ has the wavelet expansion

$$f(\mathbf{x}) = \sum_{\mathbf{k} \in \mathbb{Z}^3} c^{(0,0,0)}_{0,\mathbf{k}} \psi^{(0,0,0)}_{j,\mathbf{k}}(\mathbf{x}) + \sum_{j \in \mathbb{N}} \sum_{\mathbf{k} \in \mathbb{Z}^3} \sum_{e \in E} c^e_{j,\mathbf{k}} \psi^e_{j,\mathbf{k}}(\mathbf{x}), \tag{3}$$

where each $c^e_{j,\mathbf{k}}$ is given by

$$c^e_{j,\mathbf{k}} = \int_{\mathbb{R}^3} f(\mathbf{x}) \psi^e_{j,\mathbf{k}}(\mathbf{x})\, d\mathbf{x}.$$

Here, the index $\mathbf{k}$ denotes the spatial index of the 3D cell at resolution j. The index e is called the gender of the wavelet

and indicates which type of the 8 tensor product basis functions is being used.

3.2. Building the Indicator Function

In the case of surface reconstruction, the function f is the indicator function χ_M of the solid M. Without loss of generality, we may assume that M lies in the cube $[0,1)^3$. The coefficients $c_{0,\mathbf{k}}^{(0,0,0)}$ and $c_{j,\mathbf{k}}^e$ in the wavelet representation of χ_M are determined by

$$c_{j,\mathbf{k}}^e = \int_{\mathbb{R}^3} \chi_M(\mathbf{x}) \psi_{j,\mathbf{k}}^e(\mathbf{x})\, d\mathbf{x}$$

$$= 2^{3j/2} \int_M \psi^{e_1}(2^j x_1 - k_1)\psi^{e_2}(2^j x_2 - k_2)\psi^{e_3}(2^j x_3 - k_3)\, d\mathbf{x}.$$

Notice that, unlike standard wavelet applications in image processing, we do not have samples of χ_M from the interior of M, but only from its boundary. Therefore it is not obvious how to use wavelets to construct an approximation to the unknown function χ_M. In [Kaz05] Kazhdan observed in the context of computing the Fourier coefficients of χ_M that integrals of this form can be represented as surface integrals using the Divergence Theorem

$$\int_M \nabla \cdot \vec{F}(\mathbf{x})\, d\mathbf{x} = \int_{p \in \partial M} \vec{F}(p) \cdot \vec{n}(p)\, d\sigma, \tag{4}$$

where $\vec{F} = (F_1, F_2, F_3)$ is a vector valued function on $\mathbb{R}^3$, $\vec{n}(p)$ is the outward unit normal to the surface ∂M at point p and $d\sigma$ is the differential surface area of ∂M. To apply this theorem to compute each of the coefficients $c_{j,\mathbf{k}}^e$ and $c_{0,\mathbf{k}}^{(0,0,0)}$, we must construct vector valued functions $\vec{F}_{j,\mathbf{k}}^e$, $e \in E$, and $\vec{F}_{0,\mathbf{k}}^{(0,0,0)}$ that satisfy

$$\nabla \cdot \vec{F}_{j,\mathbf{k}}^e(\mathbf{x}) = \psi^{e_1}(2^j x_1 - k_1)\psi^{e_2}(2^j x_2 - k_2)\psi^{e_3}(2^j x_3 - k_3). \tag{5}$$

Given such a function $\vec{F}_{j,\mathbf{k}}^e$ satisfying (5), we can discretize (4) over the point samples $(p_i, \vec{n}_i)$

$$c_{j,\mathbf{k}}^e \approx 2^{3j/2} \sum_i \vec{F}_{j,\mathbf{k}}^e(p_i) \cdot \vec{n}_i d\sigma_i, \tag{6}$$

where $d\sigma_i$ is an estimate of the differential surface area associated with the sample point p_i.

There are many choices of functions $\vec{F}_{j,\mathbf{k}}^e$ that satisfy (5); however, many will **not** have compact support even if the associated wavelet basis does. This lack of locality will ruin any effeciency gains we obtained from using compactly supported (biorthogonal) wavelets as the computation of each coefficient $c_{j,\mathbf{k}}^e$ will be influenced by all points p_i on the surface. We present a construction of functions $\vec{F}_{j,\mathbf{k}}^e$, which can be found in the Appendix, that creates compactly supported $\vec{F}_{j,\mathbf{k}}^e$, $e \in E$, for *any* compactly supported (bi)orthogonal wavelet basis. Furthermore, the functions in $\vec{F}_{j,\mathbf{k}}^e$, $e \in E$ will have the same support as the underlying wavelet basis.

Therefore, (6) can be written as

$$c_{j,\mathbf{k}}^e \approx 2^{3j/2} \sum_{p_i \in \partial M \cap supp\, \psi_{j,\mathbf{k}}^e} \vec{F}_{j,\mathbf{k}}^e(p_i) \cdot \vec{n}_i d\sigma_i$$

where $supp\, \psi_{j,\mathbf{k}}^e$ denotes the support of $\psi_{j,\mathbf{k}}^e$. Notice that as opposed to globally supported bases like the Fourier basis, this summation does not involve all point samples, but only those that belong to the support of $\psi_{j,\mathbf{k}}^e$. The only coefficients that involve summation over all points p_i are the $c_{0,\mathbf{k}}^{(0,0,0)}$. In this case, we only need to compute those coefficients $c_{0,\mathbf{k}}^{(0,0,0)}$ that correspond to basis functions $\varphi(x_1 - k_1)\varphi(x_2 - k_2)\varphi(x_3 - k_3)$ whose support overlaps the region of interest $[0,1)^3$ containing M. Fortunately, this is a small, constant number of coefficients that depends on the support of the scaling function φ.

Finally, the computation of the coefficients in (6) requires an evaluation of $d\sigma_i$, associated with the point p_i. There are many ways of estimating $d\sigma_i$ and [Kaz05] provides one such method based on Gaussian weighting. We use a simple, octree-based method to compute $d\sigma_i$ that handles non-uniformly sampled data points. We refine all octree cells containing sample points until we reach some maximum depth d_{max} specified by the user. Once all points are inserted into the octree, we prune leaves of this tree until every leaf is adjacent to at least 3 octree cells of the same depth that also contain sample points. Our cutoff of exactly 3 adjacent cells is not arbitrary since it guarantees the minimum number of cells necessary to form a connected piece of surface (a tetrahedron). Then the value of $d\sigma_i$, associated with a point p_i inside a leaf from the octree is given by $d\sigma_i = 2^{-2d_i}/m$, where m is the total number of points in this leaf and d_i is the depth of the leaf. This quantity is exactly the area of the side of the leaf divided by the number of points in that leaf.

The computation of $d\sigma_i$ creates an octree in whose cells we store not only the number of points inside the cell, but also the wavelet coefficients that are indexed by this cell. The cells of the octree at each level j and position $k \in \mathbb{R}^3$ store the non-zero coefficients $c_{j,\mathbf{k}}^e$ for all $e \in E$. Note that we may require octree cells outside of $[0,1)^3$ to store coefficients $c_{j,\mathbf{k}}^e$ of $\psi_{j,\mathbf{k}}^e$ whose support overlaps the sample points.

4. Surface Extraction

As described in Section 3.1, we compute an approximation $\tilde{\chi}_M$ to χ_M, and recover a solid $\widetilde{M}$ that is a level-set of the $\tilde{\chi}_M$. Since $\tilde{\chi}_M \approx \chi_M$, which is 0 outside M and 1 inside of M, extracting the solid at level $\frac{1}{2}$ is a reasonable choice. For poorly scanned surfaces or point sets with high amounts of noise, we can choose a data-dependent iso-value using the average value of $\tilde{\chi}_M$ over the sample points.

4.1. Polygon Generation

The ability of wavelets to detect discontinuities naturally creates an adaptive refinement of the octree near the bound-

Figure 2: *Surface reconstruction using Haar wavelets (left) results in a noisy surface because the basis functions are discontinuous. Smoothing the indicator function results in a substantially smoother surface at a small cost to speed and approximation quality (right).*

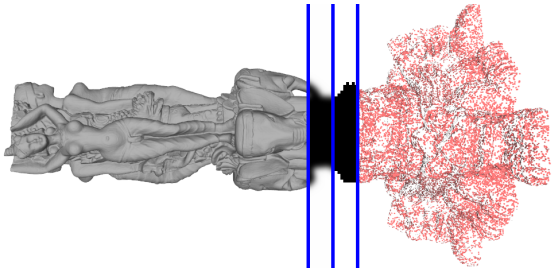

Figure 3: *Depiction of our streaming implementation. We first construct the wavelet coefficients of the indicator function, smooth the function and then extract the iso-surface.*

ary ∂M of M. We use this octree to construct a polygonal model of the boundary $\partial\tilde{M}$ by applying the octree contouring method from [SW04]. This algorithm computes the dual cell structure of the octree through a recursive octree walk. The method uses the values of $\tilde{\chi}_M$ at the vertices of the dual cells, located at the centers of the corresponding octree cells. We then run Marching Cubes [LC87] on the dual cells to create a water-tight, adaptive contour, which is guaranteed to produce a topological and geometrical manifold. Since the number of dual cells is proportional to the size of the octree, the running time of this contouring method is also proportional to the size of the octree.

4.2. Post-processing of the Indicator Function

The smoothness of the (biorthogonal) wavelet used will impact the smoothness of $\tilde{\chi}_M$ and, therefore, the smoothness of the level set $\tilde{M}$. Wavelets with small support yield higher performance algorithms because fewer wavelet coefficients in (6) need to be calculated and each coefficient is influenced by a smaller number of sample points. However, smaller support negatively impacts the quality of the resulting surface. Instead of increasing the support of the wavelet to improve the quality of the reconstructed surface, one alternative is to perform a post-processing smoothing step on the indicator function $\tilde{\chi}_M$. With this method, we can retain the time efficiency associated with wavelets of small support and achieve a more visually appealing surface. We have compared the Hausdorff error obtained by this method with the error produced by other techniques. The results are summarized in Table 3 and show that our method, with and without post-processing, outperforms other techniques in terms of accuracy.

Smoothing a function is typically performed by convolving the function with a small smoothing kernel over a uniform grid. However, for the large data sets that we target, convolution over uniform grids is too expensive. Therefore, prior to polygon generation, we perform an approximate smoothing pass, modified to operate over octrees, whose complexity is proportional to the size of the octree. We use a small convolution mask that only involves adjacent cells and is the tensor product of the mask $(\frac{1}{4}, \frac{1}{2}, \frac{1}{4})$ in $\mathbb{R}^3$.

We smooth the function $\tilde{\chi}_M$, given by (3), where the summation over the dyadic levels j is up to a user specified depth d_{max}. Given a cell at depth d, we compute the value of $\tilde{\chi}M$ at the center of the cell and its 26 neighbors at the same level using (3) by summing up to depth d. If a neighbor does not exist, then the wavelets indexed by this neighbor do not contribute to the sum. We then perform uniform convolution over this locally uniform grid and we treat the obtained value as the value of the smoothened function at the center of that cell. Figure 2 shows an example of a surface reconstructed using Haar wavelets (see Section 5) without (left) and with (right) this smoothing step.

4.3. Streaming Implementation

The storage space for $\tilde{\chi}_M$ is proportional to the surface area of the reconstructed surface because refinement is only performed near point samples. However, extremely large data sets may require more space than can fit into the memory of most desktop machines. To combat this effect, we develop a streaming version of our algorithm. Like most streaming algorithms, we require that the input points are sorted in one of the Euclidean directions. If the points are not sorted, we preprocess them by sorting along the longest Euclidean direction of their bounding box using an out-of-core merge sort. We assume, without loss of generality, that the sort is in the z-direction. For a data size of about 205 million points, the sort takes 20 minutes and the sorting time is negligible compared to the time for surface reconstruction.

Our streaming algorithm builds a low resolution, in-core approximation of χ_M down to some depth $d_m < d_{max}$, where d_{max} is the maximal depth of the octree and encodes subtrees

corresponding to high-resolution details in a streaming fashion. Note that the coefficients $c_{0,\mathbf{k}}^{(0,0,0)}$ depend on all sample points and, therefore, $\tilde{\chi}_M$ cannot be evaluated until all points are processed at least once. Our solution is to perform two passes over the data. The first pass constructs all coefficients down to depth d_m. The second, streaming pass builds the non-zero wavelet coefficients $c_{j,\mathbf{k}}^e$ of $\tilde{\chi}_M$ for $d_m < j \leq d_{max}$ and $\ell 2^{-d_m} \leq k_3 < (\ell+1)2^{-d_m}$ for $0 \leq \ell < 2^{d_m}$. For each slice ℓ, we build the corresponding wavelet coefficients by inserting points into the octree and refining the tree only where (6) creates non-zero coefficients $c_{j,\mathbf{k}}^e$. We then smooth the function values and create polygons before deleting the subtrees corresponding to the slice from memory. Figure 3 depicts this streaming process.

Any choice of d_m will work with our streaming algorithm, but its value will affect the amount of memory required to reconstruct the surface. Obviously, choosing $d_m = 0$ or d_{max} requires the entire tree to fit into memory. Assume we have L leaves of the octree at depth d_{max}. Since the size of the octree at each level is proportional to the surface area of ∂M, the number of cells at depth j is approximately $L4^{-(d_{max}-j)}$. Therefore, our streaming algorithm stores approximately

$$\sum_{j=0}^{d_m} \frac{L}{4^{d_{max}-j}} + \frac{\beta}{2^{d_m}} \sum_{j=d_m+1}^{d} \frac{L}{4^{d_{max}-j}}$$

cells in memory at any one time because we keep β slices in memory for the different stages of our algorithm, where β is dependent on the support of the wavelet used. For the Daubechies wavelets we consider in Section 5, $\beta = 4$. Minimizing this sum with respect to d_m yields an optimal value of $d_m \approx .69 d_{max}$. While this technique does not allow us to process arbitrarily deep octrees (the octree to depth d_m must still be stored), it does allow us to process much deeper trees than a strictly in-core algorithm. We have been able to reconstruct surfaces down to depth 14 in memory (see Section 6). At that resolution, a single cross-section of the grid at the maximal depth has over 250 million cells and allows us to process the largest data sets that we could obtain.

5. Implementation

In the applications we are interested, computational time is of major concern, which motivates us to explore wavelets with small support such as the Haar and D4 Daubechies wavelets. Despite the lack of smoothness of these wavelets, the reconstructed surface is of good quality and is a good approximation to the original surface (see Section 6).

The Haar wavelet is given by the formula

$$\psi(t) = \begin{cases} 1, & 0 \leq t < 1/2, \\ -1, & 1/2 \leq t < 1, \end{cases}$$

and its corresponding scaling function φ is

$$\varphi(t) = \begin{cases} 1, & 0 \leq t < 1, \\ 0, & \text{elsewhere.} \end{cases}$$

The Daubechies D4 wavelet is determined by (1), where the coefficients come from the scaling relationship

$$\varphi(t) = \begin{pmatrix} \frac{1+\sqrt{3}}{4} & \frac{3+\sqrt{3}}{4} & \frac{3-\sqrt{3}}{4} & \frac{1-\sqrt{3}}{4} \end{pmatrix} \begin{pmatrix} \varphi(2t) \\ \varphi(2t-1) \\ \varphi(2t-2) \\ \varphi(2t-3) \end{pmatrix}.$$

Figure 4 depicts plots of both the scaling function and D4 wavelet.

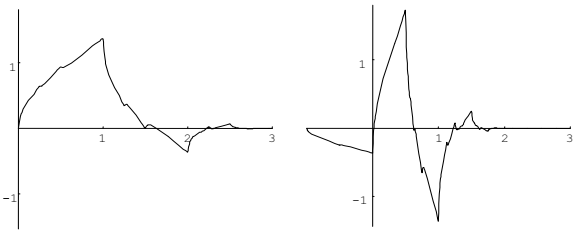

Figure 4: *The D4 scaling function φ (left) and the corresponding wavelet ψ (right).*

Given that the D4 wavelet ψ has no analytical representation, we evaluate the functions $\vec{F}_{j,\mathbf{k}}^e$ (see the Appendix) at the sample points p_i using a piecewise linear interpolant of φ, ψ, Φ and Ψ based on the exact values of these functions on a uniform grid. The exact values on uniform grid are found using standard techniques of evaluation of functions that satisfy a scaling relationship with finite number of non-zero scaling coefficients, see for example [DM93, BCU00]. Note that the functions Φ and Ψ are in this category as well. For example, Φ satisfies the scaling relation

$$\Phi(t) = \begin{pmatrix} \frac{1+\sqrt{3}}{8} & \frac{3+\sqrt{3}}{8} & \frac{3-\sqrt{3}}{8} & \frac{1-\sqrt{3}}{8} \end{pmatrix} \begin{pmatrix} \Phi(2t) \\ \Phi(2t-1) \\ \Phi(2t-2) \\ \Phi(2t-3) \end{pmatrix},$$

derived by integrating the scaling relation for φ. In our implementation, we use a uniform rational grid with spacing $\frac{1}{64}$ to represent these functions, but grids with different spacing can easily be used.

6. Results

Here, we present the results obtained by our method using the Haar and the D4 wavelets, in terms of speed, memory efficiency and accuracy. We compare our method with some of the best known methods for surface reconstruction based on point cloud data. We show that, in terms of time, our technique outperforms these methods by an order of magnitude (see Table 2) while producing surfaces whose accuracy is comparable to the accuracy of the surfaces obtained by the other methods (see Table 3).

We apply our algorithm to point clouds for which the separate point scans are already aligned in 3D. If the scans are

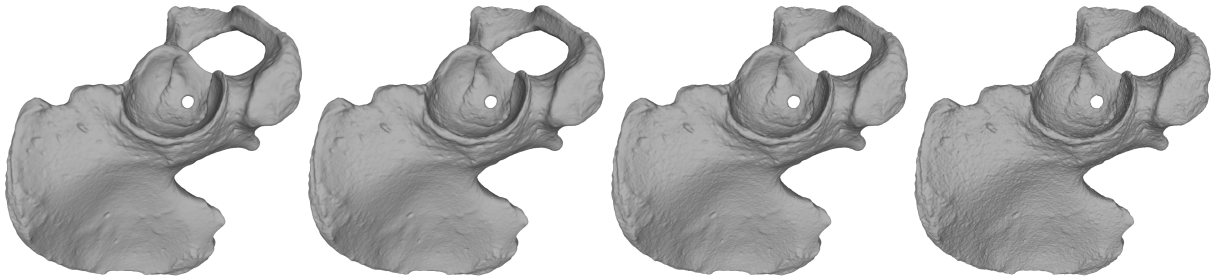

Figure 5: *Reconstruction of a hip bone using Haar wavelets with varying amounts of noise in the normals. From left to right: 0 degrees, 30 degrees, 60 degrees, 90 degrees uniform random rotational deviation in the normal direction.*

not aligned, then we perform a preprocessing step that involves a method like the one in [BR07]. We also assume that each point sample p_i on the surface has an associated outwards unit normal $\vec{n}_i$ to the surface at p_i. If the data for the normals is absent, we can estimate the normals using a local PCA method or polynomial fitting. However, many scanners produce not only point samples, but partially triangulated scans reflecting the scanner's estimate of which samples are connected from a single scanning direction (all of the data in the Digital Michelangelo Project is of this form). While these triangles are not sufficient to produce a closed, triangulated model, they do give us the ability to estimate normals efficiently. Furthermore, the orientation of the normal (inwards vs. outwards) can also be constructed robustly from these scans since the scanners rely on visibility information and the normals must be oriented in the direction of the scanner.

The process of estimation of normals for our point samples from the data provided by laser range scanners can be inaccurate and can introduce significant errors. However, we show that our technique is robust with respect to errors in the normal directions. Figure 5 shows several reconstructions where we incrementally add more noise to the input normals. Even at 90 degrees, the surface is still faithfully reconstructed. With more noise the reconstruction quality begins to suffer noticeably, but at this noise level ($>$ 90 degrees deviation) the normals are pointing inwards to the surface and begin to be meaningless.

Surface reconstruction using Haar wavelets is extremely fast. Haar wavelets create a minimal number of coefficients in the octree. These coefficients can be computed very quickly because the scaling and wavelet functions as well as their integrals have an analytical form and the support of the wavelet is small. Figure 6 shows a depth 14 reconstruction of Michelangelo's Awakening statue using Haar wavelets. This data set is one of the largest we obtained and contains 381 million points. Despite its size, our method is able to produce a faithful reconstruction in about 81 minutes.

For many real-world data sets, Haar wavelets create pleasing surface reconstructions when the scans are well-aligned

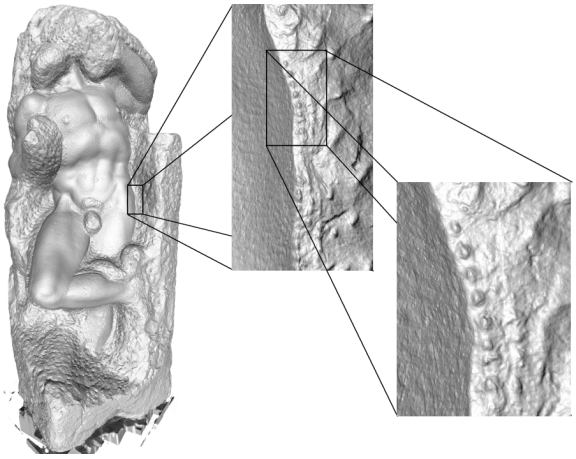

Figure 6: *"Awakening" with* 381 *million points (*8.51 *GB of data) reconstructing using Haar wavelets at depth 14 took about 81 minutes and produced over* 590 *million polygons. We show two extreme zooms indicating that even small chisel marks are reconstructed with a high degree of accuracy.*

and the noise is relatively low. However, in some cases, the reconstruction begins to fit noise in the data, due to the small support of the Haar wavelet, and results in the appearance of artifacts. Surface reconstruction using D4 wavelets provides a much higher reconstruction quality compared to Haar wavelets. The support of the basis functions is larger, which makes the method more resilient to noise in the input data. The larger support of this basis also increases the number of coefficients we have to store and their computation time. Furthermore, using D4 wavelets roughly triples the number of octree cells needed for the Haar wavelet. Nevertheless, reconstruction times are still quite fast. Figures 1, 8 and 7 (bottom right) are all reconstructed using D4 wavelets.

Table 1 shows reconstruction times for several popular surface reconstruction methods for which implementations are freely available, all operating on the same data set of 4.5 million points from David's head and at a maximal octree

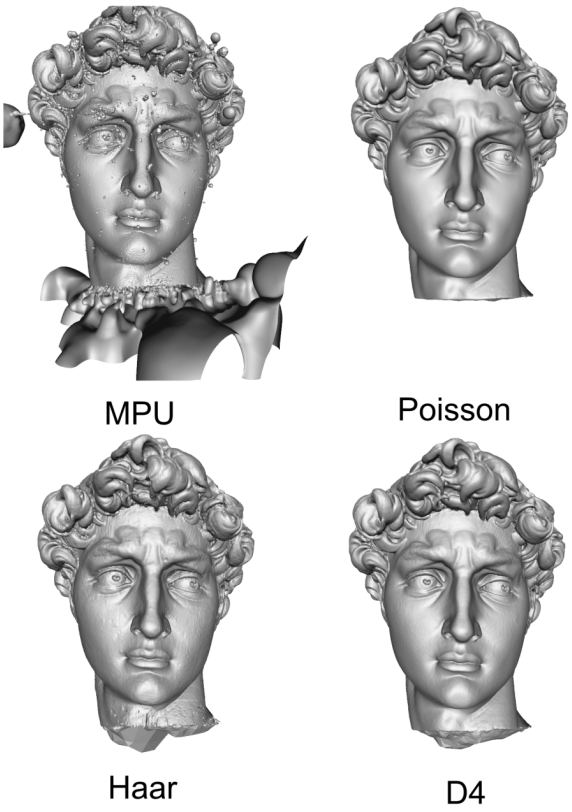

Figure 7: *Comparison of reconstructions of David's head with* 4.5 *million points (103 MB of data) at depth* 9 *with MPU Implicits (top left), Poisson reconstruction (top right), Haar wavelets (bottom left) and D4 wavelets (bottom right).*

Method	Time(sec)	Memory(MB)	Polygons
Ohtake et al	551	750	1582380
Bolitho et al	289	57	1257980
Haar Wavelet	17	13	1357872
D4 Wavelet	82	43	1377858

Table 1: *Reconstruction of David's head consisting of* 4.5 *million points at depth* 9 *with various methods.*

depth of 9 where appropriate. All tests are run on an Intel 6700 with 2GB of RAM. [OBA*03] is typically considered a fast surface reconstruction algorithm, yet our method using D4 wavelets is roughly 6 times faster and Haar wavelets over 30 times faster. Due to our streaming implementation, our memory requirements are very low as well.

Table 2 includes reconstruction times for some of the largest data sets that we could find. Many of these data sets contain well-aligned scans and Haar wavelet reconstruction performs well without many errors due to noise or misaligned scans. Our method using Haar wavelets was able to process each of these data sets in under an hour at depth 12 and under 2 hours at depth 14. Using D4 wavelets was slower but we were still able to complete even Atlas with 410 million points at depth 12 in under 2.5 hours.

Model	Points	Haar,12	Haar,13	Haar,14	D4,12
Barbuto	329M	38.7/100	58.3/252	81.6/777	111.9/329
Awakening	381M	45.5/100	62.4/187	80.8/573	133.3/339
Atlas	410M	51.7/133	59.0/351	97.6/1188	148.4/448

Table 2: *Reconstruction of various models using Haar and D4 wavelets at various depths. Data is of the form time/memory where time is measured in minutes and memory in MB.*

It is difficult to measure the accuracy of a surface reconstruction if only point scans are available. Fitting the points exactly may yield an undesirable surface away from the point samples. Furthermore, gaps/holes in the data set make the development of a good, two-sided error metric challenging. We overcome this problem by sampling points and normals densely from known polygon models and then reconstructing surfaces with each method from these points sets. We then compute the Hausdorff distance between each reconstructed surface and the original shape using Metro [CRS98]. Table 3 summaries the results. The values for each row are normalized by the maximum error among all methods to provide a relative comparison between the different techniques. MPU is on average the worst among the tested models, followed by the Poisson reconstruction, which typically performs much better. However, in all cases, Haar and D4 wavelets with and without smoothing recover the surfaces with higher degree of accuracy than the Poisson reconstruction and in only one case (the hand) does MPU outperform the wavelet methods.

These results seem somewhat counter-intuitive as the Poisson reconstruction in Figure 7 is obviously smoother and appears more desirable than either the Haar or D4 reconstructions. Good surface reconstruction routines must satisfy the Hausdorff error metric since it measures if extraneous sheets or bulgs are created; however, this error metric does not measure normal quality. Both Haar and D4 wavelets produce surfaces that may have high frequency artifacts in the normals since both wavelet bases are not smooth. As the smoothness of the wavelet basis increases, these artifacts decrease but the surface will have a higher Hausdorff error, which is why surfaces produced using D4 wavelets typically have higher Hausdorff error than those produced by Haar wavelets but appear to be higher quality. Smoothing the function mitigates the artifacts in the normals at the cost to approximation quality, which is clearly seen in Table 3. This is an illustration of a well known phenomenon where approximation accuracy is sacrificed in order to reduce oscillations.

Model	MPU	Poisson	Haar	Haar Smoothed	D4	D4 Smoothed
armadilloman	1.000000	0.259120	0.153069	0.212276	0.151215	0.224511
happy buddha	1.000000	0.364243	0.223781	0.339264	0.346450	0.356715
cow	1.000000	0.086590	0.036725	0.046528	0.071015	0.074601
dragon	1.000000	0.790500	0.602828	0.536930	0.617106	0.636463
elephant	1.000000	0.507040	0.363651	0.221071	0.320602	0.372229
hand	0.332169	1.000000	0.380563	0.565324	0.335825	0.589883
hip	1.000000	0.110895	0.064997	0.093744	0.061002	0.091113
malaysia	1.000000	0.217397	0.151879	0.189831	0.144758	0.200062
teeth	0.835987	1.000000	0.418790	0.503185	0.471338	0.702229
venus	1.000000	0.371843	0.184752	0.260992	0.198316	0.285313

Table 3: *Hausdorff distance between real surfaces and reconstructed surfaces from sampled data. Each row is normalized by the worst geometric error (lower is better).*

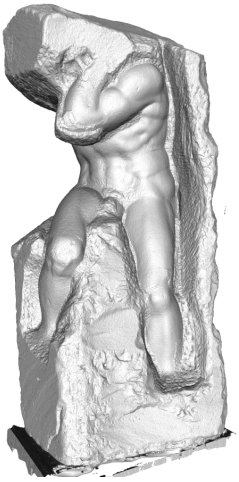

Figure 8: *Reconstruction of Michelangelo's Atlas with* 410 *million points (*9.15 *GB of data) at depth* 12 *with D4 wavelets took less than* 2.5 *hours and produced* 42.7 *million polygons.*

7. Future Work

In the future, we would like to implement other smoother basis functions whose support size is relatively small. Controlling support is important, because the size of the support of the basis functions is closely related to the computational cost and time efficiency of the algorithm, and thus a smaller support leads to a faster algorithm. On the other hand, smoother basis functions are needed for smoother reconstructions, but smother functions require larger support. We would like to understand quantitatively this dichotomy and select a basis for which these two quantities are at balance. Examples of bases we would like to explore include smoother wavelets, biorthogonal wavelets and quasi-interpolants. In general, any basis that consists of smooth compactly supported functions that allow multiscale decomposition and have vanishing moments could be used with our technique though.

Wavelets are renowned for their applications in image compression and de-noising. In 3D, their localization properties allow us to efficiently represent χ_M in terms of its wavelet coefficients and this is one of the main reasons for the speed and efficiency of the proposed algorithm. We would like to explore wavelet techniques for image de-noising and how they can be used in the context of surface reconstruction to create fast methods that are even more robust with respect to noise.

Acknowledgements
We would like to thank the Digital Michelangelo project and Szymon Rusinkiewicz for the aligned statue scans. This work was also supported in part by NSF grant CCF-07024099, DARPA grant HR0011-08-1-0014, ARO/MURI grant W911NF-07-1-0185 and DMS 0505501.

References

[ACA07] ALLÈGRE R., CHAINE R., AKKOUCHE S.: A streaming algorithm for surface reconstruction. In *Proceedings of the fifth Eurographics symposium on Geometry processing* (2007), pp. 79–88.

[ACK01] AMENTA N., CHOI S., KOLLURI R. K.: The power crust. In *SMA '01: Proceedings of the sixth ACM symposium on Solid modeling and applications* (2001), pp. 249–266.

[BCU00] BERTOLUZZA S., CANUTO C., URBAN K.: On the adaptive computation of integrals of wavelets. *Applied Numerical Mathematics 34*, 1 (2000), 13–38.

[BKBH07] BOLITHO M., KAZHDAN M., BURNS R., HOPPE H.: Multilevel streaming for out-of-core surface reconstruction. In *Symposium on Geometry Processing* (July 2007), pp. 69–78.

[BMR*99] BERNARDINI F., MITTLEMAN J., RUSHMEIER H., SILVA C., TAUBIN G.: The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics 5*, 4 (1999), 349–359.

[BR07] BROWN B., RUSINKIEWICZ S.: Global non-rigid align-

ment of 3-D scans. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 26*, 3 (2007), 21.

[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 SIGGRAPH* (2001), pp. 67–76.

[CL96] CURLESS B., LEVOY M.: A volumetric method for building complex models from range images. In *Proceedings of SIGGRAPH* (1996), pp. 303–312.

[CRS98] CIGNONI P., ROCCHINI C., SCOPIGNO R.: Metro: Measuring error on simplified surfaces. *Computer Graphics Forum 17*, 2 (1998), 167–174.

[Dau92] DAUBECHIES I.: *Ten lectures on wavelets*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1992.

[DG06] DEY T. K., GOSWAMI S.: Provable surface reconstruction from noisy samples. *Comput. Geom. Theory Appl. 35*, 1 (2006), 124–141.

[DGH01] DEY T. K., GIESEN J., HUDSON J.: Delaunay based shape reconstruction from large data. In *Proceedings of the symposium on parallel and large-data visualization and graphics* (2001), pp. 19–27.

[DM93] DAHMEN W., MICCHELLI C. A.: Using the refinement equation for evaluating integrals of wavelets. *SIAM Journal on Numerical Analysis 30*, 2 (1993), 507–537.

[HDD*92] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Surface reconstruction from unorganized points. In *Proceedings of SIGGRAPH* (1992), pp. 71–78.

[Kaz05] KAZHDAN M. M.: Reconstruction of solid models from oriented point sets. In *Symposium on Geometry Processing* (2005), pp. 73–82.

[KBH06] KAZHDAN M., BOLITHO M., HOPPE H.: Poisson surface reconstruction. In *Symposium on Geometry Processing* (June 2006), pp. 61–70.

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

[LPC*00] LEVOY M., PULLI K., CURLESS B., RUSINKIEWICZ S., KOLLER D., PEREIRA L., GINZTON M., ANDERSON S., DAVIS J., GINSBERG J., SHADE J., FULK D.: The digital michelangelo project: 3d scanning of large statues. In *Proceedings of SIGGRAPH* (2000), pp. 131–144.

[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.

[OF02] OSHER S. J., FEDKIW R. P.: *Level Set Methods and Dynamic Implicit Surfaces*. Springer, 2002.

[SBS07] SCHALL O., BELYAEV A., SEIDEL H.-P.: Error-guided adaptive fourier-based surface reconstruction. *Computer Aided Design 39*, 5 (2007), 421–426.

[Set99] SETHIAN J.: *Level set methods and fast marching methods. Evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, 2 ed. Cambridge University Press, 1999.

[SS05] SCHALL O., SAMOZINO M.: Surface from scattered points: a brief survey of recent developments. In *First International Workshop on Semantic Virtual Environments* (2005), pp. 138–147.

[SW04] SCHAEFER S., WARREN J.: Dual marching cubes: Primal contouring of dual grids. In *Proceedings of Pacific Graphics* (2004), pp. 70–76.

Appendix A: The functions $\vec{F}^{e}_{j,\mathbf{k}}$

We define

$$\Phi(t) := \int_{-\infty}^{t} \varphi(s)\,ds, \quad \Psi(t) := \int_{-\infty}^{t} \psi(s)\,ds.$$

Notice that $\Phi'(t) = \varphi(t)$ and $\Psi'(t) = \psi(t)$, and for compactly supported wavelets, Ψ will have the same support as ψ, since ψ has at least a zero-th order vanishing moment, i.e. $\int_{-\infty}^{\infty} \psi(s)\,ds = 0$. We now define the following vector functions $\vec{F}^{e}_{j,\mathbf{k}}$ for $j \in \mathbb{N}$, $\mathbf{k} \in \mathbb{Z}^3$, and $e \in E'$,

$$\vec{F}^{(0,0,0)}_{0,\mathbf{k}}(\mathbf{x}) = \begin{array}{c} \tfrac{1}{3}(\Phi(x_1-k_1)\varphi(x_2-k_2)\varphi(x_3-k_3), \\ \varphi(x_1-k_1)\Phi(x_2-k_2)\varphi(x_3-k_3), \\ \varphi(x_1-k_1)\varphi(x_2-k_2)\Phi(x_3-k_3)), \end{array}$$

$$\vec{F}^{(1,0,0)}_{j,\mathbf{k}}(\mathbf{x}) = 2^{-j}(\Psi(2^j x_1-k_1)\varphi(2^j x_2-k_2)\varphi(2^j x_3-k_3),0,0),$$

$$\vec{F}^{(0,1,0)}_{j,\mathbf{k}}(\mathbf{x}) = 2^{-j}(0,\varphi(2^j x_1-k_1)\Psi(2^j x_2-k_2)\varphi(2^j x_3-k_3),0),$$

$$\vec{F}^{(0,0,1)}_{j,\mathbf{k}}(\mathbf{x}) = 2^{-j}(0,0,\varphi(2^j x_1-k_1)\varphi(2^j x_2-k_2)\Psi(2^j x_3-k_3)),$$

$$\vec{F}^{(1,1,0)}_{j,\mathbf{k}}(\mathbf{x}) = \begin{array}{c} \tfrac{2^{-j}}{2}(\Psi(2^j x_1-k_1)\psi(2^j x_2-k_2)\varphi(2^j x_3-k_3), \\ \psi(2^j x_1-k_1)\Psi(2^j x_2-k_2)\varphi(2^j x_3-k_3),0), \end{array}$$

$$\vec{F}^{(0,1,1)}_{j,\mathbf{k}}(\mathbf{x}) = \begin{array}{c} \tfrac{2^{-j}}{2}(0,\varphi(2^j x_1-k_1)\Psi(2^j x_2-k_2)\psi(2^j x_3-k_3), \\ \varphi(2^j x_1-k_1)\psi(2^j x_2-k_2)\Psi(2^j x_3-k_3)), \end{array}$$

$$\vec{F}^{(1,0,1)}_{j,\mathbf{k}}(\mathbf{x}) = \begin{array}{c} \tfrac{2^{-j}}{2}(\Psi(2^j x_1-k_1)\varphi(2^j x_2-k_2)\psi(2^j x_3-k_3),0, \\ \psi(2^j x_1-k_1)\varphi(2^j x_2-k_2)\Psi(2^j x_3-k_3)), \end{array}$$

$$\vec{F}^{(1,1,1)}_{j,\mathbf{k}}(\mathbf{x}) = \begin{array}{c} \tfrac{2^{-j}}{3}(\Psi(2^j x_1-k_1)\psi(2^j x_2-k_2)\psi(2^j x_3-k_3), \\ \psi(2^j x_1-k_1)\Psi(2^j x_2-k_2)\psi(2^j x_3-k_3), \\ \psi(2^j x_1-k_1)\psi(2^j x_2-k_2)\Psi(2^j x_3-k_3)). \end{array}$$

Computing the divergence of $\vec{F}^{e}_{j,\mathbf{k}}$ shows that each of these functions satisfies (5). Note that, for $e \neq (0,0,0)$, the support of $\vec{F}^{e}_{j,\mathbf{k}}$ is finite and the same as the support of $\psi^{e}_{j,\mathbf{k}}$. While the smoothness of the wavelet affects the surface accuracy of the reconstruction, compact support of $\vec{F}$ is extremely important since, without it, the number of non-zero wavelet coefficients in (6) is proportional to the volume of the solid rather than the surface area of the shape. This lack of locality would render the reconstruction algorithm computationally impractical for large data sets.