Floating Scale Surface Reconstruction

Simon Fuhrmann Michael Goesele
TU Darmstadt, Germany

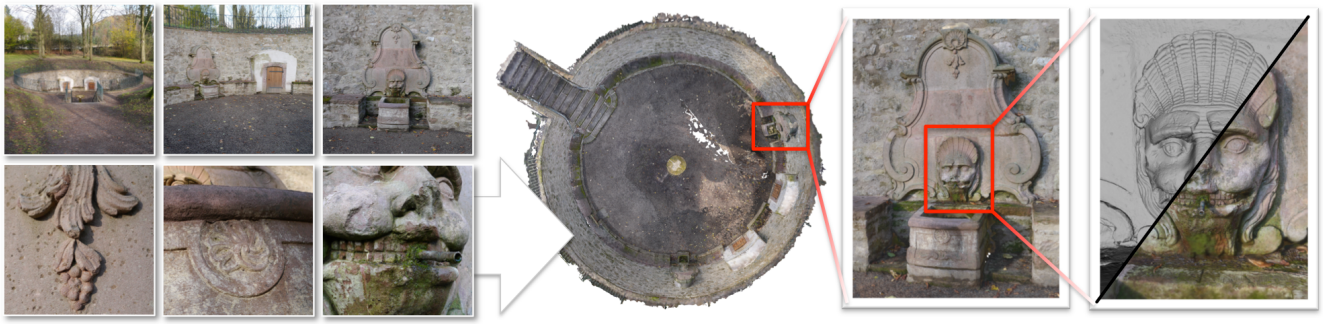

Figure 1: *Floating Scale Surface Reconstruction example. 6 out of 384 input images of a multi-scale dataset (left). Registered images are processed with multi-view stereo which yields depth maps with drastically different sampling rates of the surface. Our algorithm is able to accurately reconstruct every caputured detail of the dataset using a novel multi-scale reconstruction approach (right).*

Abstract

Any sampled point acquired from a real-world geometric object or scene represents a finite surface area and not just a single surface point. Samples therefore have an inherent scale, very valuable information that has been crucial for high quality reconstructions. We introduce a new method for surface reconstruction from oriented, scale-enabled sample points which operates on large, redundant and potentially noisy point sets. The approach draws upon a simple yet efficient mathematical formulation to construct an implicit function as the sum of compactly supported basis functions. The implicit function has spatially continuous "floating" scale and can be readily evaluated without any preprocessing. The final surface is extracted as the zero-level set of the implicit function. One of the key properties of the approach is that it is virtually parameter-free even for complex, mixed-scale datasets. In addition, our method is easy to implement, scalable and does not require any global operations. We evaluate our method on a wide range of datasets for which it compares favorably to popular classic and current methods.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems

Keywords: Surface Reconstruction

Links: DL PDF WEB CODE

SIGGRAPH '14, August 10 – 14 2014, Vancouver, Canada.

ACM 978-1-4503-2904-0/14/08 $15.00.
http://dx.doi.org/10.1145/2601097.2601163

1 Introduction

Surface reconstruction from sampled data is a long-standing and extensively studied topic in computer graphics. Consequently, there exists a broad and diverse range of methods with various strengths and weaknesses. One well-known example is VRIP [Curless and Levoy 1996], an efficient and scalable method able to create high quality models. Due to these properties, it was extensively used in the context of the Digital Michelangelo project [Levoy et al. 2000] to merge the captured range images. Since then many new techniques were developed that, e.g., use more advanced mathematical concepts, are able to smoothly interpolate holes, or employ hierarchical techniques. These approaches come, however, often at the cost of limited efficiency, scalability or certain quality issues. Moreover, they frequently treat reconstruction as completely separate from the actual sample acquisition process.

Our goal in this paper is to present a method that is able to efficiently reconstruct high quality meshes from acquired sample data even for large and noisy datasets using a virtually parameter-free method. Examples of such reconstructions from hundreds of millions of samples are the *Fountain* dataset (Figure 1) and the full-sized David statue (Figure 12) from the Digital Michelangelo project [Levoy et al. 2000]. Following on earlier work, we attach a *scale* value to each sample which provides valuable information about the surface area each sample was acquired from.

The sample scale can in general be easily derived from the acquisition process (e.g., from the sample footprint in a structured light scan or the patch size in a multi-view stereo algorithm). This definition of scale that has been used in prior work [Muecke et al. 2011; Fuhrmann and Goesele 2011]. Knowing scale allows us to reliably identify redundancy in the samples and avoid intermingling data captured at different scales (such as in multi-view stereo depth maps reconstructed from images at various distances to the geometry, as shown in Figure 1). Without scale information, datasets containing non-uniform redundancy, sample resolution or noise characteristics will, in general, lead to poor reconstructions. Many methods do adapt the reconstruction resolution to the input data in some way. These decisions, however, are often based on the density of the input data. Figure 2 shows a common case that demonstrates why density and scale are not always related: An increased sample den-

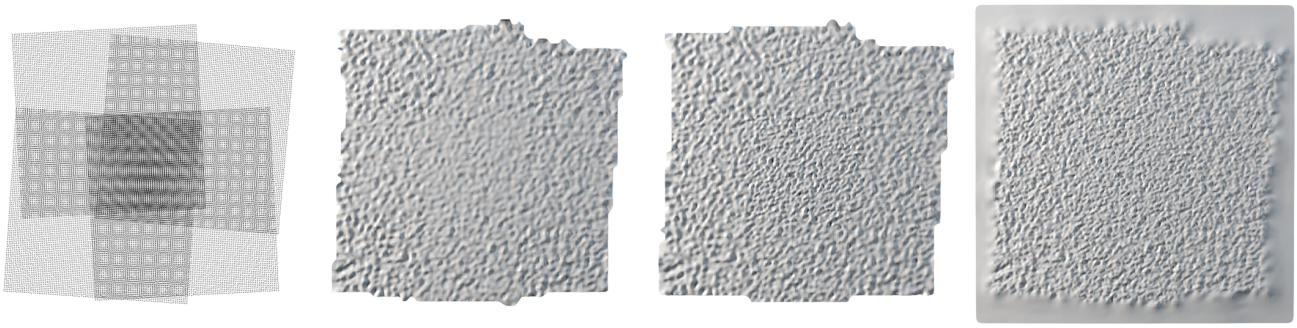

Figure 2: *Sample density versus sample scale. Noisy input samples from four synthetic, overlapping scans (left). Reconstructed surface with proper scale values (middle left) and with scale values estimated from the sample density (middle right). In the former case, redundancy is properly exploited for noise reduction. In the latter case, however, higher sample density leads to higher frequency noise in the reconstruction. Similarly, the Poisson Surface Reconstruction [2013] (right) also suffers from higher frequency noise.*

sity is often caused by data redundancy. Being able to detect this redundancy makes the difference between proper noise reduction and reconstructing higher frequency noise.

Conceptually, our method is based on reconstructing an implicit function F from the input samples. F has spatially continuous scale (*floating scale*), i.e., the scale at which surface details are represented by F varies continuously as defined by the scale of the input samples. We then define a discrete, scale-adaptive sampling of F and extract an isosurface corresponding to the zero-level set of F. The implicit function F is constructed as the sum of compactly supported basis functions. But unlike, e.g., Radial Basis Functions [Carr et al. 2001] or Smooth Signed Distance Reconstruction [Calakli and Taubin 2011] our method does not require the solution of a global problem, is computationally tractable, and the implicit function can, given the samples, readily be evaluated. The compact support leads to an approach that reconstructs open meshes and leaves holes in regions where data is too sparse for a reliable reconstruction. This is useful for scenes which cannot be completely captured, such as outdoor scenes. This stands in contrast to methods such as Kazhdan et al. [2006], which perform excellent hole-filling but often hallucinate geometry in incomplete regions, requiring manual intervention.

Our contributions are:

- The reconstruction of a continuous, signed implicit function with spatially continuous scale (*floating scale*) using a simple mathematical formulation,
- a virtually parameter-free approach that selects the appropriate reconstruction scale and automatically adapts the interpolation and approximation behavior depending on the redundancy in the data,
- no costly aggregation of samples in a pre-processing step so that the implicit function can, given the input samples, readily and rapidly be evaluated, and
- an efficient and scalable method that does not require any global operations (such as applying graph cuts or solving large systems of equations).

In the remainder of this paper we first review related work (Section 2). We then formally introduce our surface reconstruction approach (Section 3) and perform experiments on synthetic and real-world data (Section 4). Next, we describe the isosurface extraction (Section 5) and evaluate our approach (Section 6). We finally discuss the limitations of our approach (Section 7) and conclude with an outlook on future work (Section 8).

2 Related Work

We give an overview of closely related surface reconstruction algorithms with a focus on how they handle scale, whether and which parameters they require, and to what extent they use costly global optimizations to reconstruct the final mesh.

Volumetric Range Image Processing (VRIP) [Curless and Levoy 1996] averages surfaces (regardless of scale) in a regular grid using a volumetric approach based on the signed distance function. Averaging a high resolution and a low resolution surface yields an average surface quickly blurring the high resolution information. Our method is similar in that it also uses the weighted average of locally estimated functions to define the implicit surface compactly around the input data. While VRIP's implicit function is approximately a signed distance function, the interpretation of our function is more abstract and values do not represent distances. In contrast to VRIP, (Screened) Poisson Surface Reconstruction [Kazhdan et al. 2006; Kazhdan and Hoppe 2013] uses the density of the samples as indicator for scale. Thus a denser set of samples is assumed to originate from a surface sampled at a higher resolution. However, the sampling rate is not necessarily related to the sample resolution, and an increased sampling rate may simply be caused by data redundancy (see Figure 2). As a consequence, Poisson Surface Reconstruction starts fitting to the sample noise and hallucinates geometric detail. Mesh Zippering [Turk and Levoy 1994] selects a triangulated depth map for each surface region, eroding redundant triangles. It is worth noting that such an approach works with meshes at pixel resolution and is thus, at least in theory, able to select high resolution surface parts and could avoid averaging with low resolution surfaces. In practice Mesh Zippering is fragile and fails in the presence of noise and outliers.

Using basis functions for surface reconstruction is a common approach, e.g., for rendering of atomic structures [Blinn 1982] or in the area of mesh-free particle-based simulation [Yu and Turk 2013]. A scalar field is defined as the sum of radially symmetric or anisotropic basis functions, possibly with finite support, and triangulated or rendered at a fixed isovalue. Radial Basis Functions (RBFs) have been used for surface reconstruction from (oriented) point clouds [Turk and O'Brien 1999] but their work is limited to small problems and closed surfaces. Another inherent difficulty lies in defining off-surface constraints to avoid the trivial solution. Although advances made RBFs much more tractable to real world data in terms of size and handling of noise [Carr et al. 2001], RBF fitting is global in nature and a large linear system of equations must be solved to obtain the parameters of the basis functions. Similarly, Calakli and Taubin [2011] present a variational approach to recon-

struct a smooth signed distance function which requires the global solution of a linear system of equations.

A local approach is presented by Ohtake et al. [2003] who fit local shape functions to oriented points and employ weighting functions to blend together the local representations. The approach requires parameters such as the support radius for fitting the local shape functions and an error threshold that controls the refinement of the hierarchal decomposition. All of these parameters, as well as the choice of the local shape functions, depend on the density, redundancy and noise characteristics of the input samples. Their approach is "multi-scale" in the sense that features are reconstructed at different resolution, however, multi-scale input samples are not considered. The method is related to ours in that it constructs the implicit function as a weighted sum of local functions. In contrast, their functions fit multiple points using local shape priors over an octree hierarchy, whereas our functions are defined on a per-sample basis. Shen et al. [2004] presents an approach based on an implicit moving least-squares formulation. One key distinction to [Ohtake et al. 2003] is that not only point constraints are considered when fitting the input data: Integrated constraints are used over the polygons which allows the method to either interpolate or approximate polygonal data.

Mixed-Scale: Although there exists a wealth of surface reconstruction literature, few authors consider samples at different scales as input. Integrating scale in the reconstruction process allows us to identify and use redundancy to suppress noise, and to distinguish between high and low resolution samples. Given sufficient high resolution information, any amount of additional low resolution information should not degrade the high resolution reconstruction.

Muecke et al. [2011] splat Gaussians for every input sample into a grid to produce a 3D confidence map. They use normalized Gaussians so that every sample contributes the same confidence but, depending on the scale of the sample, distribute the confidence over differently sized regions. The final surface is extracted as the maximum confidence cut through a graph defined by the grid. The downside of this approach is the unsignedness of the map, and the exact maximum of the function cannot be obtained by interpolation. The global graph cut optimization is also a limiting factor. We draw inspiration from this approach in that we also use basis functions whose size change with the sample scale. In contrast, our implicit function is signed, the zero level-set can be triangulated with subvoxel accuracy, and we do not require any global optimization.

Fuhrmann and Goesele [2011] present a multi-scale depth map fusion method. The distance field of triangulated depth maps is rendered into a hierarchical signed distance field and, in contrast to VRIP, only surfaces at compatible scales are averaged. Low resolution information is discarded in regions with sufficient high resolution information. The final surface is extracted as the zero level-set of the implicit function. Although our work is inspired by the same basic idea of reconstructing multi-resolution data, the approaches are quite different. Where Fuhrmann and Goesele [2011] assume triangulated depth maps with known sensor positions as input, we rely on oriented, scale-enabled surface samples. Instead of a discretized representation of the implicit function both spatially and in scale, our implicit function can be evaluated anywhere without interpolation in scale and space, solely from the input samples. Thus scale selection becomes more flexible and is not limited to neighboring octree levels. Like VRIP, Fuhrmann and Goesele [2011] cannot extract surfaces in regions without data. Our implicit function extends beyond the input samples to some degree, which enables us to fill small holes and obtain more complete reconstructions. Finally, our isosurface extraction does not require a global Delaunay tetrahedralization, and is thus more efficient and produces meshes with fewer output triangles.

3 Floating Scale Implicit Function

In this section we describe the choice of our implicit function. We assume that N input samples are given and equipped with a position $\boldsymbol{p}_i \in \mathbb{R}^3$, a normal $\boldsymbol{n}_i \in \mathbb{R}^3$, $\|\boldsymbol{n}_i\| = 1$, and a scale value $s_i \in \mathbb{R}$. Optional attributes are the sample's confidence $c_i \in \mathbb{R}$ and a color $\boldsymbol{C}_i \in \mathbb{R}^3$. We will treat color reconstruction only as subordinate aspect of our work.

In the first step an implicit function $F(\boldsymbol{x}) : \mathbb{R}^3 \mapsto \mathbb{R}$ is defined as the weighted sum of basis functions f_i. Every sample in the input set contributes a single basis function which is parameterized by the sample's position and normal, as well as its scale value. This step does not require any preprocessing and F can readily be evaluated. The final surface is then given as the zero-set of F. In order to make the approach computationally tractable, the basis function weights w_i are compactly supported such that only a small subset of all samples need to be evaluated to reconstruct F at a position $\boldsymbol{x} \in \mathbb{R}^3$. Due to the compact support of the basis functions, the set $\{\boldsymbol{x} \mid F(\boldsymbol{x}) = 0\}$ essentially defines a surface everywhere beyond the support of the samples. We therefore only consider the zero-level set inside the support where the weight function W is strictly positive, i.e.

$$\{\boldsymbol{x} \mid F(\boldsymbol{x}) = 0 \wedge W(\boldsymbol{x}) > 0\}. \tag{1}$$

3.1 Implicit Function

Like many approaches in literature, we reconstruct a signed implicit function which is positive in front of and negative behind the surface (similar to a signed distance function). This function F is defined as a weighted sum of basis functions:

$$F(\boldsymbol{x}) = \frac{\sum_i c_i w_i(\boldsymbol{x}) f_i(\boldsymbol{x})}{\sum_i c_i w_i(\boldsymbol{x})} \qquad W(\boldsymbol{x}) = \sum_i c_i w_i(\boldsymbol{x}) \tag{2}$$

Function f_i and weight w_i are parameterized by the ith sample position $\boldsymbol{p}_i$, normal $\boldsymbol{n}_i$ and scale s_i. The optional confidence c_i essentially scales the weight function and can easily be omitted by setting it uniformly to $c_i = 1$. In the following, without loss of generality, we define f_i and w_i as a one parameter family of functions depending only on the scale s_i of the sample. The position $\boldsymbol{p}_i$ and normal $\boldsymbol{n}_i$ are considered by translating and rotating the input coordinate $\boldsymbol{x}$

$$\boldsymbol{x}_i = R_i \cdot (\boldsymbol{x} - \boldsymbol{p}_i) \tag{3}$$

with rotation matrix $R_i = R(\boldsymbol{n}_i)$ such that $\boldsymbol{x}$ is transformed into the local coordinate system (LCS) of sample i. The LCS is defined such that the sample's position is located in the origin and the normal coincides with the positive x-axis. Because the normal defines the LCS only up to a one dimensional ambiguity it is important that the basis and weight functions f_i and w_i are defined in a *rotation invariant* manner, such that the reconstruction is invariant to the choice of the LCS orthogonal to the normal. Given a rigid transformation T and reconstruction operator $\mathcal{R}$ acting on a point set P, this property ensures that $T(\mathcal{R}(P)) = \mathcal{R}(T(P))$.

3.2 Basis Function

Similar to Muecke et al. [2011], we use basis functions that, for every sample, contribute the same "confidence", or volume, to the implicit function. Depending on the scale of a sample, the volume is distributed over differently sized regions. As basis function f we use the derivative of the Gaussian f_x in the direction of the normal with $\sigma = s_i$ set to the scale of the sample. (We flip the sign of the function because it is defined to be *positive* in front of the surface,

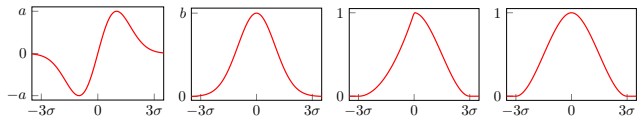

Figure 3: *The 1D components of the basis function f_x, f_y, and weight function w_x and w_y. The peaks are $a = {}^1\!/\!_{\sigma e^{-0.5}}$ and $b = {}^1\!/\!_{\sigma\sqrt{2\pi}}$.*

i.e. in the direction of the *positive* x-axis.) Normalized Gaussians f_y, f_z are used orthogonal to the normal in y and z direction.

$$f_x(x) = \frac{x}{\sigma^2} e^{\frac{-x^2}{2\sigma^2}} \quad f_y(x) = f_z(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-x^2}{2\sigma^2}} \tag{4}$$

Figure 3 illustrates the function components in 1D. This yields the basis function

$$f(\boldsymbol{x}_i) = f_x(x) f_y(y) f_z(z) = \frac{x}{\sigma^4 2\pi} \cdot e^{\frac{-1}{2\sigma^2}(x^2+y^2+z^2)} \tag{5}$$

The function is rotation invariant around the normal because $f_y f_z$ can be rewritten in terms of the distance $\sqrt{y^2+z^2}$ to the normal. The integral of the function's absolute value is 1 and thus every basis function contributes the same volume to the implicit function:

$$\iiint |f(\boldsymbol{x}_i)| \mathrm{d}\boldsymbol{x}_i = \int |f_x(x)| \mathrm{d}x \int f_y(y) \mathrm{d}y \int f_z(z) \mathrm{d}z = 1 \tag{6}$$

Since f_y and f_z are normalized Gaussians, their integrals are 1 by definition. We integrate the absolute function $|f_x|$ because the point-symmetric parts cancel each other out. $|f_x|$ does not require explicit normalization and $\int |f_x| = 1$. Figure 4 (left) illustrates the function in 2D.

3.3 Weighting Function

In the following we design a polynomial weighting function w that has compact support, falls smoothly off to zero and gives more weight to the regions in front of the surface. The justification behind this is related to free space constraints and occlusions as discussed by Curless and Levoy [1996] and Vrubel et al. [2009]: If a sample has been observed, the existence of a surface between the observer and the sample is not possible. Behind the sample, however, we cannot be sure of the existence of a surface and want to reduce the weight quickly. We observe that $f(\boldsymbol{x}_i)$ has negligible influence beyond 3σ, and thus we chose 3σ as the point beyond which the weighting function vanishes. The weighting function

$$w(\boldsymbol{x}_i) \quad = \quad w_x(x) \cdot w_{yz}(\sqrt{y^2+z^2}) \tag{7}$$

is composed of a non-symmetric component in x-direction

$$w_x(x) \quad = \quad \begin{cases} \frac{1}{9}\frac{x^2}{\sigma^2} + \frac{2}{3}\frac{x}{\sigma} + 1 & x \in [-3\sigma, 0) \\ \frac{2}{27}\frac{x^3}{\sigma^3} - \frac{1}{3}\frac{x^2}{\sigma^2} + 1 & x \in [0, 3\sigma) \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

and a rotation invariant component in y- and z-direction

$$w_{yz}(r) \quad = \quad \begin{cases} \frac{2}{27}\frac{r^3}{\sigma^3} - \frac{1}{3}\frac{r^2}{\sigma^2} + 1 & r < 3\sigma \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

$$r \quad = \quad \sqrt{y^2+z^2}. \tag{10}$$

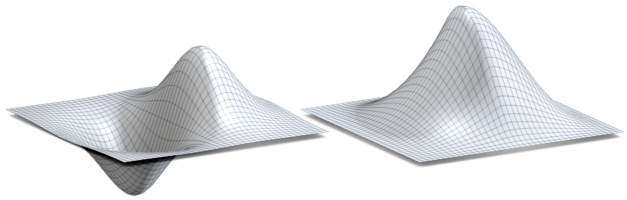

Figure 4: *The basis and weighting functions in 2D in the interval $[-3\sigma, 3\sigma]^2$. Left: $f(x,y) = f_x(x) f_y(y)$. Right: $w(x,y) = w_x(x) w_y(y)$.*

Note that the function w_{yz} is the positive domain of w_x where x is replaced with the distance to the normal r. The individual 1D components of the weighting function are illustrated in Figure 3, and a 2D illustration is shown in Figure 4 (right).

4 Analysis in 2D

There are many possible choices for both basis and weighting function. We chose the Gaussian family of functions as basis function which empirically provides excellent approximation and extrapolation behavior. We will now demonstrate these properties of the implicit function on simple synthetic 2D datasets as well as real world data. In Section 6 we discuss alternative choices for both the basis function and weighting function.

We visualize the implicit function using a color mapping where positive values are colored green and negative values are blue. Bright colors correspond to small values of F (near and also far from the isosurface) and darker colors to large values, such that the isosurface is directly visible in the images. A gray color is used outside of the support, where W is zero. Samples are indicated in red.

4.1 Synthetic Data

When designing a reconstruction algorithm, we are concerned with the interpolation, extrapolation and approximation characteristics of the reconstruction operator. On the one hand the isosurface should pass through the input samples, in particular if the points are sparse and accurate. The implicit function should gracefully fill the gaps between the samples using smooth extrapolation. On the other hand, if many redundant and noisy point samples are available, it should approximate the samples and average out the noise instead of over-fitting the data. The presented formulation of the implicit function automatically adapts to the data as demonstrated by the following 2D experiments.

We provide 2D point samples (of a curve) with normals. In the first experiment the scale is computed for each sample as the average distance to the two nearest neighbors of the sample. We then multiply the computed scale with several factors, see Figure 5. With a small factor, the function interpolates the samples but extrapolation becomes less smooth. With an increasing factor, the reconstruction will approximate the points and provide a smoother extrapolation, but a less accurate interpolation.

In Figure 6 we consider the sampling of a function with added noise to positions and normals. The noise is uniform and about 5% of the bounding box of the samples. As we increase redundancy (adding more noisy samples while keeping the scale of the samples constant), the technique starts to increasingly approximate the data, reducing the noise, until the reconstruction converges towards the original function.

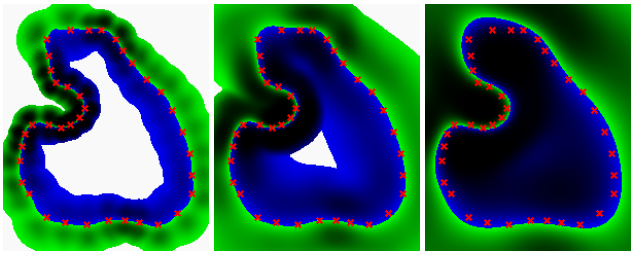

Figure 5: *Reconstructions with increasing scale factor. The larger the factor the more approximative is the reconstruction. Scale factors are* 0.5 *(left),* 1.0 *(middle) and* 2.0 *(right).*

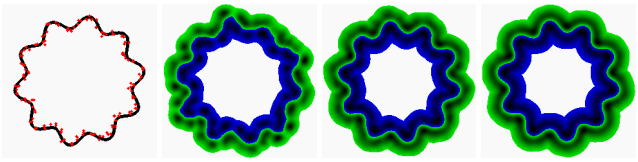

Figure 6: *Reconstructions with increasing redundancy converging towards the original function. The left image shows the original function and 150 noisy samples. The reconstructions have been computed from 50, 500 and 5000 noisy samples, respectively.*

4.2 Real-World Data

In practical cases the noise characteristics of the input data change considerably depending on how the samples have been acquired. Our intention is to demonstrate the ability of our reconstruction operator to handle both clean and extremely noisy datasets without tuning any parameters (such as noise characteristics or the octree level).

In the following experiments we use the Stanford Bunny and the Middlebury Temple dataset (see Section 6 for details how these datasets were created). For each dataset, we intersect the set of samples with a plane and select only samples whose distance to the plane is below a threshold. This yields 2D datasets with 2690 samples for the Bunny, and 157275 samples for the Temple, see Figure 7. While the Stanford Bunny contains clean, range scanned samples, the Middlebury Temple is a very noisy multi-view stereo (MVS) reconstruction with many outliers. The presence of isolated outliers as well as noise in both normals and sample positions lead to many isovalue crossings further away from the true surface. The weighting function, however, indicates which parts of the implicit function are important. In practice, only isosurfaces above a certain weight are extracted, which removes spurious isolated components. Note that this weight threshold is not a parameter to our reconstruction operator. In fact, we postpose cleaning the geometry until after the surface mesh has been extracted.

5 Sampling the Implicit Function

In this section we detail how the implicit function F is efficiently evaluated to extract the isosurface $F(\boldsymbol{x}) = 0$. All input samples are first inserted into an octree data structure according to their scale value. The resulting octree hierarchy prescribes a sampling of F by considering the positions in the corners of the octree leaf nodes. The implicit function is then evaluated at these positions. We call these sample positions *voxels* to distinguish from the input sample points. Finally, the isosurface is extracted from the octree using a variant of the Marching Cubes algorithm.

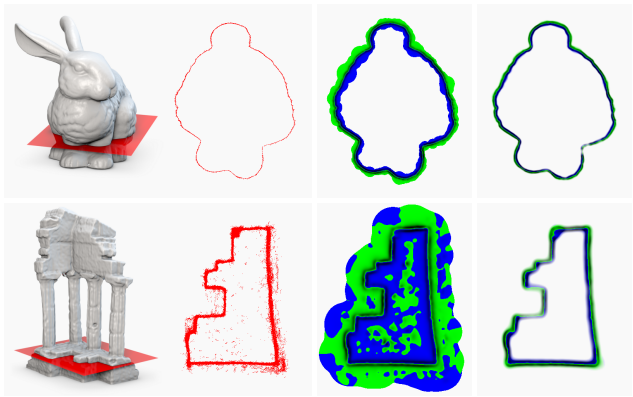

Figure 7: *Reconstruction with real world data. Top row: Slice of the Stanford Bunny. Bottom row: Slice of the Middleburry Temple. Illustration of the slice and the 2D input point set (left), normalized implicit function (middle) and the weighted implicit function (right).*

5.1 Octree Generation

In order to avoid aliasing when sampling the implicit function or evaluating the function too far from the isosurface, we set bounds on the voxel spacing according to the samples' scale values. Recall that sample i has scale value s_i and the radius of the sample's support is $3s_i$. We impose

$$S_\ell \leq s_i < S_{\ell-1} \quad \Leftrightarrow \quad S_\ell \leq s_i < 2S_\ell \tag{11}$$

where ℓ is the octree level at which the sample will be inserted, and S_ℓ is the side length of an octree node at level ℓ (i.e., the voxel spacing). This forces a sample to be inserted into an octree node with a side length S_ℓ of at most s_i but usually smaller:

$$1/2\, s_i < S_\ell \leq s_i. \tag{12}$$

We start with an empty octree without nodes. The first sample i is inserted in a newly created root node with a side length of s_i and centered around the sample's position $\boldsymbol{p}_i$. When inserting subsequent samples, three cases can occur:

1. The new sample is outside the octree. In this case the octree is iteratively expanded in the direction of the new sample until the new sample is inside the octree. The sample is then inserted using cases 2 or 3.
2. The new sample's scale is larger than the scale of the root node. Again, octree expansion is used to create new, larger root nodes until the root has a scale according to (12).
3. The new sample's scale is smaller than the scale of the root node. In this case the tree is traversed, possibly creating new nodes, until a node with a scale according to (12) is reached.

Once a node is determined, the sample is inserted into that node.

5.2 Evaluating the Implicit Function

After inserting all samples in the octree, the octree is prepared for evaluation of the implicit function. We enforce that nodes can be classified into either *inner nodes* or *leafs*. Inner nodes have all eight children allocated, and leafs have no children allocated. The current octree, however, has *mixed nodes* where only some of the children are allocated. We make the octree *regular* by allocating the remaining unallocated children of nodes which are not leafs. This creates new leafs and eliminates mixed nodes.

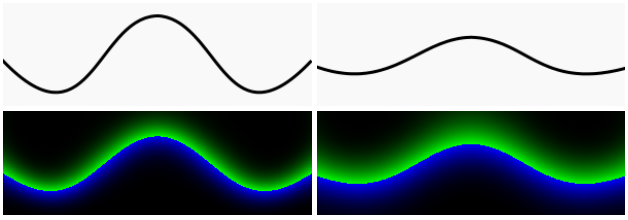

Figure 8: *Synthetic experiment: The top row shows the high resolution (HR) surface (left) and a low-pass filtered version of the high resolution (LR) surface (right). The bottom row shows the result of mixing 100 HR samples with 1000 LR samples (left) and mixing 100 HR samples with 10000 LR samples (right), causing the isosurface to degrade towards the low-pass filtered geometry.*

A list of voxels (points at which the implicit function is evaluated) is created by iterating all leaf nodes. Each leaf node generates eight voxels in the corners of the node. This is a *primal sampling* as opposed to a *dual sampling* where voxels are positioned in the center of the node. Since neighboring leaf nodes share common voxels, every voxel is identified with a unique ID and inserted into a unique set. The implicit function is then evaluated at the voxel positions.

In order to evaluate the implicit function at position $\boldsymbol{x}$, we design an efficient query on the octree that selects only samples which influence the implicit function at $\boldsymbol{x}$: The octree is recursively traversed and for every node a check is performed if the node can possibly contain a sample which influences $\boldsymbol{x}$. From Equation (11) we know that node N contains samples with a scale of at most $2S_N$, where S_N is the side-length of N. Thus, $\boldsymbol{x}$ cannot be influenced by any sample in N if

$$\|\boldsymbol{x} - \text{center}(N)\| - \sqrt{3}\frac{S_N}{2} > 3 \cdot 2S_N. \tag{13}$$

The left side of the inequality is the worst case (smallest) distance from $\boldsymbol{x}$ to any point in the node, and the right side is the largest possible influence radius of a sample in N, i.e. 3 times the largest sample scale $2S_N$. If the inequality holds, the node can be skipped without descending into child nodes. Otherwise, all samples i in the node are considered if $\|\boldsymbol{x} - \boldsymbol{p}_i\| < 3s_i$. The implicit function $F(\boldsymbol{x})$ can then be evaluated according to Equation (2) using all selected samples that influence $\boldsymbol{x}$.

Scale Selection: Limiting the number of samples for evaluating the implicit function will have two effects: It speeds up the algorithm, but more importantly, it can actually improve the quality of the reconstruction. On the one hand, the error to the ground truth geometry is decreased by exploiting redundancy to account for the sample noise. On the other hand, the surface error is increased by mixing samples with different scales: As the formation of a sample usually happens through some kind of integration process over a surface area, every sample corresponds to a low-pass filtered version of the original surface depending on the scale of the sample [Klowsky et al. 2012]. Mixing high and low resolution samples will thus have the effect of degrading the isosurface towards a low-pass filtered geometry. This is demonstrated with the synthetic experiment in Figure 8 (see also the supplemental material).

Our approach to this problem is based on the idea of balancing the positive effect of redundancy (Figure 6) with the negative effect of mixing high and low resolution samples (Figure 8). These two properties are orthogonal to each other: Noise reduction improves precision along the surface normal whereas low resolution samples have an impact along the tangent of the surface. Making a trade-off between the two is not straightforward. Fuhrmann and Goesele [2011] discard low resolution samples by locally selecting the highest supported resolution from the discretized scale-space representation. Similarly, we also discard low resolution samples. We do, however, not discretize scale and can therefore choose a continuous cut-off scale using the following heuristic.

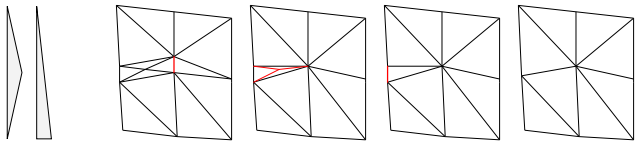

Figure 9: *Two types of degenerated triangles, caps and needles (left). The mesh cleanup procedure (right) with the initial mesh, needles cleanup, caps cleanup, and another needles cleanup. The edges to be collapsed are shown in red.*

To evaluate the implicit function at voxel $\boldsymbol{x}$, consider the set of samples whose (compact) support overlaps with $\boldsymbol{x}$. We now determine a cut-off scale value $s_{\max}$ and only consider samples i with $s_i < s_{\max}$ to reconstruct the implicit function at $\boldsymbol{x}$. Conceptually, we define $s_{\max} = s_{\boldsymbol{x}} \cdot f_{\text{noise}}$, where $s_{\boldsymbol{x}}$ is a reference scale and f_{noise} can be chosen according to the noise properties of the data. In our implementation, the reference scale $s_{\boldsymbol{x}}$ is chosen as a robust 10th percentile of those scale values affecting $\boldsymbol{x}$. (Finding the nth percentile is a linear operation and does not require sorting all samples.) We set $f_{\text{noise}} = 2$ in all of our experiments.

5.3 Isosurface Extraction

At this point, the samples are no longer required and the isosurface can be extracted from the implicit function defined at the octree voxels. This is, however, more complicated than with a regular grid. In the regular case, each cube can be processed individually using Marching Cubes [Lorensen and Cline 1987] and the result is guaranteed to be watertight. In the case of an octree, however, different decisions are made on either side of a cube face (because of depth disparity in the octree), which leaves cracks in the surface. We use the isosurface extraction algorithm proposed by Kazhdan et al. [2007] which yields a crack-free and highly adaptive mesh directly from the octree hierarchy.

The resulting surface contains many degenerated triangles, which is typical for Marching Cubes-like algorithms. To obtain a well-behaved mesh we apply a simple cleanup procedure, see Figure 9. We first identify needle triangles, which are erased by collapsing the short edge. A check that the normals of adjacent triangles do not change too much prevents topological artifacts. Afterwards cap triangles are removed by collapsing vertices with only three adjacent faces. A final pass of needle removal is performed as new needles may be created by the previous operation. This simple procedure usually reduces the number of triangles in the mesh by about 40%.

5.4 Color Reconstruction

We use a simple approach to evaluate a second implicit function that yields a color value for every position $\boldsymbol{x}$. The implicit function has form (2) but uses simpler basis functions. f_i is replaced with the constant sample color $\boldsymbol{C}_i$ and the weight function w_i is replaced with a narrow 3D Gaussian with $\sigma = 1/5 \cdot s_i$. Here, σ is chosen so small to avoid blurring the color and to obtain a crisp texture. Although this weighting function does not have compact support, the weight evaluated at $\pm 3s_i$ away from the sample is in the order of 10^{-10} and thus negligible.

6 Results

We perform a thorough evaluation of our approach on three types of datasets. In Section 6.1 we compare our results on controlled data with Mesh Zippering [Turk and Levoy 1994] and VRIP [Curless and Levoy 1996]. We use the Middlebury benchmark in Section 6.2 to rank our reconstruction on multi-view stereo data. Finally, in Section 6.3, we show the performance of our algorithm on mixed-scale data. For all datasets we also compare with the quasi-standard reconstruction algorithm, (Screened) Poisson Surface Reconstruction (PSR) [Kazhdan and Hoppe 2013]. Instead of comparing to an exhaustive number of algorithms, we limit ourselves to PSR as one representative algorithm that uses point density to estimate per-sample scale in the reconstruction process. Extensive comparison of PSR with other algorithms has been performed by Kazhdan and Hoppe [2013].

6.1 Range Scanner Data

The availability of both range data and final reconstructions in the Stanford Scanning Repository [2013] allow us to qualitatively compare our reconstructions with those from the website performed with Mesh Zippering [Turk and Levoy 1994] and VRIP [Curless and Levoy 1996]. We obtained the input point sets to our system by aligning the range data using transformations provided by the Stanford Scanning Repository. This yields one mesh per range scan in the global coordinate system. Normals are computed for every vertex from the adjacent triangles. The per-vertex scale value is set to the average length of all edges emanating from the vertex. Connectivity information of the range scans is discarded afterwards.

Stanford Models: Figure 10 compares several reconstructions from the Stanford Scanning Repository [2013] with our own reconstructions. The Stanford Bunny dataset contains 10 range scans, the Dragon 71 and the Armadillo a total of 97 range scans. Our algorithm is able to make use of the redundancy in the data without blurring the result, which reveals details unavailable in the Mesh Zippering and VRIP reconstructions. The surfaces created with PSR look visually very close to our reconstruction, so we omit a visual comparison here. Instead, we provide a quantitative evaluation in Table 1. For this evaluation we split the input point set and use 90% of the samples for reconstruction, and the remaining 10% of the samples to evaluate the RMS error and mean distance to the reconstructed surface. Our method shows performance on par with PSR on these datasets.

Error comparison on Stanford Datasets			
	Bunny	Dragon	Armadillo
RMS (PSR)	1.419789	2.950294	5.527238
RMS (ours)	1.394920	2.930433	5.439365
Mean (PSR)	0.970039	1.560911	1.706402
Mean (ours)	0.911296	1.512578	1.676785

Table 1: *Quantitative evaluation on Stanford datasets. 90% of the samples are used for reconstruction, the remaining 10% for evaluating the mean and RMS distance to the reconstructed surface. The measurements are in units of* 10^{-4}.

Incomplete Data: We now demonstrate the behavior of our method on data with holes and boundaries. Due to the local nature of the basis functions, the implicit function is undefined beyond the support of the samples. Although the implicit function is able to close small gaps in the sampling of the surface, it does not close larger holes. Figure 11 illustrates this behavior on a single range scan of the Stanford Bunny.

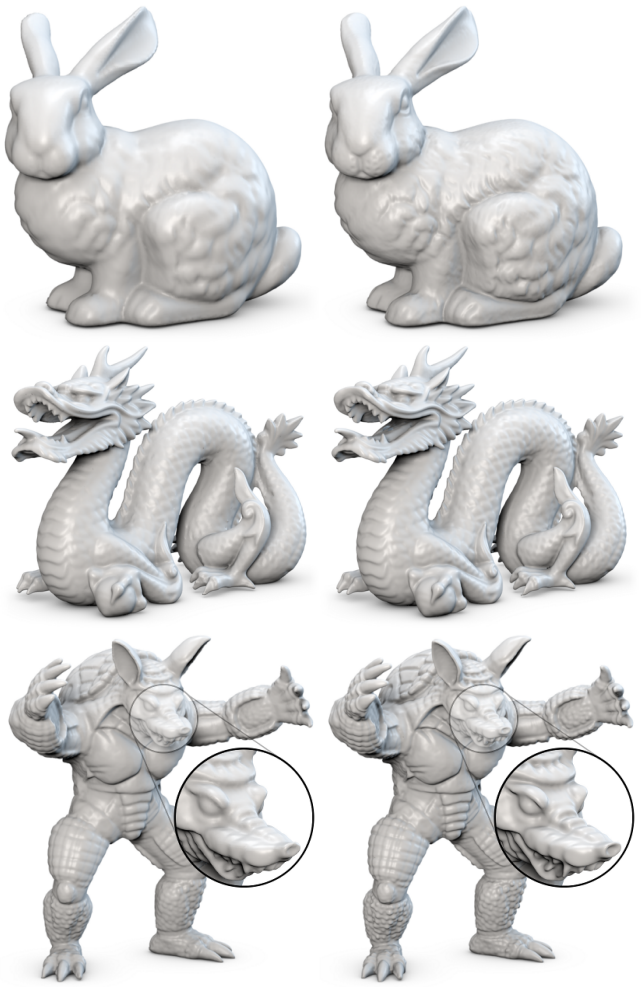

Figure 10: *Reconstruction of the Stanford models. The top row shows the Bunny reconstruction using Mesh Zippering [Turk and Levoy 1994] (left) and our reconstruction (right). The middle and bottom row show the Dragon and Armadillo reconstructed with VRIP [Curless and Levoy 1996] (left) and with our algorithm (right). Our algorithm reveals more detail on all models.*

Michelangelo's David: To showcase the scalability of our approach, we reconstruct the Michelangelo's David provided by the Stanford 3D Scanning Repository, see Figure 12. The dataset is a VRIP reconstruction of non-rigidly aligned range scans and contains a total of 472 million input samples. Although our reconstruction required a considerable amount of memory (114 GB RAM) and processing time (4 hours on a machine with 8 AMD Opteron Quad-Core processors), we were not able to process the data with PSR within a memory limit of 250 GB at any octree level larger than 11. We succeeded in running Streaming PSR [Bolitho et al. 2007] at a level of 14, which took about a day, but still resulted in a very low resolution output mesh.

6.2 Multi-View Stereo Data

Next, we evaluate our approach on multi-view stereo (MVS) data. We produce the input samples to our algorithm in the following way: A depth map is computed for every input image using the freely available MVS implementation of Goesele et al. [2007]. Similar to the range scanner data, scale is computed for every vertex

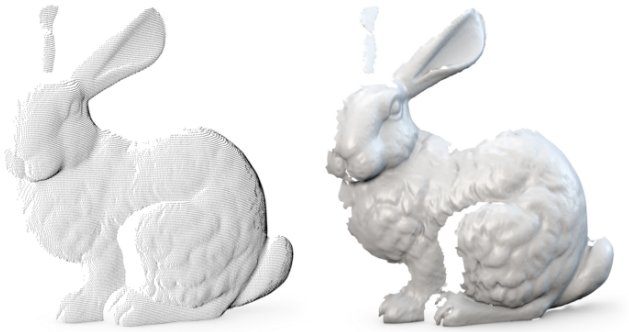

Figure 11: *Reconstruction behavior with incomplete data. The input point set from a single range image (left) and our reconstruction leaving holes in regions with insufficient sampling (right).*

Figure 12: *Reconstruction of Michelangelo's David from 472M input samples. The dataset is kindly provided by the Stanford 3D Scanning Repository.*

in the triangulated depth maps as the average length of all edges emanating from that vertex. But in contrast to the scanner data, every pixel in the depth map actually corresponds a surface region larger than the pixel: Every depth value is the result of a photo-consistency optimization on patches of a certain extent, which has a (low-pass) filtering effect on the reconstructed surface [Klowsky et al. 2012]. We used a patch size of 5x5 pixels and, empirically, found that multiplying the scale with 2.5 (i.e. the "radius" of the patch) yields good results. Finally, the union of all vertices from all depth maps is used as the input point set.

The *Temple Full* dataset from the Middlebury benchmark [Seitz et al. 2006] contains 312 images. All MVS depth maps yield a total of 23 million input samples. Our reconstruction is available as *Fuhrmann-SG14* on the Middlebury evaluation page for quantitative comparison. (Note that the final geometry does not only

Figure 13: *Reconstruction of the Middlebury Temple. The Poisson Surface Reconstruction (left), our colored reconstruction (middle) and shaded reconstruction (right).*

depend on our reconstruction technique, but also on the MVS algorithm). We visually compare our result with PSR at an octree depth of 10 in Figure 13. The PSR reconstruction looks slightly sharper around the edges but also has some geometric artifacts. In contrast to PSR, our algorithm does not require any parameter tuning.

6.3 Multi-Scale MVS Data

Our algorithm gracefully handles both clean and uniform scale datasets, but excels in handling multi-resolution datasets. In the following we perform an evaluation on a multi-scale multi-view stereo dataset where images are taken at various distances to the subject. This yields depth maps with vastly different sampling rates of the surface. In contrast to algorithms using point density, our algorithm produces sharp geometry even in the presence of many low resolution samples, and smooth results in low resolution regions. We use PSR as representative algorithm to demonstrate the shortcomings of traditional methods on multi-scale data. We then compare our results with other multi-scale approaches.

We first register the input images using a Structure-from-Motion software. Similar to the *Temple Full* dataset, we reconstruct dense depth maps using the MVS implementation by Goesele et al. [2007] and use the samples of all depth maps as the input to our algorithm. (More comparisons can be found in the supplemental material that accompanies the paper.)

Elisabeth Dataset: Due to technical limitations (memory consumption and processing time) with PSR, we prepared a smaller dataset called *Elisabeth* to perform the comparison. The dataset contains high resolution regions with detailed carvings and reliefs, as well as regions captured at a much lower resolution, see Figure 14. Although PSR at level 9 produces a smooth result in the low resolution region, it cannot reconstruct the high resolution details. PSR at level 11 reconstructs the fine details but produces a poor result in low resolution regions: It cannot reliably detect redundancy and, due to the too large octree level, reconstructs the noise in the data. A visual comparison of the reconstruction can be found in Figure 15.

Fountain Dataset: We now compare our algorithm on an MVS dataset with a much larger extent. We captured 384 photos of an old fountain yielding a total of 196 million input samples (about half the size of the David dataset). While most of the scene is captured in lower resolution, one of the two lion heads is captured with many close-up photos. Figure 16 shows some input images as well as an overview of the whole reconstruction spanning more than two orders of magnitude differences in scale. Figure 17 shows some geometric details on the fountain.

Figure 14: *Multi-scale reconstruction of the* Elisabeth *dataset. The top row shows our reconstruction with color (left), with shading (middle) and with false coloring of the scale (right). The bottom row shows 5 of 205 input images with varying scale.*

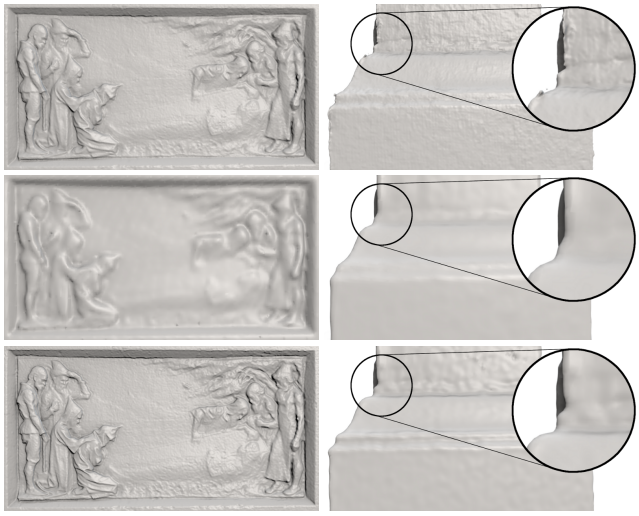

Figure 15: *Comparison with PSR on the* Elisabeth *dataset.* Top row: *Reconstruction with PSR at level 11, which reconstructs details (left) but produces noise in low resolution regions (right).* Middle row: *PSR at level 9 smoothly reconstructs low resolution regions but fails on the details.* Bottom row: *Our method reproduces both high- and low resolution regions appropriately.*

We compare our reconstruction with two other mixed-scale approaches, namely the work by Muecke et al. [2011] (*SurfMRS*) and Fuhrmann and Goesele [2011] (*DMFusion*) in Figure 18. Due to excessive use of memory with SurfMRS on the full point set, we cropped and reconstructed only the detailed region around the fountain for the comparison. Many details are lost in the SurfMRS reconstruction because the graph cut optimization often cuts through details, such as the teeth and the spout at the mouth. While DMFusion leaves small holes in the surface, our algorithm is able to deliver a watertight result. Although all algorithms managed to properly distinguish between low and high resolution regions, our algorithm achieves a more detailed yet smoother reconstruction.

6.4 Alternative Basis and Weighting Functions

In the following we present alternative basis and the weighting functions. In particular, we replace our basis function with signed distance ramps similar to VRIP [Curless and Levoy 1996], and we evaluate the radially symmetric B-spline used in the work of Ohtake et al. [2003] as weighting function.

Figure 16: *The* Fountain *dataset. The top row shows the full colored reconstruction of the site. The bottom row shows 4 of 384 input images depicting the whole site, the fountain and two details on the fountain.*

Figure 17: *Details in the* Fountain *dataset.*

Basis Function: An approximate signed distance function for sample $\boldsymbol{p}_i$ with surface normal $\boldsymbol{n}_i$ is given by

$$f_i(\boldsymbol{x}) = \langle \boldsymbol{x} - \boldsymbol{p}_i \mid \boldsymbol{n}_i \rangle. \tag{14}$$

Because this function does not attenuate orthogonal to the surface normal, it results in a smoother implicit function but less accurate sample interpolation. The integral of the function is unbounded with a constant slope in the direction of the normal, which results in large values if evaluated far away from the surface. This aspect makes the function less useful in multi-scale scenarious because low-resolution samples tend to dominate the implicit function and degrade geometric details. This is demonstrated in Figure 19, which shows a high-resolution region of a large multi-scale dataset.

Weighting Function: While we advocate the use of a weighting function with non-symmetric behavior in the direction of the

Figure 18: *Comparison of details in the* Fountain *dataset using SurfMRS (left), DMFusion (middle) and our result (right).*

normal, simpler choices are possible. Ohtake et al. [2003] use the compactly supported, radially symmetric, quadratic B-spline $B(\frac{3}{2}\frac{\|\boldsymbol{x}_i\|}{3\sigma}+1.5)$ with radius 3σ and centered around the origin. For most datasets this weighting function produces very comparable results. However, similar to Vrubel et al. [2009], the non-symmetric weighting function supresses more artifacts caused by noise and outliers, as demonstrated on the Temple dataset in Figure 20.

6.5 Runtime Performance

In this section we report runtime and memory performance of our system. Table 2 lists datasets with the number of input samples, and the time required for the reconstruction. The reconstruction time is split into sampling the implicit function, which consumes most of the time, and isosurface extraction. We also report the peak memory usage of the system, which is measured as the maximum resident memory size of the process. All benchmarks are performed on a Intel Xeon Dual CPU system with 6×2.53GHz cores per CPU. The reported wall time for evaluating the implicit function uses all cores. Isosurface extraction, however, is limited to a single core.

Dataset Name	Number of Samples	Recon. Time	Peak Memory	Output Vertices
Bunny	362 K	30s + 9s	320 MB	277 K
Dragon	2.3 M	83s + 17s	603 MB	455 K
Armadillo	2.4 M	63s + 13s	553 MB	293 K
David	472 M	247m + 38m	114 GB	81.9 M
Temple	22.8 M	5m + 5s	1.96 GB	176 K
Elisabeth	39.3 M	19m + 1m	4.39 GB	2.3 M
Fountain	196 M	178m + 6m	19.9 GB	10.2 M

Table 2: *Runtime performance for various datasets. The timings are broken down into implicit function evaluation and surface extraction. The peak memory is measured as the maximum resident memory size of the process.*

Figure 19: *Comparison of reconstructions using signed distance ramps (left) and our basis function using Gaussians (right). The distance ramps are particularly harmful in mixed-scale datasets.*

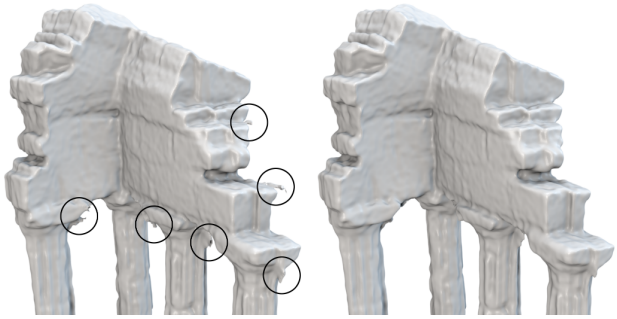

Figure 20: *Reconstruction using the radially symmetric B-Spline weighting function (left) and our non-symmetric weighting function (right). Our weighting function produces less artifacts caused by noise and outliers in the input data.*

7 Discussion

Our approach requires normals and scale information for every input sample. Several approaches for estimating normals have been proposed, e.g. by Hoppe et al. [1992] and Dey et al. [2005]. Scale values, however, cannot reliably be inferred without information about the formation of the samples. In the special (but unlikely) case, where the sample density is globally related to the scale of samples (which is assumed in many methods), the scale values can be computed from the sample spacing, for example using the average distance to the k nearest neighbors. If this is not the case, the estimation of scale will fail, and surface reconstruction can produce undesirable results. In particular, the algorithm looses the ability to exploit redundancy for noise reduction and thus reconstructs high frequency noise, as demonstrated in Figure 2.

Although our implementation scales well to huge datasets and the runtime performance is competitive with state-of-the-art methods, sampling the implicit function is a time-consuming step because the Gaussians which we use as basis functions are expensive to evaluate. Even though increasing redundancy does not considerably increase memory consumption, it does increase computation time. The reason is that for every sampling point $\boldsymbol{x}$, more samples influence $\boldsymbol{x}$ and more basis functions need to be evaluated.

Due to the local nature of our algorithm, the implicit function is not defined beyond the support of the samples. Although our approach is able to close small gaps in the surface sampling, it cannot close larger holes and leaves these regions empty. This is suitable for open scenes or geometric objects which are only partially captured. On the other hand, this behavior stands in contrast to many global approaches which perform excellent hole-filling but often hallucinate low resolution geometry in incomplete regions, requiring manual intervention.

8 Conclusion

We presented a point-based surface reconstruction method that considers the scale of every sample and enables an essentially parameter-free algorithm. It can handle both very redundant and noisy as well as controlled datasets without any parameter tuning. This flexibility comes at the price of providing a scale value for every input sample, which is typically easily obtained. The method has been shown to compute highly detailed geometry, gracefully degrades given imperfect input data such as noisy points and normals, outliers, large holes or varying point density. The mathematical concept behind the approach is very simple and will likely inspire more research in this direction. For example, studying the impact of various basis functions on the reconstruction properties can lead to new reconstruction operators.

We believe that the approach is particularly well suited for out-of-core implementation and distributed reconstruction because of the local nature of our formulation. We would like to investigate this direction in future work. This opens the door for high-quality city-scale surface reconstruction projects, impossible with current state-of-the-art approaches.

Acknowledgements

Part of the research leading to these results has received funding from the FP7 Framework Programme under grant agreements ICT-323567 (HARVEST4D), ICT-611089 (CR-PLAY) and the DFG Emmy Noether fellowship GO 1752/3-1.

References

BLINN, J. F. 1982. A Generalization of Algebraic Surface Drawing. *ACM Transactions on Graphics 1*, 3, 235–256.

BOLITHO, M., KAZHDAN, M., BURNS, R., AND HOPPE, H. 2007. Multilevel Streaming for Out-of-core Surface Reconstruction. In *Proc. SGP*, 69–78.

CALAKLI, F., AND TAUBIN, G. 2011. SSD: Smooth Signed Distance Surface Reconstruction. *Computer Graphics Forum 30*, 7, 1993–2002.

CARR, J., BEATSON, R., CHERRIE, J., MITCHELL, T., FRIGHT, W., AND MCCALLUM, B. 2001. Reconstruction and Representation of 3D Objects with Radial Basis Functions. In *Proc. SIGGRAPH*, 67–76.

CURLESS, B., AND LEVOY, M. 1996. A Volumetric Method for Building Complex Models from Range Images. In *Proc. SIGGRAPH*, 303–312.

DEY, T. K., LI, G., AND SUN, J. 2005. Normal Estimation for Point Clouds: A Comparison Study for a Voronoi Based Method. In *Eurographics Symposium on Point-Based Graphics*, 39–46.

FUHRMANN, S., AND GOESELE, M. 2011. Fusion of Depth Maps with Multiple Scales. In *Proc. SIGGRAPH Asia*, 148:1 – 148:8.

GOESELE, M., SNAVELY, N., CURLESS, B., HOPPE, H., AND SEITZ, S. M. 2007. Multi-View Stereo for Community Photo Collections. In *Proc. ICCV*.

HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. 1992. Surface Reconstruction from Unorganized Points. In *Proc. SIGGRAPH*, 71–78.

KAZHDAN, M., AND HOPPE, H. 2013. Screened Poisson Surface Reconstruction. *ACM Transactions on Graphics 32*, 3, 29.

KAZHDAN, M., BOLITHO, M., AND HOPPE, H. 2006. Poisson Surface Reconstruction. In *Proc. SGP*, 61–70.

KAZHDAN, M., KLEIN, A., DALAL, K., AND HOPPE, H. 2007. Unconstrained Isosurface Extraction on Arbitrary Octrees. In *Proc. SGP*.

KLOWSKY, R., KUIJPER, A., AND GOESELE, M. 2012. Modulation Transfer Function of Patch-based Stereo Systems. In *Proc. CVPR*.

LEVOY, M., PULLI, K., CURLESS, B., RUSINKIEWICZ, S., KOLLER, D., PEREIRA, L., GINZTON, M., ANDERSON, S., DAVIS, J., GINSBERG, J., SHADE, J., AND FULK, D. 2000. The Digital Michelangelo Project: 3D Scanning of Large Statues. In *Proc. SIGGRAPH*, 131–144.

LORENSEN, W. E., AND CLINE, H. E. 1987. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *Proc. SIGGRAPH 21*, 5, 79–86.

MUECKE, P., KLOWSKY, R., AND GOESELE, M. 2011. Surface Reconstruction from Multi-Resolution Sample Points. In *Proc. of Vision, Modeling, and Visualization*, 398 – 418.

OHTAKE, Y., BELYAEV, A., ALEXA, M., TURK, G., AND SEIDEL, H.-P. 2003. Multi-level partition of unity implicits. In *Proc. SIGGRAPH*, 463–470.

SEITZ, S. M., CURLESS, B., DIEBEL, J., SCHARSTEIN, D., AND SZELISKI, R. 2006. A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In *Proc. CVPR*, 519–528.

SHEN, C., O'BRIEN, J. F., AND SHEWCHUK, J. R. 2004. Interpolating and Approximating Implicit Surfaces from Polygon Soup. In *Proc. SIGGRAPH*, 896 – 904.

STANFORD SCANNING REPOSITORY, 2013. `http://graphics.stanford.edu/data/3Dscanrep/`.

TURK, G., AND LEVOY, M. 1994. Zippered Polygon Meshes from Range Images. In *Proc. SIGGRAPH*, 311–318.

TURK, G., AND O'BRIEN, J. F. 1999. Variational Implicit Surfaces. Tech. rep., Georgia Institute of Technology.

VRUBEL, A., BELLON, O., AND SILVA, L. 2009. A 3D Reconstruction Pipeline for Digital Preservation of Natural and Cultural Assets. In *Proc. CVPR*.

YU, J., AND TURK, G. 2013. Reconstructing Surfaces of Particle-Based Fluids using Anisotropic Kernels. *ACM Transactions on Graphics 32*, 1, 5:1–5:12.