Visibility-Consistent Thin Surface Reconstruction Using Multi-Scale Kernels

SAMIR AROUDJ, PATRICK SEEMANN, FABIAN LANGGUTH,
STEFAN GUTHE, and MICHAEL GOESELE, TU Darmstadt, Germany

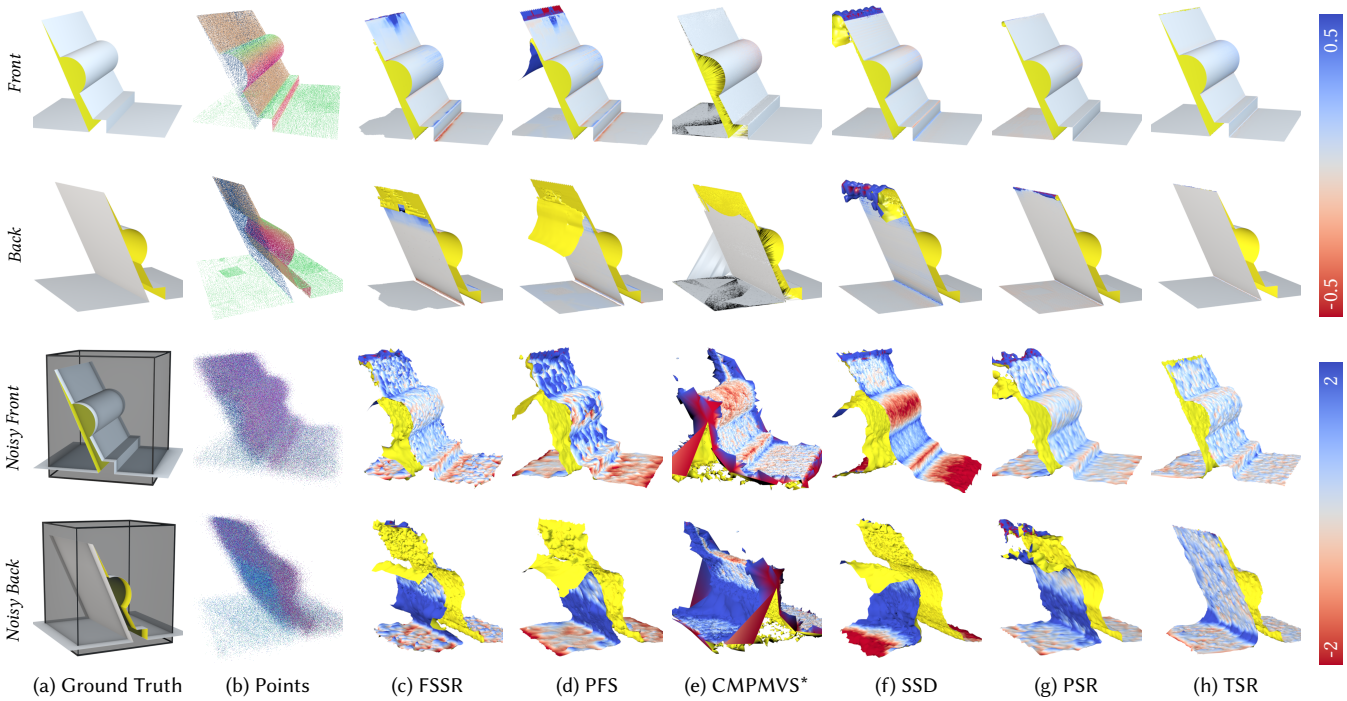


Fig. 1. Reconstructing the synthetic wedge with and without noise on the input. Reconstructions are cropped for visibility of the center part as marked by the gray box in (a). All prior methods show at least minor artifacts—even in the noise-free case—and fail catastrophically for noisy data whereas our method (TSR) reconstructs the overall shape correctly. Note the 4× difference in color scale. Back faces are drawn in yellow. See Sect. 6 for methods and implementations.

One of the key properties of many surface reconstruction techniques is that they represent the volume in front of and behind the surface, e.g., using a variant of signed distance functions. This creates significant problems when reconstructing thin areas of an object since the backside interferes with the reconstruction of the front. We present a two-step technique that avoids this interference and thus imposes no constraints on object thickness. Our method first extracts an approximate surface crust and then iteratively refines the crust to yield the final surface mesh. To extract the crust, we use a novel observation-dependent kernel density estimation to robustly estimate the approximate surface location from the samples. Free space is similarly estimated from the samples' visibility information. In the following refinement, we determine the remaining error using a surface-based kernel interpolation that limits the samples' influence to nearby surface regions with similar orientation and iteratively move the surface towards its true location. We demonstrate our results on synthetic as well as real datasets reconstructed using multi-view stereo techniques or consumer depth sensors.

CCS Concepts: • **Computing methodologies** → **Mesh geometry models**; *Reconstruction*; *Kernel methods*;

Additional Key Words and Phrases: multi-scale surface reconstruction

ACM Reference Format:
Samir Aroudj, Patrick Seemann, Fabian Langguth, Stefan Guthe, and Michael Goesele. 2017. Visibility-Consistent Thin Surface Reconstruction Using Multi-Scale Kernels. *ACM Trans. Graph.* 36, 6, Article 187 (November 2017), 13 pages.
https://doi.org/10.1145/3130800.3130851

0730-0301/2017/11-ART187 $15.00
https://doi.org/10.1145/3130800.3130851

1 INTRODUCTION

Surface reconstruction from oriented point sets has been a very active research topic in recent years. Both scanning devices and many image-based reconstruction algorithms such as multi-view stereo (MVS) [Seitz et al. 2006] generate only per-view depth information

that has to be merged into a globally consistent model. This is often difficult as the input point samples contain noise and outliers. Additionally, in uncontrolled, real-world settings the scene is often sampled irregularly, which results in varying sample density and resolution. Recent surface reconstruction techniques focus therefore, e.g., on multi-scale reconstruction [Fuhrmann and Goesele 2014], the integration of visibility constraints [Jancosek and Pajdla 2011] or robust global optimization [Calakli and Taubin 2011; Kazhdan and Hoppe 2013; Ummenhofer and Brox 2015]. However, for very thin structures, all of these algorithms show at least minor artifacts in the noise-free case and fail catastrophically for noisy input data as demonstrated in Fig. 1.

A key problem for thin structures is typically the assumption that objects need to have an *inside*, i.e., some amount of explicit interior space located behind the observed surface [Ummenhofer and Brox 2015]. Depending on the specific technique and the properties and resolution of the underlying data structure this leads to various artifacts for thin structures as input points potentially influence the reconstruction of unrelated surfaces behind them. Ummenhofer and Brox [2013] therefore proposed a point cloud processing method that yields a clean, point-based representation even for thin objects but does not extract an actual closed surface representation and cannot handle multi-scale data. We combine ideas from their work and modern surface reconstruction methods with kernel-based approaches for sample representation and data interpolation. This yields a novel, general surface reconstruction algorithm called *visibility-consistent thin surface reconstruction (TSR)* that can handle noisy, irregular, multi-scale, oriented point sets as input and is robust enough to reconstruct a consistent surface from complex, outlier-heavy data.

Our main contributions are

- on the theoretical side, an observation-dependent kernel density estimation approach, and a general, multi-scale kernel-based interpolation scheme on surfaces,
- a practical multi-scale surface and empty space estimation technique using variable bandwidth kernel density estimation for samples and visibility, and
- a surface reconstruction approach that first extracts a robust crust encapsulating the true surface and then refines the crust surface using a multi-scale approach.

One of the key benefits of our method is the fact that spatially close surface samples do not interfere with each other if their geodesic distance on the surface is large or if their normals differ significantly, allowing even the reconstruction of thin surfaces.

2 RELATED WORK

We here discuss surface reconstruction methods for oriented point sets from active or passive 3D reconstruction techniques. We focus specifically on how surfaces and free space are represented.

Early work on surface reconstruction such as mesh zippering [Turk and Levoy 1994] used the input depth maps as underlying surface representation. Overlapping depth maps are combined and averaged in order to achieve robustness to noise. While not explicitly discussed in their paper, interference between front and back surface could be avoided by taking surface normals into account. Merell et al. [2007] operate in a real-time streaming approach. They first clean depth maps using a visibility-based approach and then extract a final surface. Since they take only temporarily close depth maps into account, they will in practice never treat two sides of a thin surface simultaneously. Ball pivoting [Bernardini et al. 1999] uses the underlying point samples directly and virtually rolls a ball over the point cloud, creating a triangle whenever the ball rests on three input points. Two sides of a surface can thus never interfere with each other but noisy samples lead to an overestimation of the surface. Digne et al. [2011] first iteratively smooth the input points of accurate, non-oriented laser scan data. They then triangulate the smoothed points using ball pivoting and restore the original positions of the triangulated points to produce a final mesh that exactly interpolates the input points. Like ball pivoting, their approach suffers from noisy samples. Chen and Medioni [1995] inflate a simple initial mesh inside the object until it is bounded by sample points. They also use a set of refinement rules to locally adapt to the sample density. However, the approach does not robustly handle noise and self intersections as well as incompletely sampled outdoor scenes. Using active contour models [Kass et al. 1988] could potentially solve this problem by approximating the most likely surface. This spline-based 2D approach is, however, not easily extended to general 3D surface reconstruction. Moreover, all of the above approaches are unable to handle data with varying sample scale appropriately.

Signed distance functions (SDF) were originally introduced by Hoppe et al. [1992]. They are typically defined in a region near the input data. The surface is implicitly represented as the zero set of the SDF. Curless and Levoy [1996] used discretely sampled SDFs (nowadays also called truncated signed distance functions, TSDF) in their volumetric range image processing approach. Visibility information can be explicitly integrated by carving away surfaces along the line of sight of the capture device. Curless [1997] later demonstrated in his thesis limitations of the method including its inability to handle thin surfaces correctly. The TSDF is nevertheless widely used, e.g., in KinectFusion [Newcombe et al. 2011] and many of its follow-up papers or in the multi-scale fusion approach by Fuhrmann and Goesele [2011]. In follow-up work, Fuhrmann and Goesele [2014] use compactly supported basis functions for multi-scale samples and combine them to estimate an approximate multi-scale signed distance function. Ummenhofer and Brox [2015] frame reconstruction as a global optimization but still use a TSDF (plus normals) as underlying representation. Due to the limitations of TSDFs and the underlying discrete sampling pattern, none of the above approaches are able to handle thin surfaces. Mücke et al. [2011] propose a multi-scale extension of earlier work by Hornung and Kobbelt [2006] using unsigned distance functions. Starting from an initial coarse octree-based crust they extract a final mesh via an iterative multi-scale global graph cut optimization, steered via a global confidence map to handle multi-scale input. In practice, the approach is, however, unable to handle thin structures as the graph cut needs as input a crust with inside outside segmentation, requiring a prohibitively high crust resolution for thin structures. Also the crust is not reliably detected when there are outlier samples in free space.

Poisson Surface Reconstruction (PSR) [Kazhdan and Hoppe 2013] recovers an indicator function, separating space into interior and exterior regions. It can be extended to also incorporate visibility

constraints [Shan et al. 2014]. Calakli and Taubin [2011; 2012] use a similar continuous approach to avoid undefined gradients at the boundary of interior and exterior surface regions. All of these approaches suffer from finite size of the basis functions so that front and back side of a thin surface interfere and may also have problems with the discrete space sampling.

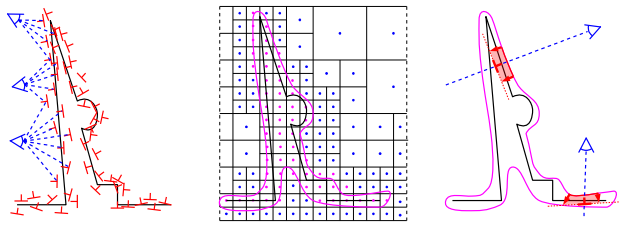
Labatut et al. [2009] propose an approach that uses different kinds of visibility and free space constraints. Like its extensions [Jancosek and Pajdla 2011; Vu et al. 2012], it is based on Delaunay tetrahedralization followed by a graph cut. While this approach is in principle able to handle multiple scales within a scene, it cannot properly handle areas covered by samples with strongly varying scale. Further, thin surfaces may be missed in the graph cut, in particular in the presence of noise, yielding empty space. Interestingly, Vu et al. [2012] augment the basic method with a photoconsistency-based refinement step that can reconstruct thin surfaces as long as they are represented in the output of the basic method. It needs, however, access to the input images to evaluate photo consistency and is thus not applicable to pure oriented point cloud data.

2.1 Reconstructing thin structures

Given an accurate, noise-free, continuous SDF, details such as thin surfaces that are smaller than the grid resolution of the marching cubes algorithm can be reconstructed by switching to the dual problem [Ohtake and Belyaev 2003; Schaefer and Warren 2004]. Since we target noisy, contradicting and discrete input samples instead of an ideal SDF, we cannot employ these techniques directly.

Given the problems thin structures pose and the known shortcomings of existing methods, several approaches emerged that are specifically tailored towards such challenging situations. Yücer et al. [2016b] treat reconstruction as a 2D-3D segmentation problem in densely sampled light fields yielding a visual hull like representation well suited for appropriately aligned thin structures. Yücer et al. [2016a] propose a gradient- and photoconsistency-based light field reconstruction technique that can represent thin structures but surface extraction is again performed using PSR. Since both techniques rely on dense light field data they cannot handle casually captured or multi-scale images as input. Savinov et al. [2016] jointly reconstruct 3D objects and a semantic labeling from the input images using exact ray potentials. Solving the energy minimization problem of semantic labeling during reconstruction implicitly supports thin structures but requires images as input data. Further, since their optimization steps are based on a regularly sampled volume, they are unable to fully support multi-scale reconstructions.

The most closely related work on thin surfaces is Ummenhofer and Brox [2013]. They operate directly on a global point cloud as input and formulate the reconstruction as a particle simulation which pulls nearby samples with similar normals towards a common plane. Opposing samples are pushed apart to remove self-intersections, introducing a clear bias in the representation. Most importantly, their method only creates a cleaner point cloud but does not extract a closed surface mesh.



(a) Input (b) Crust (c) Refinement

Fig. 2. Data and method overview. (a) Ground truth geometry (black) and input data consisting of samples (red) and (selected) views (blue). (b) Crust (magenta) extraction based on occupancy scalar field with varying resolution. (c) Surface refinement to reduce the error (red) of the surface estimate.

3 OVERVIEW

We propose a general surface reconstruction algorithm, that outputs a closed surface $\mathcal{S}$ for an input oriented point cloud with linked views as shown in Fig. 2(a). We define a view v^i as an abstract scene capture taken from a single center of projection $\mathbf{p}_v^i$. In practice, a view is a registered image, a range scan or similar. To achieve high robustness and accuracy without trading off generality, we propose a coarse-to-fine two-step approach. It handles noisy input surface samples of varying scale for a vast variety of scenes including, in particular thin structures. Both steps use compactly supported weighting functions (kernels) to account for the varying sample scales and to cancel out noise while avoiding undue sample interference. To evaluate these kernels at proper detail level, we build a hierarchical space partitioning structure automatically steered by the input samples.

For robustness, we first coarsely estimate the scene space that possibly contains a surface, see Fig. 2(b). We find an intermediate global scalar function with varying resolution using Bayesian kernel density estimation (KDE). Our KDE models both the space occupied by samples as well as their visibility and the resulting free space constraints. We call this interim implicit representation occupancy scalar field. We then extract a surface crust C from this field, which overestimates and encapsulates the final output surface $\mathcal{S}$. In essence, C helps to robustly bootstrap reconstructing $\mathcal{S}$.

We then refine the crust C to become the final surface $\mathcal{S}$, see Fig. 2(c). This multi-scale surface refinement is a constrained non-convex optimization problem. In each iteration, our algorithm shifts the current surface estimate $\mathcal{S}$ to reduce its error computed from all input samples using interpolation with kernels based on geodesic distances. Importantly, $\mathcal{S}$ constrains the optimization by separating spatially close kernels associated with different surface parts.

3.1 Input and notation

The input for our algorithm is a set of views $\{v^i\}$ with their positions (projection centers) $\{\mathbf{p}_v^i \in \mathbb{R}^3\}$ and a point cloud of N surface samples $\{s^j\}$. Each sample s^j consists of a sample position $\mathbf{p}_s^j \in \mathbb{R}^3$, normal $\mathbf{n}_s^j \in \mathbb{R}^3$, optional confidence $c_s^j \in \mathbb{R}$ and scale $\delta_s^j \in \mathbb{R}$. Moreover, each sample s^j was reconstructed for a reference view v_{ref}^j and seen from a set of views $V^j \subset \{v^i\}$ including the reference view.

We call a pair of a sample s^j and a view $v^i \in V^j$ in which s^j is visible a *linked view to sample pair* (v^i, s^j). We use subscripts to mark quantities as belonging to a view v, a sample s, a node n, the occupancy O or a surface $\mathcal{S}$. Further, we denote all 3D positions by a bold variable $\mathbf{p}$, e.g., $\mathbf{p}_s^j$, and 2D surface coordinates of a surface $\mathcal{S}$ by a bold variable $\mathbf{x}$, e.g., $\mathbf{x}^{i,j}$.

Most passive and active reconstruction techniques can naturally assign a view position to each view (e.g., the camera center capturing an image or range scan). Confidences are optional, sample normals and scales can be estimated in a preprocessing step [Fuhrmann et al. 2014]. We follow Fuhrmann et al. [2014] and define the scale (also called 3D footprint) δ_s^j of s^j as the average 3D distance of s^j to its neighboring samples in the reference view v_{ref}^j. We triangulate each depth map, reproject it into 3D space and average the length of incident 3D edges for each reprojected depth sample. Our surface reconstruction method is therefore generally as applicable as most related work, including [Curless and Levoy 1996; Jancosek and Pajdla 2011; Labatut et al. 2009].

4 CRUST COMPUTATION

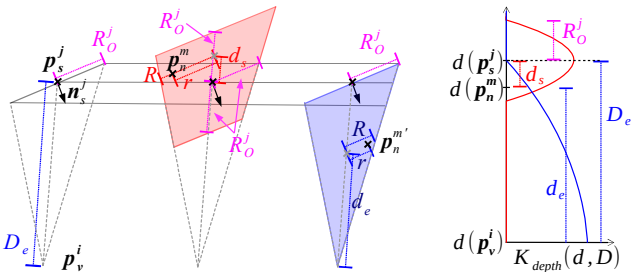
We first estimate two probability density functions (PDFs) within a sparse voxel octree (SVO) using KDE. They represent space close to surfaces inferred from samples and free space derived from visibility. We then restrict the PDF for surface samples by the free space PDF, yielding an occupancy scalar field from which we extract a coarse crust C. This KDE-based crust extraction is very robust and yields, in contrast to previous work, a reliable encapsulation of thin structures even in the presence of noise.

4.1 Sparse voxel octree

For definition of sample-driven and hence spatially varying reconstruction level of detail as in Fuhrmann and Goesele [2014], we insert all samples into a sparse voxel octree, see Fig. 2(b). Similar to Ummenhofer and Brox [2015], each center $\mathbf{p}_n^m \in \mathbb{R}^3$ of a leaf node n^m serves as an evaluation position at which we estimate an implicit scene representation, the scalar occupancy field. A sample s^j is inserted into an octree node n^m if n^m contains the sample position $\mathbf{p}_s^j$ and has a sufficiently small side length $l_n^m \in \mathbb{R}$ with $l_n^m \leq h_{\text{SVO}} \cdot \delta_s^j < 2l_n^m$. The user defined constant $h_{\text{SVO}} \in \mathbb{R}$ controls the overall scene sampling density. Since we evaluate the occupancy field at leaf node centers, we need to ensure that there are leaf node centers close enough in front of and behind the plane of a sample patch s^j. If necessary, we refine the SVO until we create evaluation positions $\{\mathbf{p}_n^m\}$ within radius $R_{\text{SVO}}^j \in \mathbb{R}$ fulfilling a similar criterion $R_{\text{SVO}}^j \leq h_{\text{SVO}} \cdot \delta_s^j$ on both sides of the sample patch plane. We balance the tree to have a maximum depth difference of one between directly neighboring leaf nodes as in Ummenhofer and Brox [2015]. This yields a smoothly varying sampling grid for our PDF estimations. Finally, we compute the dual of the SVO [Schaefer and Warren 2004] for later crust surface extraction.

4.2 Probability Density Functions

One of the key aspects in kernel density estimation is the choice of a suitable bandwidth for the kernels. It determines their spatial extend and in turn their smoothing properties. There has been extensive research in the machine learning community regarding this aspect. An important approach is spatially varying the kernel bandwidth depending on the kernel center or kernel evaluation position [Sheather and Jones 1991]. Another method is selecting the bandwidth h to fulfill an optimality criterion, such as minimizing the expected L_2 loss between the unknown density f and its kernel density estimator $\bar{f}$ (Variable KDE, [Terrell and Scott 1992]). In contrast, to represent individual sample scales, multi-scale surface reconstruction techniques typically vary the size of the corresponding basis functions or they (less explicitly) define the per-sample scale via the depth at which a sample is represented in a hierarchical data structure. Motivated by this, we generalize this per-sample variation and not only employ it for interpolation or sample splatting as in previous work but also for KDE. In particular, we select the bandwidth of each individual kernel based on the scale of the observation it represents. Importantly, an observation is not only a mere surface sample since sample scales also determine free space kernel bandwidths. We call this observation-dependent KDE (ODKDE).



(a) Sample, kernel domains (b) k_{depth}

Fig. 3. Implicit scene representation via kernels. (a) Sample with position $\mathbf{p}_s^j$ and normal $\mathbf{n}_s^j$. Surface (red) and empty space (blue) kernel domains consisting of discs parallel to the sample plane. The kernel parameters for an evaluation position $\mathbf{p}_n^m$ are defined by the intersection (gray cross) of cone axis and the disc through the node center $\mathbf{p}_n^m$. (b) The 1D depth kernels.

Our models for sample and empty space observations generally build on a kernel radius $R_O^j = h_O \cdot \delta_s^j$ at the samples' locations for KDE, which is proportional to the sample scale δ_s^j and a user defined smoothing factor h_O (similar to Fuhrmann and Goesele [2014]). We model both oblique circular cone kernel domains for free space and for surfaces by sweeping circular discs with projected radius $R = \text{proj}(v^i, R_O^j)$ along the cone center axis. All such cone discs are orthogonal to the respective sample normal $\mathbf{n}_s^j$, see Fig. 3(a).

We design the crust around the true surface as a Bayesian decision boundary. C should conservatively enclose the true surface without introducing topological artifacts such as wrongly filling gaps. Thus, we define two corresponding classes: $C^{\mathcal{E}}$ for empty space and $C^{\mathcal{S}}$ for space close to a surface. Note that $C^{\mathcal{S}}$ does not represent the space inside objects. We assign each leaf node center $\mathbf{p}_n^m$ to one of these classes using the joint Bayesian probability densities

$$p(\mathbf{p}_n^m, C^{\mathcal{E}}) = p(C^{\mathcal{E}}) \cdot p(\mathbf{p}_n^m | C^{\mathcal{E}}) \quad (1)$$

$$p(\mathbf{p}_n^m, C^{\mathcal{S}}) = p(C^{\mathcal{S}}) \cdot p(\mathbf{p}_n^m | C^{\mathcal{S}}) \quad (2)$$

and model our crust as decision boundary $p(\mathbf{p}_n^m, C^{\mathcal{E}}) = p(\mathbf{p}_n^m, C^{\mathcal{S}})$. The class conditionals $p(\mathbf{p}_n^m|C^{\mathcal{E}})$ and $p(\mathbf{p}_n^m|C^{\mathcal{S}})$ are PDFs calculated using ODKDE. The *emptiness* PDF models empty space between all linked pairs with $M = \left|\{(v^i, s^j)\}\right|$:

$$p(\mathbf{p}_n^m|C^{\mathcal{E}}) = \frac{1}{M} \sum_{(v^i,s^j)} k_e^{i,j}(\mathbf{p}_n^m) \tag{3}$$

Each kernel $k_e^{i,j} : \mathbb{R}^3 \mapsto \mathbb{R}$ maps a complete oblique circular view cone to emptiness weights, see Fig. 3(a). The *surfaceness* PDF

$$p(\mathbf{p}_n^m|C^{\mathcal{S}}) = \frac{1}{M} \sum_{(v^i,s^j)} k_s^{i,j}(\mathbf{p}_n^m) \tag{4}$$

represents the probability density of having surface samples within the vicinity of leaf node centers. Each sample s^j is similarly represented by a kernel $k_s^{i,j} : \mathbb{R}^3 \mapsto \mathbb{R}$ within a cut oblique circular cone centered at s^j.

Each 3D kernel consists of a 2D kernel, k_{disc}, with a respective projective circular disc as domain multiplied by a 1D kernel, k_{depth}, with the corresponding cone axis as domain. More specifically, we define them as

$$k_e^{i,j}(\mathbf{p}_n^m) = k_{\text{disc}}(r, R) \cdot k_{\text{depth}}(d_e, D_e) \tag{5}$$

$$k_s^{i,j}(\mathbf{p}_n^m) = k_{\text{disc}}(r, R) \cdot 0.5 \cdot k_{\text{depth}}\left(|d_s|, R_O^j\right) \tag{6}$$

$$k_{\text{disc}}(r, R) = \begin{cases} \frac{2}{\pi R^2}\left(1 - \frac{r^2}{R^2}\right) & r < R \\ 0 & \text{else} \end{cases} \tag{7}$$

$$k_{\text{depth}}(d, D) = \begin{cases} \frac{3}{2D}\left(1 - \frac{d^2}{D^2}\right) & d < D \\ 0 & \text{else.} \end{cases} \tag{8}$$

Fig. 3(a) shows the depths along a cone axis and the radial distance r within a projected circular disc of radius R. For the sampleness kernels $k_s^{i,j}$, we use the same overall shape but within a more restricted support volume. The cones for samples are cut at the beginning and centered at the sample. Further, we reuse the 1D ray kernel k_{depth} for samples but have k_{depth} centered at the samples, mirrored at the sample plane and downscaled for $\int_{-\infty}^{\infty} 0.5 \cdot k_{\text{depth}}(|d_s|, D_s)\, \mathrm{d}d_s = 1$, see Fig. 3(b). The kernels $k_e^{i,j}$ and $k_s^{i,j}$ are compactly supported, non-negative and have a constant volume integral of one and are thus suitable for PDF estimation.

The class priors $p(C^{\mathcal{E}})$ and $p(C^{\mathcal{S}})$ represent a priori probabilities of leaf node centers belonging either to empty space or space near surfaces. They express the generally higher probability of nodes to belong to free space than to space close to a surface. For the priors we first globally sum all cone main axis lengths per class to have $L^{\mathcal{E}}$ and $L^{\mathcal{S}}$. Then we estimate the priors via the total sums as

$$p(C^{\mathcal{E}}) = \frac{L^{\mathcal{E}}}{L^{\mathcal{E}} + L^{\mathcal{S}}} \qquad p(C^{\mathcal{S}}) = \frac{L^{\mathcal{S}}}{L^{\mathcal{E}} + L^{\mathcal{S}}}. \tag{9}$$

Note that we use axis lengths instead of cone volumes as they are more robust for noisy sample normals.

When choosing the exact kernel functions, there are many degrees of freedom to account for different input data properties and correspondingly many kernels are known and used [Adams and Wicke 2009]. Generally, the kernel bandwidths (kernel support ranges) are most crucial in KDE as the bandwidths define the trade-off between overfitting and oversmoothing during PDF estimation. ODKDE strongly alleviates the kernel bandwidth selection problem. In contrast, the choice of the exact kernel shapes is less important. In case of 3D surface reconstruction for which the number of linked pairs M is usually high, different but proper PDF kernels result in almost equal PDF estimates. Further, kernel parameters are partially redundant. For example, flat kernels with small support range are very similar to spiky kernels with large support range and therefore result in very similar PDF estimates.

4.3 Crust extraction

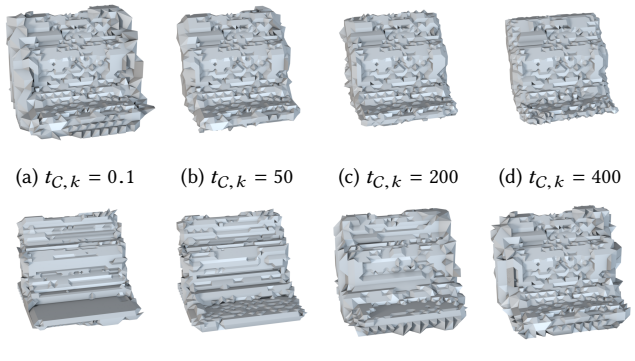
We extract a surface crust C that serves as robust initial estimate and constraint for the non-convex surface optimization detailed in Sect. 5. C is not supposed to be an accurate reconstruction but should be relatively close to the true surfaces, neither miss thin, small or weakly-supported structures nor contain false geometry from outlier clusters. Surfaces and empty space are mutually exclusive, so we can combine the two estimated PDFs of Eqs. 3 and 4 into a single implicit representation, our irregularly sampled occupancy scalar field $\{o(\mathbf{p}_n^m)\}$. For all leaf nodes we define occupancy as

$$o(\mathbf{p}_n^m) = p(\mathbf{p}_n^m, C^{\mathcal{S}}) - p(\mathbf{p}_n^m, C^{\mathcal{E}}) \tag{10}$$

with Bayesian decision boundary at isovalue zero. Further, we mark leaf nodes with too sparse data as uncertain and avoid extracting crust surfaces for them by setting $o(\mathbf{p}_n^m) = -\epsilon$ for all leaf nodes with $\sum_{i,j} k_e^{i,j}(\mathbf{p}_n^m) + k_s^{i,j}(\mathbf{p}_n^m) < t_{C,k}$. The values of $\{o(\mathbf{p}_n^m)\}$ are mainly zero or negative in free space due to the view kernels. Further, they rapidly increase close to the samples and towards the true surfaces. Thus a moderate change of $t_{C,k}$ only noticeably influences the extraction in sparsely sampled areas and defines how much of weakly supported and thus often noisy crust is cut. Given scale-dependent Gaussian noise on depth maps, Fig. 4 shows the crust for the noisy wedge dataset from Fig. 1 with varying crust threshold $t_{C,k}$ (top) and its robustness against increasing noise (bottom). Note that C is truly a closed crust. It consists of an outer and inner side at areas where objects are sufficiently thick.

Using the occupancy field $\{o(\mathbf{p}_n^m)\}$, our algorithm extracts the multi-scale crust C at the Bayesian decision boundary via dual marching cubes [Schaefer and Warren 2004]. Importantly, C always overestimates the true surface and encapsulates it. The spatial surface overestimation is proportional to the sample scales and matches the sampling by the SVO. In particular, the crust overestimation is lower for fine details and higher for coarsely sampled scene areas. Hence, we can still reconstruct holes in objects. Moreover, we circumvent the requirement of having leaf centers $\{\mathbf{p}_n^m\}$ within the exact thin structure by only requiring them to be within C. We therefore do not have the aliasing problems of previous work which explicitly represents filled space.

In addition, our algorithm deletes small isolated geometry components caused by locally concentrated outlier sample clusters using a triangle count threshold similar to Fuhrmann and Goesele [2014] (isle size $|n_{\text{isle}}| < t_{C,\text{isle}}$). Further, the normals of the initial crust C are unreliable due to voxelization artifacts. We strongly smooth C before the first refinement step using the Umbrella operator to have



(a) $t_{C,k} = 0.1$ (b) $t_{C,k} = 50$ (c) $t_{C,k} = 200$ (d) $t_{C,k} = 400$

(e) $\sigma = 0.001$ (f) $\sigma = 0.01$ (g) $\sigma = 0.05$ (h) $\sigma = 0.1$

Fig. 4. Crust extraction behavior for the synthetic wedge from Fig. 1. Noisy input data created by ray tracing the synthetic wedge ground truth mesh and adding depth noise with $d = d + \sqrt{d} \cdot \mathcal{N}(0, \sigma)$. *Top:* Varying the crust threshold $t_{C,k}$ with constant $\sigma = 0.1$ noticeably influences the extraction only in weakly sampled area as desired. *Bottom:* Increasing noise for constant $t_{C,k} = 1$ increases uncertainty of exact surface locations. Note that noise results in crust expansion instead of the removal of thin surfaces.

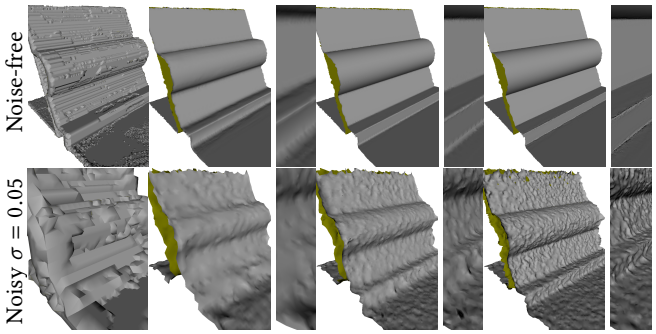
better normals [Taubin 1995]. After that, the optimization starts with the crust C as initial estimate $\mathcal{S}$.

5 SURFACE REFINEMENT

For a detailed final result, we iteratively evolve the initial surface C to an accurate surface $\mathcal{S}$. The functions $\mathbf{p}_\mathcal{S} : \mathcal{S} \mapsto \mathbb{R}^3$ and $\mathbf{n}_\mathcal{S} : \mathcal{S} \mapsto \mathbb{R}^3$ map 2D surface coordinates $\mathbf{x} \in \mathcal{S}$ to 3D coordinates and normals. We define the error E of a given surface $\mathcal{S}$ with respect to all input samples and their linked views

$$E(\mathcal{S}, \mathcal{W}) = \frac{1}{A_\mathcal{S}} \int_\mathcal{S} e_\mathcal{S}(\mathbf{x}, \mathcal{W}) d\mathbf{x} \tag{11}$$

as total reconstruction error normalized by the surface area $A_\mathcal{S}$ via the locally varying surface error function $e_\mathcal{S}$. Since our input consists of only point-based data, we need to robustly define the continuous function $e_\mathcal{S}$ for the whole surface. Thus, we interpolate samples (or other quantities) over $\mathcal{S}$ using multi-scale kernels that define the interpolation weighting functions $\mathcal{W} = \{w_\mathcal{S}^{i,j} : \mathbb{R}^2 \mapsto \mathbb{R}\}$. For each linked pair (v^i, s^j), the weighting function $w_\mathcal{S}^{i,j}$ should spread over the proper local area $\Omega_\mathcal{S}^{i,j} \subset \mathcal{S}$ reconstructed for (v^i, s^j), see Fig. 6. To implement that, we first intersect the ray from the view projection center $\mathbf{p}_v^i$ to the sample position $\mathbf{p}_s^j$ with $\mathcal{S}$, yielding 2D coordinates $\mathbf{x}^{i,j} \in \mathcal{S}$ of the projected sample $(\mathbf{p}^{i,j}, \mathbf{n}^{i,j}) = (\mathbf{p}_\mathcal{S}(\mathbf{x}^{i,j}), \mathbf{n}_\mathcal{S}(\mathbf{x}^{i,j}))$. We then find geodesics (shortest paths on $\mathcal{S}$) from $\mathbf{x}^{i,j}$ to surrounding surface points $\Omega_\mathcal{S}^{i,j}$. We modify the geodesic distances to penalize changes of surface normals from the respective sample s^j and input these modified distances into a standard normalized kernel with implicit domain $\Omega_\mathcal{S}^{i,j}$. Finally, we scale the entire kernel by two confidences quantifying the quality of the matching of (v^i, s^j) to the surface $\mathcal{S}$ and the sample confidence itself to yield the required weighting function $w_\mathcal{S}^{i,j}$. Details are given in Sect. 5.1.



(a) Crust (b) $i = 1$ (c) $i = 3$ (d) $i = 10$

Fig. 5. Surface refinement results. Note the increase in corner crispness and strong initial error reduction, especially from (a) crust C to (b) $\mathcal{S}$, $i = 1$.

Importantly, our kernels spread only within $\mathcal{S}$ and not freely within 3D space. Hence, a given surface $\mathcal{S}$ constrains our optimization problem by restricting the kernels' and thus samples' influence to relevant nearby locations, which is why we call them data-driven surface kernels. Note that we also define the kernel sizes via sample scales enabling multi-scale surface interpolation and optimization. The scene dependent constraints yield, however, a non-convex optimization problem since the surface estimate $\mathcal{S}$ depends on the surface error $E(\mathcal{S}, \mathcal{W})$ and thus on the weights $\mathcal{W}$. The weighting functions $\mathcal{W}$ in turn depend on $\mathcal{S}$. We linearize this problem by iteratively refining $\mathcal{S}$, using the crust C as robust starting point. In each linearization step, we first completely fix $\mathcal{S}$ to calculate the weighting functions $\mathcal{W}$. We then optimize $\mathcal{S}$ while keeping $\mathcal{W}$ completely fixed. Sect. 5.2 details surface update steps using our novel interpolation scheme and Fig. 5 shows an example optimization. How to implement our optimization for a discrete triangle mesh surface follows in Sect. 5.3.

5.1 Interpolation via surface kernels

Similar to previous methods [Curless and Levoy 1996; Fuhrmann and Goesele 2014; Ummenhofer and Brox 2015], we approximate a continuous function from samples via kernel-based interpolation. For all linked pairs $\{(v^i, s^j)\}$, we interpolate point measurements or surface functions $\{f_\mathcal{S}^{i,j}(\mathbf{x})\}$ of these via

$$\bar{f}_\mathcal{S}(\mathbf{x}, \mathcal{W}) = \frac{1}{W_\mathcal{S}(\mathbf{x}, \mathcal{W})} \sum_{(v^i, s^j)} w_\mathcal{S}^{i,j}(\mathbf{x}) f_\mathcal{S}^{i,j}(\mathbf{x}). \tag{12}$$

over the fixed surface $\mathcal{S}$ instead of 3D functions in scene space. $W_\mathcal{S} : \mathcal{S} \mapsto \mathbb{R}$ is the normalization function. $\mathcal{W} = \{w_\mathcal{S}^{i,j}(\mathbf{x})\}$ are corresponding pair-based weighting functions:

$$W_\mathcal{S}(\mathbf{x}, \mathcal{W}) = \sum_{(v^i, s^j)} w_\mathcal{S}^{i,j}(\mathbf{x}), \quad w_\mathcal{S}^{i,j}(\mathbf{x}) = c_s^j \mathcal{N}_\mathcal{S}^{i,j} k_\mathcal{S}^{i,j}(\mathbf{x}). \tag{13}$$

Each $w_\mathcal{S}^{i,j}$ consists of three components. For increased robustness, we use the projection confidence $\mathcal{N}_\mathcal{S}^{i,j} \in \mathbb{R}$ to weight down the respective projection $(\mathbf{p}^{i,j}, \mathbf{n}^{i,j})$ onto $\mathcal{S}$ if it disagrees with s^j (on top of the standard input confidence c_s^j). The kernel $k_\mathcal{S}^{i,j}$ computes geodesic costs from $\mathbf{x}^{i,j}$ to any $\mathbf{x} \in \Omega_\mathcal{S}^{i,j}$ and maps the costs for $\mathbf{x}$ to

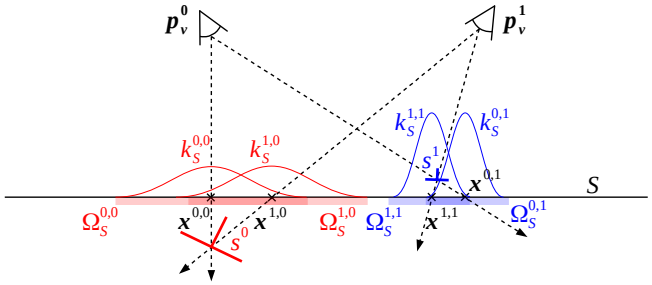


Fig. 6. Surface kernels. Small (blue) and large (red) scale kernels for four linked pairs with curves representing interpolation weights and scale-based influence areas $\Omega_{\mathcal{S}}^{i,j}$ depicted below $\mathcal{S}$.

a normalized scalar as third component of $w_{\mathcal{S}}^{i,j}(\mathbf{x})$. Next, we detail $\mathcal{N}_{\mathcal{S}}^{i,j}$ and $k_{\mathcal{S}}^{i,j}$ and describe the geodesic costs.

Sample projection confidence: Given the globally consistent estimate $\mathcal{S}$ from all pairs $\{(v^i, s^j)\}$, we compute data-driven confidences. We model the confidence $\mathcal{N}_{\mathcal{S}}^{i,j}$ as Gaussian expressing the similarity of projection $(\mathbf{p}^{i,j}, \mathbf{n}^{i,j})$ to its original sample $(\mathbf{p}_s^j, \mathbf{n}_s^j)$ via

$$\mathcal{N}_{\mathcal{S}}^{i,j} = \exp\left(-\tfrac{1}{2}\left(\frac{\alpha^{i,j}}{\sigma_\alpha}\right)^2 - \tfrac{1}{2}\left(\frac{d^{i,j}}{\sigma_d}\right)^2\right). \tag{14}$$

The angle between $\mathbf{n}^{i,j}$ and $\mathbf{n}_s^j$ is denoted $\alpha^{i,j}$. $d^{i,j}$ is the absolute distance of $\mathbf{p}^{i,j}$ to the plane of sample s^j. $\mathcal{N}_{\mathcal{S}}^{i,j}$ moderately punishes noise on samples. In addition to sample noise, there are often outliers: Either the corresponding input samples are very wrong or samples might be projected onto the wrong surface due to the initially coarse $\mathcal{S}$. The latter depends on the quality of $\mathcal{S}$ and quickly decreases during refinement. Conveniently, these outliers receive zero or very low confidences owing to their high dissimilarity.

Data-driven surface kernels: From the projection to $\mathbf{x}^{i,j}$, we spread the kernel $k_{\mathcal{S}}^{i,j} : \mathcal{S} \to \mathbb{R}$ with domain $\mathcal{S}$ as restriction. We define $k_{\mathcal{S}}^{i,j}$ using the 2D distance kernel $k_{\text{spiky}} : \mathbb{R}\times\mathbb{R} \mapsto \mathbb{R}$ from [Adams and Wicke 2009; Müller et al. 2003] and path costs function $\zeta_{\mathcal{S}}$ for costs $r_c = \zeta_{\mathcal{S}}(\mathbf{x}^{i,j}, \mathbf{x})$ from $\mathbf{x}^{i,j}$ to $\mathbf{x}$ and range $R_{\mathcal{S}}^j = h_{\mathcal{S}} \cdot \delta_s^j$:

$$k_{\mathcal{S}}^{i,j}(\mathbf{x}) = k_{\text{spiky}}\left(r_c, R_{\mathcal{S}}^j\right) = \begin{cases} \frac{10}{\pi R_{\mathcal{S}}^j \cdot R_{\mathcal{S}}^j}\left(1 - \frac{r_c}{R_{\mathcal{S}}^j}\right)^3 & r_c < R_{\mathcal{S}}^j \\ 0 & \text{else.} \end{cases} \tag{15}$$

Here, the bandwidth $h_{\mathcal{S}} \in \mathbb{R}$ again controls smoothing. The individually shaped support of $k_{\mathcal{S}}^{i,j}$ is $\Omega_{\mathcal{S}}^{i,j}$ with $\mathbf{x} \in \Omega_{\mathcal{S}}^{i,j}$ if $k_{\mathcal{S}}^{i,j}(\mathbf{x}) > 0$. The normalization of the kernel enables multi-scale input processing: We assign wide spread (large $\Omega_{\mathcal{S}}^{i,j}$) but low amplitude kernels to large samples. Equally, narrow spread (small $\Omega_{\mathcal{S}}^{i,j}$) but high-amplitude kernels represent small scale samples, see Fig. 6. Thus, more accurate small scale input data has a stronger but more local influence so that fine surface details are reconstructed.

Geodesic costs: For a geodesic (shortest surface path) $\psi_{\mathcal{S}} : \mathbb{R} \mapsto \mathcal{S}$ from $\mathbf{x}^{i,j}$ to $\mathbf{x}$ with 3D trajectory $\mathbf{p}(t) = \mathbf{p}_{\mathcal{S}}(\psi_{\mathcal{S}}(t))$ and surface normals $\mathbf{n}(t) = \mathbf{n}_{\mathcal{S}}(\psi_{\mathcal{S}}(t))$, we model the costs $\zeta_{\mathcal{S}} : \mathcal{S}\times\mathcal{S} \mapsto \mathbb{R}$ as

$$\zeta_{\mathcal{S}}(\mathbf{x}^{i,j}, \mathbf{x}) = \int_{\mathbf{x}^{i,j}}^{\mathbf{x}} \phi\left(\cos^{-1}\left\langle \mathbf{n}(t), \mathbf{n}^{i,j}\right\rangle\right) d\mathbf{p}(t) \tag{16}$$

$$\phi(\gamma) = \begin{cases} 1 + \Phi\frac{\gamma}{\Gamma-\gamma} & \gamma < \Gamma \\ \infty & \gamma \geq \Gamma \end{cases} \tag{17}$$

with costs (geodesic length) scaling via the angular deviation penalty $\phi : \mathbb{R} \mapsto \mathbb{R}$. The parameters $\Gamma, \Phi \in \mathbb{R}$ control the range and increase of ϕ. We have $\phi(0) = 1$ and hence no increase for regions oriented like the sample s^j, $\phi(0.5 \cdot \Gamma) = 1 + \Phi$ as control point and $\phi(\gamma)$ increasing to infinity for $\frac{1}{2}\Gamma < \gamma \leq \Gamma$. The latter prevents kernels from spreading over sharp features. Sect. 5.3 details the implementation of $\zeta_{\mathcal{S}}$ via Dijkstra and angle-based edge costs.

5.2 Surface errors and optimization

We interpolate errors for all linked pairs as basis for $E(\mathcal{S}, \mathcal{W})$ from Eq. 11. We define the error for a pair (v^i, s^j) and surface point $\mathbf{x}$ (shown in Fig. 7 in red or blue) explicitly as

$$\mathbf{e}_{\mathcal{S}}^{i,j}(\mathbf{x}) = \frac{\left\langle \mathbf{p}_{\mathcal{S}}(\mathbf{x}) - \mathbf{p}_s^j, \mathbf{n}_s^j \right\rangle}{\left\langle \mathbf{n}_s^j, \mathbf{n}^{i,j} \right\rangle} \mathbf{n}^{i,j}. \tag{18}$$

Shifting $\mathbf{p}_{\mathcal{S}}(\mathbf{x})$ by $-\mathbf{e}_{\mathcal{S}}^{i,j}(\mathbf{x})$ moves it exactly onto the sample plane. Combining these error functions for all pairs with the interpolation (Eq. 12) yields the continuous approximate surface error vector field $\bar{\mathbf{e}}_{\mathcal{S}} : \mathcal{S} \mapsto \mathbb{R}^3$ for the complete input point cloud and surface $\mathcal{S}$:

$$\bar{\mathbf{e}}_{\mathcal{S}}(\mathbf{x}, \mathcal{W}) = \frac{1}{W_{\mathcal{S}}(\mathbf{x}, \mathcal{W})} \sum_{i,j} w_{\mathcal{S}}^{i,j}(\mathbf{x}) \mathbf{e}_{\mathcal{S}}^{i,j}(\mathbf{x}). \tag{19}$$

Since $\mathbf{e}_{\mathcal{S}}^{i,j}(\mathbf{x})$ is unreliable for large angles α between $\mathbf{n}_s^j$ and $\mathbf{n}^{i,j}$, we ignore $\mathbf{e}_{\mathcal{S}}^{i,j}(\mathbf{x})$ if $\alpha \geq \Gamma$. Having non-zero weights for $\mathbf{e}_{\mathcal{S}}^{i,j}(\mathbf{x})$ in the scale-dependent kernel area $\Omega_{\mathcal{S}}^{i,j}$ and $\mathbf{e}_{\mathcal{S}}^{i,j}(\mathbf{x})$ ending at the sample plane results in semi locally fitting $\mathcal{S}$ to the plane of sample s^j. This prevents the error vector field $\bar{\mathbf{e}}_{\mathcal{S}}$ from becoming zero for noisy input samples, even if $\mathcal{S}$ perfectly interpolates the input samples. Thus our interpolation enforces multi-scale smoothness via the varying areas $\{\Omega_{\mathcal{S}}^{i,j}\}$. Further, $\bar{\mathbf{e}}_{\mathcal{S}}$ implicitly models errors for surface normals $\mathbf{n}_{\mathcal{S}}$ deviating from sample normals. It enforces each $\Omega_{\mathcal{S}}^{i,j}$ to match the plane orthogonal to $\mathbf{n}_s^j$.

We gradually move $\mathcal{S}$ in linearized steps. For a fixed error field $\bar{\mathbf{e}}_{\mathcal{S}}$, we update the complete surface via $\forall \mathbf{x} \in \mathcal{S} : \mathbf{p}_{\mathcal{S}}(\mathbf{x}) \to \mathbf{p}_{\mathcal{S}}(\mathbf{x}) - \bar{\mathbf{e}}_{\mathcal{S}}(\mathbf{x}, \mathcal{W})$. To detect refinement convergence via the surface error $E(\mathcal{S}, \mathcal{W})$ of Eq. 11, we finally define the normal error function $e_{\mathcal{S}} : \mathcal{S} \mapsto \mathbb{R}$ which ignores movements of surface points only within $\mathcal{S}$ itself as

$$e_{\mathcal{S}}(\mathbf{x}, \mathcal{W}) = \left\langle \bar{\mathbf{e}}_{\mathcal{S}}(\mathbf{x}, \mathcal{W}), \mathbf{n}_{\mathcal{S}}(\mathbf{x}) \right\rangle. \tag{20}$$

5.3 Update step and practical issues

Using the proposed interpolation scheme and the surface error vector field, we now implement an iterative refinement for triangle meshes. Algorithm 1 describes the structure of a single iteration. The remainder of this section details individual steps.

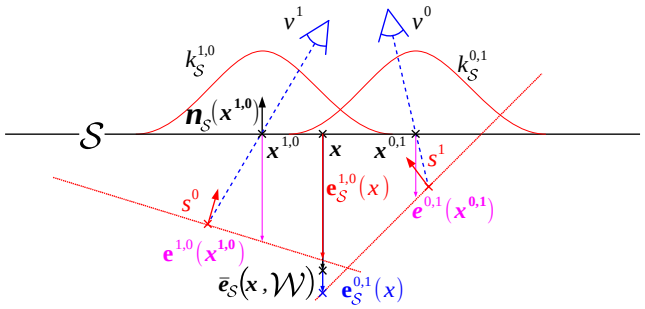


Fig. 7. $\bar{\mathbf{e}}_S(\mathbf{x}, \mathcal{W})$ (black). Point-based errors $\mathbf{e}^{0,1}(\mathbf{x}^{0,1})$ and $\mathbf{e}^{1,0}(\mathbf{x}^{1,0})$ from ray tracing (magenta). Interpolating the surface-dependent errors $e_S^{0,1}(\mathbf{x})$ (blue) and $e_S^{1,0}(\mathbf{x})$ (red) of the view sample pairs (v^0, s^1) and (v^1, s^0) at $\mathbf{x}$.

ALGORITHM 1: Each optimization step computes weights and interpolates sample quantities at each vertex $\mathbf{x}$ and robustly deforms $\mathcal{S}$ while keeping a regular multi-scale triangle distribution.

```
1  collapse() // remove needle triangles from marching cubes or moves
2  ∀(v^i, s^j): (p^{i,j}, n^{i,j}) = traceRay(p^i_v, p^j_s, S) // Embree [Wald et al. 2014]
3  ∀x: compute W, W_S(x, W), ē_S(x, W), any f̄_S(x, W) // Eqs. 12, 13, 19
4  ∀x: if W_S(x, W) < t_{S,weak} : then markUnreliable(x)
5  foreach reliable x do
6      compute e_S(x, W) // Eq. 20
7      if e_S(x, W) > 2 · e_S(x, W)_old then
8          p_S(x) = p_S(x)_old, e_S(x, W) = e_S(x, W)_old // revert
9      else
10         p_S(x) = p_S(x) − ē_S(x, W) // advect
11     end
12 end
13 if converged(E(S, W)) then return final S // Eq. 11
14 ∀x : p_S(x) = umbrellaOp(x, λ_dist) // [Taubin 1995]
15 begin // handle outlier geometry
16     eraseUnreliableIsles() // if connected triangle count n_isle < t_{S,isle}
17     patchHoles() // fill each hole ring with flat triangles
18     smoothIrregularIsles() // find and smooth badly moved vertices
19 end
20 split() // varying details: large triangles moved into small nodes?
21 ∀ unreliable x: p_S(x) = umbrellaOp(x, λ_out) // the weak follow others
```

Kernel spread: We implemented the surface kernel $k_S^{i,j}$ using a Dijkstra search over triangle edges starting at $\mathbf{p}^{i,j}$ (Line 3). For each potential step over an edge, we compute the angular deviation γ between $\mathbf{n}^{i,j}$ and the mean normal of the two triangles adjacent to the edge. The penalty $\phi(\gamma)$ (Eq. 17) then scales the edge length.

Surface support function: We use $W_S(\mathbf{x}, \mathcal{W})$ (Eq. 13) as surface support. It is low at weakly sampled areas such as scene borders and occluded object regions. $W_S(\mathbf{x}, \mathcal{W})$ is also zero for most hallucinations at back sides that are not supported by samples but where nevertheless a crust was extracted. We mark these areas via thresholding with $t_{S,\text{weak}}$ (Line 4) which strongly facilitates removing hallucinated geometry, see Fig. 13(j). In contrast to Fuhrmann and Goesele [2014], our $W_S(\mathbf{x}, \mathcal{W})$ does not only increases with sample density but depends also on the projection confidences $\{\mathcal{N}_S^{i,j}\}$ and is thus more reliable.

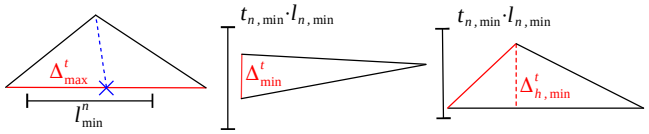


Fig. 8. To adapt the triangle size to the local level of detail and to avoid elongated triangles, an edge (red) is split (at the blue cross) if (a) $\Delta^t_{\max} > l_{n,\min}$ or collapsed if (b) $\Delta^t_{\min} < t_{\text{n, min}} \cdot l_{n,\min}$ or (c) $\Delta^t_{h,\min} < t_{\text{n, min}} \cdot l_{n,\min}$.

Triangle distribution: For proper kernel interpolation, we need regular triangles at the resolution of the input samples to ensure robust surface normal estimation and reasonable kernel spread. Both regularity and resolution must be maintained even if triangles move quite far and potentially degenerate during the mesh refinement. We ensure this by again exploiting the octree leaf nodes as the constructed SVO defines scene location-dependent reconstruction level of detail as mentioned in Sect. 4.

Fig. 8 shows our approach which works as follows. For each triangle Δ^t, we find all leaf nodes which intersect it and note the leaf node $n_{\min}$ with the smallest side length $l_{n,\min}$. If the longest triangle edge of Δ^t is larger than the smallest intersection node $\Delta^t_{\max} > l_{n,\min}$ (Fig. 8a), we split the longest edge to better match the local resolution (Line 20). On the other hand, if the smallest triangle edge length $l_{n,\min}$ of Δ^t is too small compared to the edge length of the smallest intersected node $\Delta^t_{\min} < t_{\text{n, min}} \cdot l_{n,\min}$ (Fig. 8b), we collapse the shortest edge. To avoid degenerate triangles, we find the smallest altitude length $\Delta^t_{h,\min}$ of triangle Δ^t. Again, if this altitude is too small ($\Delta^t_{h,\min} < t_{\text{n, min}} \cdot l_{n,\min}$) (Fig. 8c), we collapse the shortest edge (Line 1). Further, we apply the Umbrella operator with low λ_{dist} (Line 14), not to smooth sharp geometry but to distribute vertices tangentially and thus regularize triangles [Vu et al. 2012]. Without the Umbrella operator we would need unnecessary many edge collapses or splits to keep a regular scale-adaptive resolution of $\mathcal{S}$. Moreover, we cannot reliably refine vertices with low support using our interpolation scheme and thus smooth them with λ_{out} to make them follow well supported neighbor vertices (Line 21 for flat smooth back sides).

Outlier geometry: Weakly supported areas cannot be reliable refined and are thus excluded (Line 5). We delete small isolated unreliable vertex islands (Line 16) and close holes individually with flat geometry (Line 17). This keeps back sides (and closed meshes) but removes bad geometry including topological artifacts from crust extraction. It might change the initial topology, e.g., crust artifacts such as false connections over actual gaps might be removed.

The simple thresholding in (Line 4) does, however, not always detect such weakly supported areas, in particular if they are close to densely sampled regions. We detect such problematic cases using two criteria (Line 18): First, we calculate the average angle between the vertex normal and the neighboring trinagles' normals and average the resulting angle twice in the 1-ring neighborhood of the

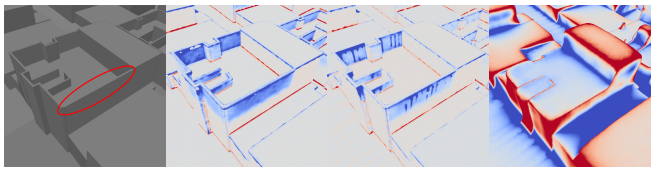
vertex in order to increase robustness. If the smoothed angle is larger than the threshold angle t_β we regard the corresponding vertex as a spiky outlier. Second, we also mark vertices as spiky outliers if their average incident edge length, i.e., their scale is more than doubled after updating the positions of the reliable vertices in Lines 5–12. Note that triangle sizes and thus edge lengths should mainly decrease during refinement: Triangles are subdivided if they are moved into smaller SVO nodes (Line 20) but are typically not moved into larger nodes or coarser sampled scene space. To robustly handle such spiky outlier vertices, which only occur as small islands, we fit them to their well-refined neighbors by smoothing them until convergence using the Umbrella operator.

6 RESULTS

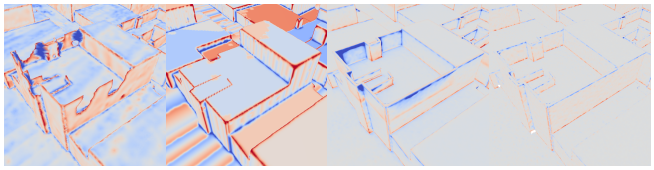
We compare TSR against Poisson Surface Reconstruction (PSR) [Kazhdan and Hoppe 2013], Floating Scale Surface Reconstruction (FSSR) [Fuhrmann and Goesele 2014], Ummenhofer and Brox's point-based reconstruction (UPoi) [2013], Point Fusion (PFS) [Ummenhofer and Brox 2015], Smooth Signed Distance Surface Reconstruction (SSD) [Calakli and Taubin 2011], Surface Reconstruction from Multi-resolution Points (SURFMRS) [Mücke et al. 2011] and CMPMVS [Jancosek and Pajdla 2011] using the original implementations of the authors except for CMPMVS* in Fig. 1 where we needed to use our own implementation. Note that we never cull back faces. Additionally, back faces of colorless reconstructions are rendered in yellow. TSR performance is in Tab. 1 with peak memory consumption measured as the maximum resident memory size of the process. Tab. 2 lists the parameter values for the presented scenes.

Synthetic tests and benchmark: Fig. 1 shows the synthetic wedge, a thin structure with optional Gaussian noise in the bottom rows ($\sigma = 0.1$). We generated only one reference view and no additional views per synthetic surface sample. Our qualitative evaluation shows that while related work already cannot accurately reconstruct the wedge given perfect samples, our approach is able to faithfully reconstruct the wedge even in the presence of noise. The visualized error shows that front and back of the wedge do not interfere with each other. Tab. 3 quantitatively details the bottom row results. To avoid punishing artifacts at the open boundary due to the finite size of the synthetic ground truth wedge, we evaluated only reconstructed surfaces within the gray box depicted in Fig. 1(a). TSR always performs best. Note that the Middlebury evaluation methodology does not consider surface orientations and may match the reconstructed front with the ground truth back (and vice versa). This falsely increases completeness values for all other methods for areas where they accurately reconstructed only one but not both sides of the thin structure.

Fig. 9 presents a large city model. We created a version textured with simplex noise, simulated 287 photographs, and fed these into MVE [Fuhrmann et al. 2014] for multi-view stereo-based point cloud reconstruction. Fig. 9 demonstrates that all other approaches have issues with correctly reconstructing the edges and planar surfaces. In particular, the thin roof railings highlighted in Fig. 9(a) are either missing (SSD, SURFMRS), incomplete (FSSR, CMPMVS) or show significant errors due to front-back interference (FSSR, PFS, PSR).



(a) Ground truth (b) FSSR (c) PFS (d) SSD



(e) CMPMVS (f) SURFMRS (g) PSR (h) TSR

	Accuracy (errors)			Completeness (%)		
	95%	90%	80%	9.50	5.00	2.50
SSD	14.04	10.33	6.68	70.9	59.0	44.3
SURFMRS	7.20	5.32	3.70	87.9	78.9	55.7
CMPMVS	7.96	4.11	2.60	95.3	90.6	78.9
PSR	3.19	1.67	0.76	**99.0**	97.5	93.6
PFS	3.82	1.20	0.55	98.9	97.2	93.9
FSSR	3.40	1.12	0.46	98.0	94.9	91.4
TSR	**1.22**	**0.61**	**0.33**	**99.0**	**98.4**	**96.4**

(i) Accuracy and completeness according to the Middlebury Mview evaluation methodology [Seitz et al. 2006]

Method	Face count	Runtime	Peak memory
PFS	24, 416, 218	1h 13m	17GB
FSSR	19, 134, 127	1h 18m	27GB
TSR	71, 532, 306	18h 30m	79GB

(j) Performance (result size, total runtime and peak memory)

Fig. 9. City dataset. (a - h) Closeups visualizing thin roof railings highlighted in (a). Note lower accuracy and missing or overestimated thin structures for previous work. (i) TSR clearly dominates the quantitative evaluation. (j) TSR, however, has highest runtime and memory demand.

(a) Image (b) Depths (c) Points, views (d) TSR

Fig. 10. Consumer depth sensor test. (a - b) Example input frame. (c) TSR input: 1000 Kinect views (frames) [Sturm et al. 2012]. (d) Reconstruction.

TSR keeps not only these structures but also consistently performs best in the quantitative evaluation, see Fig. 9(i).

As a sanity check, we submitted our results for the full temple to the Middlebury multi-view stereo evaluation [Seitz et al. 2006] and achieved state of the art results (accuracy: 0.36 mm for 90%, completeness: 99.2 % within 1.25 mm).

	Sample count	Face count	Peak memory	Total time	SVO	Occupancy	Refinement
City	361, 058, 245	71, 532, 306	79GB	18h 30m	1h 20m	2h 16m	14h 54m
Citywall	292, 415, 380	52, 379, 087	67GB	17h 22m	1h 2m	7h 21m	8h 59m
Cardboard	76, 540, 423	17, 372, 889	38GB	16h 33m	23m	2h 17m	13h 53m
Orchid	16, 237, 665	8, 429, 311	26GB	4h 7m	6m	1h 28m	2h 33m
Temple	15, 568, 210	1, 055, 670	20GB	44m	3m	15m	26m
ShellTower	7, 942, 563	1, 720, 804	20GB	24m	1m	8m	14m

Table 1. Point cloud sizes, face counts of results, peak memory consumption and runtimes for TSR. Total time from input loading to final mesh output. SVO, occupancy and refinement refer to the time from input loading to SVO output, time for occupancy scalar field computation and complete surface refinement for a server with two Intel E5-2650V2 processors.

	h_{SVO}	h_O	h_S
Wedge	0.25	1	1
(noisy)	5	3	1
City	1	1	1
Citywall	2.5	3	1
Cardboard	1	1.5	1
Orchid	1	3	2
Temple	1	1.5	0.75
ShellTower	1	1.5	1
Teddy	1	1	1

σ_α	$\pi/9$
σ_d	$2R^j_{\mathcal{S}}/3$
Γ	$5\pi/12$
Φ	0.1
$t_{C,k}$	100
$t_{C,\text{isle}}$	2500
$t_{\text{n, min}}$	0.1
$t_{\mathcal{S},\text{weak}}$	1.0
$t_{\mathcal{S},\text{isle}}$	100
t_β	$\pi/6$
λ_{dist}	0.1
λ_{out}	1.0

Table 2. TSR parameters. h_{SVO}, h_O, h_S control the smoothing strength according to input data accuracy.

	Accuracy (errors)			Completeness (%)		
	95%	90%	80%	2	1	0.5
PSR	1.43	9.78	2.79	88.9	68.9	46.2
PFS	3.48	2.73	2.00	81.8	54.4	29.6
SSD	4.59	2.60	1.76	64.4	34.1	16.4
FSSR	3.47	2.28	1.51	83.9	65.7	40.5
CMPMVS	3.11	1.97	1.39	61.0	52.3	35.9
TSR	**1.18**	**0.98**	**0.76**	**94.8**	**87.9**	**62.8**

Table 3. Synthetic wedge ($\sigma = 0.1$) reconstruction accuracy and completeness comparison (Middlebury Mview methodology [Seitz et al. 2006]).

General scenes: We show the generality of TSR by testing it on the TUM RGB-D fr3 teddy scene [Sturm et al. 2012], see Fig. 10. TSR reconstructs fine details such as the creases in the teddy's shirt (Fig. 10(d)) despite the quantized Kinect depth maps (Fig. 10(b)).

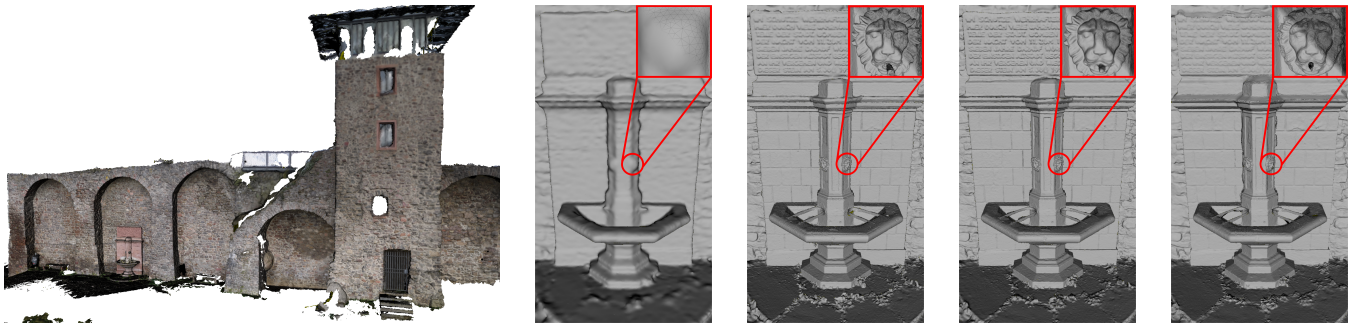
We reconstructed the CityWall scene of Fig. 11 provided with MVE for demonstration of multi-scale support. The uncontrolled data set consists of 564 images captured with strongly varying distances to the surfaces. CMPMVS is clearly inferior to the other approaches as it fails to correctly model the varying scale of the input samples. It oversmoothes the lion head and misses the thin metal bars, see Fig. 11(b). FSSR, PFS and TSR produce highly detailed geometry for the text on the wall and the lion heads. These areas are particularly challenging as they contain overlapping samples with strongly varying scale.

Thin structures: Fig. 12 shows a real 1 mm thick piece of cardboard with printed noise texture and additionally attached markers. We captured 178 photographs for point cloud reconstruction with MVE. The figures show the estimated geometry cropped to the approximate bounding box of the cardboard. PFS, FSSR and TSR reconstruct the well sampled front side well while only TSR is able to also reconstruct the sparser sampled back. The reconstructions of CMPMVS, SSD and FSSR miss much cardboard surface area. PFS, PSR and SSD hallucinate large amounts of geometry. In addition, our approach is the only one correctly capturing the cardboard thickness, see Fig. 13. We computed the histogram by rendering the distance (depth) from the camera to the front and back side of the TSR reconstruction and subtracted the resulting depth maps to get the thickness at every pixel. The histogram consists of the accumulated pixel difference values excluding areas of the attached markers. The mean of the difference values is 1,2 mm and hence close to the true thickness. Fig. 13(j) additionally presents the closed surface $\mathcal{S}$ before cropping weakly supported vertices and the corresponding support function $W_{\mathcal{S}}(\mathbf{x}, \mathcal{W})$ of Eq. 13. Fig. 13(k) highlights TSR weaknesses: topological artifacts left from crust extraction, refinement convergence to wrong local minima, and self-intersections due to noisy input samples, see also Sec. 6.1.

A further challenging scene for thin structures is the Streetsign from Ummenhofer and Brox [2013] with only one reference view per sample, see Fig. 14. Their point cloud processing techniques produce less noise in the front view. When observing the side view, we notice, however, that the sign is a single plane, basically flattening the entire geometry of the frame instead of actually reconstructing it. In contrast, TSR does not oversmooth the geometry, reconstructs the metal pillars and frame better but suffers from the many input samples clustered before or behind the street sign leading to an overestimation of the surface.

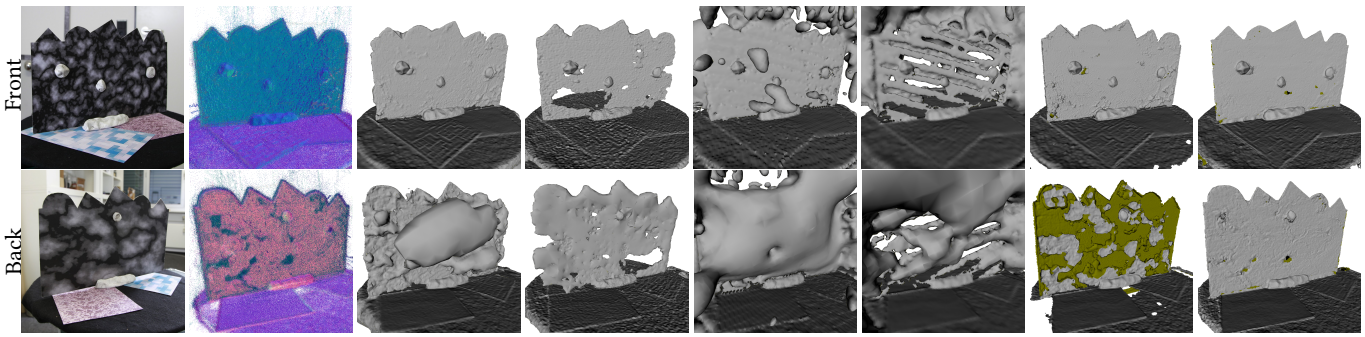
The next test scene is a point cloud of a miniature tower and a small sea shell, see Fig. 15. For this challenging scene, our approach is the only one that is able to produce a reconstruction containing only a few holes while avoiding hallucinated geometry. FSSR and PFS simply miss one of the sea shell sides while PSR and SSD produce hallucinated geometry. CMPMVS fares second best but again misses geometry for very thin structures.

Our final test scene is the point cloud of an orchid reconstructed with MVE. The reconstruction is challenging due to the thin structures as well as the amount of noise contained. Fig. 16 shows that PSR, FSSR, and SSD produce large quantities of spurious geometry while PFS and SSD miss thin or fine structures. Even though CMPMVS produces a substantially better result, it overestimate thickness of leaves and blossom petals. CMPMVS also misses geometry for



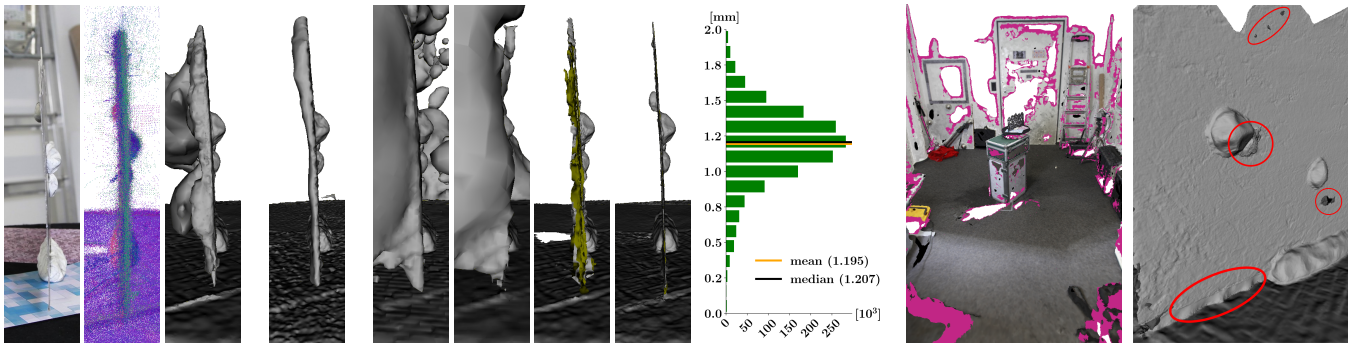
(a) Citywall (TSR) (b) CMPMVS (c) FSSR (d) PFS (e) TSR

Fig. 11. CityWall multi-scale reconstruction. TSR retained small details, avoided clutter in the fountain and accurately reconstructed the thin metal bars.



(a) Image (b) Points (c) PFS (d) CMPMVS (e) PSR (f) SSD (g) FSSR (h) TSR

Fig. 12. Cardboard front and back. Well (top) and sparsely sampled sides (bottom) showing representation issues resulting in back side artifacts except for TSR.



(a) Image (b) Input (c) PFS (d) CMPMVS (e) PSR (f) SSD (g) FSSR (h) TSR (i) TSR thickness (j) $W_S(\mathbf{x}, \mathcal{W})$ (k) Artifacts

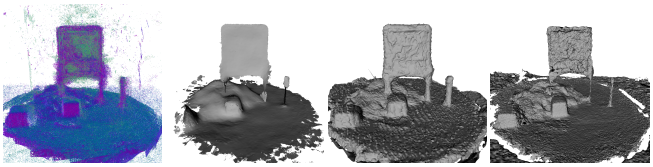
Fig. 13. Cardboard thickness, surface support function and artifacts. (a - h) Cardboard side views highlighting related work issues. (i) Thickness histogram for TSR. (j) Uncropped scene with the surface support $W_S(\mathbf{x}, \mathcal{W})$ (vertices with $W_S(\mathbf{x}, \mathcal{W}) < t_{S,\text{weak}}$ in pink). (k) Artifacts demonstrating weaknesses of TSR.

flower petals and leaves at the bottom of the plant although there were enough samples available, see Fig. 16 bottom. Only TSR reliably reconstructs both sides of leaves and blossoms, performing best in both completeness and accuracy. However, the ability of TSR to accurately reconstruct the thickness of thin parts also reveals imperfections of the input. In particular, noisy samples in combination with the independent reconstruction of front and back side can lead to self-intersections (highlighted in yellow).

6.1 Discussion

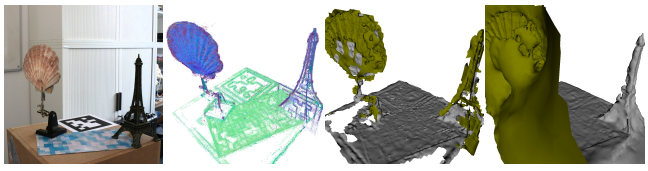
Strengths. The results clearly show the robustness of TSR and its ability to smoothly approximate the input data instead of merely interpolating it. Additionally, only our approach reliably reconstructs surface meshes containing small and thin structures via multi-scale data from uncontrolled capture setups without producing strong artifacts, see Figs. 1 and 16. Our results do not suffer from false pruning in well sampled areas and consist of largely connected geometry

(a) Image (b) Side (c) UPoi (d) TSR C (e) TSR $\mathcal{S}$

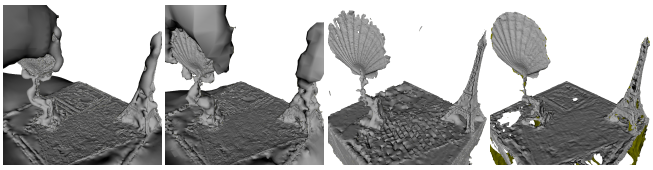


(f) Front (g) UPoi (h) TSR C (i) TSR $\mathcal{S}$

Fig. 14. Street sign comparison with UPoi. The sign reconstructed by both TSR (noisy) and UPoi (oversmooth) despite strongly inaccurate samples.



(a) Image (b) Points (c) FSSR (d) PFS



(e) PSR (f) SSD (g) CMPMVS (h) TSR

Fig. 15. Shelltower comparison. Like for the cardboard, only TSR accurately reconstructs the thin structures of the scene.

with floating parts only in areas with substantially missing data. On top of that, the crust C is always extracted as closed mesh and kept watertight during refinement. Still, we can identify and delete all hallucinated fill-in and back side geometry via the surface support function $W_S(\mathbf{x}, \mathcal{W})$, see Fig. 13(j). The iterative projective surface corrections implicitly define gradient descent step sizes depending on surface error magnitudes. Thus, the refinement converges for most surface parts after a few iterations, especially in well sampled areas, see Fig. 5. Finally, we only employ local optimization operations so that our algorithm scales well.

Weaknesses. The extracted crust C can have deficiencies. It might wrongly reach over gaps or falsely connect surfaces if respective visibility constraints are insufficient, see Fig. 13(k). This problem is scale dependent (like the crust) and finer sampling alleviates it. These artifacts can hamper the following surface evolution since the initial solution is locally too far from the surface.

The refinement has weaknesses inherent to gradient descent: slow convergence for fast changing gradients (e.g., due to contradicting samples), convergence to local optima (possible for concavities, clustered outliers, etc; see Figs. 1 and 13(k)) and no convergence guarantee owing to the non-convexity. In a few cases, triangle updates for error reduction create false creases, hampering convergence due to the angular costs function of our surface kernels. These disadvantages are the main source of missing details compared to related work and might be reduced by an added smoothness energy term.

Accurately reconstructing the boundaries of thin structures is difficult. There are usually no samples of the thin structure edges (e.g., leaf edges) bounding the extend of the structures. The input samples though require inter- and extrapolation which is thus not precisely defined at the boundaries. Further, the Umbrella operator leads to surface shrinkage at the same areas which is not reverted by surface update steps due to the lack of respective samples.

Finally, the current implementation has the highest runtime and memory usage of all competing approaches, see Fig. 9(j), leaving room for further algorithmic improvement and code optimization.

7 CONCLUSION

Despite decades of research on surface reconstruction, many modern techniques are still not able to reliably reconstruct a simple sheet of cardboard. We addressed in this paper the fundamental underlying deficiencies of models and representations preventing the reconstruction of thin surfaces. This allows TSR to faithfully reconstruct scenes with thin surface parts as well as more traditional single- and multi-scale scenes.

Besides improvements to the efficiency of the implementation we see different directions for future work. First, we would like to consistently address intersections of front and back side of thin surfaces reconstructed from noisy data without introducing a bias that increases the thickness of surfaces. Second, we plan to integrate photometric consistency into our approach by taking not only oriented point sets but also images of the scene as input for the reconstruction. Image edges might also help to better reconstruct thin and small structure borders. Third, to cope with scene-varying sample noise strength, automatic selection of the resolution and bandwidth parameters h_{SVO}, h_O and h_S is necessary. Finally, our approach is limited to static scenes. To remove this limitation and enable a larger scene variety, especially uncontrolled reconstruction of real plants, we will investigate modeling of object deformations.

ACKNOWLEDGMENTS

Part of this work was funded by the European Community's FP7 programme under grant agreement ICT-323567 (Harvest4D). We thank the FalconViz CAD team and Nils Moehrle for the City dataset.

REFERENCES

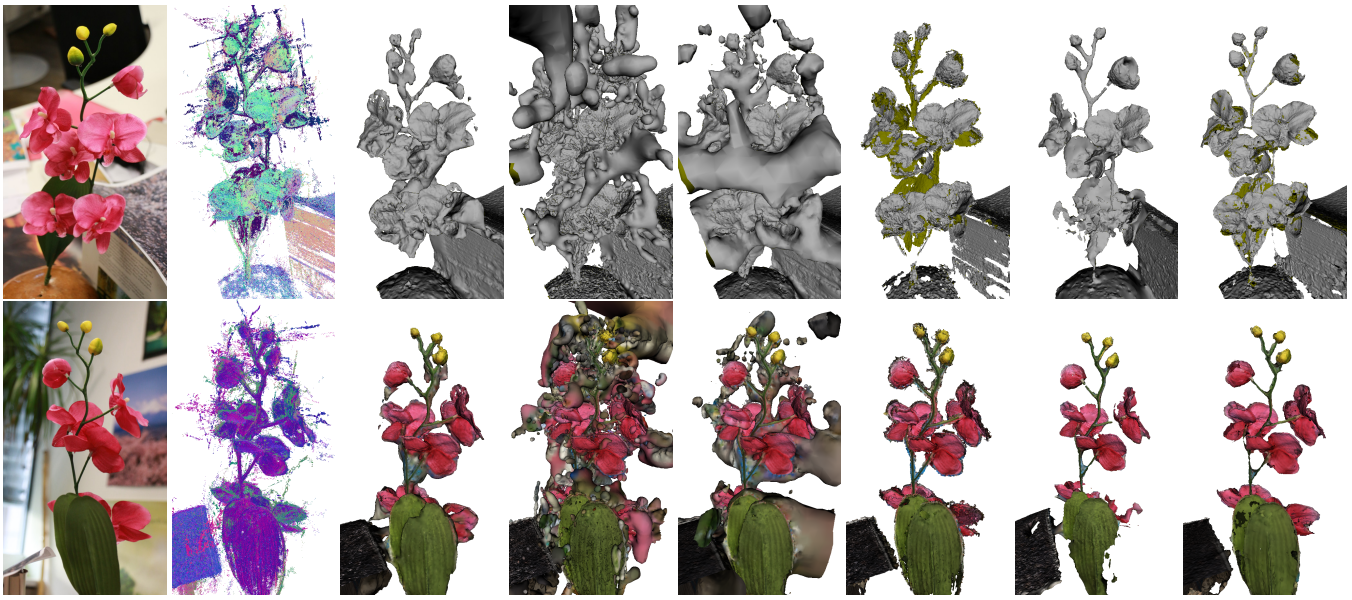
Bart Adams and Martin Wicke. 2009. Meshless Approximation Methods and Applications in Physics Based Modeling and Animation.. In *Eurographics (Tutorials)*.

Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. 1999. The Ball-Pivoting Algorithm for Surface Reconstruction. *IEEE TVCG* 5, 4 (1999).

Fatih Calakli and Gabriel Taubin. 2011. SSD: Smooth Signed Distance Surface Reconstruction. *CGF* 30, 7 (2011).

Fatih Calakli and Gabriel Taubin. 2012. SSD-C: Smooth Signed Distance Colored Surface Reconstruction. In *Expanding the Frontiers of Visual Analytics and Visualization*.

Yang Chen and Gérard Medioni. 1995. Description of complex objects from multiple range images using an inflating balloon model. *CVIU* 61, 3 (1995).



(a) Image (b) Points (c) PFS (d) PSR (e) SSD (f) FSSR (g) CMPMVS (h) TSR

Fig. 16. Comparison using the challenging orchid dataset. Most accurate and complete reconstruction by TSR. Other approaches with false pruning or thickening and hallucinated geometry.

Brian Curless and Marc Levoy. 1996. A Volumetric Method for Building Complex Models from Range Images. In *SIGGRAPH*.

Brian Lee Curless. 1997. *New methods for Surface Reconstruction from Range Images*. Dissertation. Stanford University.

Julie Digne, Jean-Michel Morel, Charyar-Mehdi Souzani, and Claire Lartigue. 2011. Scale space meshing of raw data point sets. In *CGF*, Vol. 30.

Simon Fuhrmann and Michael Goesele. 2011. Fusion of Depth Maps with Multiple Scales. *ACM TOG* 30, 6 (2011).

Simon Fuhrmann and Michael Goesele. 2014. Floating Scale Surface Reconstruction. *ACM TOG* 33, 4 (2014).

Simon Fuhrmann, Fabian Langguth, and Michael Goesele. 2014. MVE - A Multi-View Reconstruction Environment. In *GCH*, Vol. 6.

Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. 1992. Surface Reconstruction from Unorganized Points. In *SIGGRAPH*.

Alexander Hornung and Leif Kobbelt. 2006. Robust Reconstruction of Watertight 3D Models from Non-uniformly Sampled Point Clouds Without Normal Information. In *SGP*.

Michal Jancosek and Tomás Pajdla. 2011. Multi-View Reconstruction Preserving Weakly-Supported Surfaces. In *CVPR*.

Michael Kass, Andrew Witkin, and Demetri Terzopoulos. 1988. Snakes: Active contour models. *IJCV* 1, 4 (1988).

Michael Kazhdan and Hugues Hoppe. 2013. Screened Poisson Surface Reconstruction. *ACM TOG* 32, 3 (2013).

Patrick Labatut, Jean-Philippe Pons, and Renaud Keriven. 2009. Robust and Efficient Surface Reconstruction from Range Data. In *CGF*, Vol. 28.

Paul Merrell, Amir Akbarzadeh, Liang Wang, Philippos Mordohai, Jan-Michael Frahm, Ruigang Yang, David Nistér, and Marc Pollefeys. 2007. Real-Time Visibility-Based Fusion of Depth Maps. In *ICCV*.

Patrick Mücke, Ronny Klowsky, and Michael Goesele. 2011. Surface Reconstruction from Multi-resolution Sample Points. In *VMV*.

Matthias Müller, David Charypar, and Markus Gross. 2003. Particle-based Fluid Simulation for Interactive Applications. In *SCA*.

Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. 2011. KinectFusion: Real-time Dense Surface Mapping and Tracking. In *International Symposium on Mixed and Augmented Reality (ISMAR)*.

Yutaka Ohtake and Alexander G Belyaev. 2003. Dual-Primal Mesh Optimization for Polygonized Implicit Surfaces With Sharp Features. *Journal of Computing and Information Science in Engineering* 2, 4.

Nikolay Savinov, Christian Häne, L'ubor Ladický, and Marc Pollefeys. 2016. Semantic 3D Reconstruction with Continuous Regularization and Ray Potentials Using a Visibility Consistency Constraint. In *CVPR*.

Scott Schaefer and Joe Warren. 2004. Dual Marching Cubes: Primal Contouring of Dual Grids. In *PG*.

Steven M. Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. 2006. A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In *CVPR*, Vol. 1.

Qi Shan, Brian Curless, Yasutaka Furukawa, Carlos Hernandez, and Steven Seitz. 2014. Occluding Contours for Multi-View Stereo. In *CVPR*.

Simon J Sheather and Michael Chris Jones. 1991. A reliable data-based bandwidth selection method for kernel density estimation. *Journal of the Royal Statistical Society. Series B (Methodological)*.

J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. 2012. A Benchmark for the Evaluation of RGB-D SLAM Systems. In *IROS*.

Gabriel Taubin. 1995. A Signal Processing Approach to Fair Surface Design. In *SIGGRAPH*.

George R. Terrell and David W. Scott. 1992. Variable Kernel Density Estimation. *The Annals of Statistics* 20, 3 (1992).

Greg Turk and Marc Levoy. 1994. Zippered Polygon Meshes from Range Images. In *SIGGRAPH*.

Benjamin Ummenhofer and Thomas Brox. 2013. Point-Based 3D reconstruction of thin objects. In *ICCV*.

Benjamin Ummenhofer and Thomas Brox. 2015. Global, Dense Multiscale Reconstruction for a Billion Points. In *ICCV*.

Hoang-Hiep Vu, Patrick Labatut, Jean-Philippe Pons, and Renaud Keriven. 2012. High Accuracy and Visibility-Consistent Dense Multiview Stereo. *TPAMI* 34, 5 (2012).

Ingo Wald, Sven Woop, Carsten Benthin, Gregory S. Johnson, and Manfred Ernst. 2014. Embree: A Kernel Framework for Efficient CPU Ray Tracing. *ACM TOG* 33, 4, Article 143.

Kaan Yücer, Changil Kim, Alexander Sorkine-Hornung, and Olga Sorkine-Hornung. 2016a. Depth from Gradients in Dense Light Fields for Object Reconstruction. In *3DV*.

Kaan Yücer, Alexander Sorkine-Hornung, Oliver Wang, and Olga Sorkine-Hornung. 2016b. Efficient 3D Object Segmentation from Densely Sampled Light Fields with Applications to 3D Reconstruction. *ACM TOG* 35, 3 (2016).

Received May 2017; revised August 2017; final version September 2017; accepted September 2017