Hierarchical surface reconstruction from multi-resolution point samples

Ronny Klowsky, Patrick Mücke, and Michael Goesele

TU Darmstadt

Abstract. Robust surface reconstruction from sample points is a challenging problem, especially for real-world input data. We present a new hierarchical surface reconstruction based on volumetric graph-cuts that incorporates significant improvements over existing methods. One key aspect of our method is, that we exploit the footprint information which is inherent to each sample point and describes the underlying surface region represented by that sample. We interpret each sample as a vote for a region in space where the size of the region depends on the footprint size. In our method, sample points with large footprints do not destroy the fine detail captured by sample points with small footprints. The footprints also steer the inhomogeneous volumetric resolution used locally in order to capture fine detail even in large-scale scenes. Similar to other methods our algorithm initially creates a crust around the unknown surface. We propose a crust computation capable of handling data from objects that were only partially sampled, a common case for data generated by multi-view stereo algorithms. Finally, we show the effectiveness of our method on challenging outdoor data sets with samples spanning orders of magnitude in scale.

1 Introduction

Reconstructing a surface mesh from sample points is a problem that occurs in many applications, including surface reconstruction from images as well as scene capture with triangulation or time-of-flight scanners. Our work is motivated by the growing capabilities of multi-view stereo (MVS) techniques [20, 8, 9, 7] that achieve remarkable results on various data sets.

Traditionally, surface reconstruction techniques are designed for fairly high-quality input data. Measured sample points, in particular samples generated by MVS algorithms, are, however, *noisy* and contain *outliers*. Figure 1 shows an example reconstructed depth map that we use as input data in our method. Furthermore, sample points are often non-uniformly distributed over the surface and entire regions might not be represented at all. Recently, Hornung and Kobbelt presented a robust method well suited for noisy data [12]. This method generates optimal low-genus watertight surfaces within a crust around the object using a volumetric graph cut. Still, their algorithm has some major limitations regarding crust generation, sample footprint, and missing multi-resolution reconstruction which we address in this paper.

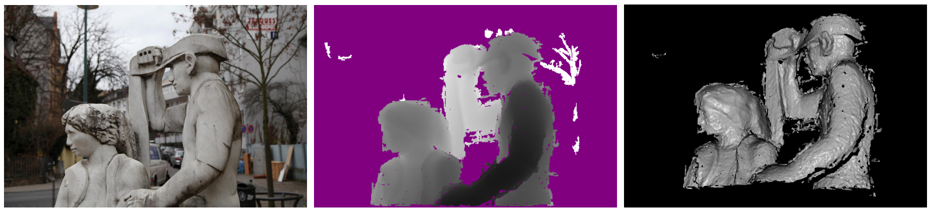


Fig. 1. *Left:* An input image to Multi-View Stereo reconstruction. *Middle:* The reconstructed depth map visualized in gray values (white: far, black: near). *Right:* The triangulated depth map rendered from a slightly different view point.

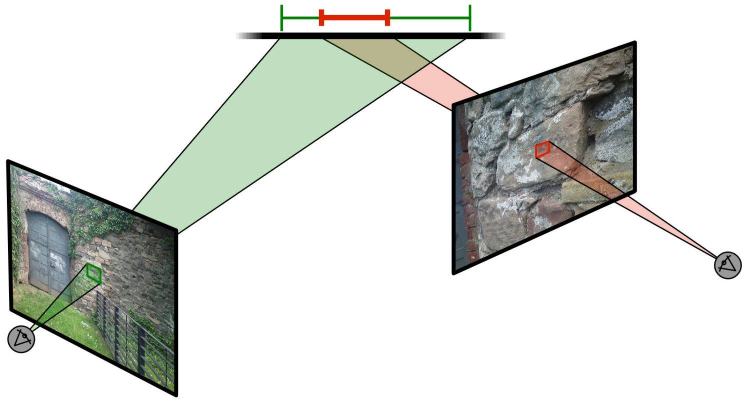


Fig. 2. Visualization of the *footprint* of a sample point: A certain pixel in the left image covers a significantly larger area than a corresponding pixel in the right image.

Hornung and Kobbelt create a surface confidence function based on unsigned distance values extracted from the sample points. The final surface S is obtained by optimizing for maximum confidence and minimal surface area. As in many surface reconstruction algorithms, the footprint of a sample point is completely ignored when computing the confidence. Every sample point, regardless of how it was obtained, inherently has a *footprint*, the underlying surface area taken into account during the measurement (see Figure 2). The size of the footprint indicates the sample point's capability to capture surface details. A method that outputs sample points with different footprints was proposed by Habbecke and Kobbelt [9]. They represent the surface with surfels (surface elements) of varying size depending on the image texture. Furukawa et al. [7] consider footprints to estimate reconstruction accuracy and Fuhrmann and Goesele [6] build a hierarchical signed distance field where they insert samples on different scales depending on their footprint. However, both methods effectively discard samples with large footprints prior to final surface extraction. In this paper, we propose a different way to model the sample footprint during the reconstruction process.

In particular, we create a modified confidence map where samples contribute differently depending on their footprints.

The confidence map is only evaluated inside a *crust*, a volumetric region around the sample points. In [12], the crust computation implicitly segments the boundary of the crust into *interior* and *exterior*. The final surface separates interior from exterior. This crust computation basically works only for completely sampled objects. Even with their proposed workaround (estimating the medial axis), the resulting crust is still not applicable to many data sets. Such a case is illustrated in Figure 3, where no proper interior component can be computed. This severely restricts the applicability of the entire algorithm. We propose a different crust computation that separates the crust generation from the crust segmentation process, extending the applicability to a very general class of input data.

Finally, as Vu et al. [24] pointed out, volumetric methods such as [12] relying on regular volume decomposition are not able to handle large-scale scenes. To overcome this problem our algorithm reconstructs on a locally adaptive volumetric resolution and finally extracts a watertight surface. This allows us to reconstruct fine details even in large-scale scenes such as the Citywall data set (see Figure 11).

This paper builds strongly on a recent publication by Mücke et al. [18] but contains the following substantial improvements.

- The sampling of the global confidence map is parallelized.
- We now employ a graph embedding modeling the 26-neighborhood which better approximates the Euclidean distance.
- Surface extraction is deferred to the end of the algorithm by using a combination of marching cubes and marching tetrahedra on a multi-resolution grid. This supersedes the need of the error-prone mesh clipping used before.

In addition, we show the effectiveness of our algorithm on a new challenging data set with high surface genus.

The remainder of the paper is organized as follows: First, we review previous work (Section 2) and give an overview of our reconstruction pipeline (Section 3). Details of the individual steps are explained in Sections 4–7. Finally, we present results of our method on standard benchmark data as well as challenging outdoor scenes (Section 8) and wrap up with a conclusion and an outlook on future work (Section 9).

2 Related work

Surface reconstruction from (unorganized) points

Surface reconstruction from unorganized points is a large and active research area. One of the earliest methods was proposed by Hoppe et al. [10]. Given a set of sample points, they estimate local tangent planes and create a signed distance field. The zero-level set of this signed distance field, which is guaranteed to be a manifold, is extracted using a variant of the marching cubes algorithm [15].

If the sample points originate from multiple range scans, additional information is available. VRIP [5] uses the connectivity between neighboring samples as well as the direction to the sensor when creating the signed distance field. Additionally, it employs a cumulative weighted signed distance function allowing it to incrementally add more data. The final surface is again the zero-level set of the signed distance field. A general problem of signed distance fields is that local inconsistencies of the data lead to surfaces with undesirably high genus and topological artifacts. Zach et al. [25] mitigate this effect. They first create a signed distance field for each range image and then compute a regularized field u approximating all input fields while minimizing the total variation of u. The final surface is the zero-level set of u. Their results are of good quality, but the resolution of both, the volume and the input images, is very limited. In their very recent paper, Fuhrmann and Goesele [6] introduce a depth map fusion algorithm that takes sample footprints into account. They merge triangulated depth maps into a hierarchical signed distance field similar to VRIP. After a regularization step, basically pruning low-resolution data where reliable higher-resolution data is available, the final surface is extracted using marching tetrahedra. Our method does not rely on triangulated depth maps and tries to merge all data samples while never discarding information from low-resolution samples. Another recent work taking unorganized points as input is called cone carving and is presented by Shalom et al. [21]. They associate each point with a cone around the estimated normal to carve free space and obtain a better approximation of the signed distance field. This method is in a way characteristic for many surface reconstruction algorithms in the sense that it is designed to work on raw scans from a commercial 3D laser scanner with rather good quality. Such methods are often not able to deal with the lower quality data generated by MVS methods from outdoor scenes containing a significant amount of noise and outliers.

Kazhdan et al. [13] reformulate the surface reconstruction problem as a standard Poisson problem. They reconstruct an indicator function marking regions inside and outside the object. Oriented points are interpreted as samples of the gradient of the indicator function, requiring accurate normals at each sample point's position which are usually not present in MVS data. The divergence of the smoothed vector field, represented by these oriented points, equals the Laplacian of the indicator function. The final surface is extracted as an iso-surface of the indicator function using a variant of the marching cubes algorithm. Along these lines, Alliez et al. [1] use the normals to derive a tensor field and compute an implicit function whose gradients best approximate that tensor field. Additionally, they present a technique, called Voronoi-PCA, to estimate unoriented normals using the Voronoi diagram of the point set.

Graph cut based surface reconstruction

Boykov and Kolmogorov [2] introduced the idea of reconstructing surfaces by computing a cut on a graph embedded in continuous space. They also show how to build a graph and set the edge weights such that the resulting surface is minimal for any anisotropic Riemannian metric. Hornung and Kobbelt [11] use the volumetric graph cut to reconstruct a surface given a photo-consistency measure defined at each point of a predefined volume space. They propose to embed an octahedral graph structure into the volume and show how to extract a mesh from the set of cut edges. In a follow-up paper [12], they present a way to compute confidence values from a non-uniformly sampled point cloud and improve the mesh extraction procedure.

An example of using graph cuts in multi-view stereo is the work of Sinha et al. [22]. They build an adaptive multi-resolution tetrahedral mesh where an estimated photo-consistency guides the subdivision. The final graph cut is performed on the dual of the tetrahedral mesh followed by a photo-consistency driven mesh refinement. Labatut et al. [14] build a tetrahedral mesh around points merged from multiple range images. They introduce a surface quality term and a surface visibility term that takes the direction to the sensor into account. From an optimal cut, which minimizes the sum of the two terms, a labeling of each tetrahedra as inside or outside can be inferred. The final mesh consists of the set of triangles separating the tetrahedra according to their labels. Vu et al. [24] replace the point cloud obtained from multiple range images with a set of 3D features extracted from the images. The mesh obtained from the tetrahedral graph cut is refined mixing photo-consistency in the images and a regularization force. However, none of the existing graph cut based surface reconstruction algorithms properly incorporates the footprint of a sample.

3 Overview

The input of our algorithm is a set of *surface samples* representing the scene (Figure 3a). Each surface sample consists of its position, footprint size, a scene surface normal approximation, and an optional confidence value. A cubic bounding box is computed from the input points or given by the user.

First, we determine the *crust*, a subset of the bounding volume containing the unknown surface. All subsequent computations will be performed inside this crust only. Furthermore, the boundary of the crust is partitioned into *interior* and *exterior*, defining interior and exterior of the scene (Figure 3b). Inside the crust we compute a *global confidence map*, such that points with high confidence values are likely to lie on the unknown surface. Each sample point adds confidence to a certain region of the volume. The size of the region and the confidence peak depend on the sample point's footprint size. Effectively, every sample point adds the same total amount of confidence to the volume but spread out differently. A volumetric graph is embedded inside the crust where graph nodes correspond to voxels and graph edges map the 26-neighborhood. A minimal cut on this graph separates the voxels into interior and exterior representing the optimal surface at this voxel resolution (Figure 3c). The edge weights of the graph are chosen such that the final surface minimizes surface area while maximizing confidence.

We then identify surface regions with sampled details too fine to be adequately represented on the current resolution. Only these regions are subdivided, the global confidence map is resampled, and the graph cut is computed

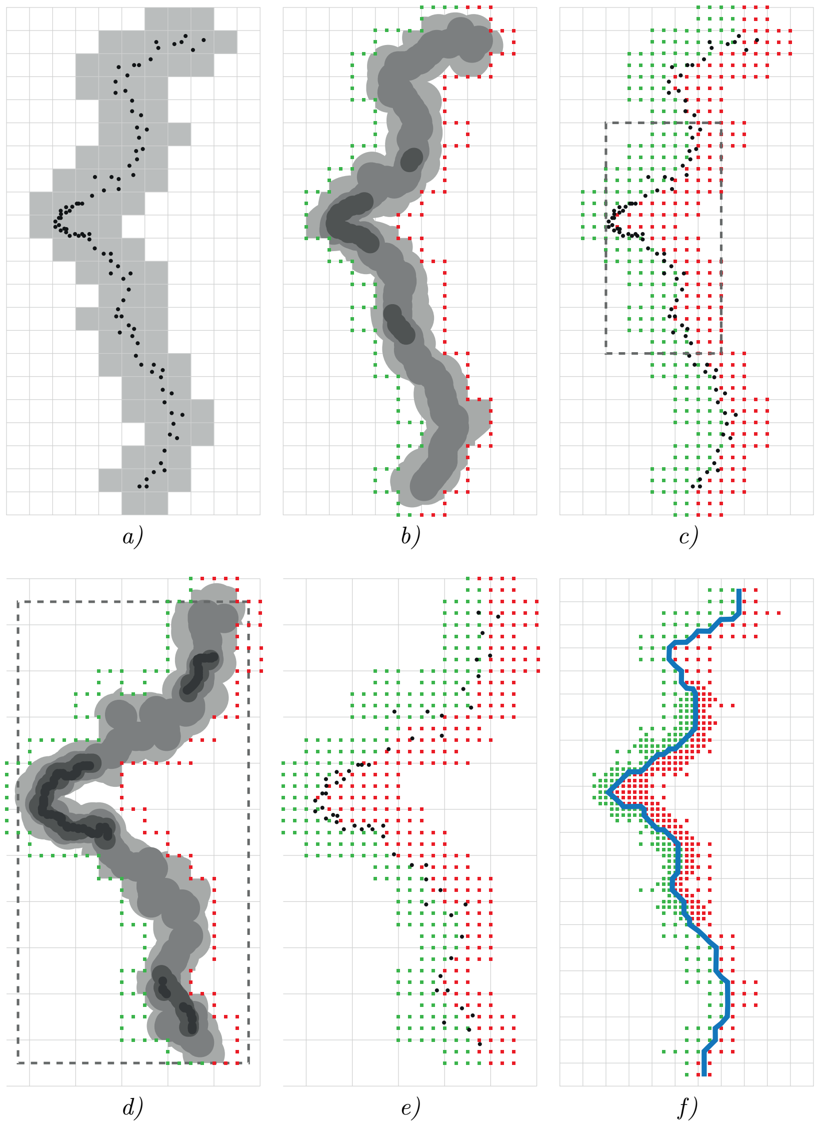


Fig. 3. Overview of our reconstruction pipeline. *a)* We compute a crust around the input samples of different footprints and varying sampling density. *b)* We segment the crust into *interior* (red) and *exterior* (green) and compute the global confidence map (GCM) to which each input sample contributes. *c)* A minimal cut on the embedded graph segments the voxel corners representing the surface with maximum confidence while minimizing surface area. We mark the areas with high-resolution samples (dashed black box) and iteratively increase resolution therein. *d+e)* In the increased resolution area we re-evaluate the GCM and perform the graph cut optimization. *f)* Finally, an adaptive triangle mesh is extracted from the multi-resolution voxel corner labeling.

on a higher resolution (Figure 3d+e). We repeat this process iteratively until eventually all fine details were captured. Finally, we extract the surface in the irregular voxel grid using a combination of marching cubes and marching tetrahedra. This results in a multi-resolution surface representation of the scene, the output of our algorithm (Figure 3f).

4 Crust computation

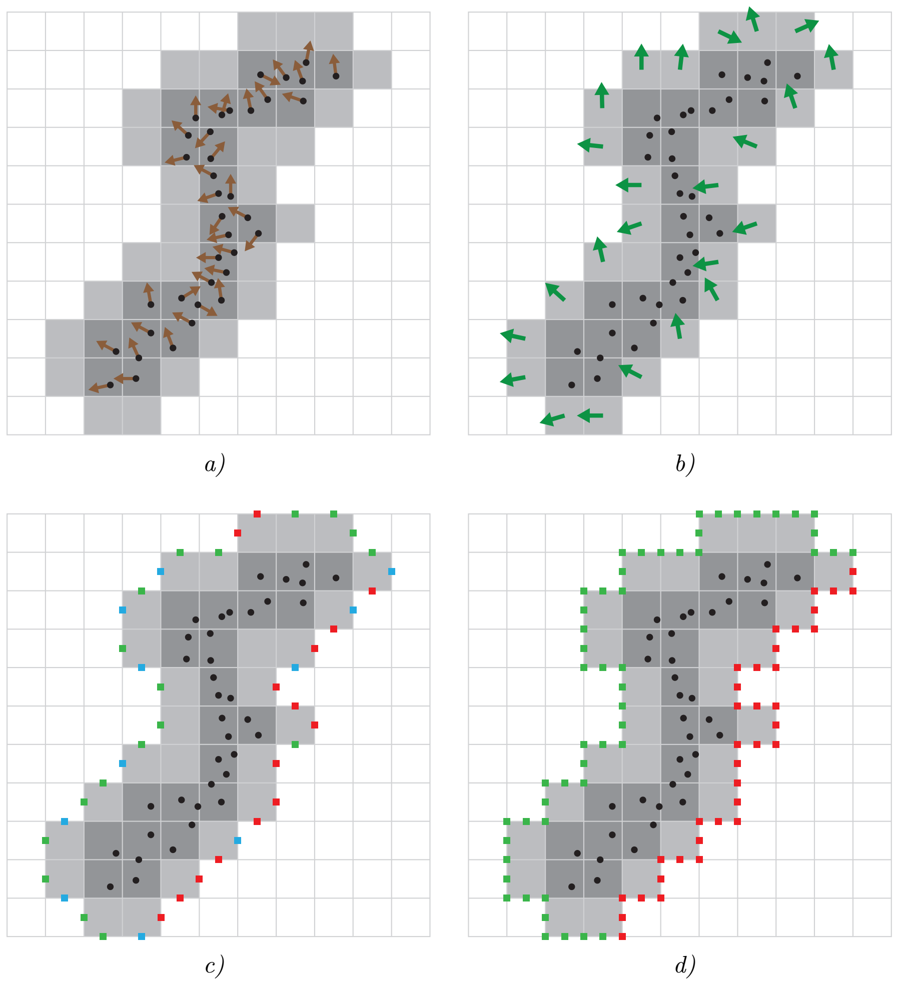
We subdivide the cubic bounding box into a regular voxel grid. For memory efficiency and to easily increase the voxel resolution, this voxel grid is represented by an octree data structure. Our algorithm iteratively treats increasing octree levels (finer resolution) starting with a user-defined low octree level ℓ_0, i.e., with a coarse resolution.

The crust $V_{crust} \subset V$ is a subset of voxels that contains the unknown surface. The crust computation is an important step in the algorithm for several reasons: The shape of the crust constrains the shape of the reconstructed surface. Furthermore, the crust has to be sufficiently large to contain the optimal surface and on the other hand as narrow as possible to reduce computation time and memory cost. We split the crust computation into two parts. First, the crust is generated, then the boundary of this crust is segmented to define interior and exterior of the scene (see Figure 4 for an overview).

Crust generation We initialize the crust on level ℓ_0 with the set of voxels on the parent octree level $\ell_0 - 1$ containing surface samples. We dilate this sparse set of voxels several times over the 6-neighborhood of voxels, followed by a morphological closing operation (Figure 4a). The number of dilation steps is currently set by the user, but the resulting crust shape can be immediately inspected, as the crust generation is fast on the low initial resolution. Subsequently, these voxels $v \in V_{crust}^{\ell_0-1}$ are once regularly subdivided to obtain the initial crust $V_{crust}^{\ell_0}$ for further computations on level ℓ_0.

Crust segmentation In this step our goal is to assign labels *interior* and *exterior* to all boundary voxel corners on level ℓ_0 to define the interior and exterior of the scene. In the following, we define $\partial V_{crust}^{\ell}$ to be the set of boundary voxels on level ℓ. We start by determining labels for voxel corners v_f that lie on the midpoints of boundary faces of parent crust voxels $v \in \partial V_{crust}^{\ell_0-1}$. The labels are determined by comparing a surface normal estimate $\boldsymbol{n}_v^{surf}$ for parent voxel v with the normals of the boundary faces $\boldsymbol{n}_{v_f}^{crust}$. The surface normal is computed for each crust voxel by averaging the normals of all sample points inside the crust voxel. Crust voxels that do not contain surface samples obtain their normal estimate through propagation during crust dilatation (Figure 4b). We determine the initial labels on the crust boundary by

$$label(v_f) = \begin{cases} exterior, & \text{if} \quad \boldsymbol{n}_{v_f}^{crust} \cdot \boldsymbol{n}_v^{surf} \geq \tau \\ interior, & \text{if} \quad \boldsymbol{n}_{v_f}^{crust} \cdot \boldsymbol{n}_v^{surf} \leq -\tau \\ \text{unknown}, & \text{otherwise} \end{cases} \tag{1}$$



a) *b)* *c)* *d)*

Fig. 4. Initial crust computation for lowest resolution: *a)* We initialize the crust with voxels containing sample points and dilate several times. *b)* Surface normals are computed for each voxel. *c)* The comparison of surface normals with the face normals of the crust voxels defines an initial labeling into *interior* (red), *exterior* (green), and *unknown* (blue). *d)* An optimization yields a homogenous crust surface segmentation.

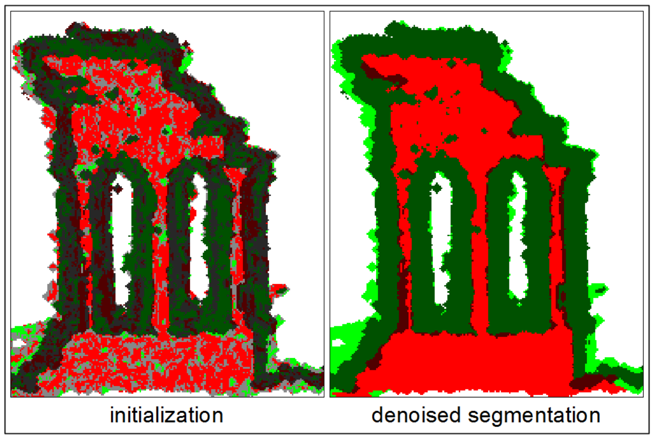


Fig. 5. Visualization of the crust surface for the Temple (cut off perpendicular to the viewing direction). The color is similar to Figure 4. Light shaded surfaces are seen from the front, dark shaded ones are seen from the back.

with $\tau \in (0,1)$ (Figure 4c). We used $\tau = 0.75$ in all experiments.

By now we have just labeled a subset of all voxel corners on level ℓ_0 (Figure 4c). Furthermore, since surface normal information of the samples may only be a crude approximation, this initial labeling is noisy and has to be regularized. We cast the problem of obtaining a homogenous labeling of the crust surface into a 2D binary image denoising problem solved using graph cut optimization as described by Boykov and Veksler [4]. We build a graph with a node per voxel corner in $\partial V_{crust}^{\ell_0}$ and a graph edge connecting two nodes if the corresponding voxel corners share a voxel edge. Additionally, 'diagonal' edges are inserted that connect the initially labeled corners in the middle of parent voxel faces with the four parent voxel corners. We also add two terminal nodes *source* and *sink* together with further graph edges connecting each node to these terminals. Note that this graph is used for the segmentation of the crust on the lowest resolution level ℓ_0 only and should not be confused with the graphs used for surface reconstruction on the different resolutions.

All edges connecting two non-terminal nodes receive the same edge weight w. Edges connecting a node n with a terminal node receive a weight depending on the labeling of the corresponding voxel corner v_c, where unlabeled voxel corners are treated as unknown:

$$w_n^{source} = \begin{cases} \mu & \text{if } v_c \text{ is labeled } \textit{interior} \\ 1-\mu & \text{if } v_c \text{ is labeled } \textit{exterior} \\ \frac{1}{2} & \text{if } v_c \text{ is unknown} \end{cases} \tag{2}$$

$$w_n^{sink} = 1 - w_n^{source} \tag{3}$$

for a constant $\mu \in (0, \frac{1}{2})$. With these edge weights the *exterior* is associated with *source*, *interior* with *sink*. A cut on this graph assigns each node either to the *source* or to the *sink* component and therefore yields a homogeneous segmentation of the boundary voxel corners of $\partial V_{crust}^{\ell_0}$ (Figure 4d and Figure 5 right). We used $w = 0.5$ and $\mu = 0.25$ in all experiments.

If two neighboring crust voxel corners obtained different labels, the reconstructed surface is forced to pass between them, as it has to separate *interior* from *exterior*. The denoising minimizes the number of such occurrences and therefore prevents unwanted surfaces from being formed. In the case of entirely sampled surfaces and a correctly computed crust, two neighboring voxel corners never have different labels. However, if the scene surface is not sampled entirely, such segment borders occur even for correct segmentations (see Figure 4d). This forces the surface to pass through the two involved voxel corners which, unlike the rest of the surface reconstruction, does not depend on the confidence values. This fixation does not affect the surface in sampled regions, though. We exploit this constraint on the reconstructed surface in our refinement step where we reconstruct particular areas on higher resolution (see Section 7).

5 Global confidence map

The *global confidence map* (GCM) is a mapping $\Gamma : \mathbb{R}^3 \to \mathbb{R}$ that assigns a confidence value to each point in the volume. Our intuition is that each sample point spreads its confidence over a region in space whose extent depends on the sample footprint. Thus, sample points with a small footprint create a focused spot whereas sample points with a large footprint create a blurry blob (see Figure 3b). We model the spatial uncertainty of a sample point as a Gaussian γ_s centered at the sample point's position with standard deviation equal to half the footprint size. If the sample points are associated with confidence values we scale the Gaussian accordingly. The *local confidence map* (LCM) γ_s determines the amount of confidence added by a particular sample point s. Consequently, the GCM is the sum over all LCMs:

$$\Gamma(x) = \sum_s \gamma_s(x). \tag{4}$$

Implementation Let ℓ be the octree level at which we want to compute the graph cut. In all crust voxels $\{x_v\}_{v \in V_{crust}^{\ell}}$ we evluate the GCM Γ at 27 positions: at the 8 corners of the voxel, at the middle of each face and edge, and at the center of the voxel. When adding up the LCMs of each sample point s we clamp the value of γ_s to zero for points for which the distance to s is larger than three times the footprint size of sample point s. Also, we sample each γ_s only at a fixed number of positions ($\approx 5^3$) within its spatial support and exploit the octree data structure by accumulating each γ_s to nodes at the appropriate octree level depending on the footprint size. After all samples have been processed, the accumulated values in the octree are propagated to the nodes at level ℓ by adding the values at a node to the children's nodes using linear interpolation for in-between positions. The support of LCMs of sample points with small footprints might be too narrow to be adequately sampled on octree level ℓ. For those

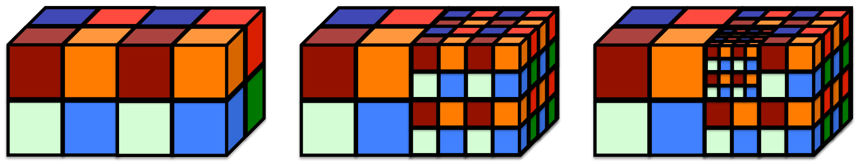


Fig. 6. Visualization of an intermediate state of the binning approach used for the parallelization of the GCM computation. Starting with two bins (left), the right bin is subdivided into eight new bins (middle). One of the new bins is subdivided again (right) resulting in a total number of 16 bins.

samples we temporarily increase the footprint for the computation of the LCM γ_s and mark the corresponding voxel for later processing at higher resolution.

5.1 Parallelization

In order to speed-up the sample insertion into the octree which is costly since each input point creates ≈ 125 samples, we parallelize the insertion at each octree level $\hat{\ell} \leq \ell$ using a binning approach. In our implementation, bins correspond to voxels. In each bin we sort the samples into eight lists representing the eight child voxels in a predefined order. We process the first list of all bins in parallel, then the second list, and so on. For this purpose samples in list x of two different bins should not interfere with each other, i.e., affect the same nodes in the octree. We start with the bounding cube as root bin containing all samples to be processed on level $\hat{\ell}$. We subdivide a bin if the following two criteria are satisfied:

1. the bin contains more than n_{max} samples, and
2. subdividing the bin maintains the property that samples out of the same list but different bins do not interfere with each other given their footprint.

When subdividing a bin the lists are effectively turned into bins and the samples are partitioned into eight smaller lists according to the same predefined order as before. The subdivision stops if a maximum number of bins has been reached or no more bins can be subdivided. Figure 6 shows the main principle of the subdivision process where the color coded voxels represent the individual lists. Note that two voxels with the same color never touch so that the LCM of samples do not interfere with each other.

6 Graph cut

As done by Hornung and Kobbelt [12] we apply a graph cut to find the optimal surface. The layout of the graph cut is however more similar to Boykov and Kolmogorov [2] since we define a graph node per voxel and edges representing the 26-neighborhood (inside the set of crust voxels V_{crust}). Note that at this

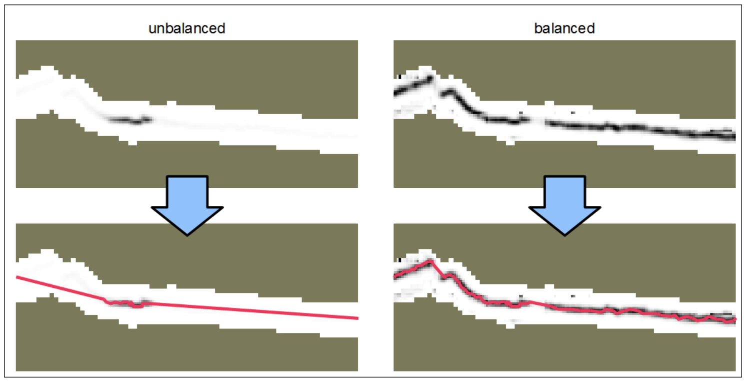


Fig. 7. The GCM values can be arbitrarily large leading to near-constant edge weights in large regions of the volume (left). Our *local GCM balancing* compensates for that allowing the final graph cut to find the correct surface (right).

stage we compute the graph cut on a certain resolution only and do not extract the surface explicitly. The edge weights w_i in the graph are derived from the GCM values $\Gamma(x_i)$ in the center of the voxel, edge, or face, respectively. Since the optimal surface should maximize the global confidence Γ we want to set small edge weights for regions with high confidence and vice versa. A straightforward way to implement this would be

$$w_i = 1 - \frac{\Gamma(x_i)}{\Gamma_{max}} + a \quad \text{with} \quad \Gamma_{max} = \max_{x \in \mathbb{R}^3} \Gamma(x) \tag{5}$$

such that all edge weights lie in $[a, 1+a]$, where a controls the surface tension. Note, that scaling all edge weights with a constant factor does not change the resulting set of cut edges. As the global maximum Γ_{max} can be arbitrarily large, local fluctuation of the GCM might be vanishingly small in relation to Γ_{max} (see Figure 7 left). Since the graph cut also minimizes the surface area while maximizing for confidence, the edge weights need to have sufficient local variation to avoid that the graph cut only minimizes the number of cut edges and thus the surface area (*shrinking bias*). In order to cope with that, we apply a technique similar to an adaptive histogram equalization which we call *local GCM balancing*. Instead of using the global maximum in Equation 5 we replace it with the weighted local maximum (LM) of the GCM at point x. We compute $\Gamma_{LM}(x)$ by

$$\Gamma_{LM}(x) = \max_{y \in \mathbb{R}^3} \left[W\left(\frac{\|x - y\|}{2^{-\ell} \cdot \mathcal{B}_{edge}} \right) \cdot \Gamma(y) \right] \tag{6}$$

where $\mathcal{B}_{edge}$ is the edge length of the bounding cube. We employ a weighting function W to define the scope in which the maximum is computed. We define W as

$$W(d) = \begin{cases} 1 - \left(\frac{d}{\frac{1}{2}\mathcal{D}} \right)^c & \text{if } d \leq \frac{1}{2}\mathcal{D} \\ 0 & \text{if } d > \frac{1}{2}\mathcal{D} \end{cases} \tag{7}$$

where $\mathcal{D}$ is the filter diameter in voxels. We used $\mathcal{D} = 11$ and $c = 4$ in all our experiments. W is continuous in order to ensure continuity of the GCM. See Figure 7 (right) to see the effect of local GCM balancing.

After the graph cut, each voxel corner on octree level ℓ is either labeled interior or exterior which we can think of as binary signed distance values. In particular, since the subdivision from level $\ell - 1$ is regular we have labels for all voxel corners, the voxel center, the center of each face and edge. This will be exploited during final surface extraction in the next Section.

7 Multi-resolution surface reconstruction

Due to memory limitations, it is often impossible to reconstruct the whole scene on a resolution high enough to capture all sampled details. An adaptive multi-resolution approach which reconstructs different scene regions on adaptive resolutions depending on the sample footprints is therefore desirable. During the GCM sampling on octree level ℓ we marked voxels that need to be processed on higher resolution. After the graph cut we dilate this set of voxels several times and regularly subdivide the resulting voxel set to obtain a new crust $V^{\ell+1}_{crust}$. The crust segmentation can be obtained from the graph cut on level ℓ, as this cut effectively assigns each voxel corner a label *interior* or *exterior*. For boundary voxel corners in $V^{\ell+1}_{crust}$ that coincide with voxel corners on level ℓ we simply transfer the label. This ensures a continuous reconstruction across level boundaries. For voxel corners that lie on a parent voxel edge or face, i.e., between two or four voxel corners on level ℓ, we obtain the conform label of the surrounding voxel corners or we leave it unknown. The new crust $V^{\ell+1}_{crust}$ is now ready for graph cut optimization on level $\ell + 1$ (see Figure 3d+e). For voxel corners that coincide with voxel corners on the lower resolution the resulting labeling on level $\ell + 1$ overwrites the labeling obtained before.

The recursive refinement stops if the maximum level ℓ_{max} is reached or no voxels are marked for further processing. Due to our refinement scheme the last subdivision in the octree is always regular, i.e., all eight octants are present. The graph cuts define the voxel corners of the finest voxels as interior or exterior.

7.1 Final surface extraction

To extract the final surface we apply a combination of marching cubes and marching tetrahedra. The decision is made voxel-by-voxel one level above the finest level. Note that the last subdivision step is always regular. If the voxel is single-resolution containing 27 labeled voxel corners, we apply classical marching cubes to all eight child voxels. We interpret the voxel corner labels as binary signed distance values. If the voxel is multi-resolution, i.e., there is a change in resolution present affecting at least one of the cube edges or faces, we apply the tetrahedralization scheme by Manson and Schaefer [16] (see Figure 8). We hereby place dual vertices at voxel corners and at the centers of edges, faces, and voxels. These positions coincide with voxel corners of the finest levels providing

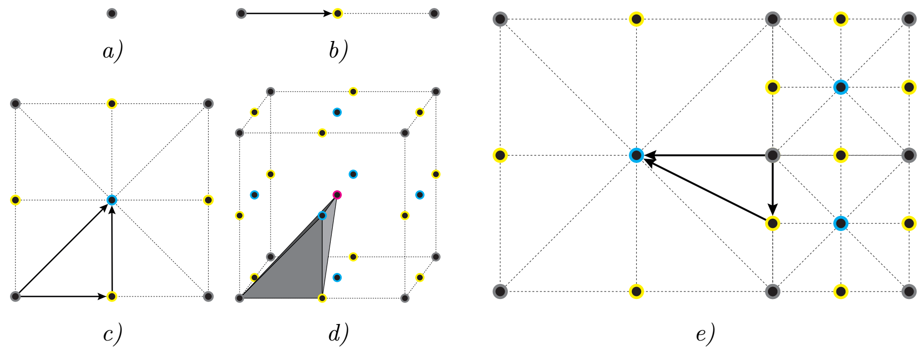


Fig. 8. Tetrahedralization of the multi-resolution grid. We connect a vertex (a) with the dual vertex of an edge (b), add a face vertex (c), and form a tetrahedron by adding the dual vertex of a cell (d). Adaptive triangulation of the multi-resolution grid (e). Tetrahedralization scheme and figures similar to Manson and Schaefer [16].

the binary signed distance values needed for the subsequent marching tetrahedra. Now, we only need to take care of voxel faces where triangles produced by marching cubes and triangles produced by marching tetrahedra meet. It is possible that T-vertices were created here but this can be easily fixed using an edge flip or vertex collapse. The final multi-resolution surface mesh is watertight and has different sized triangles depending on the details present in the corresponding areas.

8 Results

We will now present results of our method on different data sets (see Table 1). The source code is publicly available on the project page [19]. Our experiments were performed on a 2.7 GHz AMD Opteron with eight quad-core processors and 256GB RAM. All input data was generated from images using a robust structure-from-motion system [23] and an implementation of a recent MVS algorithm [8] applied to down-scaled images. We used all reconstructed points from all depth maps as input samples for our method. The footprint size of a sample is computed as the diameter of a sphere around the sample's 3D position whose projected diameter in the image equals the pixel spacing. For all graph cuts involved we used the publicly available library by Boykov and Kolmogorov [3].

The Temple is a widely used standard data set provided by the Middlebury Multi-View Stereo Evaluation Project [20, 17] and consists of 312 images showing a temple figurine. This data set can be considered to be single-resolution since all input images have the same resolution and distance to the object, resulting in the complete temple surface to be reconstructed on the same octree level in our algorithm. The reconstruction quality (Figure 9) is comparable to other state-of-the-art methods. We submitted reconstructed models created for a previous submission [18] for the TempleFull and the TempleRing variant (using only a

data set	sample points	vertices	octree level	comp. time	rel. variation in footprint
Temple	22 M	0.5 M	9	1 h	1.5
Kopernikus	32 M	3.3 M	10–12	1.5 h	38
Stone	43 M	4.3 M	8–14	4.5 h	75
Citywall	80 M	8.6 M	11–16	6 h	209

Table 1. The data sets we used and the number of sample points, the number of vertices in the resulting meshes, octree levels used for surface extraction, computation time and relative variation in footprint size.

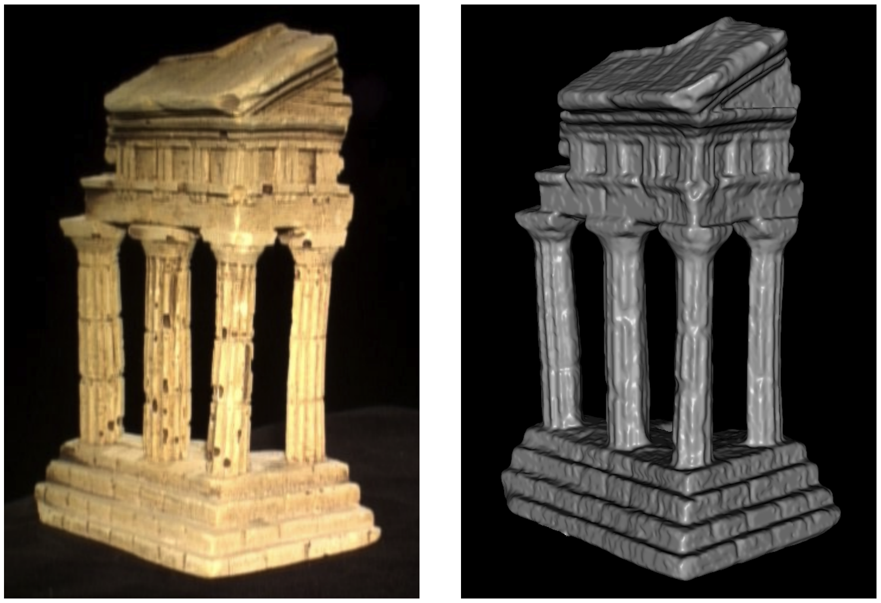


Fig. 9. An input image of the Temple data set (left) and a rendered view of our reconstructed model (right).

subset of 47 images as input to the pipeline) to the evaluation. For TempleFull we achieved the best accuracy (0.36 mm, 99.7 % completeness), for the TempleRing we achieved 0.46 mm at 99.1 % completeness.

The stone data set consists of 117 views showing a region around a portal where one characteristic stone in the wall is photographed from a close distance leading to high-resolution sample points in this region. Overall we have a factor of 75 of variation in footprint sizes. In Figure 10 we compare our reconstruction with Poisson surface reconstruction [13]. In the overall view our reconstruction looks significantly better, especially on the ground where our method results in less noise. In the close-up view also Poisson surface reconstruction shows the fine details. Due to the fact that the sampling density is much higher around

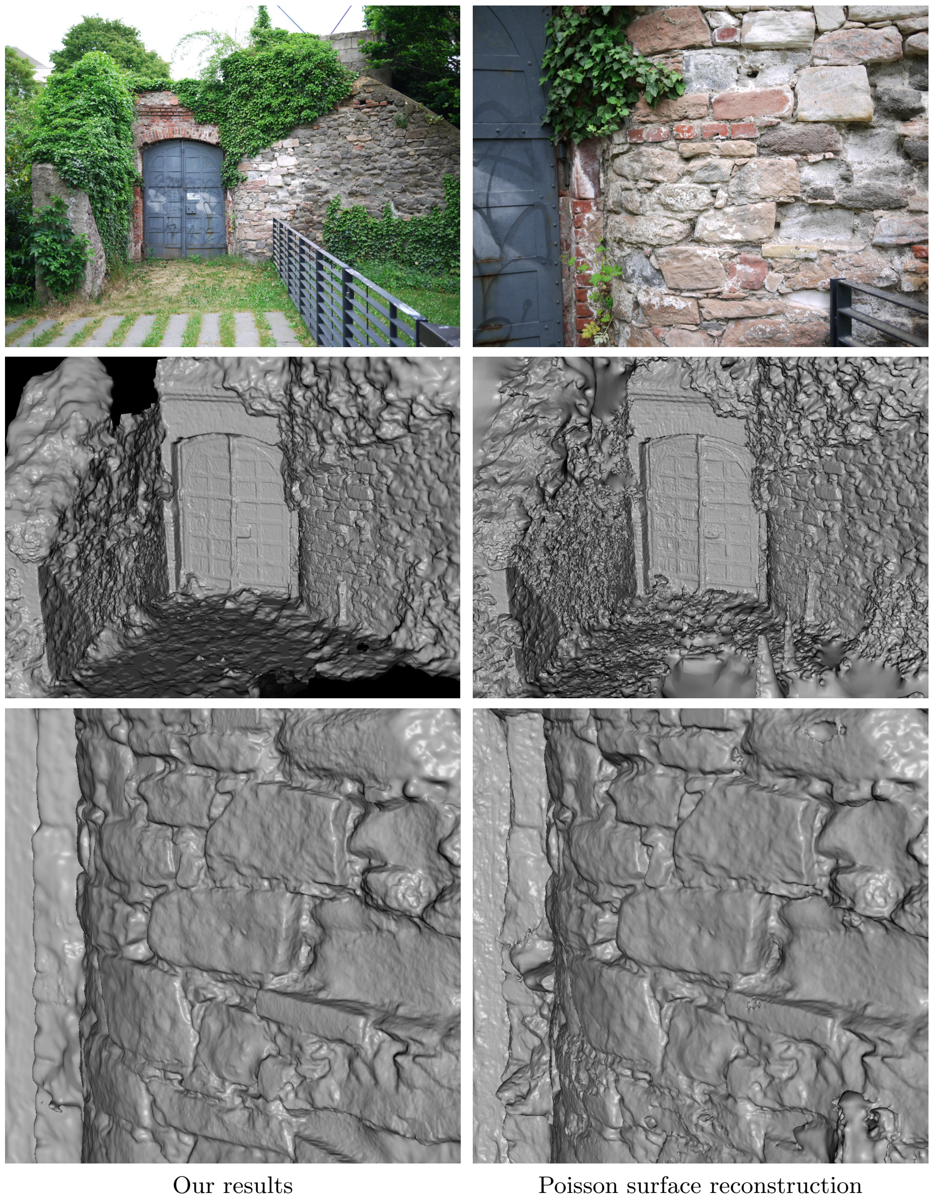


Fig. 10. *Top:* Example input images of the stone data set. *Middle + Bottom:* Comparison of our reconstruction (left) with Poisson surface reconstruction [13] (right). Although Poisson surface reconstruction does not take footprints into account the reconstruction shows fine details due to the higher sampling density. However, our surface shows significantly less noise and clutter.

the particular stone Poisson surface reconstruction used smaller triangles for the reconstruction.

The Citywall data set consists of 487 images showing a large area around a city wall. The wall is sampled with medium resolution, two regions though are sampled with very high resolution: the fountain in the middle and a small sculpture of a city to the left (Figure 11 top). Our multi-resolution method is able to reconstruct even fine details in the large scene where sample footprints differ up to a factor of 209. In consequence, the reconstruction spans six octree levels and detailed regions are triangulated about 32 times finer than low-resolution regions. The middle image of Figure 11 shows the entire mesh whereas the bottom images show close-ups of the highly detailed surface regions. One can even recognize some windows of the small buildings in the reconstructed geometry.

The Kopernikus data set (Figure 12) consists of 334 images showing a statue with a man and a women. The underlying surface geometry is particularly challenging due to its high genus. The data set is also multi-resolution in the sense that we took close-up views of the area around the hands. We compare our reconstruction against VRIP [5] and the depth map fusion by Fuhrmann and Goesele [6] (Figure 13). It is clearly visible that our model contains significantly less noise and shows no clutter around the real surface. Also, the complex topology of the object is captured very well in comparison to the other methods. However, in regions with low-resolution geometry staircase artifacts are visible due to the surface extraction from a binary signed distance field. This is also visible in the wireframe rendering in Figure 12 (bottom right) showing the dense triangulation of the women's face versus the coarse triangulation of the men's upper body.

9 Conclusion and future work

We presented a robust surface reconstruction algorithm that works on general input data. To our knowledge, except for the concurrent work of Fuhrmann and Goesele [6], we are the first to take the footprint of a sample point into account during reconstruction. Together with a robust crust computation and an adaptive multi-resolution reconstruction approach we are able to reconstruct fine detail in large-scale scenes. We presented results comparable to state-of-the-art techniques on a benchmark data set and proved our superiority on challenging large-scale outdoor data sets and objects with complex topology. The triangle meshes are manifold and watertight and show an adaptive triangulation with smaller triangles in regions where higher details were captured.

In future work, we plan to explore other ways to distribute a sample point's confidence over the volume, e.g., taking the direction to the sensor into account. This would allow us to better model the generally anisotropic error present in reconstructed depth maps.

Acknowledgements This work was supported in part by the DFG Emmy Noether fellowship GO 1752/3-1.

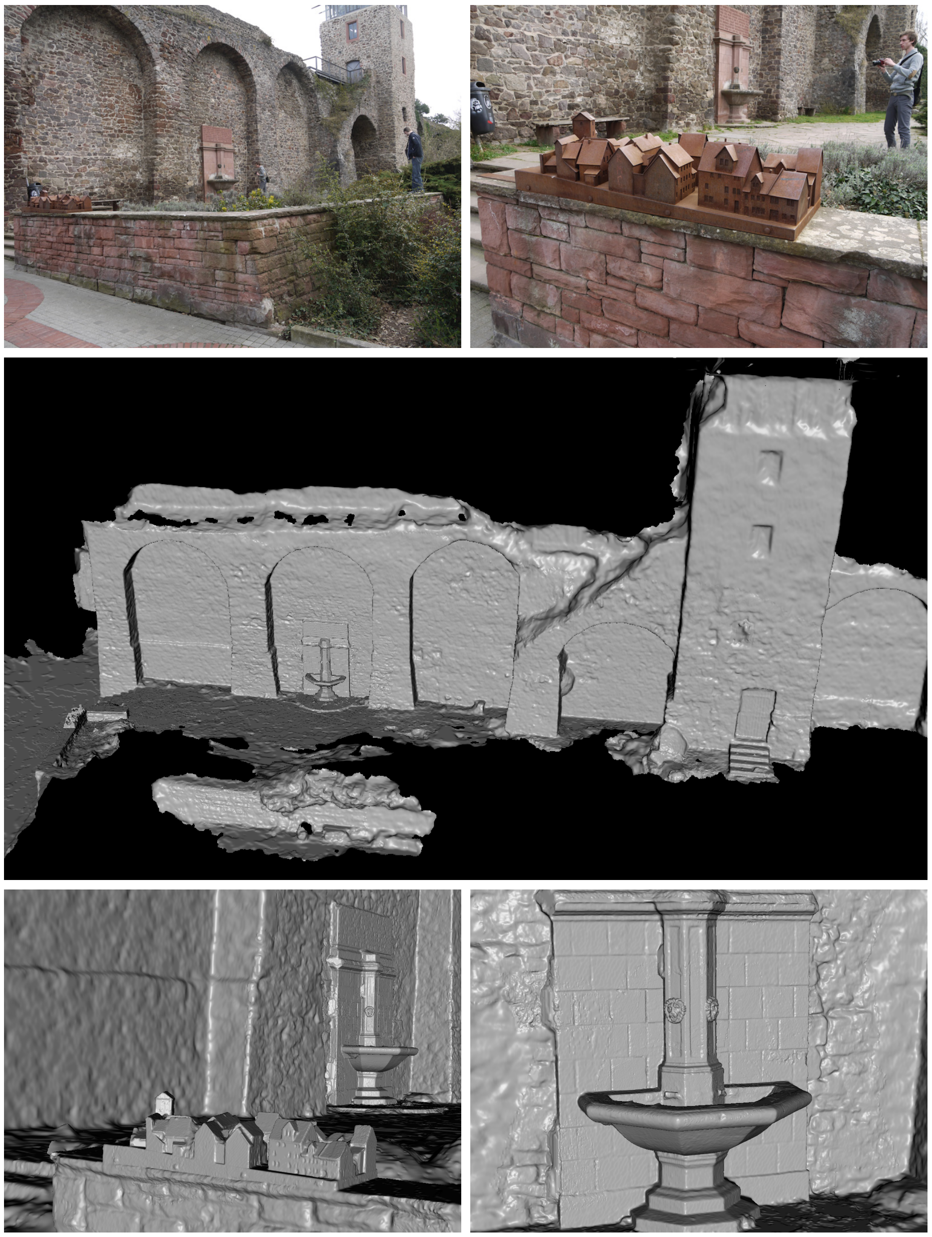


Fig. 11. Top: Two input images of the Citywall data set. Middle: Entire model (color indicates the octree level, red is highest). Bottom: Close-ups of the two detailed regions.

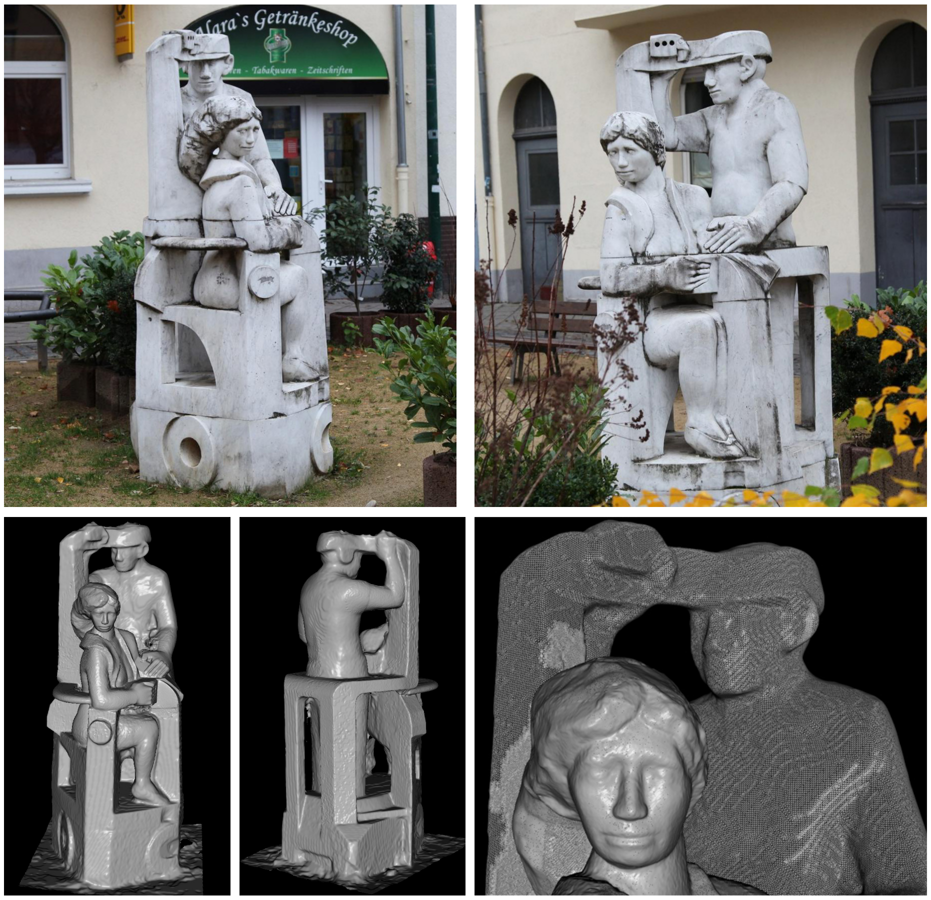


Fig. 12. Two input images of the Kopernikus data set, the complete reconstructed model from two perspectives and a close-up of the wireframe showing the adaptively triangulated mesh.

References

1. Alliez, P., Cohen-Steiner, D., Tong, Y., Desbrun, M.: Voronoi-based variational reconstruction of unoriented point sets. In: Proc. of Eurographics Symposium on Geometry Processing (2007)
2. Boykov, Y., Kolmogorov, V.: Computing geodesics and minimal surfaces via graph cuts. In: Proc. of IEEE International Conference on Computer Vision (2003)
3. Boykov, Y., Kolmogorov, V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. IEEE Transactions on Pattern Analysis and Machine Intelligence (2004)
4. Boykov, Y., Veksler, O.: Graph cuts in vision and graphics: Theories and applications. In: Handbook of Mathematical Models in Computer Vision (2006)
5. Curless, B., Levoy, M.: A volumetric method for building complex models from range images. In: Proc. of ACM SIGGRAPH (1996)

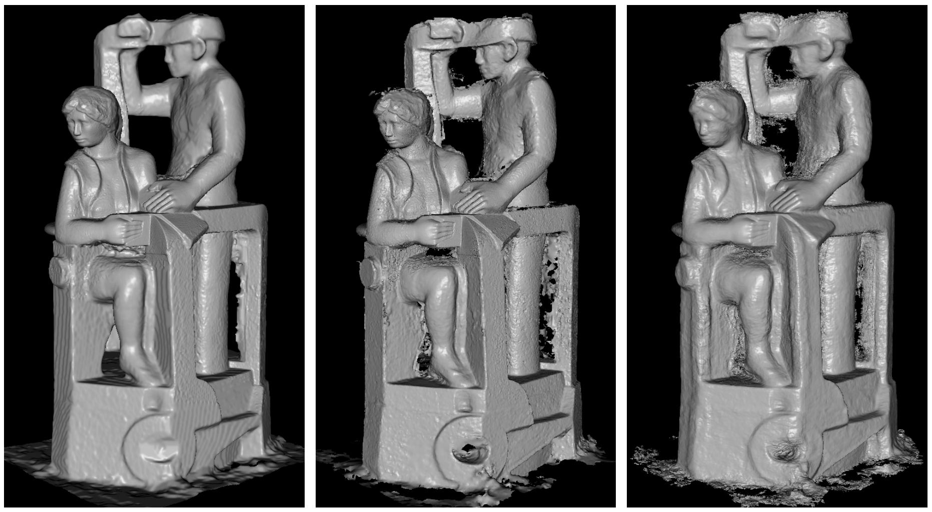


Fig. 13. Comparison of our reconstruction (*left*) with depth map fusion (*middle*) [6] and VRIP (*right*) [5].

6. Fuhrmann, S., Goesele, M.: Fusion of depth maps with multiple scales. In: Proc. of ACM SIGGRAPH Asia (2011)
7. Furukawa, Y., Curless, B., Seitz, S.M., Szeliski, R.: Towards internet-scale multi-view stereo. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition (2010)
8. Goesele, M., Snavely, N., Curless, B., Hoppe, H., Seitz, S.M.: Multi-view stereo for community photo collections. In: Proc. of IEEE International Conference on Computer Vision (2007)
9. Habbecke, M., Kobbelt, L.: A surface-growing approach to multi-view stereo reconstruction. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition (2007)
10. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W.: Surface reconstruction from unorganized points. In: Proc. of ACM SIGGRAPH (1992)
11. Hornung, A., Kobbelt, L.: Hierarchical volumetric multi-view stereo reconstruction of manifold surfaces based on dual graph embedding. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition (2006)
12. Hornung, A., Kobbelt, L.: Robust reconstruction of watertight 3D models from non-uniformly sampled point clouds without normal information. In: Proc. of Eurographics Symposium on Geometry Processing (2006)
13. Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: Proc. of Eurographics Symposium on Geometry Processing (2006)
14. Labatut, P., Pons, J.P., Keriven, R.: Robust and efficient surface reconstruction from range data. Computer Graphics Forum (2009)
15. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3D surface construction algorithm. In: Proc. of ACM SIGGRAPH (1987)
16. Manson, J., Schaefer, S.: Isosurfaces over simplicial partitions of multiresolution grids. In: Proc. of Eurographics (2010)

17. Middlebury multi-view stereo evaluation, http://vision.middlebury.edu/mview/
18. Mücke, P., Klowsky, R., Goesele, M.: Surface reconstruction from multi-resolution sample points. In: Proc. of Vision, Modeling and Visualization (2011)
19. Project page, http://www.gris.tu-darmstadt.de/projects/multires-surface-recon/
20. Seitz, S.M., Curless, B., Diebel, J., Scharstein, D., Szeliski, R.: A comparison and evaluation of multi-view stereo reconstruction algorithms. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition (2006)
21. Shalom, S., Shamir, A., Zhang, H., Cohen-Or, D.: Cone carving for surface reconstruction. In: Proc. of ACM SIGGRAPH Asia (2010)
22. Sinha, S.N., Mordohai, P., Pollefeys, M.: Multi-view stereo via graph cuts on the dual of an adaptive tetrahedral mesh. In: Proc. of IEEE International Conference on Computer Vision (2007)
23. Snavely, N., Seitz, S.M., Szeliski, R.: Skeletal sets for efficient structure from motion. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition (2008)
24. Vu, H.H., Keriven, R., Labatut, P., Pons, J.P.: Towards high-resolution large-scale multi-view stereo. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition (2009)
25. Zach, C., Pock, T., Bischof, H.: A globally optimal algorithm for robust TV-L1 range image integration. In: Proc. of IEEE International Conference on Computer Vision (2007)