Vision, Modeling, and Visualization (2011)
Peter Eisert, Konrad Polthier, and Joachim Hornegger (Eds.)

Surface reconstruction from multi-resolution sample points

Patrick Mücke, Ronny Klowsky, and Michael Goesele

TU Darmstadt, Germany

Abstract

Robust surface reconstruction from sample points is a challenging problem, especially for real-world input data. We significantly improve on a recent method by Hornung and Kobbelt [HK06b] by implementing three major extensions. First, we exploit the footprint information inherent to each sample point, that 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. Second, we propose a new crust computation making the method applicable to a substantially broader range of input data. This includes data from objects that were only partially sampled, a common case for data generated by multi-view stereo applied to Internet images. Third, we adapt the volumetric resolution locally to the footprint size of the sample points which allows to extract fine detail even in large-scale scenes. The effectiveness of our extensions is shown on challenging outdoor data sets as well as on a standard benchmark.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.5]: Computational Geometry and Object Modeling—Surface Reconstruction I.4.8 [Computer Vision]: Scene Analysis—Surface fitting

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 [SCD*06, GSC*07, HK07, FCSS10] 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*. 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 [HK06b]. 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.

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. 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 [HK07]. They represent the surface with surfels (surface elements) of varying size depending on the image texture. Furukawa et al. [FCSS10] consider footprints to estimate reconstruction accuracy. However, their method effectively discards samples with large footprints prior to final surface extraction. In this paper, we propose a way to model the footprint during the reconstruction process. In particular, we create a modified confidence map that includes the footprint information.

The confidence map is only evaluated inside a *crust*, a volumetric region around the sample points. In [HK06b], 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 1. 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. [VKLP09] pointed out, volumetric methods such as [HK06b] 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 yet producing a watertight surface. This allows us to reconstruct fine details even in large-scale scenes such as the Citywall data set (see Fig. 10).

The remainder of the paper is organized as follows: First, we review previous work and give an overview of our reconstruction pipeline (Sec. 3). Details of the individual steps are explained in Sec. 4–7. Finally, we present results of our method on standard benchmark data as well as challenging outdoor scenes (Sec. 8) and wrap up with a conclusion and an outlook on future work (Sec. 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. [HDD*92]. 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 [LC87].

If the sample points originate from multiple range scans, additional information is available. VRIP [CL96] uses the connectivity between neighboring samples as well as 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. [ZPB07] 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.

Very recently, Shalom et al. [SSZCO10] presented a technique called cone carving. 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. Such methods are often not able to deal with the lower quality data generated by MVS methods from outdoor scenes.

Kazhdan et al. [KBH06] reformulate the surface reconstruction problem into 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. [ACSTD07] use the normals to derive a tensor field and try then to compute an implicit function whose gradients best approximate that tensor field. Additionally, they presented 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 [BK03] introduced the idea of reconstructing surfaces by the means of computing a cut on a graph embedded in continuous space. They also showed 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 [HK06a] initially used the volumetric graph cut to reconstruct a surface given a photo-consistency function of a region in space. They proposed to embed an octahedral graph structure into the volume and showed how to extract a mesh from the set of cut edges. They later generalized their method to work on non-uniformly sampled point clouds and improved the mesh extraction procedure [HK06b].

An example of using graph cuts in multi-view stereo is the work of Sinha et al. [SMP07]. They build a tetrahedral mesh according to estimated photo-consistency values. The final graph cut is performed on the dual of the tetrahedral mesh followed by a photo-consistency driven mesh refinement. Labatut et al. [LPK09] build a tetrahedral mesh around points merged from multiple range images. They introduce a surface visibility term, that takes the direction to the sensor into account, and a surface quality term. 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. [VKLP09] 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 then refined mixing photo-consistency in the images and a regularization force. However, none of the existing methods properly incorporates the footprint of a sample during surface reconstruction.

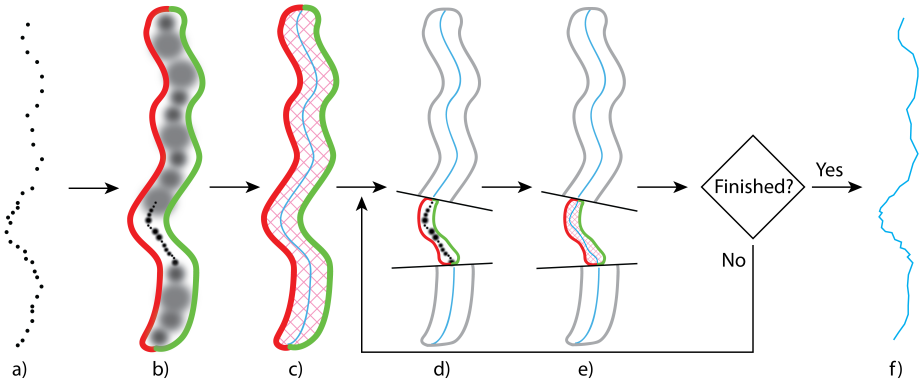

Figure 1: *Overview of our reconstruction pipeline. We first construct a segmented crust containing the input samples (a). Each input sample contributes to the global confidence map (b). From a minimal cut on the octahedral graph, embedded into the crust, we extract a surface (c). We build a finer crust in regions with high-resolution samples (d) and extract a higher resolution surface that connects to the low-resolution surface (e). This process is iterated until all surface regions are reconstructed with adequate resolution. The final surface is a composition of all surfaces with different resolutions (f).*

3. Overview

The input of our algorithm is a set of *surface samples* representing the scene (Fig. 1a). 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.

We then first 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 (see Fig. 1b). 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 (see Fig. 1c). This graph is designed, such that an s-t cut will separate interior from exterior and thus represents a surface of the scene. The edge weights of this graph are chosen such that a minimal cut represents the most confident surface w.r.t the confidence map. After the graph cut, the triangle mesh of the surface is extracted. We then identify surface regions with sampled details too fine to be adequately reconstructed on the current reconstruction resolution. Only these surface regions are then reconstructed on a higher resolution (see Fig. 1d+e). Starting on a low resolution, we repeat this process iteratively until eventually all fine details were reconstructed. Finally, we merge the surface meshes of the different resolutions into one multi-resolution mesh of the scene, which is the output of our algorithm (see Fig. 1f).

4. Crust computation

We subdivide the cubic bounding box into a regular voxel grid. For memory efficiency and to be able to easily increase the voxel resolution, this voxel grid is represented by an octree data structure. Surface extraction in each iteration is performed on a single octree level.

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, 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 Fig. 2 for an overview).

Crust generation We initialize the crust at octree level ℓ_{start} as the set of voxels containing surface samples (Fig. 2 top left). We dilate this sparse set of voxels several times over the 6-neighborhood of voxels, followed by a morphological closing operation (Fig. 2 top right). 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.

Crust segmentation Let ∂V_{crust} be the set of voxel faces on the crust boundary. By assigning the labels *interior* and *exterior* to these faces we define the interior and exterior of the scene. The labels are determined per crust voxel v on the crust boundary by comparing two normals. The *crust normal* $\vec{n}_v^{crust}$ is obtained by averaging the normal vectors of all faces of v in ∂V_{crust} (Fig. 2 middle left). The *surface normal* $\vec{n}_v^{surf}$ (Fig. 2 middle right) is computed by averaging the

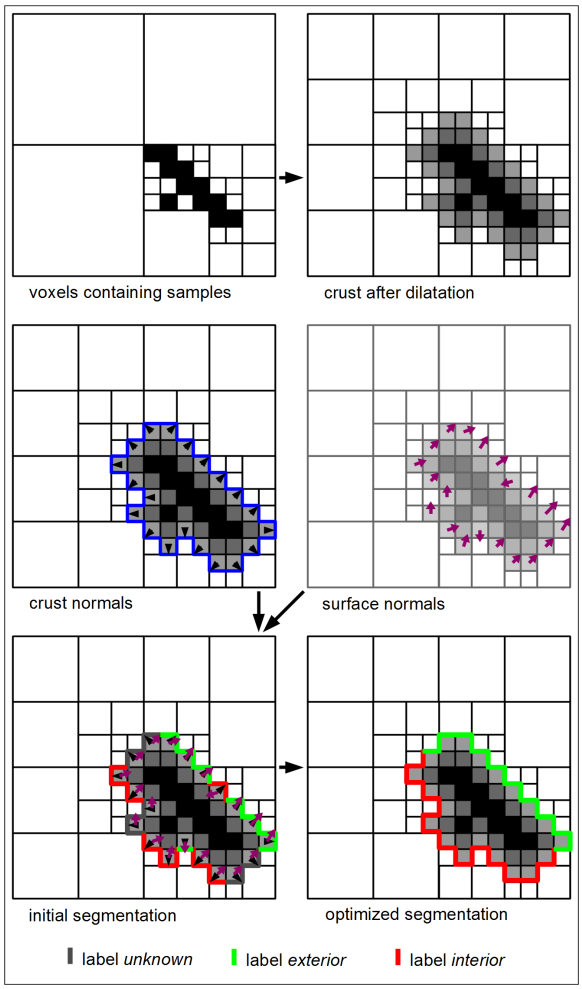

Figure 2: *Initial crust computation for lowest resolution: We initialize the crust with voxels containing sample points (top left) and dilate several times (top right). Crust normals (middle left) and surface normals (middle right) are computed for each voxel to determine an initial labeling (lower left). Optimization yields a homogenous crust surface segmentation (lower right).*

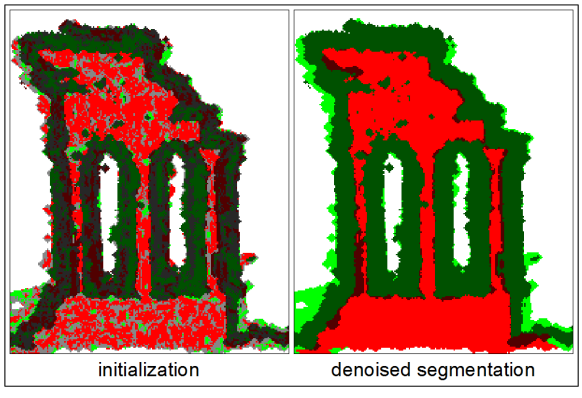

Figure 3: *Visualization of the crust surface for the Temple (cut off perpendicular to the viewing direction). The color is similar to Fig. 2. Light shaded surfaces are seen from the front, dark shaded ones are seen from the back.*

normals of all sample points in v. Voxels that do not contain any samples receive their surface normal through propagation during the crust dilatation. We determine the label for voxel v on the crust boundary by

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

with $\tau \in (0,1)$. We used $\tau = 0.7$ in all experiments.

Since surface normal information of the samples may only be a crude approximation, this initial labeling is noisy and has to be regularized (Fig. 2 bottom left and Fig. 3 left). We cast the problem of obtaining a homogenous crust labeling into a binary image denoising problem, which we solve using a graph cut optimization as described by Boykov and Veksler [BV06]. We build a graph with a node per voxel face in ∂V_{crust} and a graph edge connecting two nodes if the corresponding faces share a voxel edge. Since each face in ∂V_{crust} has four neighboring faces, the resulting graph is 4-regular. 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 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 face f_n:

$$w_n^{source} = \begin{cases} \mu & \text{if } f_n \text{ is labeled } interior \\ 1-\mu & \text{if } f_n \text{ is labeled } exterior \\ \frac{1}{2} & \text{if } f_n \text{ is labeled } unknown \end{cases} \quad (2)$$

$$w_n^{sink} = 1 - w_n^{source} \quad (3)$$

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 faces in ∂V_{crust} (Fig. 2 bottom right and Fig. 3 right). We used $w = 1$ and $\mu = 0.35$ in all experiments.

If different crust segments touch, the reconstructed surface is forced to go through these segment borders, as it has to separate *interior* from *exterior*. The denoising minimizes the number of segment borders and therefore prevents unwanted surfaces to be formed. If the scene surface is not sampled entirely, segment borders occur even for correct segmentations (see Fig. 2 bottom right). This forces the surface to pass through the segment border 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 to force meshes with different resolutions to align for easy stitching (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 Fig. 1b). 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: $\Gamma(x) = \sum_s \gamma_s(x)$.

Implementation In practice we set each LCM γ_s to zero for all points for which the distance to x_s is larger than three times the footprint size. Now, let ℓ be the octree level at which we want to extract the surface. Our goal is to evaluate the GCM Γ at the center positions of the crust voxels $\{x_v\}_{v \in V_{crust}}$ on octree level ℓ. In order to do that we iterate once over all samples s and add their LCM γ_s to the corresponding octree nodes.

For efficiency, we sample each γ_s only at a small number of positions ($\approx 5^3$) within its spatial support. We exploit the octree data structure and accumulate each γ_s to nodes at an 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 in a depth-first octree traversal (by adding the value at a node to all children's nodes). LCMs of sample points with small footprints might be too narrow to be adequately sampled on octree level ℓ. For those samples we temporarily increase the footprint α_s for the computation of the LCM γ_s and mark the corresponding voxel for later processing at higher resolution.

6. Graph cut and surface extraction

To reconstruct the optimal surface we use the same octahedral graph layout as used by Hornung and Kobbelt [HK06b]. The graph embedding is designed such that each voxel v contains 12 edges (see Fig. 4), all with the same weight w_v. In our case, the optimal surface should maximize the global confidence Γ. Therefore we want to obtain small edge weights for voxels with high confidence and vice versa. A straightforward way to implement this would be

$$w_v = 1 - \frac{\Gamma(x_v)}{\Gamma_{max}} + a \quad \text{with} \quad \Gamma_{max} = \max_{x \in \mathbb{R}^3} \Gamma(x) \qquad (4)$$

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

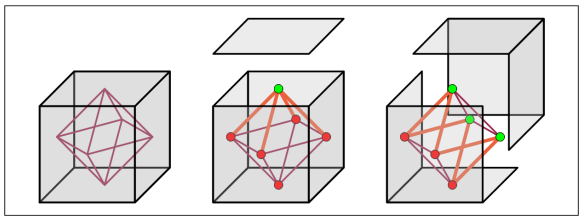

Figure 4: *The octahedral graph layout and different configurations of cut edges.*

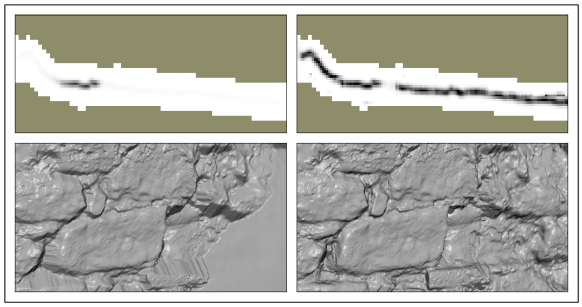

Figure 5: *The GCM values can be arbitrarily large leading to near-constant edge weights in large regions of the volume (left). Our* local GCM balancing *compensates for that allowing the final graph cut to find the correct surface (right).*

fluctuation of the GCM might be vanishingly small in relation to Γ_{max} (see Fig. 5 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 4 we replace it with the weighted local maximum (LM) of the GCM at point x. We compute $\Gamma_{LM}(x)$ by

$$\Gamma_{LM}(x) = \max_{y \in \mathbb{R}^3} \left[W\left(\frac{\|x-y\|}{2^{-\ell} \cdot \mathcal{B}_{edge}} \right) \cdot \Gamma(x) \right] \qquad (5)$$

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

$$W(d) = \begin{cases} 1 - \left(\frac{d}{\frac{1}{2}\mathcal{D}}\right)^c & \text{if } d \le \frac{1}{2}\mathcal{D} \\ 0 & \text{if } d > \frac{1}{2}\mathcal{D} \end{cases} \qquad (6)$$

where $\mathcal{D}$ is the filter diameter in voxels. We used $\mathcal{D} = 9$ and $c = 6$ in all our experiments. W is continuous in order to ensure continuity of the GCM. See Fig. 5 (right) to see the effect of local GCM balancing.

After the graph cut, the mesh $\mathcal{M}_\ell$ for octree level ℓ is extracted from the set of cut edges. A mesh vertex is placed in

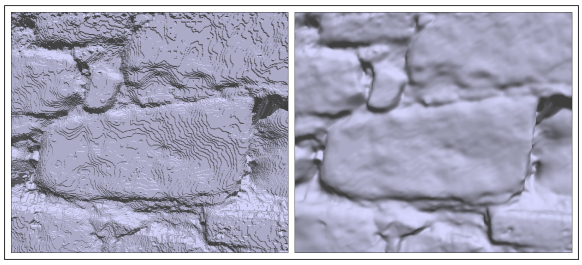

Figure 6: *The extracted mesh from the graph cut with staircase artifacts (left) and the smoothed final mesh (right).*

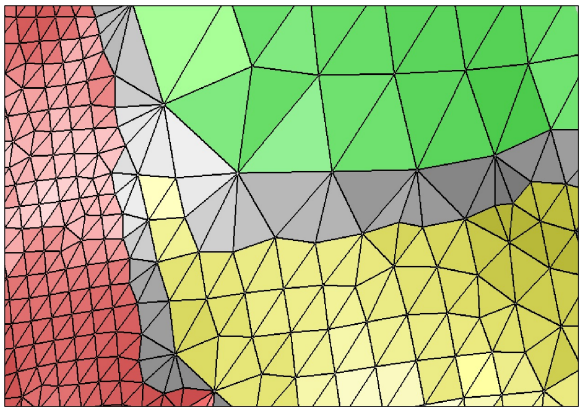

Figure 7: *The resulting meshes from different reconstruction resolutions are stitched together to the final mesh.*

the center of each voxel containing a cut edge. By traversing each block of 2x2x2 voxels, triangles are created according to the specific cut edge configuration, see [HK06b] for details. Since mesh vertices are only placed at voxel centers the mesh contains staircase artifacts. We therefore perform Laplacian smoothing after the mesh extraction (see Fig. 6).

7. Multi-resolution 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 reconstruction which reconstructs different scene regions on different resolutions depending on their sample resolution is therefore desirable (see Fig. 1).

During surface reconstruction on octree level ℓ we marked voxels for processing on higher resolution. We dilate this set of voxels several times and subdivide the resulting voxel set to obtain a new crust $V_{crust}^{\ell+1}$ for surface extraction on octree level $\ell+1$. The crust surface segmentation can be obtained from the graph cut on level ℓ, as this cut effectively assigns each voxel face a label *interior* or *exterior*. We set the label of a face in $\partial V_{crust}^{\ell+1}$ to the label of the parent face on level ℓ. $V_{crust}^{\ell+1}$ is now ready for reconstruction on level $\ell+1$.

data set	sample points	vertices / triangles	octree level	comp. time	RAM peak
Temple	24 M	1.5 / 3.0 M	10	2 h	∼15 GB
Wall	7 M	3.0 / 6.0 M	7–14	1 h	∼15 GB
Notre Dame	102 M	4.2 / 8.5 M	7–14	15 h	∼85 GB
Citywall	63 M	6.5 / 13.0 M	8–16	15 h	∼80 GB

Table 1: *The data sets we used and the number of sample points, the number of vertices and triangles in the resulting meshes, octree levels used for surface extraction, computation time and estimated peak memory usage.*

After the finest level ℓ_{max} is reached or no voxels are marked for further processing we create the final mesh $\mathcal{M}$ from the meshes $\mathcal{M}_\ell$. We start by adding $\mathcal{M}_{\ell_{start}}$ to $\mathcal{M}$ and iteratively add meshes of higher resolution ℓ while discarding lower resolution triangles in that area. To be more precise, we keep all triangles that have at least one vertex in $V_{crust}^{\ell-1} \setminus V_{crust}^{\ell}$. The meshes are clipped by contracting vertices of the high resolution mesh to vertices of lower resolutions where the meshes meet (see Fig. 7). Let w^j be a voxel on level $j < \ell$ with vertex $\hat{w}^j$ in $\mathcal{M}$ (i.e., vertex $\hat{w}^j$ was added on reconstruction level j). If $v^\ell \subseteq w^j$ is a child voxel of w^j also having a vertex $\hat{v}^\ell$ in $\mathcal{M}$, every occurrence of $\hat{v}^\ell$ is replaced by $\hat{w}^j$ in $\mathcal{M}$. Some triangles became invalid (i.e., not all three vertices are distinct) in this process and need to be removed from $\mathcal{M}$. The resulting mesh is closed and merges reconstructions of different resolutions.

If the boundary of the high resolution mesh passes through voxels which do not contain a low resolution mesh boundary, this part of the high resolution mesh cannot be stitched to the low resolution mesh, resulting in a hole in the multi-resolution mesh. The meshes of different resolutions therefore have to be constrained, such that their borders align. This can be enforced by the labeling of the crust surface. In our current implementation the segmentation of the level $\ell+1$ crust is obtained solely from the level ℓ graph cut, not from the actual mesh geometry. This slightly inaccurate labeling causes sparse and small holes in our multi-resolution mesh. Due to their limited affect on the output mesh, these holes can be easily filled by post processing.

8. Results

We will now present results of our method on different data sets (see Table 1). All input data was generated from images using a robust structure-from-motion system [SSS08] and an implementation of a recent MVS algorithm [GSC*07]. We used all reconstructed points from all depth maps as input samples for our method. We set the footprint size of a sample to 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 we used the publicly available library by Boykov and Kolmogorov [BK04].

Figure 8: *An input image of the Temple data set (left) and a rendered view of our reconstructed model (right).*

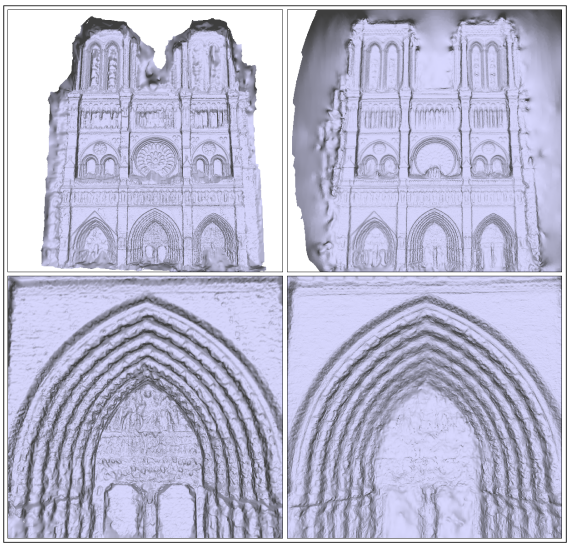

Figure 9: *Comparing the reconstruction of the Notre Dame data set using our method (left) and Poisson surface reconstruction (right).*

The Temple is a widely used standard data set provided by the Middlebury Multi-View Stereo Evaluation Project [SCD*06, Mid] 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 (see Fig. 8) is comparable to other state-of-the-art methods. We submitted reconstructed models for the TempleFull and the TempleRing variant (using only a 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 Notre Dame data set consists of 591 images automatically downloaded from the Internet showing the façade of the Notre Dame Cathedral in Paris. The data set is highly inhomogenous as the images are taken with different cameras and strongly varying vantage points. It can therefore be seen as a multi-resolution data set with strongly varying footprint sizes. Furthermore, the images were taken under varying lighting conditions, contain foreground clutter, and some were even "photoshopped". Consequently, the MVS data is very noisy and contains outliers. Our system creates good quality reconstructions for this challenging data set (see Fig. 9 left). We compare our method to the widely used Poisson surface reconstruction [KBH06] using the same data points (see Fig. 9 right). While both methods show comparable results on the entire façade our reconstruction shows clearly more details in the portal region.

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 (see Fig. 10). Our multi-resolution method is able to reconstruct even fine details in the large scene, spanning 9 levels of resolution. This means, that the detail regions are triangulated 512 times finer than low-resolution regions. The middle image of Fig. 10 shows a color-coded mesh visualizing the reconstruction resolution of different surface regions.

The Wall data set (see Fig. 11) is a multi-resolution data set consisting of 47 images. One characteristic stone in the wall is photographed from a close distance leading to high-resolution sample points in this region. Note how our algorithm increases resolution towards the stone.

9. Conclusion and future work

We presented a robust surface reconstruction algorithm that works on general input data. To our knowledge we are the first to take the footprint of a sample point into account during reconstruction. Together with a new crust computation and a multi-resolution extension we enhanced a recently published method substantially. We presented results comparable to state-of-the-art techniques on a benchmark data set and proved our superiority on challenging outdoor data sets, e.g., sample points obtained from MVS applied to images from the Internet.

In future work we would like to explore other ways to distribute a sample point's confidence over the volume, e.g., taking the direction to the sensor into account. We would also like to study different graph layouts that better approximate the euclidean metric in space. On the computational side, a parallelization of the global confidence map computation would lead to a significant speed up of our algorithm.

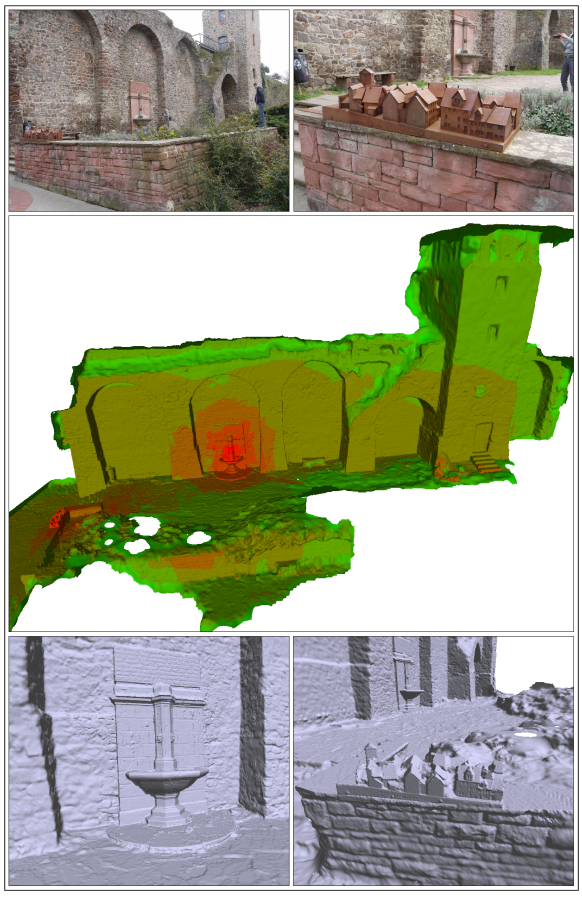

Figure 10: *Top: Two input images of the Citywall data set. Middle: Entire model (color indicates the octree level, red is highest). Bottom: Close-ups of the two detailed regions.*

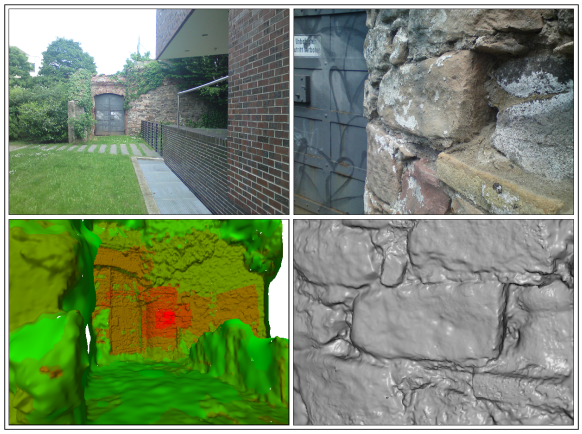

Figure 11: *Two input images of the Wall data set, the entire color-coded mesh, and a close-up of the characteristic stone.*

Acknowledgments This work supported by the DFG Emmy Noether fellowship GO 1752/3-1.

References

[ACSTD07] ALLIEZ P., COHEN-STEINER D., TONG Y., DESBRUN M.: Voronoi-based variational reconstruction of unoriented point sets. In *SGP* (2007). 2

[BK03] BOYKOV Y., KOLMOGOROV V.: Computing geodesics and minimal surfaces via graph cuts. In *ICCV* (2003). 2

[BK04] BOYKOV Y., KOLMOGOROV V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. on Pattern Analysis and Machine Intelligence* (2004). 6

[BV06] BOYKOV Y., VEKSLER O.: Graph cuts in vision and graphics: Theories and applications. In *Handbook of Mathematical Models in Computer Vision*. 2006. 4

[CL96] CURLESS B., LEVOY M.: A volumetric method for building complex models from range images. In *SIGGRAPH* (1996). 2

[FCSS10] FURUKAWA Y., CURLESS B., SEITZ S. M., SZELISKI R.: Towards internet-scale multi-view stereo. In *CVPR* (2010). 1

[GSC*07] GOESELE M., SNAVELY N., CURLESS B., HOPPE H., SEITZ S. M.: Multi-view stereo for community photo collections. In *ICCV* (2007). 1, 6

[HDD*92] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Surface reconstruction from unorganized points. In *SIGGRAPH* (1992). 2

[HK06a] HORNUNG A., KOBBELT L.: Hierarchical volumetric multi-view stereo reconstruction of manifold surfaces based on dual graph embedding. In *CVPR* (2006). 2

[HK06b] HORNUNG A., KOBBELT L.: Robust reconstruction of watertight 3D models from non-uniformly sampled point clouds without normal information. In *SGP* (2006). 1, 2, 5, 6

[HK07] HABBECKE M., KOBBELT L.: A surface-growing approach to multi-view stereo reconstruction. In *CVPR* (2007). 1

[KBH06] KAZHDAN M., BOLITHO M., HOPPE H.: Poisson surface reconstruction. In *SGP* (2006). 2, 7

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH* (1987). 2

[LPK09] LABATUT P., PONS J.-P., KERIVEN R.: Robust and efficient surface reconstruction from range data. *CGF* (2009). 2

[Mid] Middlebury multi-view stereo evaluation project. http://vision.middlebury.edu/mview/. 7

[SCD*06] SEITZ S. M., CURLESS B., DIEBEL J., SCHARSTEIN D., SZELISKI R.: A comparison and evaluation of multi-view stereo reconstruction algorithms. In *CVPR* (2006). 1, 7

[SMP07] SINHA S. N., MORDOHAI P., POLLEFEYS M.: Multi-view stereo via graph cuts on the dual of an adaptive tetrahedral mesh. In *ICCV* (2007). 2

[SSS08] SNAVELY N., SEITZ S. M., SZELISKI R.: Skeletal sets for efficient structure from motion. In *CVPR* (2008). 6

[SSZCO10] SHALOM S., SHAMIR A., ZHANG H., COHEN-OR D.: Cone carving for surface reconstruction. In *SIGGRAPH Asia* (2010). 2

[VKLP09] VU H.-H., KERIVEN R., LABATUT P., PONS J.-P.: Towards high-resolution large-scale multi-view stereo. In *CVPR* (2009). 2

[ZPB07] ZACH C., POCK T., BISCHOF H.: A globally optimal algorithm for robust TV-L1 range image integration. In *ICCV* (2007). 2