Global, Dense Multiscale Reconstruction for a Billion Points

Benjamin Ummenhofer and Thomas Brox
Computer Vision Group
University of Freiburg, Germany
{ummenhof, brox}@cs.uni-freiburg.de

Abstract

We present a variational approach for surface reconstruction from a set of oriented points with scale information. We focus particularly on scenarios with non-uniform point densities due to images taken from different distances. In contrast to previous methods, we integrate the scale information in the objective and globally optimize the signed distance function of the surface on a balanced octree grid. We use a finite element discretization on the dual structure of the octree minimizing the number of variables. The tetrahedral mesh is generated efficiently from the dual structure, and also memory efficiency is optimized, such that robust data terms can be used even on very large scenes. The surface normals are explicitly optimized and used for surface extraction to improve the reconstruction at edges and corners.

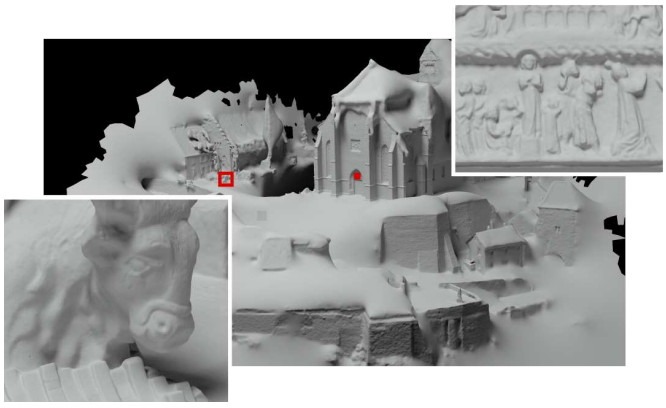


Figure 1. Reconstruction of a scene with large differences in scale. The top right and bottom left corner show a close-up view of the respectively marked spots.

1. Introduction

Current structure from motion pipelines can create sparse reconstructions of large scenes with thousands of images. Even city scale reconstructions have become feasible [4]. Such large scenes come along with several challenges for dense reconstruction. These include (1) an efficient scene representation, both in terms of memory and computational costs, (2) reconstruction of surfaces observed at different levels of detail, and (3) graceful handling of noisy and missing data.

In this paper, we deal with all three aspects. We propose for the first time to optimize a global cost function that takes the scale of the imaged points into account. Such scale adaptive reconstructions become important for scenes that are too large to be modeled at a single scale or where the focus of attention is concentrated on small parts of the scene as shown in Fig. 1.

Especially in these cases, the scene representation must be efficient in memory and should adapt to the scene structure rather than to the size of the input data. Therefore, we propose an octree representation in conjunction with point aggregation and a fast finite element discretization of the octree domain into a tetrahedral mesh based on the dual octree. Our discretization does not introduce additional nodes, thus allowing us to state the optimization problem with a minimum number of variables with respect to the the tree structure. Together with the small memory footprint this makes our approach suitable for the dense reconstruction from a billion points.

By casting the reconstruction as a global optimization problem we can complement the data fusion with regularization to deal with noisy or missing measurements. To this end, we use robust norms in the cost function. This is only possible because we counter the high memory requirements of robust norms by efficient storage of the data. To obtain a faithful reconstruction also of edges and corners at all levels of detail in the octree representation, we explicitly model and optimize the surface normals.

The input to our method is a point cloud with normal and scale information describing the size of the underlying pixel or patch in the world coordinate system. Such data can be obtained with off-the-shelf disparity estimation methods and structure-from-motion packages. In particular, we used the VisualSfM software [21, 20] to compute camera param-

eters and [8] for estimating the depth maps.

Based on this input, in a first pass, we compute a balanced octree representation of the scene by taking into account the sampling density and the scale of the points. The scale information determines the octree levels to which each individual point contributes.

In a second pass, we aggregate the points of each node in the octree. We treat each point as a signed distance function with compact support, the size of which is determined by the point's scale. After aggregation of the signed distance values and normals in each node we obtain a compact representation that does not require access to the original point cloud anymore.

Once the octree is built, we generate the dual octree structure and its tetrahedral mesh. We then minimize a discretized version of our energy functional on the tetrahedral mesh. The result of this optimization is a regularized signed distance function and its gradient. We generate a triangle mesh of its zero level set with a dual contouring approach to visualize the reconstruction.

2. Related Work

Many algorithms have been proposed for volumetric integration of depth data. We focus here on the most closely related ones.

Kazhdan and Hoppe [10] and Calakli and Taubin [1] take as input a point cloud with normal information and globally optimize a surface representation. However, they do not take the scale of points into account and without robust norms they are not robust to erroneous data. We compare to [10] and [1] in Section 8. Recently, Estellers *et al*. [3] proposed a version of [10] which uses a robust norm to penalize deviations from the point normals but uses least squares for the positions. They also do not deal with the scale.

To avoid artifacts due to different point resolutions, Fuhrmann and Goesele [5] propose a fusion method that averages samples from depth maps at compatible scales within an octree data structure. They compute a weighted average of signed distances similar to the VRIP method by Curless and Levoy [2]. In contrast to VRIP, Fuhrmann and Goesele use the scale information of each depth sample to select an appropriate octree level. In a later work Fuhrmann and Goesele [6] use basis functions with compact support that depend on the position, normal and scale of the point samples. These basis functions are aggregated in an octree representation. Both approaches by Fuhrmann and Goesele are local approaches and lack global regularization capabilities. We compare to them in Section 8.

Our regularization is strongly related to Pock *et al*. [15], who compute the signed distance function with robust energy terms for the data and a regularizer based on the total variation. They propose a convex functional with a second order total generalized variation (TGV) regularizer. The TGV regularization introduces an additional vector-valued function, which increases the computational complexity. Moreover, their robust data terms based on the Huber norm require storing all samples in memory during optimization. Zach [22] proposed a memory efficient robust approach using histograms to store the signed distance values. Both methods work on a regular grid, which prevents them to be applied to large scenes or scenes with large differences in scale.

Although some of the discussed methods [10, 1, 6] use the normal information to compute the implicit function, none of these methods uses normal information directly for the surface extraction. In contrast, we explicitly optimize the surface orientation *and* use this additional information for extracting the surface, which yields a more faithful representation of edges and corners.

3. Octree Generation

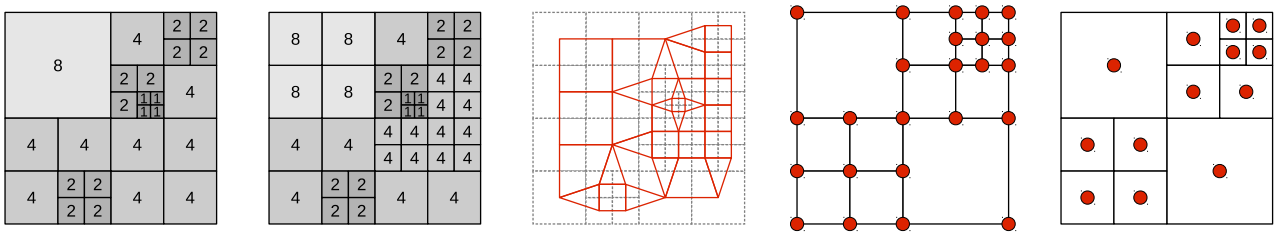
We represent the scene with a linear octree implementation analogous to [7]. To process the input data in parallel we use the concurrent hash map implementation of Li *et al*. [13] to store the location keys for the octree nodes.

Our goals for building an octree representation of the scene are twofold. First, the octree should adapt to the scene structure to reduce the required memory. Second, the octree should allow a fast discretization of the energy functional (9). To achieve this second goal we restrict the octree to be balanced, i.e., the depth difference of adjacent leaf nodes must be at most one depth level. This permits us to build a tetrahedral mesh with lookup tables and guarantees that there are no degenerated tetrahedra.

We start with building an unconstrained octree based on an estimate of the distribution of the point density in space and scale. With this estimate we decide where and at which scale to create the leaf nodes. The input scale of the points and the scene bounding box allow us to assign each point to an octree level. With the edge length L of the cubic bounding box, the edge length of an octree voxel at depth d is $l = L/2^d$. We assign a point with input scale σ to the highest octree level $d \in \{0, .., d_{\text{max}}\}$ with $2\sigma \leq L/2^d$. Together with the position of the point we compute the location key for each input point. The location key describes the depth and position of the voxel in the octree containing the point sample. Sorting the points with respect to the location keys yields a linear octree that only stores nodes containing points. For each node we accumulate the density contribution of each point. We compute the point density for a voxel with edge length l as

$$\rho = \sum_{i \in P} \frac{\sigma_i^3}{l^3}, \tag{1}$$

where P is the set of points with the same location key as the voxel. The assignment of a point to a compatible octree



(a) (b) (c) (d) (e)

Figure 2. **(a)** Quadtree before balancing. Each node stores the value of the scale function s. **(b)** Quadtree after balancing. The difference of the quadtree level of adjacent nodes is limited to 1 by recursive splitting of nodes. Splitted nodes keep the original values of the scale function. **(c)** Balanced quadtree (dashed) and its corresponding dual structure (solid red). Each node center becomes a vertex of the dual structure. In contrast to octrees, the polygonal cells of the dual structure of a quadtree can be converted to a triangle mesh by just splitting cells with 4 vertices into 2 triangles. **(d)** Primal sampling of a quadtree. Function values are stored in the vertices of the quadtree cells. **(e)** Dual sampling of a quadtree. Function values are stored in the center of each cell. The dual sampling results in a lower number of samples therefore reduces the memory and computational complexity for minimizing the energy functional (9)

level based on the scale limits the maximum contribution to the point density to $1/8$. This discards points with a scale too large to describe the surface at the specific octree level. On the other hand, we want to keep high resolution point samples at a smaller scale. To include these points in the density estimate, we recursively add the (averaged) density of child nodes to their parents. This procedure creates all missing parent nodes up to the root node.

The sparsity of the octree is improved by removing nodes that fall below a user defined density threshold. Nodes that are ancestors of at least one node that passes the threshold are kept to preserve a valid tree structure.

3.1. Octree Balancing

We balance the octree by splitting nodes recursively until all leaf nodes satisfy the balancing criterion. Alternatively, leaf nodes could be merged, but this can cause artifacts in the reconstruction.

To preserve the resolution information of the unconstrained octree we define a scale function s over the octree. s stores the scale of the reconstruction at each node and is initialized with the edge length of the corresponding octree voxel. We use s to define a scale aware energy model.

Splitting a leaf node creates the 8 child nodes and turns the former leaf into an inner node. We assign the scale value of the split node to the new leaf nodes. Passing on the scale value prevents an artificial increase in the reconstruction's resolution. We also eliminate mixed nodes by adding missing children. Fig. 2 shows an example of a quadtree before (a) and after balancing (b).

3.2. Point Aggregation

For each node in the balanced octree, we aggregate the data of the input point cloud affecting this node. This yields aggregated signed distance functions f_n and aggregated orientations $\mathbf{g}_m$ that are used in the cost function in Section 4. In Fuhrmann and Goesele [5], this aggregation is the only form of regularization. Since we globally optimize a cost function, in our case the aggregation mainly serves the efficient representation of the raw input data.

We treat each point sample as a signed distance function with a compact window function W. The support radius of the weighting function W is the point's scale σ_i; see also Fig. 3. Moreover, we assign to each voxel a soft window R with a compact support radius h proportional to the scale of the voxel. This relates the influence computation to the scale of the voxel and avoids missing points with a small scale σ_i. For most of our experiments we use $h = 3s(\mathbf{c})$ with $\mathbf{c}$ as the voxel center. The signed distance value at the voxel center for point i is

$$f_i(\mathbf{c}) = \frac{1}{w_i} \int R_h(\mathbf{x}-\mathbf{c}) W_{\sigma_i}(\mathbf{x}-\mathbf{p}_i) \langle \mathbf{n}_i, \mathbf{c}-\mathbf{x} \rangle \, \mathrm{d}\mathbf{x}, \quad (2)$$

where $\langle \cdot, \cdot \rangle$ denotes the dot product, $\mathbf{n}_i$ is the measured surface normal of the point, and $\mathbf{p}_i$ is the position of the point. The associated weight is

$$w_i = \int R_h(\mathbf{x}-\mathbf{c}) W_{\sigma_i}(\mathbf{x}-\mathbf{p}_i) \, \mathrm{d}\mathbf{x}. \quad (3)$$

Gaussian windows R and W would not have compact support, and considering all points for all voxels would be prohibitively slow. Besides, evaluating the integrals of (2) and (3) can become computationally expensive even in case of a closed form solution. To speed up computation, we approximate f_i and w_i using the window function proposed in [14]:

$$R_h(\mathbf{r}) = \begin{cases} \frac{315}{64\pi h^9}(h^2 - \|\mathbf{r}\|^2)^3 & \text{if } \|\mathbf{r}\| \le h \\ 0 & \text{else.} \end{cases} \quad (4)$$

It is fast to evaluate and frequently used in the smoothed particle hydrodynamics literature.

We define the weighting function W for the point sample as

$$W_{\sigma_i}(\mathbf{r}) = \begin{cases} 1 & \text{if } \|\mathbf{r}\| \leq \sigma_i \\ 0 & \text{else.} \end{cases} \tag{5}$$

With R and W as above

$$w_i(\mathbf{c}) \approx \frac{4}{3}\pi\sigma_i^3 R_h(\mathbf{p}_i - \mathbf{c}) \text{ , and} \tag{6}$$

$$f_i(\mathbf{c}) \approx \frac{1}{w_i}\frac{4}{3}\pi\sigma_i^3 R_h(\mathbf{p}_i - \mathbf{c}) \langle \mathbf{n}_i, \mathbf{c} - \mathbf{p}_i \rangle = \langle \mathbf{n}_i, \mathbf{c} - \mathbf{p}_i \rangle . \tag{7}$$

Since the orientation does not depend on the distance to the point, the orientation $\mathbf{g}_i$ induced by point i simply reads

$$\mathbf{g}_i(\mathbf{c}) = \mathbf{n}_i. \tag{8}$$

Due to the compact support of R, w_i is zero for point samples outside the radius h and the points can be ignored. We reuse the spatial sorting of the input points with respect to the location keys during density computation to accelerate the radial search for candidates.

Again we discard points with a too large scale $2\sigma_i > s(\mathbf{c})$ but consider high resolution points with low scale values. In contrast to [6], where sample contribution only depends on a window centered at the points, we can aggregate the data also reliably for voxels at coarser levels.

Efficient Storage Instead of storing values for each point inside the octree nodes, we use histograms and k-means clustering to store a fixed number of f_n, w_n and $\mathbf{g}_m$, w_m for each voxel. We store the signed distance values f_i and the corresponding weights w_i for each point in the histogram f_n, w_n with 8 bins. We use soft binning to reduce quantization effects. The minimum and maximum values of the bin levels f_n are bound individually for each voxel by $\pm h$. We reduce the number of normal hypotheses to 10 using an online k-means clustering. We start with evenly distributed cluster centers for 20 orientations. Each time we add a normal we update the cluster centers $\mathbf{g}_m$ and the weights w_m. After adding all points for a voxel we pick the 10 clusters with the largest weights. To further decrease the size in memory, we quantize the normal direction of the selected clusters with 8 bits for inclination and azimuth. We store all weights of the histograms and the normal clusters in 16 bit half-precision floating-point format. The total memory footprint of the data term is 64 byte per voxel including fields for averaging color information.

4. Energy Model

The final reconstruction is obtained by minimizing the energy

$$E(u, \mathbf{v}) = \lambda_1 E_{\text{data}_u} + \lambda_2 E_{\text{data}_\mathbf{v}} + \alpha_1 E_{\text{coupling}} + \alpha_2 E_{\text{smooth}} \tag{9}$$

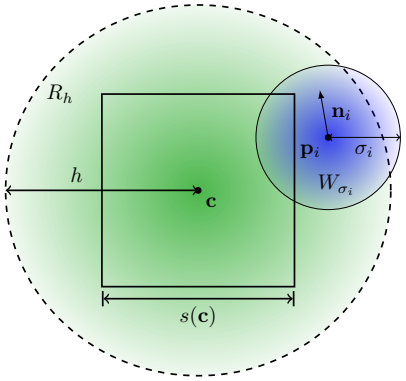


Figure 3. Data aggregation for a voxel centered at $\mathbf{c}$. The aggregation window R_h is depicted by the outer circle shaded in green. Its support h depends on the value of the scale function at the voxel center $s(\mathbf{c})$. The weight function W_{σ_i} of a point sample at position $\mathbf{p}_i$ is shown as the smaller circle in blue. Its support depends on the point's input scale σ_i.

over the signed distance function $u(\mathbf{x})$ and the normal vector field $\mathbf{v}(\mathbf{x})$, where $\mathbf{x}$ is a coordinate in $\mathbb{R}^3$. In the following, we drop $\mathbf{x}$ in the notation. The factors $\lambda_{1,2}$ and $\alpha_{1,2}$ steer the relative importance of the terms, which are defined as

$$E_{\text{data}_u}(u) = \int \frac{1}{s} \sum_n w_n |u - f_n| \, \mathrm{d}\mathbf{x} \tag{10}$$

$$E_{\text{data}_\mathbf{v}}(\mathbf{v}) = \int \sum_m w_m \|\mathbf{v} - \mathbf{g}_m\| \, \mathrm{d}\mathbf{x} \tag{11}$$

$$E_{\text{coupling}}(u, \mathbf{v}) = \int \|\nabla u - \mathbf{v}\|^2 \, \mathrm{d}\mathbf{x} \tag{12}$$

$$E_{\text{smooth}}(\mathbf{v}) = \int s \|\mathbf{J}_\mathbf{v}\| \, \mathrm{d}\mathbf{x}. \tag{13}$$

The norm $\|\cdot\|$ denotes the Frobenius norm. It is not squared, i.e., measurements can be ignored if the majority of data points contradict them. This makes the energy robust to erroneous points in the input data.

The term E_{coupling} couples the functions u and $\mathbf{v}$. The squared term ensures that $\mathbf{v}$ stays close to the gradient of the signed distance function and vice versa.

Finally, the smoothness term E_{smooth} adds a regularization to the vector field $\mathbf{v}$ by penalizing its Jacobian $\mathbf{J}_\mathbf{v}$. The norm is non-quadratic, i.e., E_{smooth} favours piecewise constant vector fields corresponding to planar surfaces. This preserves discontinuities in the vector field, for instance, at object edges or corners.

To make the energy functional aware of the reconstruction scale we introduce the scale function s in (10) and (13). s defines the scale of the reconstruction at each position. Low values result in a reconstruction with a high spatial resolution while high values correspond to a coarse reconstruction. The scale function relates the signed distance values

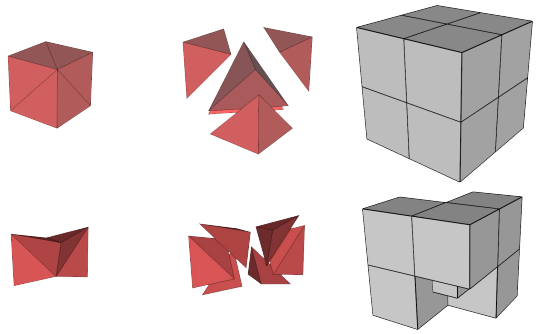


Figure 4. Two of the 27 possible cell configurations after rotation normalization. **Top** Triangulation of a cubic cell with 5 tetrahedra (left), exploded view (center), corresponding octree (right). **Bottom** Triangulation of a nonconvex cell with 6 tetrahedra (left), exploded view (center), corresponding octree (right).

of u and f_n in the energy E_{data_u} to the reconstruction scale. Without knowing the reconstruction scale we cannot tell if a deviation of one meter between u and f_n is significant or not. We can also see s as a spatially varying weighting parameter, giving more weight to the data term in regions with higher resolution. In E_{smooth} we increase the smoothing strength proportional to the scale to obtain a coarser reconstruction and decrease it to obtain a reconstruction with fine details. The functions $\mathbf{v}$, $\mathbf{g}_m$ and ∇u describe directions and therefore are scale independent. While the vectors in $\mathbf{g}_m$ are forced to unit length, we do not enforce this for $\mathbf{v}$ and ∇u to keep the functional convex.

5. Problem Discretization

To find the minimizer of (9), we discretize the functional using a finite element and finite volume discretization. We create a discrete problem with a finite dimensional search space based on the dual sampling of the octree, as shown in Fig. 2(e). Dual sampling describes the domain by sampling functions at the center of each octree node. This leads to a smaller number of degrees of freedom and reduces complexity.

We use the following approximations for the functions u and $\mathbf{v}$ in the coupling term (12) and the smoothness term (13)

$$\begin{aligned} u(\mathbf{x}) &\approx \tilde{u}(\mathbf{x}) = \textstyle\sum_k^N U_k \phi_k(\mathbf{x}) \\ \mathbf{v}(\mathbf{x}) &\approx \tilde{\mathbf{v}}(\mathbf{x}) = \textstyle\sum_k^N \mathbf{V}_k \phi_k(\mathbf{x}) \end{aligned}, \tag{14}$$

where N is the number of nodes and U_k, $\mathbf{V}_k$ are the discrete degrees of freedom of the respective approximations. The global shape functions Φ_k define the interpolation of the approximate functions $\tilde{u}$ and $\tilde{\mathbf{v}}$. Each shape function is 1 at its own node and 0 at all other nodes. We describe each global shape function as a composition of local shape functions defined on the tetrahedral elements. The local shape functions defined on the tetrahedra are the linear barycentric coordinate functions.

For the data terms (10) and (11) we use the approximations

$$\begin{aligned} u(\mathbf{x}) &\approx \hat{u}(\mathbf{x}) = \textstyle\sum_k^N U_k I_k(\mathbf{x}) \\ \mathbf{v}(\mathbf{x}) &\approx \hat{\mathbf{v}}(\mathbf{x}) = \textstyle\sum_k^N \mathbf{V}_k I_k(\mathbf{x}) \end{aligned} \tag{15}$$

with the indicator functions I_k as shape functions. I_k is 1 inside the cubic voxel and 0 otherwise. Setting up the linearized system requires us to compute the volume integrals over the shape functions. For the shape functions ϕ_k we must compute the integrals over the tetrahedra, while the volume integral for I_k is simply the volume of the voxel.

5.1. Tetrahedral Mesh Generation

Before triangulation (the generation of the tetrahedral mesh) we compute the dual octree. We use the parallel algorithm presented by Lewiner *et al*. [12] to create the cells of the dual octree. Fig. 2(c) shows the dual of a quadtree. Creating the dual swaps the roles of vertices and cells of the octree. Each vertex in the primal octree becomes a cell in the dual, and each vertex in the dual becomes a cell in the primal; see Fig. 2(d,e) for the positions of primal and dual vertices of a quadtree.

We can generate a triangulation by decomposing the cells of the dual octree. In the 2D quadtree example, as shown in Fig. 2(c), we can create a triangulation by splitting cells with four vertices into two triangles.

For octrees, the decomposition of the polyhedral cells into tetrahedrons is more involved. This is due to the nonplanar faces with four vertices of some dual cells. Choosing a triangulated surface for a nonplanar face makes at least one of the adjacent cells nonconvex. Since the triangulation of nonconvex polyhedra is a NP-complete problem [16], we use a lookup table approach and precompute the triangulation for all possible cell configurations.

We normalize the possible configurations for rotations to keep the lookup table down to 27 entries. Fig. 4 shows the triangulation for two of the 27 configurations of the polyhedral cells. The range of the created tetrahedra per configuration is 2-6. To compute the triangulations, we have used a naive algorithm as initialization. For the failure cases we have manually generated the tetrahedral decomposition. We encode the configurations in a compact 32-bit key. The key uses quantized edge lengths of the cell and the octree level of the vertices.

Our lookup table guarantees a tetrahedral mesh without holes and without intersecting tetrahedra under the assumption that the octree is balanced as described in Section 3.1.

6. Energy Minimization

Two difficulties arise with the minimization of energy (9). First, the energy functional consists of nondifferentiable functions and nonlinear terms. Second, the

problem size requires a minimization algorithm that makes best use of the available memory and computing resources.

We first address differentiability by regularizing the non-squared data terms (10), (11), and the smoothness term (13). For the data term E_{data_u} we replace the absolute value in (10) with its regularized version $|a|_\delta = \sqrt{a^2 + \delta^2}$. The function $|a|_\delta$ is differentiable everywhere and has similar properties as the Huber norm. In case of the data term (10), we set the parameter δ individually to the histogram bin widths of the voxels. This avoids quantization artifacts in the reconstruction. Analogously, we replace the Frobenius norm with a modified version. We set $\delta = 10^{-3}$ for (11) and (13).

To deal with nonlinearity we use an iterative reweighted least squares approach. Within each iteration we solve a linearized least squares problem using a parallel 8-color Gauss-Seidel scheme, which is an in-place method and, thus, has a small memory footprint. To speed up convergence we employ a coarse-to-fine scheme on top. Transitions from a coarser grid to a finer grid use the shape functions ϕ_k for interpolation.

7. Surface Extraction

Once the signed distance function u is computed we can extract the surface as the zero level set. We use the dual contouring algorithm proposed by Ju *et al*. [9]. The algorithm can be applied to adaptive grids and additional information can be used to improve the vertex placement. We use the additional information about the surface orientation from the vector function $\mathbf{v}$ to compute vertex positions.

We compute the improved vertex position $\mathbf{q}$ as the minimizer of the quadratic error function

$$\mathbf{q} = \arg\min_{\mathbf{x}} \left(\frac{1}{N} \sum_{i}^{N} \langle \mathbf{n}_i, \mathbf{x} - \mathbf{p}_i \rangle^2 + \|\mathbf{x} - \mathbf{m}\|^2 \right). \tag{16}$$

For each edge intersecting the surface, we add two plane constraints with the intersection point $\mathbf{p}_i$ and the normals of the adjacent edge vertices $\mathbf{n}_i$. The normals $\mathbf{n}_i$ equal the degrees of freedom $\mathbf{V}_k$ of the respective dual vertex.

The point $\mathbf{m}$ is the center of gravity of the edge intersections $\mathbf{p}_i$. In degenerate cases, where the normals $\mathbf{n}_i$ are very similar, $\mathbf{m}$ stabilizes the solution and avoids vertex positions far away from the cell. Since $\mathbf{m}$ does not depend on the normals, it can be used for algorithms without normal information. Fig. 5 compares the positions of $\mathbf{q}$ and $\mathbf{m}$.

8. Results

We present results on synthetic and real data sets. For the real data sets we use the Multi-View-Environment [19] and [8] to create point clouds with scale information. Camera parameters have been computed with [21], [20].

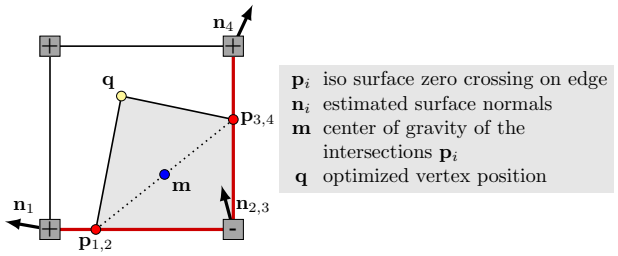


Figure 5. Computation of the vertex position. The point $\mathbf{q}$ minimizes the distances to the planes defined by $(\mathbf{n}_i, \mathbf{p}_i)$ and to the point $\mathbf{m}$. Each edge intersecting the surface adds two plane equations to the quadratic error function (16).

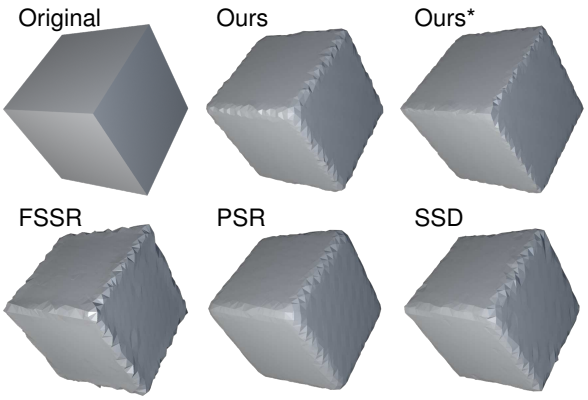


Figure 6. Reconstruction of a synthetic cube. **Original:** Original cube. **Ours:** Our reconstruction using the center of mass $\mathbf{m}$ to place vertices. **Ours*:** Our reconstruction solving the QEF (16) to compute vertex positions. **FSSR:** Floating Scale Surface Reconstruction using marching cubes as in [11]. **PSR:** Poisson surface reconstruction also using [11]. **SSD:** Smoothed Signed Distance reconstruction using dual marching cubes [17]. The maximum octree depth was set to 5 for PSR, SSD and our method.

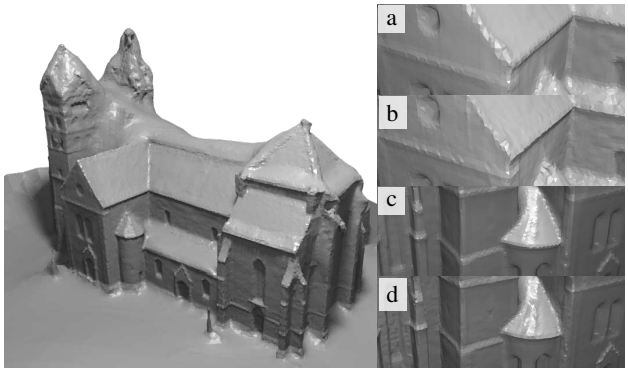


Figure 7. Effect of vertex positions being computed by solving (16) on a real data set. **Left:** Reconstruction with limited octree depth of 9 using dual contouring with improved vertex positions. **Right:** Close-up views comparing the reconstruction of edges using the center of mass (a),(c) and the improved vertex positions using normal information (b),(d). Using normal information to compute vertex positions leads to a more faithful reconstruction of edges and improves visual quality.

We compare to common state of the art methods like Poisson Surface Reconstruction (PSR) [10], Smoothed Signed Distance Reconstuction (SSD) [1] and Floating Scale Surface Reconstruction (FSSR) [6]. We show that our method achieves state of the art performance on single scale as well as multiscale data sets. In addition, our method compares favourably on data sets with many erroneous points, thus making our method applicable to a wider range of reconstruction problems.

Sharp Features We compare surface meshes generated with vertices placed at the average of the intersections $\mathbf{m}$ and the improved position $\mathbf{q}$ as shown in Fig. 5. We also compare with the marching cubes implementation of Kazhdan *et al*. [11], which is used by the PSR and FSSR algorithm, and with the dual marching cubes algorithm by Schaefer and Warren [17] used by SSD. We show reconstructions of a synthetic cube in Fig. 6 and a real scene in Fig. 7. Our reconstruction with improved vertex positions gives the best reconstruction. PSR and SSD yield a cube with rounded edges and corners. FSSR exaggerates the edges, which is an artifact of the large radius of the basis functions. Our reconstruction using the point $\mathbf{m}$ for positioning the vertices gives a similar result as SSD.

Multiscale Data Sets Fig. 8 shows a comparison of reconstructions on the citywall data set provided with the MVE tool [19]. Our method generates a dense, high detail surface for all scales.

Following [6], we show the behaviour of SSD, FSSR and our method for different point densities in Fig. 10. FSSR and our method correctly handle the regions with high density to reduce noise, while SSD adapts to the noise. The experiment shows also the importance of regularization in multiscale data sets. Our smoothness and data terms complement each other giving a more uniform reconstruction.

Next we demonstrate that our method can be applied to large scenes. The Breisach data set shown in Fig. 9 contains 1.5 billion points and models an area of about 10000m^2. Reconstruction with our method took about 4 days on a machine with 24 cores (Intel X7460 CPU @ 2.66GHz, 2008) with a peak memory consumption of 151.8 GB. We were able to reconstruct the scene with FSSR in only 3 days with a memory peak of 164.5 GB on the same machine. Remember that our method performs a global optimization on the whole octree with robust data terms. We could not reconstruct the scene with PSR and SSD due to limited memory.

Temple Data Set We show that our method is competitive with respect to accuracy on the standard benchmark [18]. To give a fair comparison, all methods have been evaluated on the *Temple full* data set using the same input point samples. The results in Table 1 show that our approach achieves a similar performance as the tested state of the art methods.

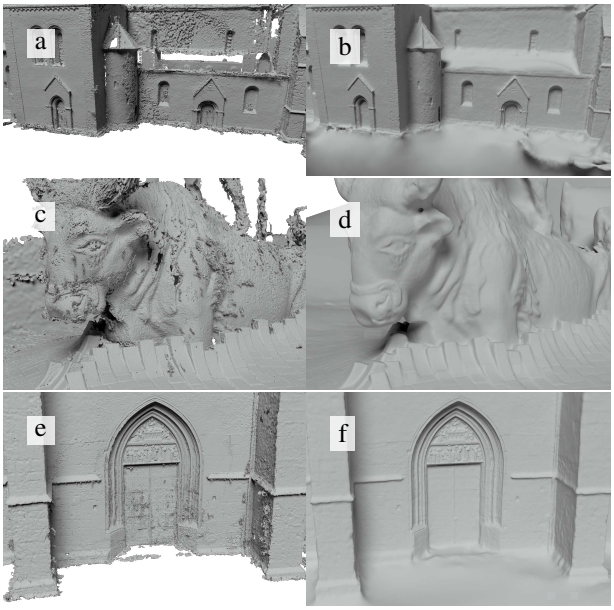


Figure 9. Reconstruction of the Breisach data set with 1.5 billion points. The left column shows the reconstruction with FSSR, the right column shows our method. We have thresholded the FSSR result with the provided clean-up tool to remove clutter from the scene. A higher threshold starts to dissolve the reconstructed surfaces. Our method is able to reasonably fill holes in the reconstruction like the ground in (f) or the roof in (b). In regions without data the regularization can extend surfaces in a wrong way like the walls as seen in (b) at the bottom. The artifacts in (a,c,e) partially stem from misaligned depth samples, which are problematic for local methods like FSSR. Our method is more robust to such small misalignments due to the regularization.

	Accuracy		
	90% Thr.	97% Thr.	99% Thr.
PSR	0.36	0.56	0.84
SSD	0.38	0.56	0.75
FSSR	0.40	0.63	0.84
Ours	0.42	0.61	0.78

Table 1. Accuracy on the MVS Middlebury *Temple Full* data set.

Noise Robustness We demonstrate the robustness to noise by reconstructing a car in Fig. 11. The point cloud contains many erroneous measurements and a high noise ratio due to reflections and transparent surfaces. The robust terms in our energy model suppress noise and preserve details better than the other methods.

9. Conclusion

We have presented a global method for surface reconstruction from point cloud data with scale information that

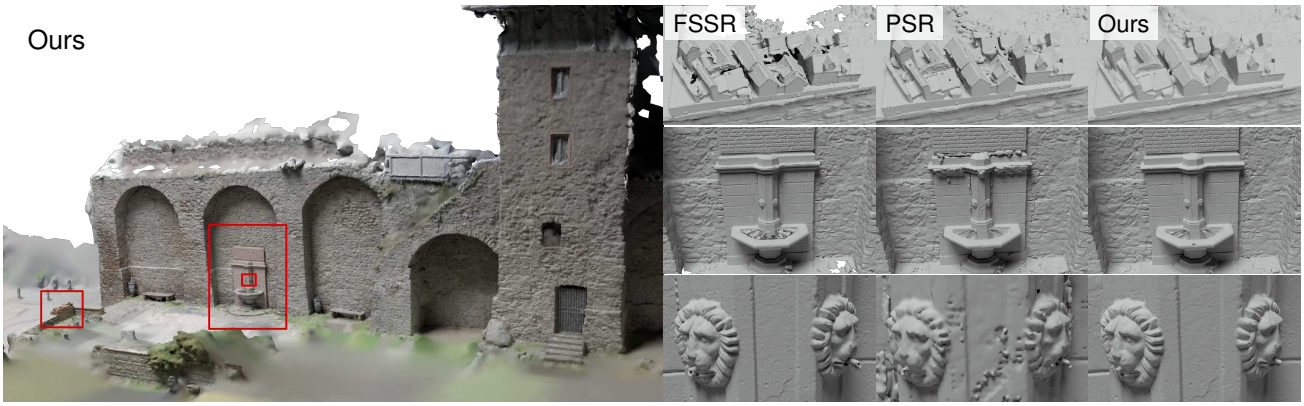


Figure 8. Reconstruction of the Citywall data set from [19] with 256 million points. **Left:** Colored reconstruction of the whole scene with our method. The annotations mark the three close-up views which are (top to bottom) the city model, the fountain and the lion heads. **FSSR:** FSSR generates a detailed reconstruction of the scene but has problems with holes in the city model and shows artifacts in the basin of the fountain. The artifacts are linked with the rest of the scene and cannot be removed easily. Of all methods, FSSR shows the best reconstruction near the mouths of the lion heads. **PSR:** The reconstruction with PSR contains many noise artifacts. Some artifacts could be removed with the provided clean-up tool in exchange for causing holes in the reconstruction as seen in the city model. The reconstruction of the lion heads is too smooth due to a maximum octree depth of 14. Memory did not permit a reconstruction using a deeper octree. **Ours:** Our method generates a surface without holes for the city model and preserves most of the concave structure between the houses. The basin of the fountain has some small artifacts. In contrast to FSSR, the view to the bottom of the basin is not blocked by reconstruction artifacts. The lion heads show all details but the surface of the taps of the lions is slightly underestimated. Results for SSD are not shown because the method did not finish after multiple days.

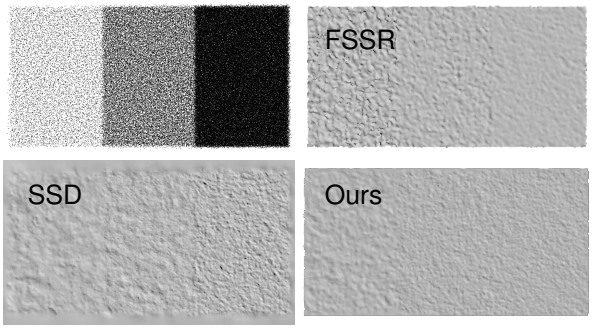


Figure 10. Reconstruction of a plane with three regions, each with a different uniform point density. The scale value assigned to the points corresponds to the sampling density of the central region. Gaussian white noise was added to the points' position and normal. **Top left:** Input point cloud. **Top right:** With increasing density, FSSR effectively cancels out noise. In the low density region it suffers from a too sparse sampling. **Bottom left:** SSD adapts the scale to the point density and therefore models the noise in the high density region. In the low density region noise is suppressed by using a coarser scale for reconstruction. **Bottom right:** Our reconstruction looks more even in the high density region than that of the other methods. The high density leads to a strong data term but is also effective to cancel out noise. In the low density region the smoothness term dominates and the reconstruction looks smoother than with SSD.

is efficient enough to handle 1.5 billion points. The method is robust to noise and can fill-in missing data in the reconstruction. It combines good properties of previous methods and is, thus, applicable to a wider range of scenes. We will make the software publicly available.

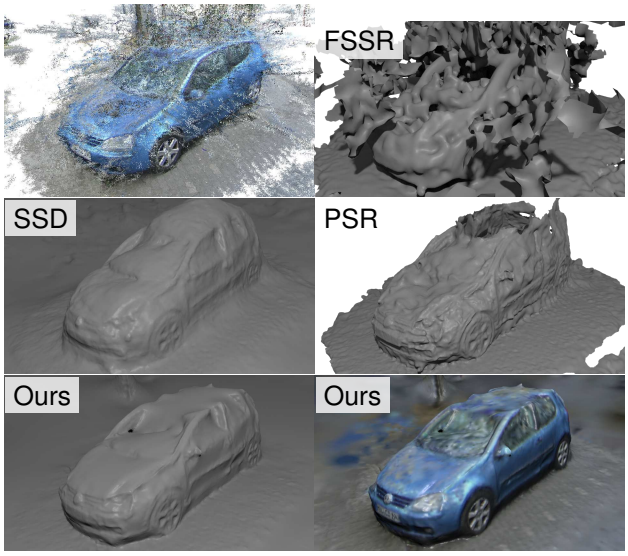


Figure 11. **Top left:** Input point cloud. **FSSR:** The FSSR method shows strong artifacts. We used a scale parameter of 8 and applied the provided clean-up tool to threshold the mesh. Choosing a higher threshold dissolves the car. **SSD:** The SSD method loses details such as the side mirror. The roof of the car is too high and the object boundary to the floor is blurred. **PSR:** The surface generated with PSR has noise artifacts. The provided clean-up tool removes the roof of the car and the background. Smaller thresholds introduce wrong geometry on top of the car and clutter. **Ours:** Our reconstruction generates a mesh with smooth surfaces and details such as the side mirror. The boundary between car and ground exhibits an edge in the mesh.

References

[1] F. Calakli and G. Taubin. SSD: Smooth Signed Distance Surface Reconstruction. *Computer Graphics Forum*, 30(7):1993–2002, 2011. 2, 7

[2] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 303–312, New York, NY, USA, 1996. ACM. 2

[3] V. Estellers, M. Scott, K. Tew, and S. Soatto. Robust Poisson Surface Reconstruction. In J.-F. Aujol, M. Nikolova, and N. Papadakis, editors, *Scale Space and Variational Methods in Computer Vision*, number 9087 in Lecture Notes in Computer Science, pages 525–537. Springer International Publishing, May 2015. 2

[4] J.-M. Frahm, P. Fite-Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, and M. Pollefeys. Building Rome on a Cloudless Day. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *Computer Vision ECCV 2010*, number 6314 in Lecture Notes in Computer Science, pages 368–381. Springer Berlin Heidelberg, 2010. 1

[5] S. Fuhrmann and M. Goesele. Fusion of depth maps with multiple scales. In *Proceedings of the 2011 SIGGRAPH Asia Conference*, SA '11, pages 148:1–148:8, New York, NY, USA, 2011. ACM. 2, 3

[6] S. Fuhrmann and M. Goesele. Floating Scale Surface Reconstruction. *ACM Trans. Graph.*, 33(4):46:1–46:11, July 2014. 2, 4, 7

[7] I. Gargantini. Linear octtrees for fast processing of three-dimensional objects. *Computer Graphics and Image Processing*, 20(4):365–374, Dec. 1982. 2

[8] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. Seitz. Multi-View Stereo for Community Photo Collections. In *IEEE 11th International Conference on Computer Vision, 2007. ICCV 2007*, pages 1–8, 2007. 2, 6

[9] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual Contouring of Hermite Data. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, pages 339–346, New York, NY, USA, 2002. ACM. 6

[10] M. Kazhdan and H. Hoppe. Screened Poisson Surface Reconstruction. *ACM Trans. Graph.*, 32(3):29:1–29:13, July 2013. 2, 7

[11] M. Kazhdan, A. Klein, K. Dalal, and H. Hoppe. Unconstrained Isosurface Extraction on Arbitrary Octrees. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, SGP '07, pages 125–133, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association. 6, 7

[12] T. Lewiner, V. Mello, A. Peixoto, S. Pesco, and H. Lopes. Fast Generation of Pointerless Octree Duals. *Computer Graphics Forum*, 29(5):1661–1669, July 2010. 5

[13] X. Li, D. G. Andersen, M. Kaminsky, and M. J. Freedman. Algorithmic Improvements for Fast Concurrent Cuckoo Hashing. In *Proceedings of the Ninth European Conference on Computer Systems*, EuroSys '14, pages 27:1–27:14, New York, NY, USA, 2014. ACM. 2

[14] M. Müller, D. Charypar, and M. Gross. Particle-based Fluid Simulation for Interactive Applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '03, pages 154–159, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association. 3

[15] T. Pock, L. Zebedin, and H. Bischof. TGV-Fusion. In C. S. Calude, G. Rozenberg, and A. Salomaa, editors, *Rainbow of Computer Science*, number 6570 in Lecture Notes in Computer Science, pages 245–258. Springer Berlin Heidelberg, Jan. 2011. 2

[16] J. Ruppert and R. Seidel. On the difficulty of triangulating three-dimensional Nonconvex Polyhedra. *Discrete & Computational Geometry*, 7(1):227–253, Dec. 1992. 5

[17] S. Schaefer and J. Warren. Dual marching cubes: primal contouring of dual grids. In *12th Pacific Conference on Computer Graphics and Applications, 2004. PG 2004. Proceedings*, pages 70–76, 2004. 6, 7

[18] S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 519–528, 2006. 7

[19] F. L. Simon Fuhrmann and M. Goesele. MVE - A Multi-View Reconstruction Environment. In *Proceedings of the Eurographics Workshop on Graphics and Cultural Heritage (GCH)*, 2014. 6, 7, 8

[20] C. Wu. Towards Linear-Time Incremental Structure from Motion. In *2013 International Conference on 3D Vision - 3DV 2013*, pages 127–134, June 2013. 1, 6

[21] C. Wu, S. Agarwal, B. Curless, and S. Seitz. Multicore bundle adjustment. In *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3057–3064, June 2011. 1, 6

[22] C. Zach. Fast and high quality fusion of depth maps. In *Proceedings of the International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*, volume 1, 2008. 2