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 nonuniform 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 with a lookup table which allows to map octree cells to the nodes of the finite elements. We optimize memory efficiency by data aggregation, 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.

1. Introduction

Current structure from motion pipelines are able to create sparse reconstructions of large scenes with thousands of images. Even city scale reconstructions have become feasible (Frahm et al, 2010). Such large scenes come along with several challenges for dense reconstruction. These include (i) an efficient scene representation, both in terms of memory and computational costs, (ii) reconstruction of surfaces observed at different levels of detail, and (iii) 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.

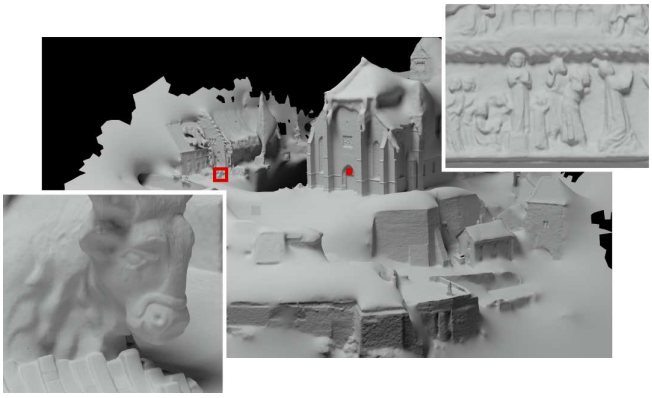

Figure 1. Reconstruction of a scene with large differences in scale. The top right and bottom left corner show a close-up view of the respectively marked spots.

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. The octree allows us to adaptively store information at different spatial resolutions. Our aggregation scheme ensures that the memory requirements only depend on the scene structure, hence the octree structure, and stays independent of the number of points. We propose a fast finite element discretization of the octree into a tetrahedral mesh. Our discretization is based on the dual structure of the balanced octree. It does not introduce additional nodes, thus allowing us to state the optimization problem with a minimum number of variables with respect to the balanced 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 functions for the surface position and orientation and jointly optimize them.

Fig. 2 gives a coarse overview of our method. 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 (Wu et al, 2011; Wu, 2013) to compute camera parameters and (Goesele et al, 2007) 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 as in (Fuhrmann and Goesele, 2011).

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 (2013) and Calakli and Taubin (2011) 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 (Kazhdan and Hoppe, 2013) and (Calakli and Taubin, 2011) in Section 8. Recently, Estellers et al (2015) proposed a modification of (Kazhdan and Hoppe, 2013) 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 (2011) 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 (1996). 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 (2014) 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.

Sagawa et al (2005) present a robust method for merging range images with an octree structure. Subdivision of the octree follows a similar scheme as in the dual marching cubes algorithm (Schaefer and Warren, 2004), which is steered by the geometric complexity rather than density.

Bolitho et al (2007) and Manson et al (2008) have proposed out of core algorithms for the reconstruction of large data sets. Both methods use a sliding window approach to make the memroy requirements independent of the input data. Due to the window approach optimization is limited to the active window or requires multiple passes over the volume.

Labatut et al (2009) treat surface reconstruction as a tetrahedra labelling problem. They compute the Delaunay triangulation of the input point cloud and use the graph cut framework to label the tetrahedra either as inside or outside. Faces connecting inside and outside tetrahedra describe the surface triangle mesh. The reconstructed surfaces show discretization artifacts due to the fixed Delaunay triangulation and the binary labelling. Hiep et al (2009) counter this problem by increasing the number of points and adding a variational refinement step, which makes the algorithm more involved. Bailer et al (2012) make use of the scale information to optimize the point cloud before the Delaunay triangulation step, by selecting point samples with the best resolution. Further they also use point samples of similar scales to optimize the positions of the selected points. Jancosek and Pajdla (2011) modify the edge weights in the graph to improve the reconstruction of surfaces that are only weakly supported by the input point cloud. This results in reconstructions that are more complete. We compare to them in Section 8.

Like (Jancosek and Pajdla, 2011), we also use a tetrahedral net as discretization of the volume. We use the finite element method (FEM) to discretize our en-

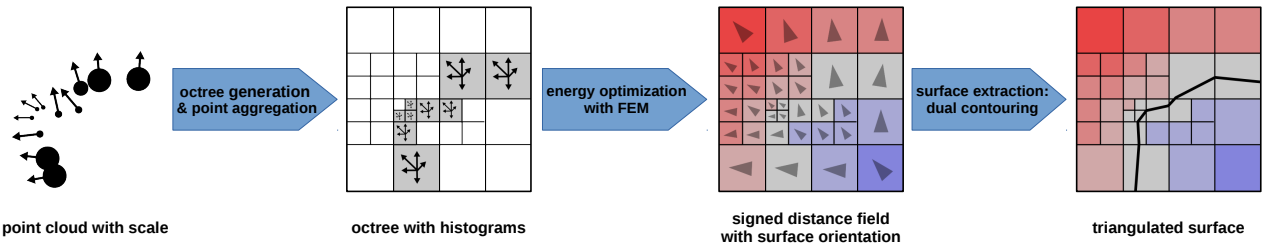

Figure 2. Method overview. The input data is a point cloud with scale and normal information. First, we generate an octree based on the scale information in the point cloud and aggregate the information into orientation and distance histograms. Second, the implicit representation is computed, which is a signed distance function and a vector field describing the surface orientation. Finally, we extract the surface making use of the signed distance function and the vector field.

ergy model. There exist several approaches for generating meshes suited for the FEM. There are advancing front algorithms (Blacker and Meyers, 1993), Delaunay triangulation based approaches (Shewchuk, 1998) or combined methods (Mavriplis, 1995). While these methods generate high quality elements, they add additional nodes which increases the number of variables. Further they do not take advantage of the underlying octree structure. Marchal (2009) creates a hexahedral mesh from the dual octree structure. Therefore, the generated mesh is related to the underlying octree. To avoid hanging nodes they also introduce additional nodes. We propose a fast lookup table approach, which generates a tetrahedral net for the dual octree and does not add additional nodes. This allows us to directly map octree cells to the nodes in the tetrahedral net. Similarly to (Marchal, 2009), we require the octree to be balanced.

Our regularization is strongly related to Pock et al (2011), 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 (2008) 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 (Kazhdan and Hoppe, 2013; Bolitho et al, 2007; Estellers et al, 2015; Manson et al, 2008; Calakli and Taubin, 2011; Fuhrmann and Goesele, 2014), 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.

A preliminary version of this work appeared in (Ummenhofer and Brox, 2015). This version features more experiments and an extended analysis of our method. The resulst section has been extended with a comparison with CMPMVS (Jancosek and Pajdla, 2011), which uses an unstructured tetrahedral grid. Additionally, we include a reconstruction experiment on the roman forum internet photo collection data set. We evaluate the extended marching cubes as an alternative to the dual contouring. We now give an analysis of the runtimes on the Breisach data set. Further we give more details on the parallelization.

3. Octree Generation

We represent the scene with a linear octree implementation analogous to (Gargantini, 1982). To process the input data in parallel we use the concurrent hash map implementation of Li et al (2014) 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), which is a non-trivial process for adaptive volumetric grids. 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. Leaf nodes are adjacent if they share a face, an edge or a corner. This permits us to build a tetrahedral mesh with lookup tables as described in Section 5.1.

We begin with building an unconstrained octree, which is 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. Similar to (Fuhrmann and Goesele, 2011),

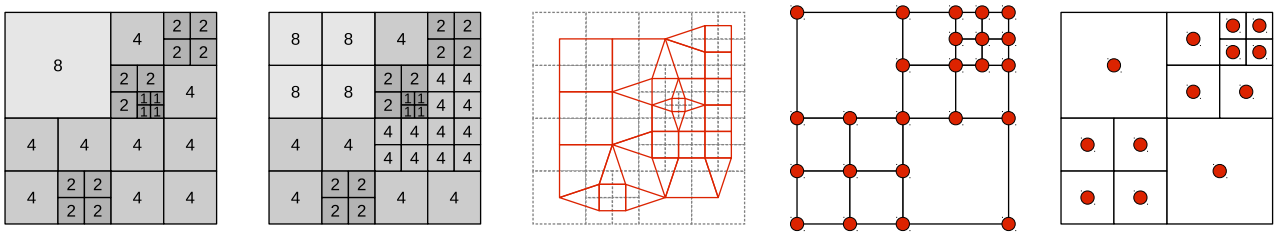

Figure 3. **(a)** Quadtree before balancing. Each node stores the value of the scale function s. **(b)** Quadtree after balancing. The difference of the quadtree level of adjacent nodes is limited to 1 by recursive splitting of nodes. Splitted nodes keep the original values of the scale function. **(c)** 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)

the input scale of the points and the scene bounding box allow us to assign each point to an octree level. The scale information is the spatial extent of a point and is defined by the distance to the camera center and the internal camera parameters. 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 $d \in \{0, .., d_{\max}\}$ 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 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. This is in contrast to (Fuhrmann and Goesele, 2011), which uses a coarse to fine pass that uses information at coarse scales to regularize voxels with low density at finer scales.

As a last step, we improve the sparsity of the octree 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.

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 add missing child nodes for inner nodes. Fig. 3 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, 2011), 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 window function W is the point's

scale σ_i; see also Fig. 4. 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 (Müller et al, 2003):

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

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

We define the window function W for the point sample as

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

With R and W as above

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

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

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

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

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 (Fuhrmann and Goesele, 2014), contribution only depends on a single window function centered at the points, thus points with a small scale may not contribute at all to voxels at coarse scale, because the voxel vertices lie outside the window. With the two window functions R and W, we can aggregate the data also reliably for voxels at coarser levels.

Efficient Storage Instead of storing the information of 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 20 evenly distributed cluster centers which represent 20 orientations. The distance to the cluster is defined by the angular difference between the represented orientation and the normal. 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

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

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,

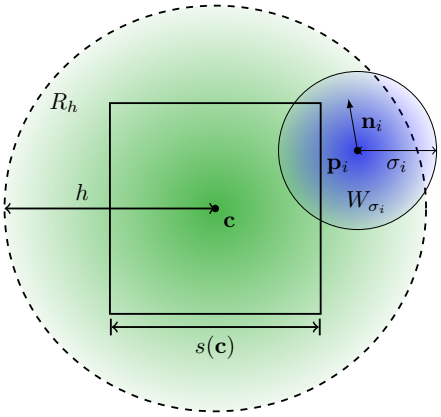

Figure 4. Data aggregation for a voxel centered at $\mathbf{c}$. The aggregation window R_h is depicted by the outer circle shaded in green. Its support h depends on the value of the scale function at the voxel center $s(\mathbf{c})$. The weight function W_{σ_i} of a point sample at position $\mathbf{p}_i$ is shown as the smaller circle in blue. Its support depends on the point's input scale σ_i.

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 nonquadratic, 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. This type of regularization has been used in (Pock et al, 2011). A quadratic version of this regularization was used in (Calakli and Taubin, 2011).

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 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.

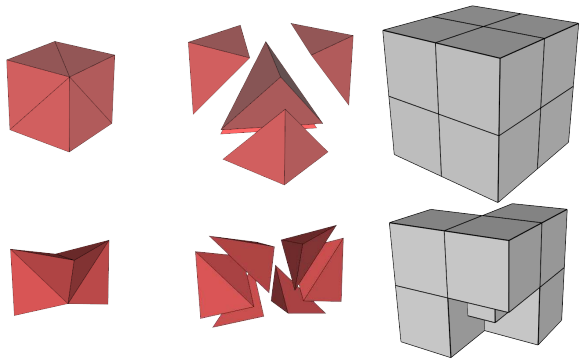

Figure 5. Two of the 27 possible cell configurations after rotation normalization. **Top** Triangulation of a cubic cell with 5 tetrahedra (left), exploded view (center), corresponding octree (right). **Bottom** Triangulation of a non-convex cell with 6 tetrahedra (left), exploded view (center), corresponding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

5. Problem Discretization

To find the minimizer of (9), we discretize the functional using a finite element and finite volume discretization. Finite element discretization has also been used in (Calakli and Taubin, 2011) for the octree voxels, which is a primal sampling as shown in Fig. 3(d). We create a discrete problem with a finite dimensional search space based on the dual sampling of the octree, as shown in Fig. 3(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 smooth-

ness term (13)

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

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

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

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 (2010) to create the cells of the dual octree. Fig. 3(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. 3(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. 3(c), we can create a triangulation by splitting cells with four vertices into two triangles.

For octrees, the decomposition of the polyhedral cells into tetrahedra 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 (Ruppert and Seidel, 1992), 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. 5 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 nonsquared 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 Gauss-Jacobi scheme. We iterate over the dual cells to process all tetrahedra. This results in better cache efficiency since the data of the cell vertices is used several times by the up to six tetrahedra per cell. To allow parallel iteration over the cells, we create a partitioning with a simple heuristic.We count the number of vertices that can be reached from a node in the octree and start with the root node as first partition. We then iteratively replace the node with the largest number of vertices with its children until the

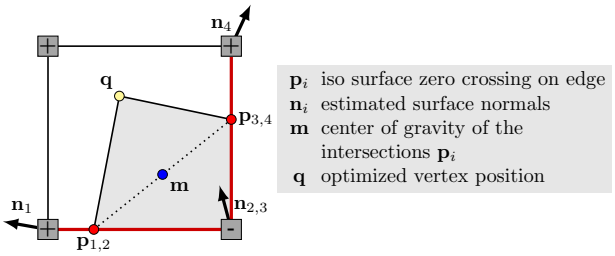

Figure 6. Computation of the vertex position. The point $\mathbf{q}$ minimizes the distances to the planes defined by $(\mathbf{n}_i, \mathbf{p}_i)$ and to the point $\mathbf{m}$. Each edge intersecting the surface adds two plane equations to the quadratic error function (16).

desired number of partitions is met.

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. Our lookup table approach allows us to update the grid in-place and avoids high memory peaks during grid transitions.

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 (2002). 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. 6 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 (Fuhrmann et al, 2014) and (Goesele et al, 2007) to create point clouds with scale information. Camera parameters have been computed with (Wu et al, 2011; Wu, 2013).

We compare to common state of the art methods like Poisson Surface Reconstruction (PSR) (Kazhdan

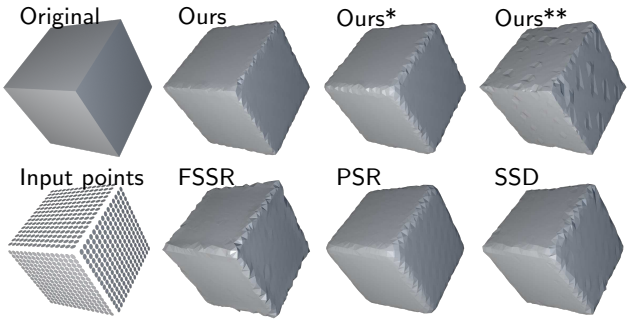

Figure 7. Reconstruction of a synthetic cube. **Original/Input points:** The input data is cube modelled with 2454 points. The image shows only the front facing points for better clarity. **Ours:** Our reconstruction solving the QEF (16) to compute vertex positions (3070 verts, 6140 tris). **Ours*:** Our reconstruction using the center of mass $\mathbf{m}$ to place vertices (3070 verts, 6140 tris). **Ours**:** Our reconstruction using extended marching cubes (Kobbelt et al, 2001) (3440 verts, 6876 tris). **FSSR:** Floating Scale Surface Reconstruction using marching cubes as in (Kazhdan et al, 2007) (5099 verts, 10194 tris). **PSR:** Poisson surface reconstruction also using (Kazhdan et al, 2007) (2934 verts, 5864 tris). **SSD:** Smoothed Signed Distance reconstruction using dual marching cubes (Schaefer and Warren, 2004) (2992 verts, 5980 tris). The maximum octree depth was set to 5 for PSR, SSD and our method.

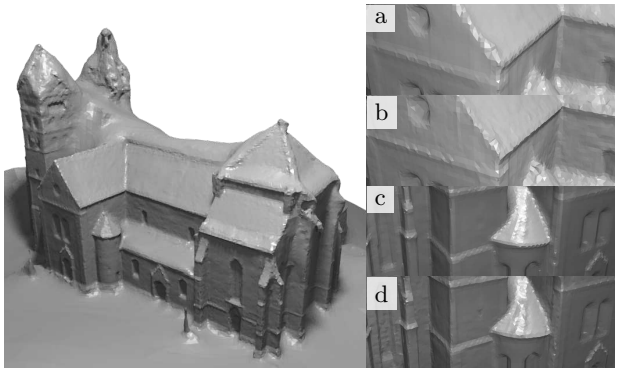

Figure 8. Effect of vertex positions being computed by solving (16) on a real data set. **Left:** Reconstruction with limited octree depth of 9 using dual contouring with improved vertex positions. **Right:** Close-up views comparing the reconstruction of edges using the center of mass (a),(c) and the 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.

and Hoppe, 2013), Smoothed Signed Distance Reconstuction (SSD) (Calakli and Taubin, 2011), Floating Scale Surface Reconstruction (FSSR) (Fuhrmann and Goesele, 2014) and the implementation of Jancosek and Pajdla (2011) (CMPMVS). 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. 6 on synthetic and real data. Fig. 7 shows the reconstructed meshes of a synthetic cube for PSR, FSSR, SSD and our method. PSR and FSSR use the marching cubes implementation of Kazhdan et al (2007). SSD uses the dual marching cubes algorithm by Schaefer and Warren (2004). Our reconstructions use dual contouring with and without normal information. We also show that the estimated normals can be used with the extended marching cubes algorithm by Kobbelt et al (2001). 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. Our reconstruction with dual contouring and the improved vertex positions gives the best reconstruction. The mesh generated with extended marching cubes yields also sharp corners and edges but shows some artifacts on the cube faces, which stem from imperfections in the signed distance function. The dual contouring approach uses up to eight values which makes the vertex positions less prone to disturbances in the signed distance field. Fig. 8 shows the effect of the improved vertex postition on real data.

Surface Extrapolation We show that the combination of the regularizer (13) acting on the vector field $\mathbf{v}$ and the coupling term (12) allows to extrapolate surfaces. This property is especially useful in cases where the data is incomplete as can be seen in Fig. 12. We show the effect for different α_1, α_2 in Fig. 9.

Multiscale Data Sets Fig. 10 shows a comparison of reconstructions on the citywall data set provided with the MVE tool (Fuhrmann et al, 2014). Our method generates a dense, high detail surface for all scales.

Following (Fuhrmann and Goesele, 2014), we show the behaviour of SSD, FSSR and our method for different point densities in Fig. 11. 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.

		Duration
Octree generation	Density estimation	74.6 min
	Balancing	7.9 min
	Histograms	782.4 min
Surface comp.	Dual grid generation	19.9 min
	Energy minimization	3678.0 min
	Dual contouring	16.3 min
Other		23.5 min
Total		4602.9 min

Table 1. Runtime breakdown for the Breisach data set. The most time consuming part during octree generation is the computation of the histograms, which takes about 17 percent of the total runtime. About 80 percent of the time is spent on energy minimization. The dual grid generation is very fast thanks to the balancing criterion and the lookup table approach. The remaining time consists mainly of disk I/O and other operations.

Next we demonstrate that our method can be applied to large scenes. The Breisach data set shown in Fig. 12 contains 1.5 billion points and models an area of about 10000m^2. Reconstruction with our method took about 77 hours on a machine with 24 cores (Intel X7460 CPU @ 2.66GHz, 2008) with a peak memory consumption of 64.0 GB. See Table 1 for more details on the runtime. We were able to reconstruct the scene with FSSR in only 71 hours 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, SSD due to limited memory and CMPMVS[1] did not generate a surface mesh.

We used our method to compute a dense reconstruction of the roman forum from an internet photo collection. We use the data set and the sparse reconstruction from (Wilson and Snavely, 2014) and (Goesele et al, 2007) to compute the point cloud. The point cloud shows strong differences in density as most points describe the individual buildings while only a small set describes the surroundings. Fig. 13 shows our reconstruction as a single connected mesh. We could reconstruct this scene with FSSR but the individual buildings are separated and most of the ground is lost.

[1] We used CMPMVS 0.6.0 with the *largeScale* option.

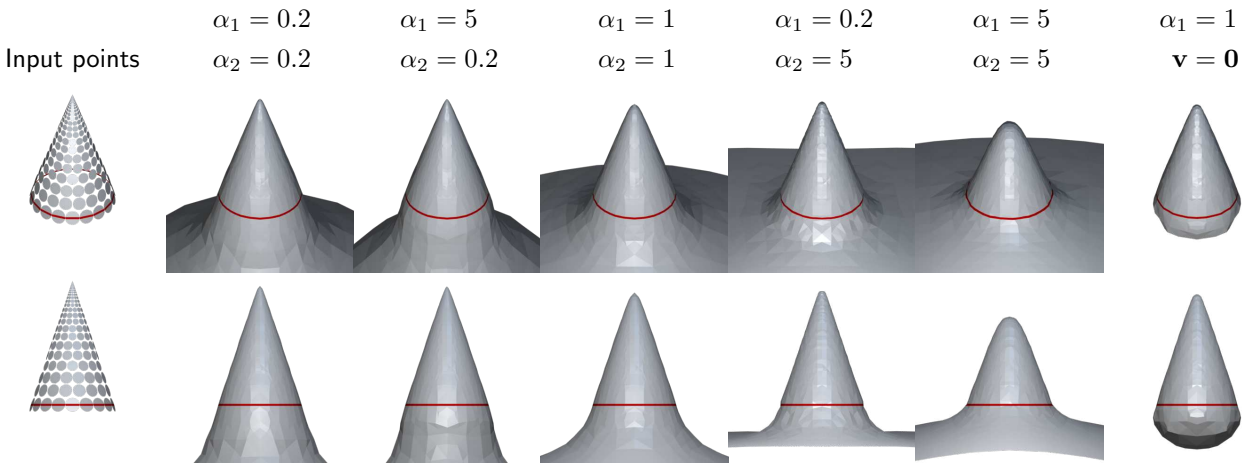

Figure 9. Influence of the smoothness parameters. The input data is a cone modelled with 479 points. There are no points at the bottom of the cone. The red line indicates the same height in all images. Both parameters influence the pointiness of the tip and how the cone surface is extended towards the bottom. $\lambda_1 = \lambda_2 = 10$ for all experiments. Larger regularization values α_2 yield a rounder tip than smaller values. The effect is strongest if we choose a large α_1 which couples the vector field $\mathbf{v}$ and the signed distance function u. Our method extends the cone surface towards the bottom. In all cases the slope of the surface below the red line is getting shallower. This is due to the average of all input point normals is pointing upwards. With a stronger regularization α_2 the bending of the surface starts closer to the cone defined by the points. The last column shows the effect of setting $\mathbf{v}$ to zero. In this case the coupling term becomes a regularizer which prefers small surfaces and the cone must be closed at the bottom.

	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
Ours*	0.43	0.61	0.78

Table 2. Accuracy on the MVS Middlebury *Temple Full* data set for different threshold levels. **Ours:** This mesh was generated with dual contouring and uses the normal information to compute optimized vertex positions. **Ours*:** This mesh does not make use of the normal information.

Temple Data Set We show that our method is competitive with respect to accuracy on the standard benchmark (Seitz et al, 2006). 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 2 show that our approach achieves a similar performance as the tested state of the art methods. They also show that making use of the normal information in the surface extraction improves the accuracy. At the time of writing, our method is ranked 11th for the 90% threshold with normals and ranked 15th without normals. For the other threshold levels both variants get adjacent ranks.

Noise Robustness We demonstrate the robustness to noise by reconstructing a car in Fig. 14. 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 is efficient enough to handle billions of 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. Our software is publicly available online[2].

Acknowledgements We acknowledge funding by the ERC Starting Grant VideoLearn and the EU project Trimbot2020.

References

Bailer C, Finckh M, Lensch HPA (2012) Scale Robust Multi View Stereo. In: Fitzgibbon A, Lazebnik S, Perona P, Sato Y, Schmid C (eds) Computer Vision

[2] http://lmb.informatik.uni-freiburg.de/people/ummenhof/multiscalefusion

Figure 10. Reconstruction of the Citywall data set from (Fuhrmann et al, 2014) with 256 million points. **Overview:** 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. **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 limits did not permit a reconstruction using a deeper octree. **CMPMVS:** The CMPMVS software correctly models most of the empty space between the houses in the city model. Some surfaces near the fountain show noise artifacts and seem to model noise. The resolution of the underlying tetrahedral net was chosen too coarse to reconstruct the lion heads with all details. Note that the CMPMVS software computes its own depth maps at full image resolution to generate its own input point cloud. **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. **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. We set $\lambda_{1,2} = 1$, $\alpha_{1,2} = 2$ for this data set. Results for SSD are not shown because the method did not finish after multiple days.

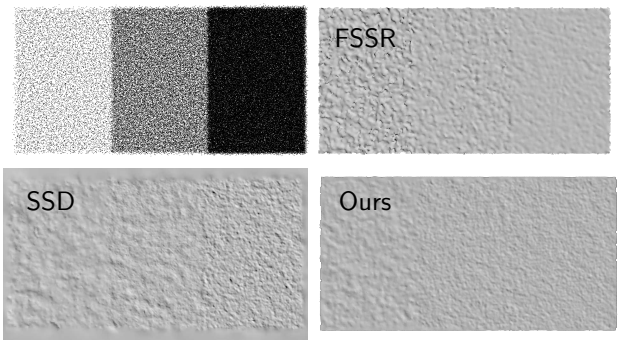

Figure 11. Reconstruction of a plane with three regions, each with a different uniform point density. The scale value assigned to the points corresponds to the sampling density of the central region. Gaussian white noise was added to the points' position and normal. **Top left:** Input point cloud. **FSSR:** With increasing density, FSSR effectively cancels out noise. In the low density region it suffers from a too sparse sampling. **SSD:** 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. **Ours:** 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.

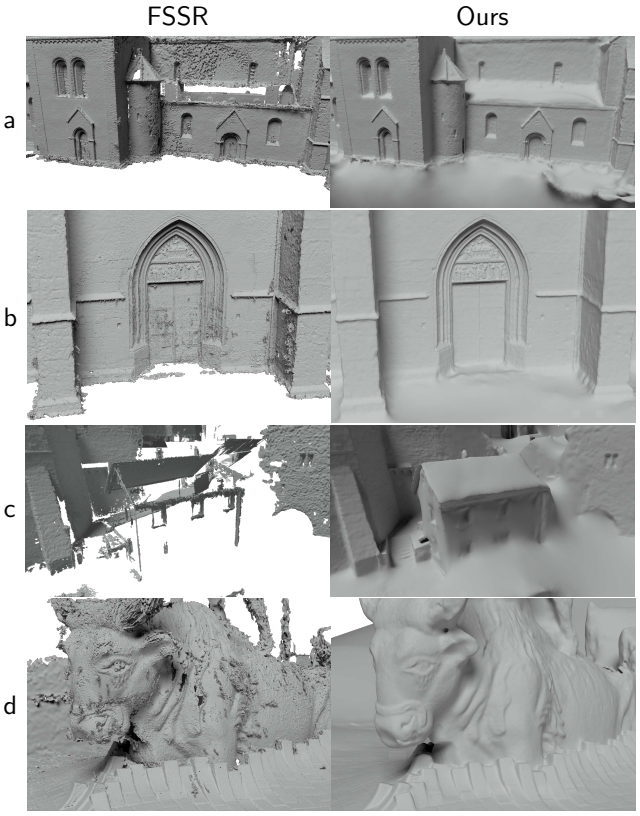

Figure 12. Reconstruction of the Breisach data set with 1.5 billion points. We have thresholded the FSSR result with the provided clean-up tool to remove clutter from the scene. A higher threshold starts to dissolve the reconstructed surfaces. Our method is able to reasonably fill holes in the reconstruction like the ground in (b) or the roof in (a). An extreme example is shown in (c), where FSSR just reconstructs the outline of the house. In regions without data the regularization can extend surfaces in a wrong way like the walls as seen in (a) at the bottom. The artifacts that can be seen in the left column of (a),(b) and (d) 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. We set $\lambda_{1,2} = \alpha_{1,2} = 1$ for this data set.

ECCV 2012, no. 7574 in Lecture Notes in Computer Science, Springer Berlin Heidelberg, pp 398–411

Blacker TD, Meyers RJ (1993) Seams and wedges in plastering: A 3-D hexahedral mesh generation algorithm. Engineering with Computers 9(2):83–93, DOI 10.1007/BF01199047

Bolitho M, Kazhdan M, Burns R, Hoppe H (2007) Multilevel Streaming for Out-of-core Surface Reconstruction. In: Proceedings of the Fifth Eurographics Symposium on Geometry Processing, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SGP '07, pp 69–78

Calakli F, Taubin G (2011) SSD: Smooth Signed Distance Surface Reconstruction. Computer Graphics Forum 30(7):1993–2002, DOI 10.1111/j.1467-8659.2011.02058.x

Curless B, Levoy M (1996) A volumetric method for building complex models from range images. In: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, ACM, New York, NY, USA, SIGGRAPH '96, pp 303–312, DOI 10.1145/237170.237269

Estellers V, Scott M, Tew K, Soatto S (2015) Robust Poisson Surface Reconstruction. In: Aujol JF, Nikolova M, Papadakis N (eds) Scale Space and Variational Methods in Computer Vision, no. 9087 in Lecture Notes in Computer Science, Springer International Publishing, pp 525–537

Frahm JM, Fite-Georgel P, Gallup D, Johnson T, Raguram R, Wu C, Jen YH, Dunn E, Clipp B, Lazebnik S, Pollefeys M (2010) Building Rome on a Cloudless Day. In: Daniilidis K, Maragos P, Paragios N (eds) Computer Vision ECCV 2010, no. 6314 in

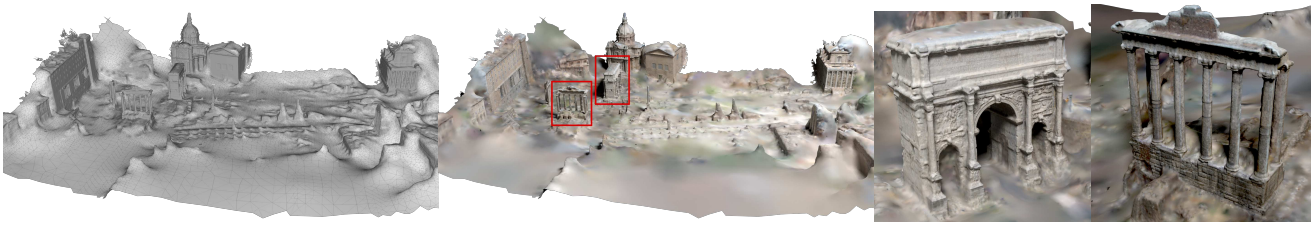

Figure 13. Dense reconstruction of the roman forum with our method. The input point cloud contains 638 million samples. **Left:** Wireframe rendering. Sights like the arch have a very high resolution in the octree and in the resulting surface mesh. The density of the triangle mesh changes smoothly due to the balancing criterion we imposed on the octree. **Center:** Colored reconstruction. **Right:** Close-up of the arch and temple.

Lecture Notes in Computer Science, Springer Berlin Heidelberg, pp 368–381

Fuhrmann S, Goesele M (2011) Fusion of depth maps with multiple scales. In: Proceedings of the 2011 SIGGRAPH Asia Conference, ACM, New York, NY, USA, SA '11, pp 148:1–148:8, DOI 10.1145/2024156.2024182

Fuhrmann S, Goesele M (2014) Floating Scale Surface Reconstruction. ACM Trans Graph 33(4):46:1–46:11, DOI 10.1145/2601097.2601163

Fuhrmann S, Langguth F, Goesele M (2014) MVE - A Multi-View Reconstruction Environment. In: Proceedings of the Eurographics Workshop on Graphics and Cultural Heritage (GCH)

Gargantini I (1982) Linear octtrees for fast processing of three-dimensional objects. Computer Graphics and Image Processing 20(4):365–374, DOI 10.1016/0146-664X(82)90058-2

Goesele M, Snavely N, Curless B, Hoppe H, Seitz S (2007) Multi-View Stereo for Community Photo Collections. In: IEEE 11th International Conference on Computer Vision, 2007. ICCV 2007, pp 1–8, DOI 10.1109/ICCV.2007.4408933

Hiep VH, Keriven R, Labatut P, Pons JP (2009) Towards high-resolution large-scale multi-view stereo. In: IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009, pp 1430–1437, DOI 10.1109/CVPR.2009.5206617

Hirschmüller H (2005) Accurate and efficient stereo processing by semi-global matching and mutual information. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol 2, pp 807–814 vol. 2, DOI 10.1109/CVPR.2005.56

Jancosek M, Pajdla T (2011) Multi-view reconstruction preserving weakly-supported surfaces. In: 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 3121–3128, DOI 10.1109/CVPR.2011.5995693

Ju T, Losasso F, Schaefer S, Warren J (2002) Dual Contouring of Hermite Data. In: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, ACM, New York, NY, USA, SIGGRAPH '02, pp 339–346, DOI 10.1145/566570.566586

Kazhdan M, Hoppe H (2013) Screened Poisson Surface Reconstruction. ACM Trans Graph 32(3):29:1–29:13, DOI 10.1145/2487228.2487237

Kazhdan M, Klein A, Dalal K, Hoppe H (2007) Unconstrained Isosurface Extraction on Arbitrary Octrees. In: Proceedings of the Fifth Eurographics Symposium on Geometry Processing, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SGP '07, pp 125–133

Kobbelt LP, Botsch M, Schwanecke U, Seidel HP (2001) Feature Sensitive Surface Extraction from Volume Data. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, ACM, New York, NY, USA, SIGGRAPH '01, pp 57–66, DOI 10.1145/383259.383265

Labatut P, Pons JP, Keriven R (2009) Robust and Efficient Surface Reconstruction From Range Data. Computer Graphics Forum 28(8):2275–2290, DOI 10.1111/j.1467-8659.2009.01530.x

Lewiner T, Mello V, Peixoto A, Pesco S, Lopes H (2010) Fast Generation of Pointerless Octree Duals. Computer Graphics Forum 29(5):1661–1669, DOI 10.1111/j.1467-8659.2010.01775.x

Li X, Andersen DG, Kaminsky M, Freedman MJ (2014) Algorithmic Improvements for Fast Concur-

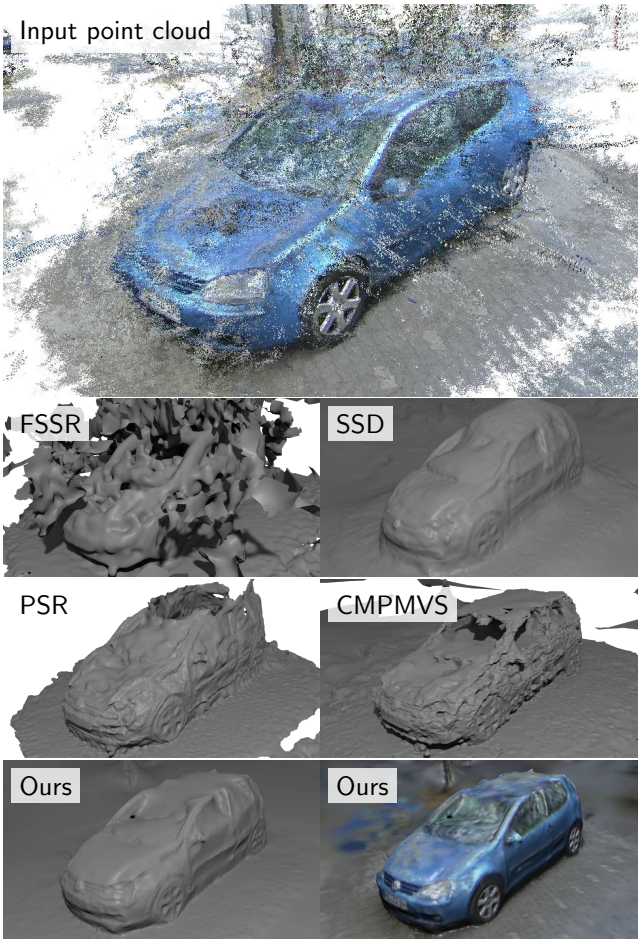

Figure 14. **Top:** Input point cloud used for FSSR, SSD, PSR and our method. **FSSR:** The FSSR method shows strong artifacts. We used a scale parameter of 8 and applied the provided clean-up tool to threshold the mesh. Choosing a higher threshold dissolves the car. **SSD:** The SSD method loses details 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. **CMPMVS:** The reconstruction with CMPMVS recovers the side mirror but also generates a bumpy surface for most of the car. There is also a dent in the hood. There is a clear boundary between the ground and the car. Note that CMPMVS does compute its own input point cloud with plane sweeping stereo and semi global matching Hirschmüller (2005), which tends to generate a denser point cloud. **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. We set $\lambda_{1,2} = 1$ and $\alpha_{1,2} = 5$ to give more weight to the regularization for this data set.

rent Cuckoo Hashing. In: Proceedings of the Ninth European Conference on Computer Systems, ACM, New York, NY, USA, EuroSys '14, pp 27:1–27:14, DOI 10.1145/2592798.2592820

Manson J, Petrova G, Schaefer S (2008) Streaming Surface Reconstruction Using Wavelets. In: Proceedings of the Symposium on Geometry Processing, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SGP '08, pp 1411–1420

Marchal L (2009) Advances in Octree-Based All-Hexahedral Mesh Generation: Handling Sharp Features. In: Clark BW (ed) Proceedings of the 18th International Meshing Roundtable, Springer Berlin Heidelberg, pp 65–84

Mavriplis DJ (1995) An Advancing Front Delaunay Triangulation Algorithm Designed for Robustness. Journal of Computational Physics 117(1):90–101, DOI 10.1006/jcph.1995.1047

Müller M, Charypar D, Gross M (2003) Particle-based Fluid Simulation for Interactive Applications. In: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SCA '03, pp 154–159

Pock T, Zebedin L, Bischof H (2011) TGV-Fusion. In: Calude CS, Rozenberg G, Salomaa A (eds) Rainbow of Computer Science, no. 6570 in Lecture Notes in Computer Science, Springer Berlin Heidelberg, pp 245–258

Ruppert J, Seidel R (1992) On the difficulty of triangulating three-dimensional Nonconvex Polyhedra. Discrete & Computational Geometry 7(1):227–253, DOI 10.1007/BF02187840

Sagawa R, Nishino K, Ikeuchi K (2005) Adaptively merging large-scale range data with reflectance properties. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(3):392–405, DOI 10.1109/TPAMI.2005.46

Schaefer S, Warren J (2004) Dual marching cubes: primal contouring of dual grids. In: 12th Pacific Conference on Computer Graphics and Applications, 2004. PG 2004. Proceedings, pp 70–76, DOI 10.1109/PCCGA.2004.1348336

Seitz S, Curless B, Diebel J, Scharstein D, Szeliski R (2006) A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In: 2006 IEEE Computer Society Conference on Computer Vision

and Pattern Recognition, vol 1, pp 519–528, DOI 10.1109/CVPR.2006.19

Shewchuk JR (1998) Tetrahedral Mesh Generation by Delaunay Refinement. In: Proceedings of the Fourteenth Annual Symposium on Computational Geometry, ACM, New York, NY, USA, SCG '98, pp 86–95, DOI 10.1145/276884.276894

Ummenhofer B, Brox T (2015) Global, Dense Multiscale Reconstruction for a Billion Points. In: IEEE International Conference on Computer Vision (ICCV)

Wilson K, Snavely N (2014) Robust Global Translations with 1dsfm. In: Fleet D, Pajdla T, Schiele B, Tuytelaars T (eds) Computer Vision ECCV 2014, no. 8691 in Lecture Notes in Computer Science, Springer International Publishing, pp 61–75

Wu C (2013) Towards Linear-Time Incremental Structure from Motion. In: 2013 International Conference on 3D Vision - 3DV 2013, pp 127–134, DOI 10.1109/3DV.2013.25

Wu C, Agarwal S, Curless B, Seitz S (2011) Multicore bundle adjustment. In: 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 3057–3064, DOI 10.1109/CVPR.2011.5995552

Zach C (2008) Fast and high quality fusion of depth maps. In: Proceedings of the International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT), vol 1