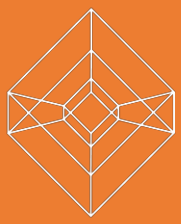


HARVEST4D

HARVESTING DYNAMIC 3D WORLDS FROM COMMODITY SENSOR CLOUDS

Publications for Task 5.3

Deliverable 5.31

Date:	30.06.2016
Grant Agreement number:	EU 323567
Project acronym:	HARVEST4D
Project title:	Harvesting Dynamic 3D Worlds from Commodity Sensor Clouds

Document Information

Deliverable number	D5.31
Deliverable name	Publications for Task 5.3
Version	1.0
Date	2016-06-30
WP Number	5
Lead Beneficiary	TUDA
Nature	R
Dissemination level	PU
Status	Final
Author(s)	TUDA (Samir Aroudj), DELFT (Timothy Kol)

Revision History

Rev.	Date	Author	Org.	Description
0.1	22.06.2016	Samir Aroudj	TUDA	Initial Version
0.2	27.06.2016	Samir Aroudj	TUDA	2.4 + Corrections
0.3	30.06.2016	Aroudj, Kol	TUDA, DELFT	Corrected Version

Statement of originality

This deliverable, although of public dissemination level, may at the time of delivery still contain original unpublished work (e.g., accepted papers that are not public yet, or papers under revision). Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

TABLE OF CONTENTS

1 Executive Summary 1
1.1 Introduction 1
1.2 Publications 1
2 Description of Publications 2
2.1 Overview 2
2.2 Accurate Isosurface Interpolation with Hermite Data 2
2.3 Simplification of Multi-Scale Geometry using Adaptive Curvature Fields 3
2.4 Automatic Reconstruction of Parametric Building Models from Indoor Point Clouds 4
3 Other Results 6
4 Appendix 7

1 EXECUTIVE SUMMARY

1.1 INTRODUCTION

This deliverable describes the publications that resulted from task 5.3 and how they fit into the work plan of the project.

The objective of task 5.3 is to improve basic multi-scale algorithms developed for or related to task 5.2. For this reason, we have devised new or adapted existing implementations to achieve results of higher quality or targeted performance issues, such as high computational demands, high memory consumption, etc.

There are three publications which are mainly attributed to task 5.3. They are in the appendix of this deliverable. Additionally, there are five publications that are related to this deliverable. We briefly mention these documents but do not discuss them in depth since they are extensively described in other deliverables. For the latter, related papers are available via the Harvest4D web site or in the deliverables to which they mainly belong.

1.2 PUBLICATIONS

The following three main publications of task 5.3 can be found in the appendix:

- Patrick Seemann, Simon Fuhrmann, Fabian Langguth, Stefan Guthe and Michael Goesele
  Simplification of Multi-Scale Geometry Using Adaptive Curvature Fields.
  Submitted to the 21st International Symposium on Vision, Modeling and Visualization (VMV), Bayreuth, Germany, 2016.
- Sebastian Ochmann, Richard Vock, Raoul Wessel and Reinhard Klein
  Automatic Reconstruction of Parametric Building Models from Indoor Point Clouds.
  In: Computers & Graphics, 2015.
- Simon Fuhrmann, Misha Kazhdan and Michael Goesele
  Accurate Isosurface Interpolation with Hermite Data.
  In: Proceedings of the International Conference on 3D Vision, Lyon, France, 2015.

The following related publications are on the Harvest4D web site or in other deliverables:

- Jean-Marc Thiery, Emilie Guy, Tamy Boubekeur and Elmar Eisemann
  Animated Mesh Approximation with Sphere-Meshes.
  In: ACM Transactions on Graphics (Proceedings of SIGGRAPH), 2016.
- Leonardo Scandolo, Pablo Bauszat and Elmar Eisemann
  Compressed Multiresolution Hierarchies for High-Quality Precomputed Shadows.
  In: Computer Graphics Forum (Proceedings of Eurographics), 2016.

- Murat Arikan, Reinhold Preiner, Claus Scheiblauer, Stefan Jeschke and Michael Wimmer
  Large-Scale Point Cloud Visualization through Localized Textured Surface Reconstruction.
  In: IEEE Transactions on Visualization & Computer Graphics, 2014.
- Murat Arikan, Reinhold Preiner and Michael Wimmer
  Multi-Depth-Map Raytracing for Efficient Large-Scene Reconstruction.
  In: IEEE Transactions on Visualization & Computer Graphics, 2015.
- Thierry Guillemot, Andrès Almansa and Tamy Boubekeur
  Covariance Trees for 2D and 3D Processing.
  In: Computer Vision and Pattern Recognition (CVPR, Oral), 2014.

2 DESCRIPTION OF PUBLICATIONS

2.1 OVERVIEW

The multi-scale data structures and algorithms that we implemented for task 5.2 underwent improvements or augmentations to fulfill the goals of the successive task 5.3. As initially planned, we focused on implementation and scalability issues. That is why we investigated problematic parts of our multi-scale approaches and searched fitting enhancements w.r.t. quality of reconstructions or scalability.

Particularly, we improved the isosurface extraction method of the algorithm *Floating Scale Surface Reconstruction* to increase the quality of our multi-scale mesh reconstructions. Moreover, we devised a novel, automatic and adaptive curvature estimation algorithm for such multi-scale models to enable reasonable application of post processing steps like mesh simplification for less consumption of resources. Finally, we developed an automatic approach for creation of multi-level and parametric building models from indoor point clouds.

2.2 ACCURATE ISOSURFACE INTERPOLATION WITH HERMITE DATA

As described in deliverable 5.22, *Floating Scale Surface Reconstruction* (FSSR) [Fuhrmann et al. 2014] takes a dense point cloud that represents the scene as input for creating a globally consistent multi-scale surface mesh. The algorithm computes a local basis function for each input surface sample. FSSR aggregates all of these local functions in an implicit and approximate surface distance function to get an intermediate representation of the complete scene. We finally employ a sparse octree in order to efficiently sample the global implicit function in a multi-scale fashion. This enables a hierarchical marching cubes (MC) surface extraction algorithm.

In contrast to the previous FSSR approach [Fuhrmann et al. 2014] that we describe in deliverable 5.21, the work of the follow-up publication [Fuhrmann et al. 2015] investigates a novel MC surface extraction algorithm. Previously, we employed an MC algorithm that performs contouring of a global implicit function by means of linear interpolation. Hereby, isovertices are found by

finding zero-crossings in linear interpolations along cell edges of signed implicit function values defined at octree cell corners. These isovertices are then triangulated to get a surface mesh. Figure 1 illustrates this approach.

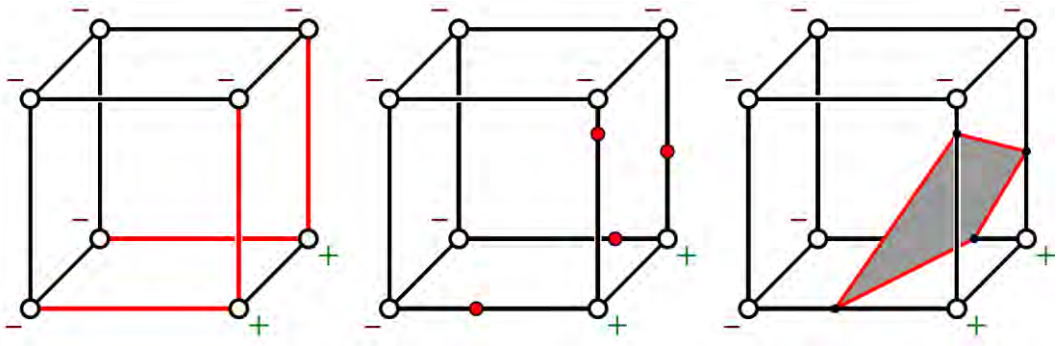


Figure 1. [*Fuhrmann et al. 2015*]: There are four edges with a sign change (left), interpolation of the signed implicit function values results in four corresponding isovertices (middle) which are used for polygonization (right).

The linear interpolation works well in cases where the underlying function is piecewise linear or close to that. Global implicit functions produced by FSSR or *Poisson Surface Reconstruction* are clearly not piecewise linear [Fuhrmann et al. 2015]. This leads to distance errors between the implicit representation and the final surface. We solve this problem by computing implicit function gradients to obtain Hermite data and perform non-linear interpolation, which results in a closer fit of the extracted surface to its underlying implicit function. Figure 2 shows a comparison that presents the increased quality of reconstructions.

Figure 2. [*Fuhrmann et al. 2015*], Miniature *City*: Geometric quality difference using linear (left) and cubic interpolation (right) for contouring.

2.3 SIMPLIFICATION OF MULTI-SCALE GEOMETRY USING ADAPTIVE CURVATURE FIELDS

In this publication, we present a novel algorithm to compute multi-scale curvature fields on triangle meshes. Our algorithm is based on finding robust mean curvatures using the ball neighborhood, where the radius of a ball corresponds to the scale of the features. The challenge lies in finding a good radius for each ball to obtain a reliable curvature estimation. Figure 3 exemplarily shows that using a single scale for curvature estimation on true multi-scale meshes

fails. It results in either overfitting due to ball neighborhoods that are too small and fit to noise, or oversmoothing due to ball neighborhoods which are too large and coarse for smaller and enclosed features.

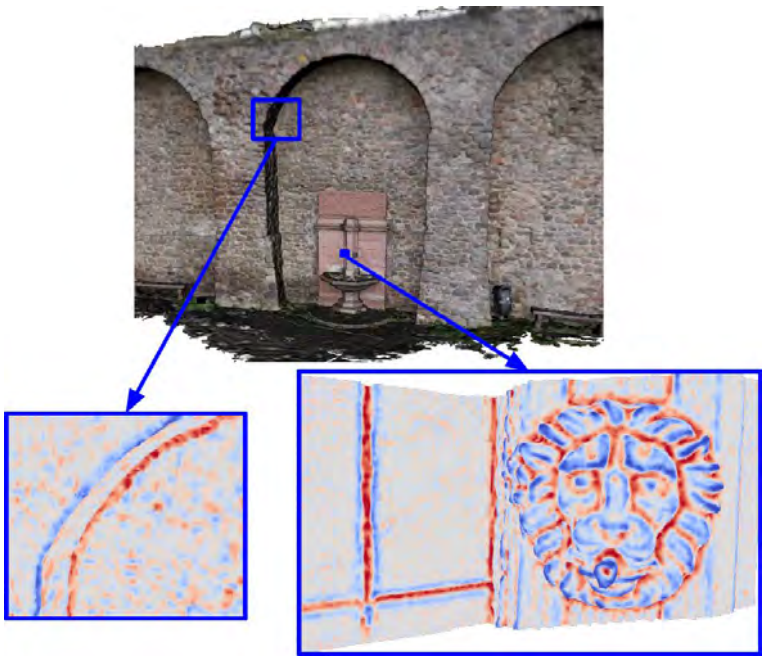


Figure 3. [*Seemann et al. 2016*]: *City Wall* dataset (top) and two close-ups of the mean curvature field computed at a single scale which preserves small features of the lion head (right). The highlights show two example areas in which the mean curvature has the same magnitude even though the scale of the mesh is much smaller at the lion head than on the edge (left). Note also the noise in planar regions.

We propose an algorithm that finds suitable radii in an automatic way. In particular, our algorithm is applicable to meshes produced by image-based reconstruction systems, such as MVE and FSSR. The resulting meshes often contain geometric features at various scales, for example if certain regions have been captured in higher detail. We also show how to convert such a multi-scale curvature field to a density field which can subsequently be used to guide applications like mesh simplification.

2.4 AUTOMATIC RECONSTRUCTION OF PARAMETRIC BUILDING MODELS FROM INDOOR POINT CLOUDS

We present a novel, automatic approach for the reconstruction of parametric 3D building models from indoor point clouds [Ochmann et al. 2015]. Figure 4 briefly presents an overview of our method. In contrast to previous approaches, such parametric building models additionally incorporate contextual information about the scene such as large-scale, global wall connectivity. Our global optimization approach reconstructs wall elements shared between rooms while it simultaneously maintains plausible connectivity between all wall elements. Our automatic prior segmentation of the input point clouds into rooms and outside areas filters large-scale outliers

and yields priors for the definition of labeling costs for our energy minimization. Detected doors and windows further enrich the reconstructed model.

First, the resulting parametric models allow for an efficient high-level scene editing and even prototyping in terms of, e.g., wall removal or room reshaping, which always result in a topologically consistent representation. Second, our parametric models also enable determination of low-level details like wall thickness or room areas. Thanks to these multi-scale properties of our building models, they are relevant for architects and engineers. Furthermore, the global connectivity information enables path finding in whole buildings that are particularly relevant for tasks, such as simulation and optimization of escape routes.

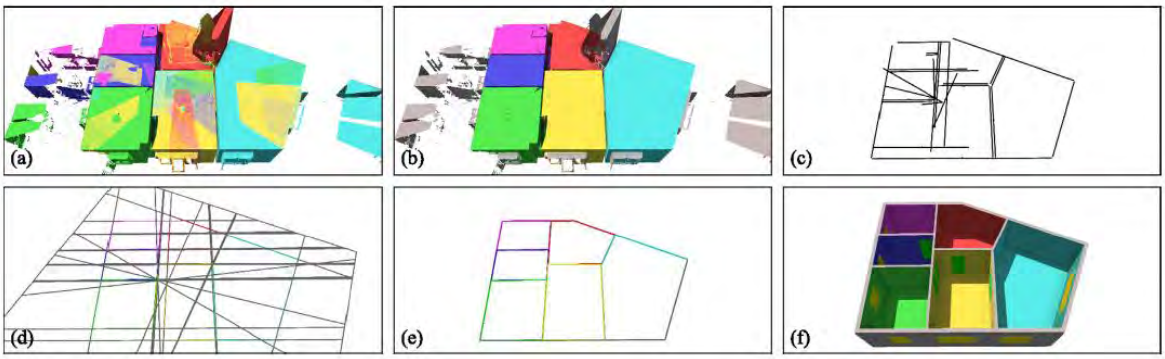


Figure 4. [*Ochmann et al. 2015*]: (a) Input point cloud with assignment of points to differently colored scans. (b) Refined assignment after automatic segmentation. (c) Detected vertical planes transferred to the ground plane. (d) Wall candidates derived from single and pairs of projected planes. (e) Pruned ground plan via graph labeling. (f) Final model with detected and classified wall openings, (green) doors and (yellow) windows.

3 OTHER RESULTS

A number of additional publications that mainly belong to other deliverables and which are discussed in these also make use of multi-scale aspects. Here, we briefly explain their relationship to task 5.3.

Covariance Trees [Guillemot et al. 2014] target the issue that Gaussian Mixture Models have not been adopted highly despite their performance breakthroughs in patch-based image denoising and restoration problems. Important reasons for that are simply the high computational demand for learning such models on large image databases and their need for tedious parameter trimming. Contrary to that, Covariance Trees are flexible and generic tools that can handle such models and circumvent the mentioned computational penalty or tedious manual trimming. The Covariance Trees achieve this by their hierarchical multi-scale data organization. It allows for queries at different levels of scale around any point in feature space.

Multi Mesh Texturing [Arikan et al. 2014] and Multi-Depth-Map Raytracing [Arikan et al. 2015] provide a high-quality visualization of large input point clouds that are augmented with high-resolution images taken from different positions and viewing angles within the scene. In both approaches, the geometry is represented at the scale of the best available photograph, and stored along with the RGB data in a set of textured depth maps, which are triangulated and stitched [Arikan et al. 2014] or ray traced [Arikan et al. 2015] to a complete image at render time. Similar in spirit to FSSR, this scene representation in its entirety consists of a composition of data from multiple scales. The multi-scale composition is guided by available images and automatically adapts to the optimal scale to cope with large data sets and keep demand of resources low.

Sphere meshes enable extreme simplification of an animated triangle mesh [Thiery et al. 2016] by indexing a number of spheres. It offers a tradeoff between performance and accuracy to the user. A higher accuracy that preserves small-scale animation features is achieved by increasing the number of spheres, which in turn, naturally, decreases the performance and increases memory consumption. On the other hand, the user can sacrifice accuracy for lower resource demand by employing fewer spheres when features of larger scale are already sufficient.

High-quality precomputed shadows can be significantly compressed by exploitation of multi-scale properties [Scandolo et al. 2016] which enables wider application of high-resolution shadow maps. We achieve high compression rates using multi-resolution hierarchies that are computed efficiently on the GPU. Moreover, the hierarchical structure of our GPU data enables real-time rendering.

4 APPENDIX

The following pages contain all the publications listed in Section 1.2 that primarily belong to this deliverable. Other publications, which we reference here, are on the public Harvest4D web site or in other deliverables.

Vision, Modeling, and Visualization (2016), pp. 1–8
M. Hullin, M. Stamminger, and T. Weinkauf (Eds.)

Simplification of Multi-Scale Geometry using Adaptive Curvature Fields

paper1001

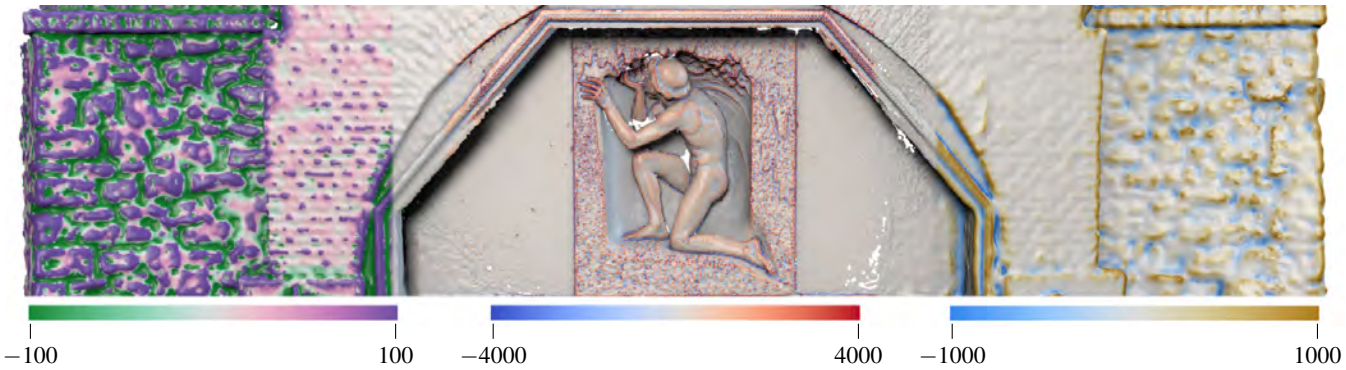


Figure 1: A single mean curvature field visualized on the mesh surface. In multi-scale meshes, the scale between the curvature values vary by several orders of magnitude. This is illustrated here using three colormaps that show the curvature field at different scales.

Abstract

We present a novel algorithm to compute multi-scale curvature fields on triangle meshes. Our algorithm is based on finding robust mean curvatures using the ball neighborhood, where the radius of a ball corresponds to the scale of the features. The essential problem is to find a good radius for each ball to obtain a reliable curvature estimation. We propose an algorithm that finds suitable radii in an automatic way. In particular, our algorithm is applicable to meshes produced by image-based reconstruction systems. These meshes often contain geometric features at various scales, for example if certain regions have been captured in greater detail. We also show how such a multi-scale curvature field can be converted to a density field and used to guide applications like mesh simplification.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric Algorithms, Languages, and Systems

1. Introduction

Triangle meshes are the most common geometry representation and their properties have been studied extensively in order to visualize, analyze, and modify them effectively. An important geometric property is surface curvature. In differential geometry it is readily defined via the second derivative of the surface. However, due to the discrete nature of triangle meshes the computation of their curvature values is non-trivial. In practical scenarios noise can have a strong influence on the output of estimation algorithms. To cope with these problems recent techniques [YLHP06, SHBK10, APM15] apply a smoothing operator which successfully removes noise but ultimately also affects the geometric detail. The biggest problem is to select an appropriate scale for this operator. If the scale is chosen too small, the noise will interfere with curvature estimation; if the scale is chosen too large, surface details will be smoothed away. This problem is even more pronounced for multi-scale geometry. Image-based geometry acquisition pipelines using multi-view stereo (e.g., [FLG14]) can generate surfaces on vastly different scales depending on the camera resolution and its distance to the real-world objects as illustrated in Figure 2. The resulting triangle meshes then contain geometric features and noise on different levels of detail. A single scale curvature estimation cannot capture the true properties of the whole surface.

In this work we present a novel algorithm that estimates the curvature field of multi-scale triangle meshes. Previous methods [YLHP06, SHBK10] compute curvatures by evaluating a neighbor-

hood around a given vertex using the ball neighborhood, which we also use in our work. Integral invariants (Section 3.1) can then be used to compute the mean curvature using the neighborhood of a vertex within the ball radius. The chosen radius defines the scale at which features are preserved and noise is smoothed. If the radius of the ball is fixed, the operator uses a uniform scale and cannot adapt to the scale variations of the surface. As a result the operator smoothes too much detail or retains too much noise.

Our main contribution is the independent and automatic selection of an appropriate ball radius for each vertex. Our method is robust against large variations in scale and can effectively distinguish between noise and geometric features. It operates directly on the mesh representation and does not require a volumetric shape representation. It is able to handle difficult input data, such as the meshes produced by image-based reconstruction techniques, which usually have varying level of detail and contain many holes.

A direct application of our method is mesh simplification. Particularly in image-based reconstruction scenarios the resulting meshes often contain millions of triangles because the vertex sampling is determined by the resolution of input images, not the geometric properties of the surface. We show how our estimated curvature field can be effectively transformed into a density field that guides the simplification process. As a result, the simplification algorithm does not need to be concerned with preserving geometric features of the surface. Instead, its task reduces to producing a vertex distribution prescribed by the density field, thus preserving more geometric detail in regions of higher curvature.

2. Related Work

Curvature estimation on discrete surfaces has been thoroughly studied and can be classified into local fitting methods, methods based on the angles between edges, and integral invariant-based methods, which integrate over larger surface regions. The latter methods are most promising in our scenario and usually perform better if the meshes are large, have geometric features at various scales, varying level of detail and noise. Many methods estimate curvature on a user-provided scale and compute curvature using a neighborhood with fixed radius. These methods do not perform well on multi-scale geometry because a suitable radius does not exist. Multi-scale methods, on the other hand, try to determine a suitable radius for each vertex.

2.1. Curvature Estimation on a Fixed Scale

There are many algorithms for computing curvature on a fixed scale [YZ13, ASWL11, MOG09]. The scale is usually provided by the user as input. Seibert et al. [SHBK10] make use of geometric algebra and compute principal curvatures directly on point set surfaces. Their approach estimates curvature at each vertex x by fitting osculating circles in uniformly sampled directions around x to a fixed local neighborhood of points. The principal curvatures for each vertex are obtained by combining the radii of the osculating circles for all directions. Because their approach relies on least squares fitting, a dense vertex sampling is required, and noise and outliers quickly degrade the curvature estimation. Their algorithm is not applicable to multi-scale geometry because it operates on a fixed local neighborhood around each vertex.

Andreadis et al. [APM15] compute geometric features (such as mean curvature [PWY*07]) at multiple scales by first transforming the input mesh into a parametric space. This decouples the computational complexity from the underlying geometry in order to produce a GPU-friendly, highly performant algorithm. However, their method relies on a mesh parameterization which has to be precomputed and is more difficult for less controlled meshes (higher genus, boundaries and holes in the surface, etc). The scale at which the curvature is computed is fixed and provided as user input.

Other approaches are based on integral invariants. Yang et al. [YLHP06, PWY*07] were the first to use integral invariants for robust estimation of curvature information of 3D meshes. To this end, the authors define the ball, sphere and surface-patch neighborhoods and perform a principal component analysis (PCA) on each neighborhood. They derive formulas to calculate the two principal curvatures and the mean curvature based on volume integral invariants from the PCA. Their definition yields the notation of curvatures at a scale r, where r corresponds to the radius of the neighborhood. The authors claim that their approach is more robust than normal cycles [CSM03] and local fitting methods like osculating jets [CP05]. In particular, the ball neighborhood seems suitable for noisy input data. Our approach is based on the ball neighborhood, and we extend their method by robust and automatic, per-vertex scale selection over a large range of scales.

2.2. Multi-Scale Curvature Estimation

Multi-scale algorithms try to choose an appropriate scale for each vertex at which the curvature is estimated. Usually, the user specifies a lower and upper bound instead of a single scale. Lai et al. [LHF09] also use integral invariants based on the ball neighborhood to compute multi-scale principal curvatures. Instead of relying on user input to specify the scale of interest, an iterative algorithm adjusts the radius r of the ball neighborhood for each vertex independently. A series of n subsequent ball radii between by a lower and upper are evaluated, and principal curvatures are computed for each radius. The algorithm uses the pre-computed curvatures and interpolates new, refined radii until convergence.

Choosing the lower and upper bound, however, remains a challenging problem. The authors propose to determine these values as factors of the average edge length in the mesh. While this solution works in the authors scenario where evaluation is performed on meshes with almost identical triangle sizes, we target true multi-scale meshes where the triangle sizes vary substantially. Thus, a global starting radius cannot be defined, this is illustrated in Figure 2. Our method improves this aspect and selects the per-vertex radius using the edge lengths in the local neighborhood of a vertex.

Another drawback of their method is that it requires a voxel representation of the model to approximate the volume of the ball neighborhood. However, a voxelization (e.g., using scan conversion) causes problems when the input mesh is not closed or has many holes. Our algorithm computes the ball neighborhood at a given vertex without relying on a volumetric representation.

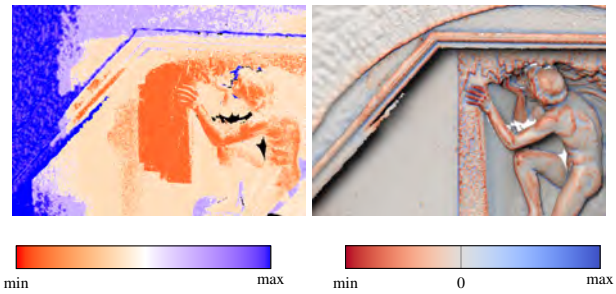


Figure 2: The average edge length in a 1-ring around a vertex (left) varies drastically throughout the mesh. Our mean curvature field (right) is not influenced by the different triangle sizes and produces correct values. To increase readability, the average edge length was clamped to the 1^{th}- and 90^{th}-percentiles and plotted in log-scale. In this mesh, the smallest triangles are about 42 times smaller than the largest ones.

3. Algorithm

Curvature is defined as the second-order derivative of the mesh surface and thus inherently depends on its scale. Given the discrete nature of a mesh, it must be evaluated numerically over an appropriately sized neighborhood. Additionally, if the mesh contains noise, a large enough neighborhood must be found that cancels out the noise while maintaining important surface details.

The first and most involved step of our algorithm computes the curvature field on the mesh surface. For each vertex of the mesh, a ball with an appropriate radius is found and used to compute a signed mean curvature value using integral invariants. We then motivate how the curvature field can be used for the purpose of mesh simplification by converting it to a density field. The density field prescribes the relative vertex sampling density to faithfully represent the mesh surface at a given vertex budget, by distributing more vertices in regions of higher curvature. As a result, the simplification algorithm is not concerned with preserving geometric features during decimation. Instead, it merely selects vertices for decimation which have a small amount of density associated with it.

3.1. Integral Invariants

A detailed analysis of integral invariants was published by Manay et al. [MHYS04]. In essence, they are used to compute integral quantities (as opposed to differential ones) over different types of neighborhoods. The invariance depends on the function that is used to compute it. In our case, we compute the volume integral, which yields the invariance to mesh rotation and translation. Because one does not have to compute higher order derivatives, integral invariants are more stable in the presence of noise. The neighborhood on triangle meshes is defined by the surface of the mesh and the volume which it represents. Pottmann et al. [PWY*07] propose the sphere, ball, and the surface-patch neighborhoods. They analyze these neighborhoods with respect to their noise properties and conclude that the ball neighborhood performs best. We use this neighborhood when computing our volume integral invariants.

The formal definition of the volume integral invariant using the ball neighborhood is as follows. Let D be a domain and Φ its boundary surface. For any point $p \in \Phi$ and ball $B(r,p)$ with radius r centered at p one can compute the intersection volume for the neighborhood around r: $V_b(r,p) = D \cap B(r,p)$. Pottmann et al. [PWY*07] derive a formula for computing the mean curvature H from the ball neighborhood at a given radius r:

$$H_{ball}(r,p) = \frac{4}{\pi r^4}\left(\frac{2\pi}{3}r^3 - V_b(r,p)\right) \quad (1)$$

This discrete mean curvature converges to the actual continuous mean curvature for $r \to 0$. For larger radii, the mean curvature is smoothed. As Lai et al. [LHF09] noted, the radius r is of special importance since larger radii produce smoother results which are thus more robust to noise. On the other hand, there is a risk of smoothing away small geometric features when the radius is too large. Therefore, a single radius is not applicable for meshes with multiple scales. Because in some regions the radius will be too large and smoothes away important surface details. In other regions, the radius will be too small and produces an unwanted response in the presence of noise.

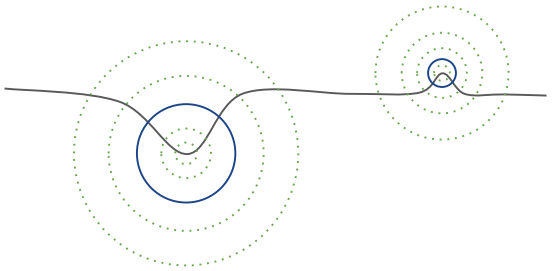


Figure 3: For each vertex, we compute the mean curvature at different radii (green circles) and then select the correct radius for each vertex (blue circles).

3.2. Curvature Field Computation

To compute the curvature field of a triangle mesh, we first evaluate the mean curvature at each vertex using Equation 1 on multiple radii (see Section 3.5) and eventually select the correct radius for each vertex, see Figure 3. For the calculation of the integral invariant, the volume of the ball neighborhood has to be evaluated for each individual radius. We approximate this volume using a triangulated sphere with an adaptive tessellation along the intersection border (see Section 3.4). To intersect the sphere with the surface of the mesh, a circular surface patch consisting of all faces within the the current radius (see Section 3.3) is used. For finding the final radius, we fit a cubic polynomial to the collected data as described in Section 3.5.

3.3. Surface Patch

For calculating the volume of the ball neighborhood $V_b(v_c,r)$ around a vertex v_c, we first find the surface patch contained within the radius r using a region growing approach. We start by iterating over all faces f_i in a 1-ring around v_c. For each vertex v_j in f_i we check whether it is inside the radius by calculating the Euclidean

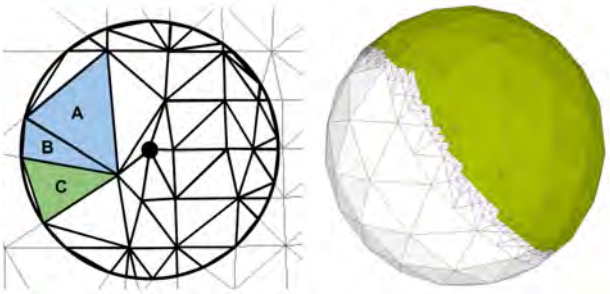


Figure 4: Triangulation of a circular patch (left): New triangles are created where the circle intersects a face. Two triangles (A and B) are created if one vertex is outside the radius. One triangle (C) is created if two vertices are outside. Faces fully within the radius are simply copied into the patch. The right picture shows the adaptive intersection (green faces) of a sphere with the surface.

distance to the center of the sphere: $\|v_j - v_c\| \leq r$. We then add every vertex where this inequality holds to a list of vertices we visit in the next iteration. If all three vertices of the face f_i are within the radius, we simply add f_i to the patch. If some vertices are outside the radius, we cut the face f_i at the intersection points of the radius with the edges of the face and create new triangles with the intersection points. There can only be two cases: Either one or two vertices of f_i are outside the radius. See Figure 4 for an illustration. When the algorithm terminates, the result is a circular surface patch. As a final step, we calculate the face and vertex normals of all triangles in the patch. These will later be used to compute the intersection volume.

3.4. Volume Computation using a Triangulated Sphere

The volume of the ball neighborhood is approximated by intersecting a triangulated sphere with the triangulated shape. The surface is represented by the surface patch that was computed in the previous step. Using the triangles of the patch as well as all faces of the sphere which are behind the surface, the volume of the resulting polyhedron can be computed by the following formula:

$$V_{approx.} = \frac{1}{6} \sum_{i=0}^{N-1} a_i \cdot \hat{n}_i \tag{2}$$

where N is the number of triangles and $\hat{n}_i$ is the unnormalized normal of triangle $f_i = (a,b,c)$ calculated by $\hat{n}_i = (b-a) \times (c-a)$.

For numerical stability, we first translate the surface patch to the origin. Thus, the sphere is also centered at the origin and scaled by the current radius. Computing the actual intersection of the sphere and the patch is non-trivial and costly. We therefore approximate the intersection by finding all faces of the sphere which lie behind the surface. We check whether each vertex of the sphere is behind or in front of the surface by finding its nearest neighbor vertex on the surface patch and perform an inside/outside check based on the scalar product of both vertex normals. We accellerate this nearest neighbor lookup, using a k-D tree data structure with all patch vertices. To compute the final volume, we use Equation 2 and sum over all patch faces as well as all faces of the sphere behind the mesh surface.

The triangulated sphere is generated using two Loop subdivision iterations [Loo87] on an Icosahedron, which result in 162 faces. The sphere faces along the intersection with the mesh surface are further subdivided as illustrated in Figure 4. Experiments have shown that six subdivisions along the border result in negligible approximation error for the intersection boundary.

3.5. Radius Sampling

To get a suitable initial radius for the current vertex v, we use the average edge lengths in a 1-ring neighborhood around v. Let $s(v)$ denote the scale of a vertex v. The starting scale s_0 of v is then computed by

$$s_0(v) = \frac{\sum_{w \in N(v)} \|w - v\|}{|\,N(v)\,|} \tag{3}$$

where $N(v)$ is the set of all neighbors of v. This approach already produces a good starting radius for each vertex. In order to cope with regions where the edge lengths are extremely small or large, we smooth the initial radius afterwards. For this, we perform n multiple smoothing iterations and update the initial scale of each vertex v based on its surrounding vertices:

$$s_{i+1}(v) = s_i(v) + \sum_{w \in N(v)} \lambda \frac{s_i(w) - s_i(v)}{|\,N(v)\,|} \tag{4}$$

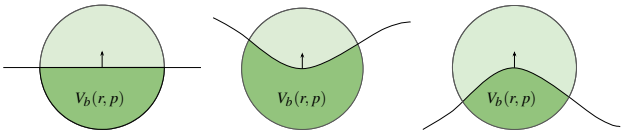
where λ is a smoothing factor. A larger value of λ increases the influence of neighboring vertices on v. If overall smoother results are desired, the starting radius can be multiplied with an additional factor > 1.

To sample the mean curvature at different radii $r_0 \ldots r_n = s_0 \ldots f_{max} \cdot s_0(v)$, we exponentially increase the radius by f_{inc} until it reaches a predefined maximum f_{max}. In each iteration i, we compute the mean curvature for a vertex v at radius $r_i = r_{i-1} \cdot f_{inc} = r_0 \cdot f_{inc}^i$. For all of our experiments we use $f_{inc} = 1.3$ and $f_{max} = 10$ (corresponding to $n = 8$) which results in a large enough sampling region to distinguish small features from noise while still being able to detect planar regions.

This sampling yields two dimensional data $D_v = \{(r_i, H(r_i, v)) \mid r_i \in \{r_0, \ldots, r_n\}\}$ for each vertex v. Because the sampling is discrete, we fit a function to D_v in order to make a decision for the final radius in continuous space. This fit, however, must be performed on the normalized curvature to not be influenced by the smoothing introduced through the radius of the ball neighborhood. Using Equation 1 we get the normalized curvature:

$$H_{norm}(r, V_b) = r \cdot H_{ball}(r, p) = \frac{8}{3} - \frac{4}{\pi r^3} V_b. \tag{5}$$

Equivalently to the unnormalized case, H_{norm} is zero when V_b is exactly half of the ball volume; for larger and smaller values we get $H_{norm} < 0$ and $H_{norm} > 0$ respectively, see Figure 5 for an illustration. This leads to scale invariant, normalized data $\hat{D}_v = \{(r_i, H_{norm}(r_i, v)) | r_i \in \{r_0, \ldots, r_n\}\}$ for each radius increase. We fit a cubic function in the least-squares sense to the data $\hat{D}_v$ for each vertex. In our experiments a quadratic fit often does not represent the data well enough while higher order polynomials cause



(a) $H_{norm} = 0.0$ (b) $H_{norm} < 0.0$ (c) $H_{norm} > 0.0$

Figure 5: The surface is either planar and has zero mean curvature or it is curved and thus has a negative or positive mean curvature. $V_b(r,p)$ is the volume of the ball neighborhood with radius r at point p.

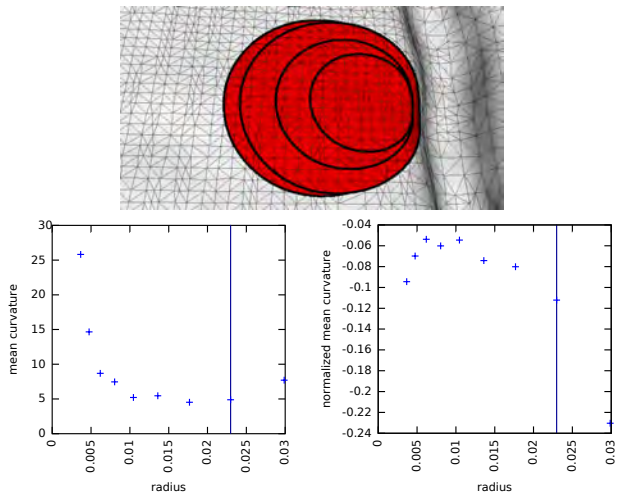


Figure 6: The final surface patch for four vertices which were classified to be in a planar region. The graphs show the mean curvature (left) and the normalized curvature (right) for a single vertex. Here, the curvature at radius 0.03 is treated as an outlier.

problems because of overfitting. Thus, we check if the error of the fit is small (below 2%). Otherwise we try to optimize it by iteratively removing data samples starting with the sample at the largest radius.

3.6. Radius Selection

The final radius for each vertex is chosen in different ways depending on the local surface properties. We first decide if the vertex lies on a planar or non-planar region. In the latter case, we analyze the extrema of the fitted polynomial which can have up to two extrema in the given interval. The following cases are considered.

Planar regions: If a surface region is planar the normalized curvature values are close or equal to zero. We use the average of the normalized curvatures of the current vertex and check whether it is below the planar threshold of $t_p = 0.2$, which is a good choice for most meshes. If the surface of the triangle mesh is particularly noisy, t_p can be increased, which will result in more smoothing. Whenever a planar region is detected, we select the largest radius at which the normalized curvature is below the threshold, see Figure 6.

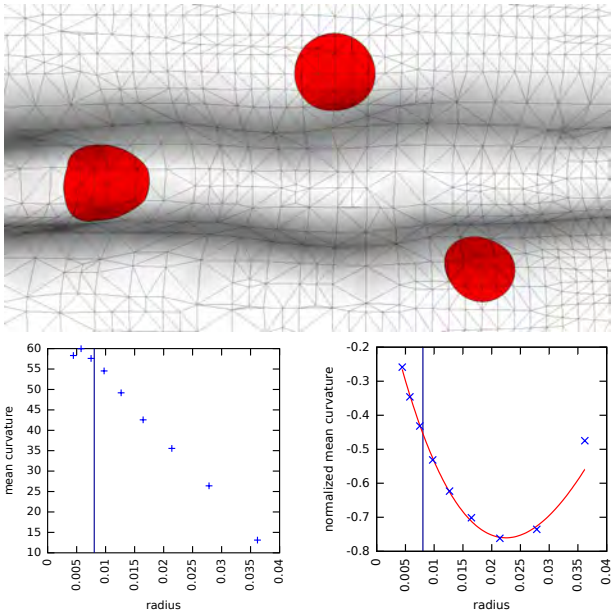


Figure 7: Example vertices around a valley which lie in a non-planar region. The extreme point of the fitted polynomial is used to select the final radius by interpolating between the smallest radius and the radius at the extreme point based on the edge smoothing factor f_{es}.

Single extrema: One extreme point within the sampled radius range is an indicator for a region which starts off with high curvature. Because the smoothing performed by the integral invariant based mean curvature computation, we cannot simply choose the radius at the extreme point as the final radius because the closer we choose a radius towards the extreme point, the smaller the mean curvature gets (Figure 7). We therefore introduced a global "edge smoothing" factor f_{es}. This factor is used to control the final radius in such a case where $f_{es} = 0.0$ results in the smallest radius r_0 and $f_{es} = 1.0$ corresponds to the radius at the extreme point.

Two extrema: This indicates geometric detail at the first extrema and another significant enough surface feature at the second. The first extrema is used to compute the final radius to preserve surface detail. We select this final radius by applying the edge smoothing factor as described in the previous paragraph.

No extrema: This can be observed when the polynomial either has no extreme points or when both extreme points are outside of the radius range. E.g., consider an edge where H_{norm} converges towards $-\frac{4}{3}$ or $\frac{4}{3}$, for a 90°or 270°edge respectively. Note that H_{norm} is not constant because edges in meshes resulting from surface reconstruction algorithms are typically not sharp in contrast to meshes used in Computer Aided Design. If there is a saddle point, we use it as a reference and interpolate between it and the smallest radius based on f_{es}. Otherwise we use the middle radius.

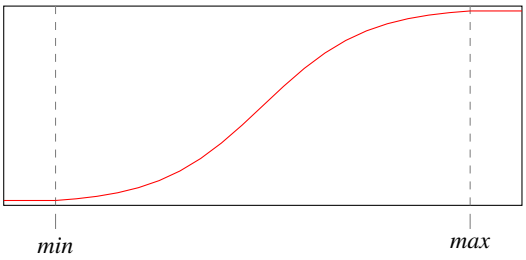


Figure 8: To create a density field useful for mesh simplification, the curvature values are remapped using the density function $d(x)$.

3.7. Density Field Computation

For mesh simplification the curvature field is remapped to a range more suitable for this application. We need to define the importance of a vertex as a single positive number. The sign of the curvature is not important, so we only consider the absolute mean curvature value. The curvature values are remapped (Figure 8) such that the density values are constant up a minimum and then increased up to a maximum using a sigmoid function.

The exact definition of the mapping we used is as follows:

$$d(x) = \begin{cases} d_{min} & \text{for } x \leq min \\ d_{scale} \cdot \left(\frac{1}{1+e^{-4\hat{x}}} - \frac{1}{1+e^{4}} \right) + d_{min} & \text{for } min < x \leq max \\ d_{max} & \text{for } x > max \end{cases}$$

with $\hat{x} = 2\frac{x-min}{max-min} - 1$ and $d_{scale} = \frac{d_{max}-d_{min}}{\frac{1}{1+e^{-4}} - \frac{1}{1+e^{4}}}$.

d_{min} and d_{max} correspond to the minimum and maximum density values and should be chosen according to the mesh simplification algorithm that is used. A final smoothing on the density field helps to gradually change the triangle sizes from low to higher density regions. In our experiments, however, we found that this smoothing is not absolutely necessary and depends on the application.

4. Results

We evaluate our algorithm on several real-world datasets which were created using the open source image-based reconstruction pipeline of [FLM*15]. Mesh simplification was performed using the *Remesher* tool from [FAKG10] together with our density fields. The mesh properties and the runtimes of our implementation are summarized in Table 1.

In Figure 9 we show the *Bronze Akt* dataset. The statue has a very rough surface and the reconstruction produced different levels of geometric noise, resulting from pictures taken from various distances to the model. In order to extract curvatures in a meaningful way, the radius for the ball neighborhood must be chosen such that small scale features which can be found in the platform region are preserved while noise on the overall model is smoothed. Choosing a single radius that preserves small-scale features results in a noisy curvature estimate on the rest of the surface. Using a larger radius removes most of the noise but also smoothes away important features. Our algorithm preserves the features while also removing noise by adaptively varying the radius per vertex.

Figure 9: *Bronze Akt*: Our curvature field (left) as well as three closeups of the platform region. Our result (top), curvatures computed using a fixed small radius (middle) and fixed larger radius (bottom).

Simplification of multi-scale geometry is a direct application of using the density field described in Section 3.7. Figure 10 shows a model of the *Fountain* dataset, which has many small-scale features at the statue. We simplified this mesh to 4% of the original number of vertices to achieve a size that is manageable for real-time or mobile applications. Using a single-scale density field results in a direct loss of small-scale features. Our multi-scale density field guides the simplification to preserve important features such as the hand of the statue.

The effects of using a single-scale radius can be also be seen in the *Goethe-Fountain* dataset, Figure 11. This dataset is another example with significant scale differences. A globally selected radius cannot cover all of the geometry features that are captured in different resolutions. During simplification we reduced the amount of vertices to 3%. The details on the fountain head are lost or edges are smoothed too much when a single-scale curvature is used. Our multi-scale curvature estimate leads to a significantly better simplification result which still retains most of the geometric detail from the original mesh.

The *Owl* dataset does not contain scale differences. Here we show that our algorithm gracefully degenerates to estimating curvature at a single scale.

Figure 10: *Fountain*: Simplification without density field, single-scale curvature field (top), our multi-scale curvature field (middle), simplification using the single-scale curvature field (bottom left) and our simplification (bottom right). We simplified the original mesh to 4% in both cases.

Dataset Name	**# Vertices**	**Runtime**
Owl	280 752	~11 min
Fountain	2 068 619	~65 min
Bronze Akt	3 666 075	~111 min
Goethe-Fountain	4 675 851	~159 min

Table 1: Runtime of our algorithm for estimating multi-scale curvatures. The total computation time increases linearly with the number of vertices.

Figure 11: *Goethe-Fountain*: overview shot (top), curvature fields (left column) and corresponding simplifications (right column): single-scale curvature large radius (top) and small radius (middle), our multi-scale result (bottom)

Figure 12: *Owl*: Our multi-scale curvature (left) is visually indistinguishable from the curvature field computed using a carefully hand-selected scale. The absolute differences (heavily amplified) are shown on the right.

5. Conclusion

In this paper, we revisited the problem of robust curvature estimation with a focus on multi-scale triangle meshes. Compared to previous approaches which ignore the presence of varying feature and noise scales, our algorithm is designed to take the local scale of the vertices into account. Our main contribution is the automatic computation of a mean curvature field, meaning that the radius of the ball is chosen for each vertex independently. We reviewed the performance and usefulness of our approach on several multi-scale datasets with respect to robust curvature estimation as well as adaptive mesh simplification using a density field as guidance. Even though our algorithm performs favorably on multi-scale data, we showed that this is not a requirement and that our algorithm also works on single-scale input data (Figure 12). In theory, the algorithm takes five optional parameters as input. In practice however, only the planar threshold t_p and the initial radius factor $f_{initial}$ may be increased to achieve more smoothing if desired.

We see future work in optimizing our algorithm to reduce its runtime. In regions with very small triangles, many duplicate computations are performed when moving from one vertex to a neighboring. In a planar region that is highly tessellated, many CPU cycles are therefore wasted computing very similar curvatures. Furthermore, with a robust multi-scale curvature field at hand, other applications such as adaptive mesh smoothing seem promising.

References

[APM15] ANDREADIS A., PAPAIOANNOU G., MAVRIDIS P.: A parametric space approach to the computation of multi-scale geometric features. *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications* (2015). 1, 2

[ASWL11] AN Y., SHAO C., WANG X., LI Z.: Estimating principal curvatures and principal directions from discrete surfaces using discrete curve model. *Journal of Information & Computational Science 8*, 2 (2011), 296–311. 2

[CP05] CAZALS F., POUGET M.: Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design 22* (2005), 121–146. 2

[CSM03] COHEN-STEINER D., MORVAN J.-M.: Restricted delaunay triangulations and normal cycle. In *Proceedings of the nineteenth annual symposium on Computational geometry* (2003), ACM, pp. 312–321. 2

[FAKG10] FUHRMANN S., ACKERMANN J., KALBE T., GOESELE M.: Direct Resampling for Isotropic Surface Remeshing. In *Proceedings of Vision, Modeling and Visualization (VMV)* (2010), VMV, pp. 9–16. 6

[FLG14] FUHRMANN S., LANGGUTH F., GOESELE M.: MVE - A Multi-View Reconstruction Environment. In *Proceedings of the Eurographics Workshop on Graphics and Cultural Heritage (GCH)* (2014). 1

[FLM*15] FUHRMANN S., LANGGUTH F., MOEHRLE N., WAECHTER M., GOESELE M.: MVE – An Image-Based Reconstruction Environment. *Computer and Graphics* (2015). 6

[LHF09] LAI Y.-K., HU S.-M., FANG T.: Robust principal curvatures using feature adapted integral invariants. In *2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling* (2009), ACM, pp. 325–330. 2, 3

[Loo87] LOOP C.: *Smooth subdivision surfaces based on triangles*. Master's thesis, University of Utah, 1987. 4

[MHYS04] MANAY S., HONG B.-W., YEZZI A. J., SOATTO S.: *Integral invariant signatures*. Springer, 2004. 3

[MOG09] MÉRIGOT Q., OVSJANIKOV M., GUIBAS L.: Robust voronoi-based curvature and feature estimation, 2009 siam. In *ACM Joint Conference on Geometric and Physical Modeling* (2009). 2

[PWY*07] POTTMANN H., WALLNER J., YANG Y.-L., LAI Y.-K., HU S.-M.: Principal curvatures from the integral invariant viewpoint. *Computer Aided Geometric Design 24*, 8 (2007), 428–442. 2, 3

[SHBK10] SEIBERT H., HILDENBRAND D., BECKER M., KUIJPER A.: Estimation of curvatures in point sets based on geometric algebra. In *VISAPP (1)* (2010), pp. 12–19. 1, 2

[YLHP06] YANG Y.-L., LAI Y.-K., HU S.-M., POTTMANN H.: Robust principal curvatures on multiple scales. In *Symposium on Geometry Processing* (2006), pp. 223–226. 1, 2

[YZ13] YANG X., ZHENG J.: Curvature tensor computation by piecewise surface interpolation. *Computer-Aided Design 45*, 12 (2013), 1639–1650. 2

Computers & Graphics 54 (2016) 94–103

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Special Issue on CAD/Graphics 2015

Automatic reconstruction of parametric building models from indoor point clouds

Sebastian Ochmann *, Richard Vock, Raoul Wessel, Reinhard Klein

Institute of Computer Science II, University of Bonn, Germany

ARTICLE INFO

Article history:
Received 7 April 2015
Received in revised form
11 July 2015
Accepted 12 July 2015
Available online 22 July 2015

Keywords:
Indoor scene reconstruction
Point cloud processing
Parametric models

ABSTRACT

We present an automatic approach for the reconstruction of parametric 3D building models from indoor point clouds. While recently developed methods in this domain focus on mere local surface reconstructions which enable e.g. efficient visualization, our approach aims for a volumetric, parametric building model that additionally incorporates contextual information such as global wall connectivity. In contrast to pure surface reconstructions, our representation thereby allows more comprehensive use: first, it enables efficient high-level editing operations in terms of e.g. wall removal or room reshaping which always result in a topologically consistent representation. Second, it enables easy taking of measurements like e.g. determining wall thickness or room areas. These properties render our reconstruction method especially beneficial to architects or engineers for planning renovation or retrofitting. Following the idea of previous approaches, the reconstruction task is cast as a labeling problem which is solved by an energy minimization. This global optimization approach allows for the reconstruction of wall elements shared between rooms while simultaneously maintaining plausible connectivity between all wall elements. An automatic prior segmentation of the point clouds into rooms and outside area filters large-scale outliers and yields priors for the definition of labeling costs for the energy minimization. The reconstructed model is further enriched by detected doors and windows. We demonstrate the applicability and reconstruction power of our new approach on a variety of complex real-world datasets requiring little or no parameter adjustment.

1. Introduction

Digital 3D building models are increasingly used for diverse tasks in architecture and design such as construction planning, visualization, navigation, simulation, facility management, renovation, and retrofitting. Especially for legacy buildings, suitable models are usually not available from the initial planning. Point cloud measurements are often used as a starting point for generating 3D models in architectural software. But despite fast scanning devices and modern software, the generation of models from scratch still are largely manual and time-consuming tasks which make automatic reconstruction methods highly desirable.

Reconstruction of indoor environments poses specific challenges due to complex room layouts, clutter and occlusions. Furthermore, planning and maintenance tasks often require models which give deeper insight into a building's structure on the level of building elements such as walls, and their relations like wall connectivity. This enables high-level editing for prototyping planned changes and simulations requiring information like room neighborhood or wall thickness. While previous reconstruction methods are able to faithfully recover partially observed surfaces from indoor point clouds and generate accurate boundary representations in the form of mesh models, a plausible decomposition into parametric, globally interrelated, volumetric building elements yet remained an open challenge. Existing approaches either represent walls, floors and ceilings as sets of unconnected planar structures detected in the point cloud [13,1,15,21,7] (Fig. 1(a)), or as collections of closed 3D boundaries of either the whole building [12], or separate rooms [4,19,18,8] (Fig. 1(b)). While the method in [20] reconstructs volumetric walls, their thickness is defined manually instead of being estimated from the input data.

To overcome the limitations of previous approaches, we propose a novel reconstruction method in which the representation of buildings using parametric, interrelated, volumetric elements (Fig. 1(c)) is an integral component. Our approach automatically reconstructs walls between adjacent rooms from opposite wall surfaces observed in the input data while simultaneously taking into account globally plausible connectivity of all elements. Together with a faithful estimation of wall thickness, the result is a high-level editable model of volumetric wall elements. The reconstruction is formulated as an energy minimization problem which simultaneously optimizes costs for assigning rooms to areal

* Corresponding author.
E-mail address: ochmann@cs.uni-bonn.de (S. Ochmann).

http://dx.doi.org/10.1016/j.cag.2015.07.008

regions of the building, and costs for separating adjacent rooms by volumetric wall elements. In contrast to previous approaches, this has the advantage that reasonable binary costs for the assignment of pairs of room labels to adjacent areal regions of the building – and thus the selection of suitable wall elements – is directly incorporated into the global optimization. To make our method robust against large-scale clutter outside the building, outliers are automatically filtered prior to reconstruction. Finally, doors and windows are detected, classified and assigned to the respective wall elements to further enrich the model. Our evaluation using various real-world indoor scans shows that our method rapidly provides models which can be used for e.g. planning of retrofitting, especially since our method requires little or no parameter adjustment.

Applications: The distinguishing feature of our approach is that it directly captures important properties and relations of building elements. Since architectural Building Information Modeling (BIM) formats (e.g. Industry Foundation Classes, IFC) are based on similar relational paradigms, exporting our results to architectural software is straight forward. This enables a whole range of processing and analysis tasks in industry-standard software. We exemplify some applications for e.g. planning of retrofitting in Fig. 2 which can directly be implemented using our results: since the incidence and adjacency relations of walls and rooms are inherently known, selecting e.g. all walls enclosing a room or manipulating whole walls while maintaining overall room topology is easily possible (Fig. 2(a)). This allows for quick, high-level prototyping of changes on the level of semantically meaningful construction element groups. The available information also enables more complex queries for e.g. the subset of wall elements that are simultaneously incident to two adjacent rooms (Fig. 2(b)). Together with directly available properties like wall thickness, openings, room and wall areas, this provides important information for performing acoustic or thermal simulations. The global connectivity information further allows us to perform pathfinding in the whole building story (Fig. 2(c)) for e.g. simulating and optimizing escape routes.

2. Related work

Okorn et al. [13] generate 2D floor plans from 3D point clouds. A histogram of the vertical positions of all measured points is built. Peaks in this histogram are considered to be large horizontal planar structures (i.e. floor and ceiling surfaces). After removing points belonging to the detected horizontal structures, a line fitting on the remaining points is performed. The resulting line segments constituting the floor plan are not connected and do not provide e.g. closed boundaries of rooms. Budroni and Boehm [4] extract planar structures for floors, ceilings and walls by conducting a plane sweep. Using a piecewise linear partitioning of the x–y-plane, they classify cells of this partitioning as inside and outside by determining the occupancy of the cells by measured points and considering densely occupied cells as inside. The result is a 2.5D extrusion of the determined room boundary. In the approach by Sanchez and Zakhor [15], points are classified into floor, ceiling, wall, and remaining points using the point normal orientations. For floor, ceiling and wall points, planar patches are fitted and their extents are estimated using alpha shapes. Parametric staircase models are fitted to the set of remaining points. The resulting mesh models consist of unconnected planar surfaces. Monszpart et al. [7] propose a method for extracting planar structures in point clouds which follow regularity constraints. Their optimization approach balances data fitting and simplicity of the resulting arrangement of planes. A method for generating visually appealing indoor models is proposed by Xiao and Furukawa [20]. An inverse-CSG approach is used for reconstructing the building's geometry by detecting planar structures and then fitting cuboid primitives. These primitives are combined using CSG operations; the quality of the resulting model is tested using an energy functional. Finally, the resulting mesh model is textured using captured images. A drawback is that the building needs to be sufficiently well approximated by the used cuboid primitives. Adan and Huber [1] reconstruct planar floor, ceiling, and wall surfaces from multi-story point clouds by first detecting the modes of a histogram of point

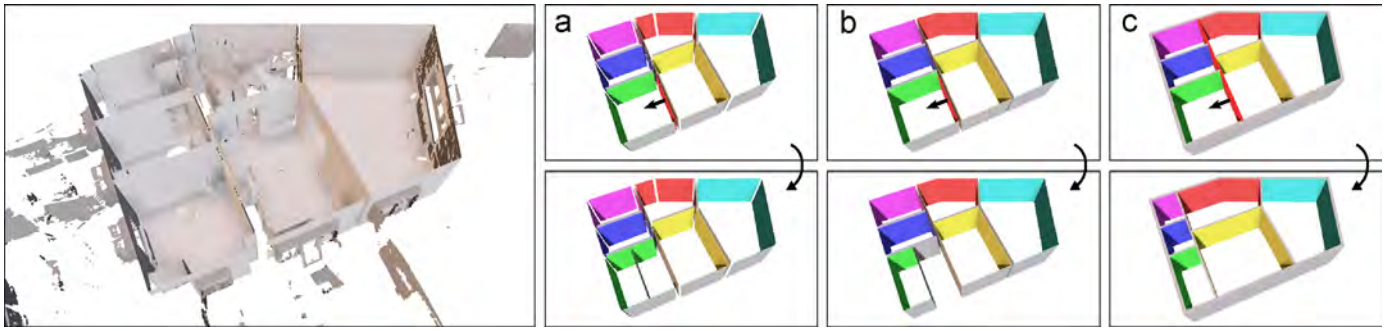


Fig. 1. Schematic of editing capabilities of different kinds of reconstructions. The input point cloud is shown on the left. The remaining columns exemplify editing operations, i.e. elements are moved in the directions of the arrows. Surface representations without (column (a)) or with (column (b)) connectivity information do not allow intuitive editing on the level of wall elements. Our reconstruction (column (c)) maintains room topology and global wall connectivity.

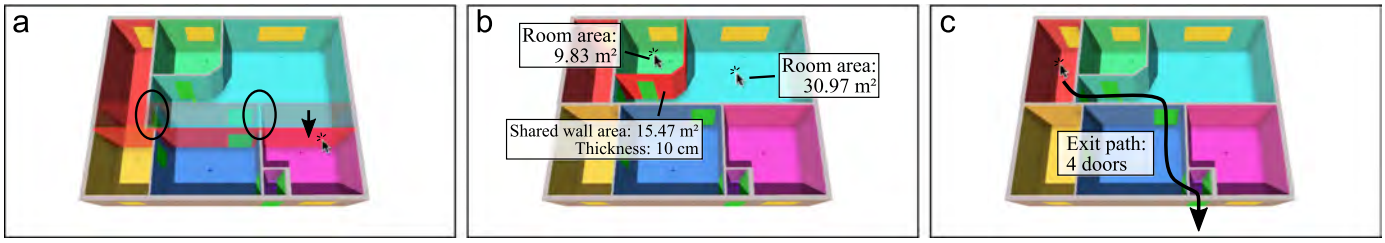


Fig. 2. Example operations which are easily implemented using our results. (a) Relations between walls and rooms enable editing while maintaining room topology. Note how incident walls are adjusted automatically. (b) Automatic determination of wall elements shared between rooms together with automatic measurements enable e.g. acoustic or thermal simulations. (c) Global connectivity enables pathfinding for e.g. simulation and optimization of escape routes.

height values to find horizontal planes, and then detecting vertical planes by means of Hough transform. They recover occluded parts of reconstructed surfaces and perform an opening detection by means of Support Vector Machine (SVM) learning. Xiong et al. [21] extend this approach by classifying detected planar patches as floor, ceiling, wall or clutter using a stacked learning approach, also taking into account contextual information of neighboring patches. Mura et al. [8] reconstruct indoor scenes with arbitrary wall orientations by building a 3D Delaunay tetrahedralization of the input dataset and partitioning inside and outside using a diffusion process governed by affinities of tetrahedron pairs. A binary space partitioning is also done by Oesau et al. [12] by first splitting the input dataset horizontally at height levels of high point densities and then constructing 2D arrangements of projections of detected wall surfaces. The space partitioning into inside and outside is performed by means of Graph-Cut. Other approaches not only perform binary space partitioning but label different rooms: Turner and Zakhor [19] generate 2.5D watertight meshes by first computing an inside/outside labeling of a triangulation of wall points and a subsequent partitioning into separate rooms using a Graph-Cut approach. This method is further developed by Turner et al. in [18], improving the texture mapping capabilities of the algorithm. The results are well-regularized, watertight, textured mesh models. Mura et al. [9] first extract candidate wall elements while taking into account possibly occluded parts of the surfaces to determine the real wall heights for filtering out invalid candidates. After constructing a 2D line arrangement, they use a diffusion embedding to establish a global affinity measure between faces of the arrangement, and determine clusters of faces constituting rooms. The result is a labeled boundary representation of the building's rooms. Many of these methods build upon a spatial partitioning defined by detected wall surfaces and a subsequent classification of regions of this partitioning. Although the resulting models have applications like visualization, navigation or energy monitoring [17], they do not realize a reconstruction of volumetric, interconnected building elements like walls.

3. Approach

The starting point of our approach is a registered point cloud of one building story consisting of multiple indoor scans including scanner positions. Registration is usually done using the scanner software and is outside the scope of this paper. The unit of measurement and up direction are assumed to be known. Surface normals for each point are estimated.

We argue that the wall structure of most building stories can be represented as a piecewise-linear, planar graph in which edges represent wall elements and vertices are locations where walls are incident (Fig. 3(e)). Wall thickness is a scalar edge attribute. Conversely, faces of this graph represent the spatial room layout. There obviously exists a duality between the story's room layout and its wall constellation, i.e. one representation can directly be derived from the other. The main idea of our approach is that – while both representations are essentially equally hard to reconstruct – we can derive important hints (priors) for the room layout from indoor point cloud scans since they are a sampling of the inner surfaces of room volumes. It is therefore meaningful to base our reconstruction on the derivation of a suitable room layout from which the constellation of walls is immediately obtained due to the duality.

We extract priors for the room layout as follows: assuming that each room was scanned from one position (or few positions), separate scans yield a coarse segmentation of the point cloud into separate rooms (Fig. 3(a)). We improve this segmentation using a diffusion process which eliminates most overlapping regions between scans (Fig. 3(b)) and automatically filters out clutter outside of the building. As further described below, the determination of a suitable room layout is then formulated as a labeling problem of the regions of a suitable partitioning of the horizontal plane (using labels for different rooms and the outside area). This directly follows the aforementioned duality principle: after determining a suitable labeling, connected components of identically labeled cells are rooms, and edges separating differently labeled regions are wall elements.

Since our goal is to extract a piecewise-linear graph of walls, we construct a partitioning based on potential wall surfaces: we first detect vertical planes as candidates for wall surfaces and project them to the horizontal plane (Fig. 3(c)). Similar to previous approaches [9,11,12] we then construct an arrangement of (infinitely long) lines from the set of possible wall surfaces (Fig. 3(d)). In contrast to previous approaches, edges of this arrangement represent wall *centerlines* instead of wall *surfaces*. Furthermore, arrangement lines are not only constructed from *single* wall surfaces but also from *pairs* of parallel surfaces which yield candidates for walls separating adjacent rooms. This subtle but crucial difference allows us to go beyond the reconstruction of *separate* room volumes as done in previous works (Fig. 1(b)) by enabling the algorithm to reconstruct room-separating wall elements directly. In order to guide the selection of adequate wall elements, we retain the information from which supporting measured points each edge originates. This yields wall selection priors encouraging the reconstruction of wall elements which were constructed from surfaces belonging to the *same* pair of rooms that the wall separates.

The determination of a globally plausible labeling is then formulated as an energy minimization problem. This allows us to incorporate room layout priors and wall selection priors as unary and binary costs into one optimization. After an optimal labeling has been determined, only retaining edges separating differently labeled regions are the sought wall structures (Fig. 3(e)). Extruding walls according to estimated room heights and a detection and classification of openings yields the final parametric model (Fig. 3(f)).

4. Point cloud segmentation

To obtain priors for the localization of rooms in subsequent steps, each point of the input point cloud is automatically assigned a label for a room or the outside area. Our approach is based on the method by Ochmann et al. [10] which we will briefly summarize before describing our modifications: the original method assumes at least one scan within each room; multiple scans per room are merged manually such that a one-to-one mapping between (merged) scans and rooms is obtained. The initial assignment of each point to one of the (merged) scans (Fig. 3(a)) provides a coarse segmentation of the point cloud into rooms. However, openings such as open doors lead to severe overlaps between scans, causing large areas of the point cloud to contain a mix of differently labeled points. To obtain a point labeling that roughly corresponds to the building's room layout and is homogeneous within each room (Fig. 3(b)), an automatic labeling refinement is performed. The process is based on the assumption that most points that are *visible* from the position of a point p are already labeled correctly. By determining which points are visible from the position of p and averaging the observed labels, a new (soft) labeling of p is obtained. After iterating this procedure, the label with the highest confidence is assigned to p. This process can be interpreted as a diffusion of point labels between points governed by mutual visibility. In practice, a stochastic ray casting from the

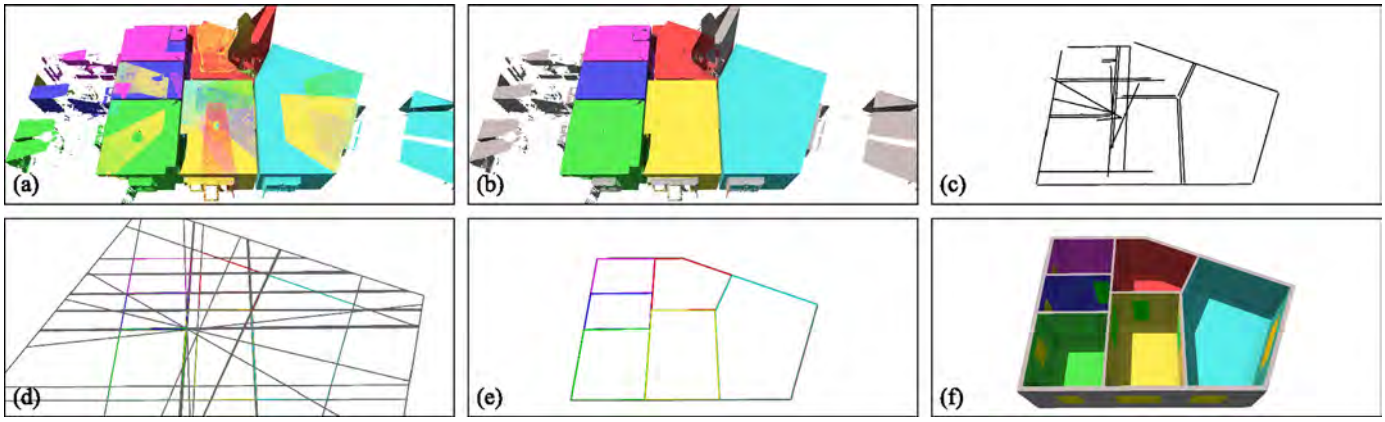


Fig. 3. Overview of our approach (see also Section 3). (a) Input point cloud; assignment of points to scans shown in different colors. (b) Refined assignment after automatic segmentation. (c) Detected vertical planes transferred to the horizontal plane. (d) Candidates for walls are derived from single and pairs of projected planes. Intersecting their centerlines yields a planar graph whose faces are subsequently assigned labels for rooms or outside area. (e) Only edges separating differently labeled faces are retained. (f) The final model with detected and classified wall openings, e.g. doors (green) and windows (yellow). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

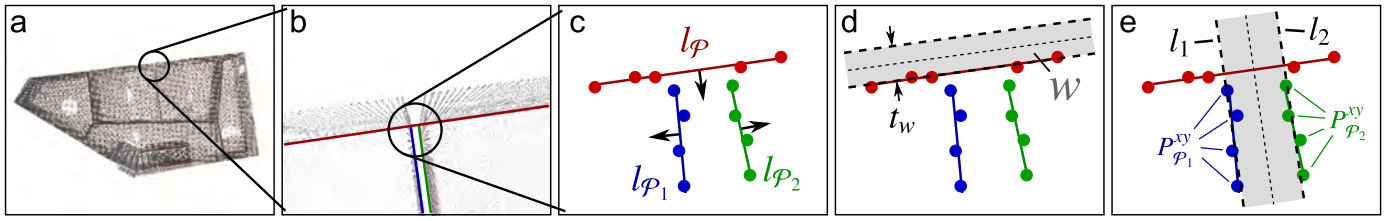


Fig. 4. Wall candidate generation. (a) and (b) Detected vertical planes in the 3D point cloud are projected into the horizontal plane. (c) Different wall surface lines including the respective (projected) support points and surface normals. (d) For each single wall surface, an infinitely long wall candidate w for a wall separating a room from outside area is generated. In this case, the thickness t_w is user-specified. (e) For each pair of approximately parallel wall surfaces, a candidate for separating adjacent rooms is generated. In this case, wall thickness is estimated from the data.

position of p into the hemisphere around the normal of p is performed.

We extend this method in two ways: first, we automatically filter out clutter outside of the building which is often caused by windows or mirrors. We argue that for a point p that is part of clutter outside of the building, most rays cast from p into the hemisphere around the normal of p do not hit any interior wall surfaces. In this case we assign a high value for an additional outside label to p. This modification proves to be highly effective in our experiments as demonstrated in Fig. 3(b) (gray points have been assigned the outside label). Second, we do not require that multiple scans per room are merged manually. Instead, we run the reconstruction using all scans as separate labels. In case of multiple scans in a room, this leads to implausible walls within rooms which are subsequently removed as described in Section 7.

5. Generation of wall candidates

Candidates for wall elements are derived from vertical surfaces observed in the scans. They constitute possible locations of walls for the optimization in Section 6. Since wall heights and lengths are not regarded in this step, the following 2D representation is used: each wall candidate $w=(t_w, n_w, d_w)$ is defined by a thickness $t_w \in \mathbb{R}^{\geqslant 0}$ and an infinite centerline in the horizontal plane given in Hesse normal form $\langle n_w, x\rangle - d_w = 0$. Wall heights and lengths will be determined later.

In a first step, planes in the 3D point cloud are detected using a RANSAC implementation by Schnabel et al. [16]. Nearly vertical planes ($\pm 1^\circ$) with a sufficiently large approximate area ($\geqslant$1.5 m^2) are considered as potential wall surfaces. For a plane $\mathcal{P}$ fulfilling these constraints, let $n_{\mathcal{P}} \in \mathbb{R}^3$ be the plane normal and $P_{\mathcal{P}}$ the set of measured points supporting $\mathcal{P}$. Each extracted plane $\mathcal{P}$ is transferred to the horizontal plane as a *wall surface line* $l_{\mathcal{P}}$ defined by $\langle n_{l_{\mathcal{P}}}, x\rangle - d_{l_{\mathcal{P}}} = 0$. A schematic example for the extraction of wall surface lines is shown in Fig. 4(a)–(c). The normal $n_{l_{\mathcal{P}}}$ is approximated by the projection of $n_{\mathcal{P}}$ into the horizontal plane,

$$n_{l_{\mathcal{P}}} := \frac{((n_{\mathcal{P}})_x, (n_{\mathcal{P}})_y)}{\|((n_{\mathcal{P}})_x, (n_{\mathcal{P}})_y)\|_2}.$$

The distance to the origin $d_{l_{\mathcal{P}}}$ is determined by least squares fitting to the set $P^{xy}_{\mathcal{P}}$ of support points projected to the horizontal plane using the fixed normal $n_{l_{\mathcal{P}}}$ such that $\sum_{p \in P^{xy}_{\mathcal{P}}} (\langle n_{l_{\mathcal{P}}}, p\rangle - d_{l_{\mathcal{P}}})^2$ is minimized. From the wall surface lines, we then generate two kinds of wall candidates as we do not know at this point which types of candidates will yield a globally plausible reconstruction:

Outside walls: For each *single* wall surface line $l_{\mathcal{P}}$, we construct a candidate for a wall separating a room from the outside area (Fig. 4(d)). Since the real wall thickness cannot be determined automatically from a single surface, a user-specified thickness is used (in our experiments, $t_w = 20$ cm). The centerline of the candidate is constructed such that the side of the wall candidate that points towards the inside of the room is identical to $l_{\mathcal{P}}$, i.e. the centerline is defined by $\langle n_{l_{\mathcal{P}}}, x\rangle - d_w = -t_w/2$.

Room-separating walls: To generate candidates for walls separating adjacent rooms, each *pair* of wall surface lines fulfilling certain constraints is considered as two opposite surfaces of a wall separating adjacent rooms (Fig. 4(e)). Let $l_{\mathcal{P}_1}$ and $l_{\mathcal{P}_2}$ be two wall surface lines that are approximately parallel ($\pm 1^\circ$) and have opposing normal orientations. To prune invalid pairs, a coarse check is performed whether the projected support pointsets of the originating planes $P^{xy}_{\mathcal{P}_1}, P^{xy}_{\mathcal{P}_2}$ (partially) overlap. To this end, the support pointsets are projected onto the respective opposite line. If support points are present near the projected points, their

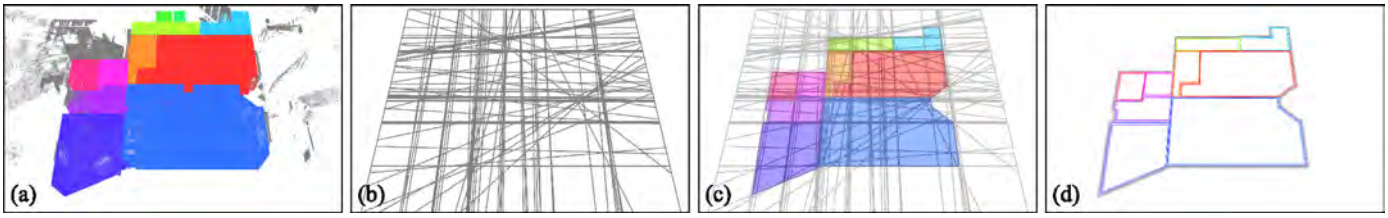


Fig. 5. Determination of suitable wall candidate segments. (a) Input point cloud after segmentation. (b) Intersecting all wall candidate centerlines yields a planar graph. We determine an assignment of all faces to rooms or outside area such that connected components of identically labeled faces are rooms and edges between differently labeled faces are wall elements. (c) Resulting labeling of faces after optimization; colors indicate room labels. (d) Retaining only edges separating differently labeled faces yields a subgraph representing the sought wall elements and their connectivity. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

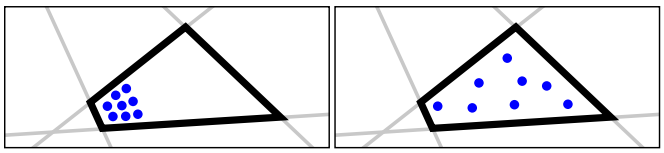


Fig. 6. Considering only the number of projected points within a face for unary costs does not take into account their spatial distribution.

support is considered overlapping. For each pair fulfilling these constraints, a wall candidate is generated by fitting to $l_{\mathcal{P}_1}$ and $l_{\mathcal{P}_2}$ simultaneously: The candidate's normal n_w is first determined as the average of the normals n_1, n_2 of $l_{\mathcal{P}_1}, l_{\mathcal{P}_2}$, weighted with the cardinality of the support pointsets,

$$n_w := \frac{|P^{xy}_{\mathcal{P}_1}|\, n_1 + |P^{xy}_{\mathcal{P}_2}|\,(-n_2)}{\|(|P^{xy}_{\mathcal{P}_1}|\, n_1 + |P^{xy}_{\mathcal{P}_2}|\,(-n_2))\|_2}.$$

Using the common normal n_w, two parallel lines $l_i, i \in \{1,2\}$ defined by $\langle n_w, x\rangle - d_i = 0$ are fitted to the respective support pointsets such that $\sum_{p \in P^{xy}_{\mathcal{P}_i}} (\langle n_w, p\rangle - d_i)^2$ is minimized. The centerline of the wall candidate is constructed midway between the parallel lines, $\langle n_w, x\rangle - \frac{1}{2}(d_1 + d_2) = 0$, and the candidate's thickness is defined as the distance between them, $t_w = |d_1 - d_2|$. Candidates with a thickness above a threshold are discarded (in our experiments, $t_w > 60$ cm).

6. Determination of an optimal room and wall layout

From the infinitely long wall candidates, we determine a set of wall segments which yields a plausible reconstruction of the building's walls. To this end, we consider the intersection of all wall candidate centerlines in the horizontal plane which yields a planar graph $W' = (V', E')$ (Fig. 5(b)). Faces of W' are regions of the building's layout (i.e. parts of rooms or outside area), edges E' are segments of possible walls, and vertices V' are possible locations where walls are incident. We follow the intuition that walls separate different regions, i.e. adjacent rooms, or rooms and the outside world. Consequently, a classification of the faces of W' implies locations of walls in the following sense: connected components of identically labeled faces are rooms (or outside areas), and edges between differently labeled faces are walls. Fig. 5(c) shows an example for a face labeling from which connected wall elements as shown in Fig. 5(d) are extracted. Wall thickness of an edge e is set to the thickness of the wall candidate from which e originates.

We formulate the face classification as a labeling problem which is solved using an energy minimization approach. The target functional has two terms: unary costs for the assignment of labels to faces of W' and binary costs for the assignment of label pairs to adjacent faces. Unary costs provide hints where rooms (or outside areas) are located and binary costs guide the selection of adequate edges for separating differently labeled faces. In particular, if two adjacent rooms share a common wall, a wall candidate constructed from wall surfaces of *these* rooms should separate them. We will now formalize the problem. Let $W = (V, E)$ be the dual graph of W' and let $\{l_1, \ldots, l_k, l_o\}$ be the set of labels where $l_i, i \in \{1, \ldots, k\}$ are labels for each scan and l_o is the outside label. For clarity, we assume for now that each room was scanned from exactly one position and thus k equals the number of rooms; the more general case of multiple scans per room will be discussed later. Given a unary cost function $U_v(l_v)$ yielding the cost for assigning label l_v to a vertex $v \in V$, and a binary cost function $B_{v,w}(l_v, l_w)$ yielding the cost for assigning the (unordered) pair of labels l_v, l_w to $v, w \in V$, we minimize the total cost for a labeling l, i.e.

$$E(l) = \sum_{v \in V} U_v(l_v) + \sum_{(v,w) \in E} B_{v,w}(l_v, l_w) \to \min. \quad (1)$$

Applying the minimization algorithm to the dual graph W of W' allows us to determine a labeling of the faces of W' by finding an optimal labeling of the vertices of W. The problem stated in Eq. (1) is solved using the algorithm by Boykov et al. [3,6,2]. We now define unary and binary cost functions for label assignments. In the following, the notation for *label vectors*

$$\mathcal{L}(\cdot) = (c_1, \ldots, c_k, c_o), \quad \forall\, i : c_i \geqslant 0, \quad \|\mathcal{L}(\cdot)\|_1 = 1$$

will be used for soft label assignments to different entities, e.g. points, faces and edges. The coefficient c_i of $\mathcal{L}(\cdot)$ corresponding to label l_i will be denoted $\mathcal{L}_i(\cdot)$. As a shorthand, let $\mathcal{I}_i$ denote a hard label vector with $c_i = 1$, and let $\mathcal{I}_{ij} := \frac{1}{2}(\mathcal{I}_i + \mathcal{I}_j)$.

Unary costs: Intuitively, the cost $U_v(l_v)$ shall be low iff the area spanned by face f in W' is likely to belong to l_v. We first estimate a label vector $\mathcal{L}(f)$ whose coefficients reflect the probabilities that the area covered by f belongs to each room or the outside area. A naive approach would be to project all measured points into the horizontal plane and to determine how many points of each room (with respect to the point labels obtained in Section 4) are located within f. The first problem is that non-uniform distributions of measured points (Fig. 6, left) yield a similar probability estimate like a uniform distribution (Fig. 6, right) although the latter provides stronger evidence that the whole face belongs to a certain room. The second problem is that we need to estimate the probability that f is located in the outside area which is not represented by measured points.

We therefore propose a stratified sampling method which takes the spatial distribution of projected measured points into account and yields an estimate for the outside label. All measured points are projected into a uniform 2D grid in the horizontal plane. The side length of the grid cells is chosen as twice the point cloud subsampling density (see Section 8). The label vectors of all points within a grid cell are averaged and empty cells are assigned the outside label vector $\mathcal{I}_o$. Subsequently, the label vector $\mathcal{L}(f)$ of f is estimated by picking in the grid at uniformly sampled positions within f and averaging the resulting label vectors. The number of samples within f is chosen proportionally to the face area (at least

one sample is enforced). The unary cost function is then defined as

$$U_v(l_v) := \alpha \cdot area(f) \cdot \|\mathcal{L}(f) - \mathcal{I}_v\|_1, \quad (2)$$

where v is the vertex of W corresponding to face f in W', and α is a weighting factor (see Section 8). $\mathcal{L}(f)$ is the *estimated* labeling of face f, and $\mathcal{I}_v$ is the ideal *expected* label vector for label l_v. The distance between these label vectors is weighted proportionally to the area of f in order to mitigate the impact of differently sized faces in the sum of total labeling costs.

Binary costs: For the binary cost $B_{v,w}(l_v, l_w)$, consider edge e in W' to which the edge (v,w) in W corresponds. Intuitively, the cost for assigning labels l_v, l_w to $v \in V$ and $w \in V$ shall be low iff the surfaces of the wall represented by e are supported by measured points with labels l_v, l_w (in the case of wall bordering the outside area, there should be no support on the exterior side). In other words, for the separation of faces with different labels l_v, l_w, wall elements whose surfaces are supported by points with labels l_v, l_w shall be preferred. For estimating the label vector for an edge e, a sampling strategy similar to the face label vectors is used. Consider edge e originating from up to two wall surface lines $l_{\mathcal{P}_1}, l_{\mathcal{P}_2}$ (see Section 5) with according projected support points $P^{xy}_{\mathcal{P}_1}, P^{xy}_{\mathcal{P}_2}$. If e originates from a single wall surface line $l_{\mathcal{P}_1}$, we set $P^{xy}_{\mathcal{P}_2} = \varnothing$. Analogous to the 2D grid in the horizontal plane, we construct a one-dimensional grid on e. The support points $P^{xy}_{\mathcal{P}_1} \cup P^{xy}_{\mathcal{P}_2}$ are projected into the grid and their point labels are averaged per cell. Empty cells are assigned the outside label. The label vector $\mathcal{L}(e)$ is now estimated by sampling uniformly distributed points on e and averaging the label vectors obtained by picking in the grid at the sample positions. We then define the binary costs as

$$B_{v,w}(l_v, l_w) := \begin{cases} \beta \cdot len(e) \cdot (\|\mathcal{L}(e) - \mathcal{I}_{vw}\|_1 + \gamma \mathcal{L}_o(e)) & \text{if } l_v \neq l_w, \\ 0 & \text{otherwise,} \end{cases} \quad (3)$$

where v, w are the vertices of W corresponding to faces f, g in W' that are separated by edge e, $len(e)$ is the Euclidean length of edge e, and β, γ are weighting factors (see Section 8) respectively. Similar to the unary costs, weighting the distance between the observed and ideal label vectors by edge length mitigates the influence of different edge lengths. The additional term $\mathcal{L}_o(e)$ penalizes usage of edges with a high outside prior. We found that this term helps to select correct edges with support points on both sides for separating adjacent rooms. After the face labeling is determined, only edges which separate differently labeled faces are retained. The resulting subgraph $\overline{W}$ of W' (Fig. 5 (d)) is used in Section 7 for reconstructing connected wall elements.

Multiple scans within one room: We previously assumed that each room was scanned from exactly one position within that room. In the case of more than one scan, one room is represented by a set of different labels. Fig. 7(a) shows an example of a hallway scanned from three positions. After segmentation (Section 4), the hallway is split into multiple regions represented by differently labeled points (Fig. 7(b)). The graph labeling optimization separates these sections by implausible walls (Fig. 7(c)). We remove such walls (Fig. 7(d)) as part of the opening detection in the next section.

7. Model generation and opening detection

From the determined graph, the final model can now be derived in a straight forward manner. The model is further enriched by detected window and door openings.

Walls: For each edge $\overline{e} = (\overline{v}, \overline{w})$ of $\overline{W}$, a wall element $\mathcal{W}$ is constructed with centerline endpoints located at $\overline{v}$ and $\overline{w}$. The thickness of $\mathcal{W}$ is determined by the thickness of the wall candidate from which $\overline{e}$ originates. Endpoints of wall elements are connected iff the corresponding edges are incident to a common vertex. For vertical extrusion, we first estimate floor and ceiling heights for each face $\overline{f}$ in $\overline{W}$ separately using the following heuristic: consider all approximately *horizontal* planes detected during wall candidate generation (Section 5). For each plane, the number of support points located within $\overline{f}$ is determined. The elevation of the plane with the largest support within $\overline{f}$ and upwards- (resp. downwards-) facing normal is chosen as the floor height $h_{fl}(\overline{f})$ (resp. ceiling height $h_{cl}(\overline{f})$) of $\overline{f}$. The vertical extent of a wall represented by edge $\overline{f}$ separating faces $\overline{f}_1, \overline{f}_2$ is then defined to span the heights of both adjacent rooms: $[\min(h_{fl}(\overline{f}_1), h_{fl}(\overline{f}_2)), \max(h_{cl}(\overline{f}_1), h_{cl}(\overline{f}_2))]$.

Opening detection: Openings in walls either arise from doors and windows, or because a reconstructed wall was artificially introduced due to multiple scans within one room as described in Section 6. By classifying detected openings accordingly, we further enrich the model by doors and windows, and determine which walls to remove for handling multiple scans within rooms. To locate potential openings, we determine intersection points between reconstructed walls and simulated laser rays from the scan positions to the measured points. The intersection points are clustered in the 2D domain of the wall surfaces (a simple greedy, single-linkage clustering based on distances between intersection points yielded satisfactory results); see Fig. 8 (b) for an example. The clusters are then classified as doors, windows, virtual (i.e. openings due to excess walls) or invalid (i.e. clutter) by means of supervised learning using libsvm [5]. Six-dimensional feature vectors with the following features are used to characterize openings: cluster bounding box width and height, distance from lower and upper wall bounds, approximate coverage by intersection points, and a binary feature indicating whether the associated wall is adjacent to outside area. Clusters recognized as doors or windows are assigned to the respective wall elements. Adjacent faces of $\overline{W}$ separated by wall elements containing at least one virtual opening (magenta clusters in Fig. 8(b)) are merged by removing all edges to which both faces are incident. To account for changes after a wall removal, the determination of room heights, intersection points, clusters and opening classes is performed iteratively until no more virtual openings exist.

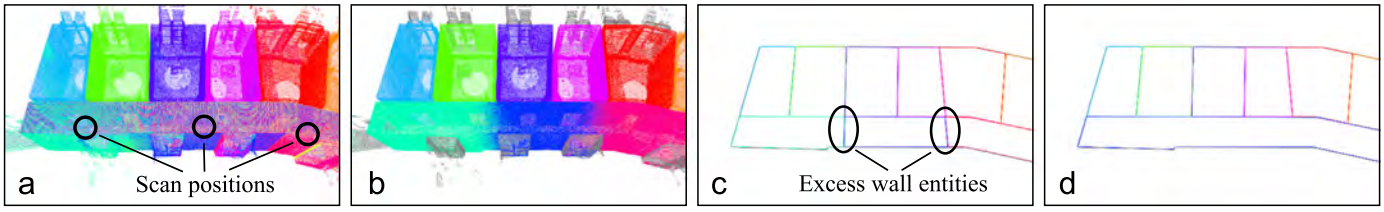


Fig. 7. Multiple scans within a single room. (a) The hallway has been scanned from three positions; room labels are mixed within that room. (b) After segmentation (Section 4), the hallway is still split into multiple sections. (c) The labeling algorithm separates these regions by wall elements that are not part of the building's true walls. (d) By detecting and removing excess wall elements, faces are merged to larger rooms.

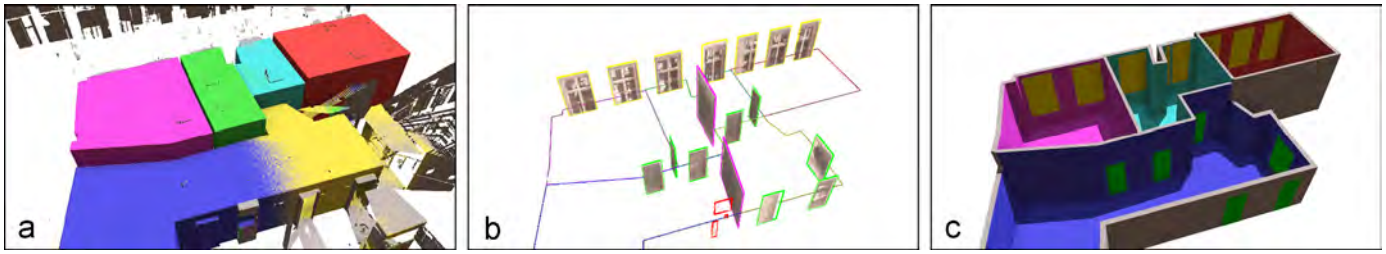


Fig. 8. Opening detection and classification. (a) The input point cloud after segmentation. (b) Detected clusters of intersections between reconstructed walls and simulated laser rays between scanner positions and measured points. Clusters are classified as doors (green), windows (yellow), "virtual" clusters indicating walls to be removed for merging multiple scans within a room (magenta), and invalid (red). (c) The final model after removal of walls containing "virtual" openings. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

Table 1
Datasets used in our experiments. Figure references marked with "∗" indicate that only a subset of the dataset is shown.

Dataset	Points	Scans	Time (s)	Figures
Building A, storey 1	9,524,724	23	84.9	12*
Building A, storey 2	19,365,622	33	215.7	16(a)
Building B, storey 1	826,229	5	6.0	–
Building B, storey 2	1,676,486	6	11.9	9*
Building B, storey 3	1,673,919	6	12.2	1, 3
Building B, storey 4	2,203,670	8	16.0	2
Building B, storey 5	2,470,678	11	17.7	5
Building C, storey 1	4,749,565	9	39.7	14*
Building C, storey 2	22,757,718	67	486.3	7*, 8*, 16(c)
Building C, storey 3	23,883,396	63	449.6	–
Building D, storey 1	17,712,659	34	252.2	15
Building E, storey 1	14,399,907	37	189.3	13*
Building E, storey 2	19,769,647	51	319.9	–
Building E, storey 3	17,104,101	43	241.5	16 (b)

8. Evaluation

We tested our approach on real-world point clouds of 14 stories from 5 different buildings; statistics are given in Table 1. The shown number of points is after subsampling with the Point Cloud Library [14] using a resolution of $\varepsilon = 0.02$ cm (i.e. in a voxel grid with a resolution of ε, at most one point in each voxel is retained). Normals are estimated by means of local PCA using point patches of 48 nearest neighbors. Normals are flipped towards the respective scanner position.

Parameter selection: The first set of crucial parameters affects plane detection in the extraction of wall surfaces (Section 5). For classifying planes as vertical (wall surfaces) or horizontal (floor and ceiling surfaces) we chose a threshold on the angular deviation of $\pm 1^\circ$ from the ideal orientations. We ignore planes with less than 500 support points or an approximate areal coverage by support points below 1.5 m^2. Also, vertical planes resulting in line segments below 0.5 m are ignored. These parameters control a tradeoff between avoiding clutter and ignoring small details: high thresholds only consider larger (but potentially more stable) planes as candidates for wall surfaces. Conversely, low thresholds may introduce clutter due to incorrectly detected planes. Fig. 9 demonstrates different choices. The second important set of parameters consists of the weights α,β,γ in Eqs. (2) and (3). In our experiments, we found that a ratio of α/β of 4/1, and $\gamma = 4$ yielded good results (Fig. 10(c)). The effects of setting either α, β, or γ to zero are shown in Fig. 10(d)–(f). We also found that smoothing the 2D and 1D grids used for the determination of face and edge label vectors in Section 6 using a large Gaussian kernel usually improves results.

Robustness: Quality and robustness of our reconstruction depend on plane detection quality which is influenced by e.g. scanner noise, point density, registration accuracy, and clutter outside and inside of the building. As our datasets were captured using professional laser scanners, noise level and sampling density were no issue. Registration errors directly influence the position of detected planes and thus the generated wall candidates. Our algorithm adapts well to small misalignment; stronger translational or rotational alignment errors have specific effects as exemplified in Fig. 11. Clutter outside of the building is effectively eliminated by our automatic filtering method. Clutter inside of rooms and scan holes pose big challenges when working with indoor scans. Except for extreme cases (e.g. completely unobserved wall surfaces or objects which span the whole story height and thus yield planes that are indistinguishable from walls), our algorithm proves to be robust against e.g. furniture within rooms as shown in Fig. 12: despite the presence of large scan holes, using all available points (from e.g. furniture) as priors for room localization closes holes, and the smoothness property of the used graph-cut-based optimization yields well-regularized walls. Furthermore, as our approach uses infinitely long wall candidates, small or medium sized holes in the support pointsets of wall surfaces caused by occlusions are automatically bridged in a plausible manner (Fig. 13). We also found that the algorithm is very robust against errors in the segmentation step (Section 4), especially in the interior of the building, i.e. if overlaps between scans of adjacent rooms still exist. However, filtering out large-scale clutter outside the building is important in order to avoid erroneous classification of outside area as rooms.

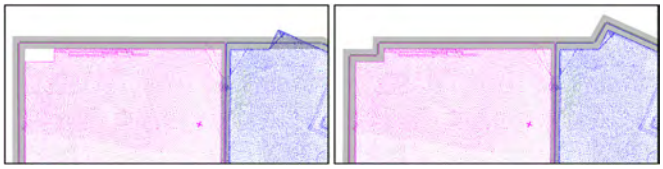


Fig. 9. Different plane detection options: Allowing smaller planes as potential wall surfaces allows for more detailed structures (right-hand side) at the cost of possibly detecting incorrect candidates in clutter.

Opening detection accuracy: Our method for opening detection consists of three parts (see Section 7): (1) determination of intersection points, (2) clustering of intersection points, and (3) classification of these clusters using supervised learning. Regarding (1), we found that the determined intersection points indicate locations and extents of openings well, with a low number of false positives. For (2), a simple single-linkage clustering based on point distances already yielded good results due to the low number of clutter intersection points which may cause chaining effects. However, a more sophisticated clustering method could improve results in some cases, e.g. multiple neighboring windows are sometimes recognized as a single cluster. Concerning (3), our training examples comprise 269 doors, 306 windows, 118 virtual, and 415 invalid clusters which were obtained by manually correcting a heuristic classification. During training, all stories originating from the building being classified were removed from the training set. Average cross validation rate of the training sets

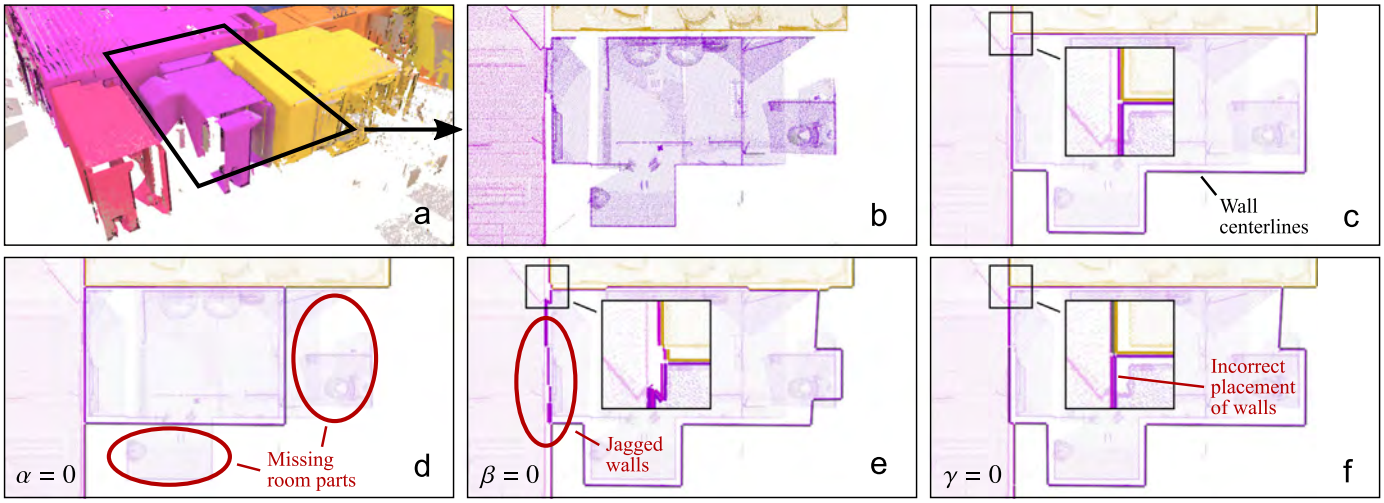


Fig. 10. Different choices for α, β, γ in Eqs. (2) and (3). (a) and (b) Perspective and orthographic view of an example situation. (c) Parameters chosen as described in Section 8. Wall centerlines are well-regularized and common wall elements have been reconstructed between rooms. (d) Without unary costs ($\alpha := 0$). While the resulting walls are well-regularized, parts of rooms are missing despite high areal support by measured points. (e) Without binary costs ($\beta := 0$). Walls are located similar to (c) but are overly complex due to missing regularization and preference for correctly labeled edges. (*f*) Without penalty for high outside labeling ($\gamma := 0$). The algorithm does not prefer common walls for separating adjacent rooms.

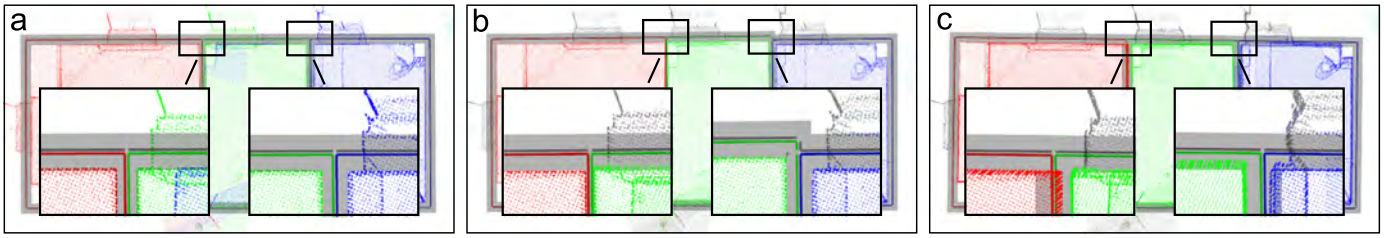


Fig. 11. Effects of registration errors. (a) Result without alignment errors. Wall volumes are shown in gray together with the respective wall centerlines. (b) Translational registration errors may result in offset walls to which the algorithm adapts accordingly (right detail view). Wall thickness may also change (wall separating the red and green rooms in the left detail view). (c) Rotational registration errors may lead to wall surface pairs not to be associated to common walls. Wall thickness is incorrect since the wall candidates do not originate from wall surfaces pairs. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

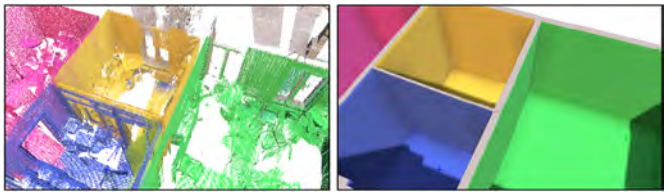


Fig. 12. Highly cluttered rooms. Left: clutter and transparent surfaces (windows) cause large scan holes; wall surfaces are only partially scanned. Right: reconstructed walls still are well regularized and separate rooms correctly.

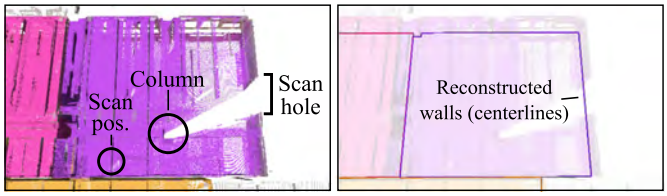


Fig. 13. Our cost minimization approach and infinitely long wall candidates automatically bridge scan holes in a plausible manner.

was 90.34%, average classification accuracy was 85.02%. This small yet significant gap indicates a generalization performance below optimum which we believe is caused by systematic differences between e.g. the used windows in different buildings, causing the feature vectors to not be i.i.d. Given the limited number of test data, we think that our approach is promising, especially since newly obtained examples can be fed back into the algorithm.

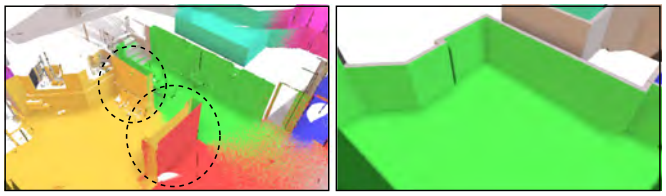


Fig. 14. Wall elements which are not connected at both ends to other walls are currently not representable by our reconstruction.

Comparison to manually generated models: A visual comparison between our reconstruction and a professional, manually generated model is shown in Fig. 15. Locations and thickness of wall elements, and locations of doors are generally good; a few walls are missing either due to the fact that (small) rooms were not scanned separately and thus room labels are missing, or because openings were misclassified as "virtual" clusters.

Time and memory requirements: Our experiments were run on a 6-core Intel Core i7-4930K (32 GB RAM) with a NVIDIA GeForce GTX 780 (3 GB RAM). Processing times of our prototypical implementation are shown in Table 1. Peak RAM usage (incl. visualization) for the largest dataset (Fig. 16c) was about 16 GB.

Limitations: If rooms are not completely enclosed by walls (e.g. balconies or partially scanned staircases), points might erroneously be classified as outside area during the segmentation step which may lead to missing parts in the reconstruction. Due to the current formulation of our approach, wall elements which are not connected to other walls at *both* ends cannot be represented.

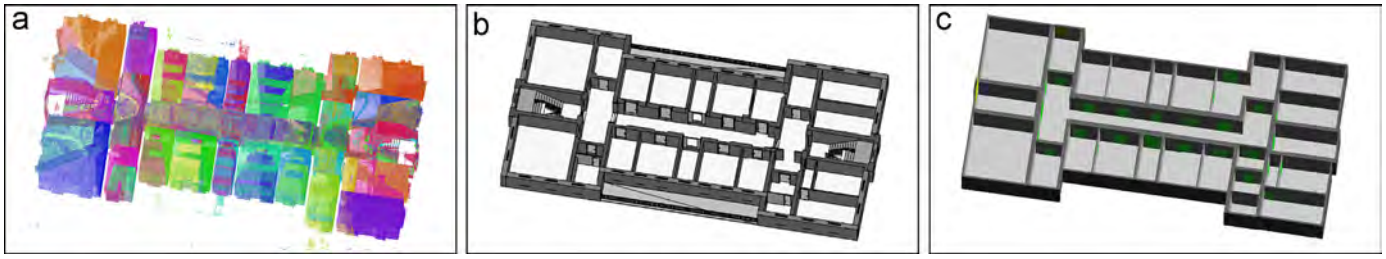


Fig. 15. Visual comparison of reconstruction and manually generated model. (a) Input point cloud; scans are shown in different colors. (b) Professionally, manually generated model. (c) Reconstructed model. Locations and thickness of walls, and locations of doors are generally good. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

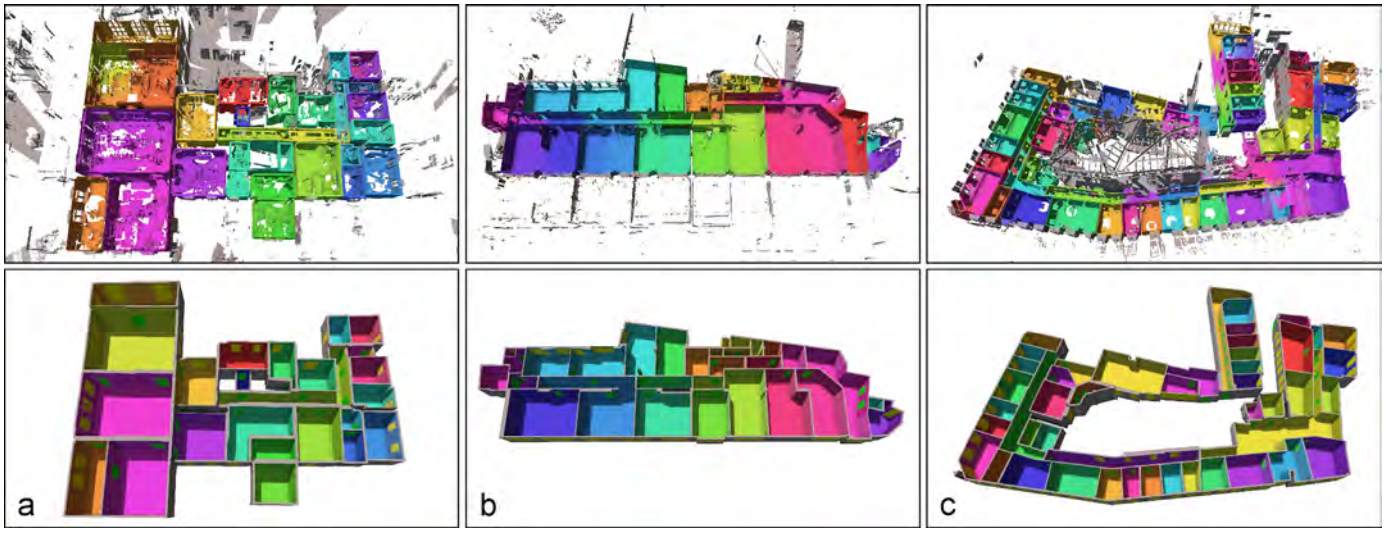


Fig. 16. Example results on point clouds with 33, 43, and 67 scans. Upper row: point clouds after segmentation step; most ceiling points (i.e. points with downwards-facing normals) are removed for visualization. Lower row: reconstructed models; detected windows are shown in yellow, doors are shown in green. Most wall elements are faithfully reconstructed; some excess walls have not been removed (see e.g. the large room in the lower-right corner of the second column). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

As a consequence, they are either missing (Fig. 14), or erroneously connected to other wall elements. Also, since we only consider planar wall surfaces and linear wall candidates, only piecewise linear wall structures can be reconstructed.

9. Conclusion and future work

We presented the first automatic method for the reconstruction of high-level parametric building models from indoor point clouds. The feasibility of our approach was demonstrated on a variety of complex real-world datasets which could be processed with little or no parameter adjustments. In the future, a more thorough comparison of reconstruction results with existing, manually generated models would help to analyze reconstruction results quantitatively. A generalization to multiple building stories poses specific challenges but would enable the reconstruction of multi-story models without the need to process stories separately. Also, the usage of different capturing devices (e.g. mobile devices) and real-time handling of streamed data are topics for future investigation.

Acknowledgments

This work was partially funded by the German Research Foundation (DFG) under grant KL 1142/9-2 (*MoD*), as well as by the European Community under FP7 grant agreements 600908 (DURAARK) and 323567 (Harvest4D).

Appendix A. Supplementary material

Supplementary data associated with this article can be found in the online version at http://dx.doi.org/10.1016/j.cag.2015.07.008.

References

[1] Adan A, Huber D. 3d reconstruction of interior wall surfaces under occlusion and clutter. In: Proceedings of international conference on 3D imaging, modeling, processing, visualization and transmission (3DIMPVT); 2011. p. 275–81.
[2] Boykov Y, Kolmogorov V. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. IEEE Trans Pattern Anal Mach Intell 2004;26(9):1124–37.
[3] Boykov Y, Veksler O, Zabih R. Fast approximate energy minimization via graph cuts. IEEE Trans Pattern Anal Mach Intell 2001;23(11):1222–39.
[4] Budroni A, Boehm J, Automated 3d reconstruction of interiors from point clouds. Int J Archit Comput; 2010, http://dx.doi.org/10.1260/1478-0771.8.1.55.
[5] Chang CC, Lin CJ. LIBSVM: A library for support vector machines. ACM Trans Intell Syst Technol 2011;2:27(27):1–27 Software available at: http://www.csie.ntu.edu.tw/cjlin/libsvm.
[6] Kolmogorov V, Zabin R. What energy functions can be minimized via graph cuts? IEEE Trans Pattern Anal Mach Intell 2004;26(2):147–59.
[7] Monszpart A, Mellado N, Brostow G, Mitra N. RAP ter: rebuilding man-made scenes with regular arrangements of planes. In: ACM SIGGRAPH 2015; 2015.
[8] Mura C, Jaspe Villanueva A, Mattausch O, Gobbetti E, Pajarola R. Reconstructing complex indoor environments with arbitrary wall orientations. In: Proceedings of EG posters; 2014.
[9] Mura C, Mattausch O, Jaspe Villanueva A, Gobbetti E, Pajarola R. Automatic room detection and reconstruction in cluttered indoor environments with complex room layouts. Comput Graph 2014;44.
[10] Ochmann S, Vock R, Wessel R, Tamke M, Klein R. Automatic generation of structural building descriptions from 3D point cloud scans. In: GRAPP; 2014.

[11] Oesau S, Lafarge F, Alliez P. Indoor scene reconstruction using primitive-driven space partitioning and graph-cut. In: Eurographics workshop on urban data modelling and visualisation; 2013.
[12] Oesau S, Lafarge F, Alliez P. Indoor scene reconstruction using feature sensitive primitive extraction and graph-cut. ISPRS J Photogramm Remote Sens 2014;90 (April (68–82)).
[13] Okorn B, Xiong X, Akinci B, Huber D, Toward automated modeling of floor plans. In: Proceedings of the symposium on 3D data processing, visualization and transmission, vol. 2; 2010.
[14] Rusu RB, Cousins S. 3D is here: point cloud library (PCL). In: IEEE international conference on robotics and automation (ICRA), Shanghai, China; May 9–13, 2011.
[15] Sanchez V, Zakhor A. Planar 3d modeling of building interiors from point cloud data. In: 2012 19th IEEE international conference on image processing (ICIP); 2012. p 1777–80.
[16] Schnabel R, Wahl R, Klein R. Efficient RANSAC for point-cloud shape detection. In: Computer graphics forum, vol. 26. Wiley Online Library; 2007. p. 214–226.
[17] Tamke M, Blümel I, Ochmann S, Vock R, Wessel R. From point clouds to definitions of architectural space. In: eCAADe 2014; September 2014.
[18] Turner E, Cheng P, Zakhor A. Fast automated scalable generation of textured 3d models of indoor environments. IEEE J Sel Top Signal Process 2015;9(April (3)):409–21.
[19] Turner E, Zakhor A. Floor plan generation and room labeling of indoor environments from laser range data. In: GRAPP; 2014.
[20] Xiao J, Furukawa Y. Reconstructing the world's museums. In: Computer vision, ECCV 2012. Springer; 2012. p. 668–81.
[21] Xiong X, Adan A, Akinci B, Huber D. Automatic creation of semantically rich 3d building models from laser scanner data. Autom Constr 2013;31:325–37.

Accurate Isosurface Interpolation with Hermite Data

Simon Fuhrmann
TU Darmstadt

Michael Kazhdan
Johns Hopkins University

Michael Goesele
TU Darmstadt

Abstract

In this work we study the interpolation problem in contouring methods such as Marching Cubes. Traditionally, linear interpolation is used to define the position of an isovertex along a zero-crossing edge, which is a suitable approach if the underlying implicit function is (approximately) piecewise linear along each edge. Non-linear implicit functions, however, are frequently encountered and linear interpolation leads to inaccurate isosurfaces with visible reconstruction artifacts. We instead utilize the gradient of the implicit function to generate more accurate isosurfaces by means of Hermite interpolation techniques. We propose and compare several interpolation methods and demonstrate clear quality improvements by using higher order interpolants. We further show the effectiveness of the approach even when Hermite data is not available and gradients are approximated using finite differences.

1. Introduction

Implicit functions are a popular surface representation for many reconstruction algorithms. As opposed to explicit representations, implicit functions are agnostic to topology and more easily support blending between shapes, boolean queries, and morphological operations such as erosion and dilation. Contouring of implicit functions, i.e., extracting an explicit surface from the implicit representation, is an important application in computer graphics.

An implicit function $F\colon \mathbb{R}^3 \rightarrow \mathbb{R}$ associates a scalar value $F(\mathbf{x})$ to every point in space $\mathbf{x}$. The function implicitly defines a surface S as the *level-set* $S = \{\mathbf{x} \mid F(\mathbf{x}) = d\}$ with respect to the *isovalue* d. The surface is guaranteed to be manifold if d is not a singular value of the function (i.e., the gradient ∇F does not vanish on the level-set). The level-set S is also called the *isosurface* of F. Without loss of generality, we assume $d = 0$ and note that choosing a different d is equivalent to subtracting d from F. A popular example for an implicit function is the *signed distance function* (SDF), which describes for every point in space the distance to the closest point on the shape, with the sign of the function indicating whether the point is interior or exterior to the surface. The surface is then defined as the zero level-set of the implicit function.

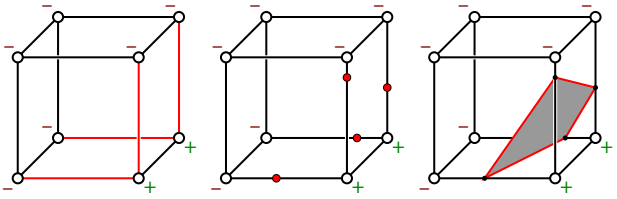


Figure 1. Marching Cubes: Four edges of the cube contain a sign change (left), interpolation of isovertices (middle) and the final polygonal surface (right).

An implicit function is often represented on a regular lattice, i.e., sampled at uniformly spaced positions, and Marching Cubes [15] is often the contouring algorithm of choice. As a regular sampling of F is unsuitable for the representation of large or multi-scale shapes, octrees and tetrahedralizations have been used, and the isosurface is obtained with more general algorithms [1, 19, 4, 24, 11].

For these representations, the implicit function is sampled at discrete positions and the *isovertex* positions are determined by interpolation along edges, and triangulations connecting the isovertices are computed per cell, see Figure 1. Traditionally, this interpolation is performed using linear approximation, i.e., by finding the zero-crossing of a linear function along each edge. If the implicit function is actually non-linear along the edges, the interpolated isovertex positions poorly estimate the actual position of the zero-crossing along the edge, see Figure 2. Depending on the type of implicit function, these inaccuracies often manifest themselves in structured patterns on the final surface, which can appear as ringing or undulating artifacts. This is caused by alternating between over- and underestimation of the correct zero-crossing.

A simple example of a non-linear implicit function for the sphere with radius r is $F_q(\mathbf{x}) = x^2 + y^2 + z^2 - r^2$. Contouring F_q with linear interpolation leads to a larger reconstruction error than contouring the signed distance function $F_\ell(\mathbf{x}) = \sqrt{x^2 + y^2 + z^2} - r$, although both functions have the same zero level-set. Note that near the isosurface, the signed distance function has small second derivatives and is approximately linear. Figure 3 visualizes this reconstruc-

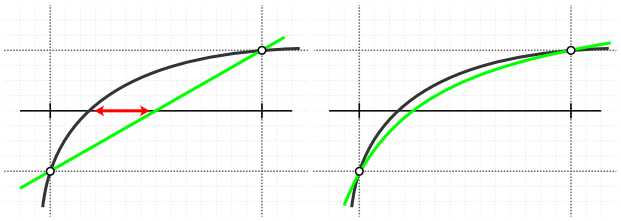


Figure 2. For non-linear functions (black), the linear interpolant (green) is often a poor fit (left). Interpolation with higher-order functions leads to much higher accuracy (right).

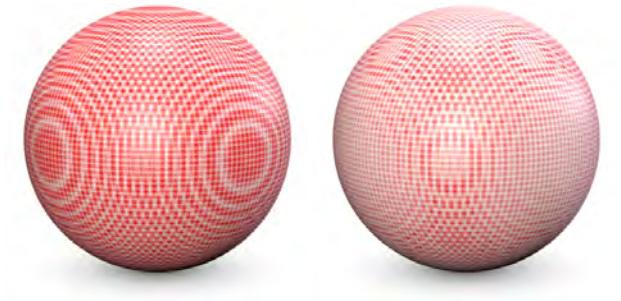


Figure 3. Visualization of the reconstruction error of the quadratic function F_q (left) and the signed distance function F_ℓ (right) on a $64 \times 64 \times 64$ voxel grid. Red corresponds to a larger error.

tion error. While both reconstructions have high-frequency errors due to discretization, reconstruction of F_q exhibits considerably more pronounced low-frequency ringing artifacts around the axes of the coordinate system.

A piecewise linear approximation to a smooth function has an approximation error that depends on the sampling density of the function. The approximation error decreases as $\mathcal{O}(h^2)$ with the sample spacing h, i.e., if the sampling spacing is halved, the approximation error is reduced by $1/4$. This quadratic behavior suggests that increasing the sampling density will be effective in reducing these artifacts. However, even in the cases where the continuous implicit function is available for re-sampling, such a refinement has a dramatic impact on memory consumption and runtime.

In this work we argue that contouring non-linear implicit functions requires additional data in order to obtain accurate, artifact-free results for high quality reconstruction. We advocate the use of Hermite data, i.e., utilizing the implicit function gradient $\nabla F(\mathbf{x})$ in addition to the values $F(\mathbf{x})$ at the sampling positions $\mathbf{x}$. Depending on the application, ∇F can be analytically computed or estimated using finite differences. We present several formulations for incorporating the derivatives in the interpolation scheme and evaluate the quality of the obtained results on higher-order implicit functions. Further, we incorporate all interpolation methods in *Poisson Surface Reconstruction* (PSR) [9, 10] and the more recent *Floating Scale Surface Reconstruction* (FSSR) [5], and analyze the impact of the different formulations on the reconstruction accuracy. We demonstrate clear surface quality improvements on synthetic and real-world data compared to linear interpolation. This improvement has motivated the incorporation of one Hermite interpolation technique in the early PSR code [18]. This work is the first to compare the different non-linear interpolation techniques and evaluate the practical implications on surface quality.

2. Related Work

The most popular method for contouring implicit functions is the *Marching Cubes* algorithm [15] which uses linear interpolation to place isovertices along the zero-crossing edges of a regular lattice and then defines a triangulation by connecting the isovertices within each cell, see Figure 1.

Although initially proposed for regular hexahedral grids, the Marching Cubes algorithm has been extended to adaptive space partitions including octrees [1, 19, 24, 23, 11] and (graded) tetrahedralizations of space [4]. To define the implicit surface, all these approaches require estimating the position of the isovertex along a zero-crossing edge, and linear interpolation is the technique most commonly used.

Many extensions of Marching Cubes have been proposed [20], including the reconstruction of bicubic spline surfaces [7] or continuous quadratic implicit functions for visualization purposes [17]. In contrast, our method does not compute a higher-order surface representation. It uses similar ideas for the purpose of reducing reconstruction artifacts, but is restricted to a one dimensional interpolation problem along the zero-crossing edges.

Hermite data has been used in the dual contouring method by Ju et al. [8], by Manson and Schaefer [16], and the primal/dual hybrid approach by Kobbelt et al. [12]. They use Hermite data to construct planes that are tangent to the surface and minimize a quadratic error function (QEF) to solve for an isovertex position in the interior of the cell. The minimizer is often a poor estimate of the actual function, and may require additional function evaluations [16]. There are numerical difficulties in solving the linear system of equations induced by the QEFs, e.g., the minimizer is not guaranteed to be in the interior of the cell (see [22, 8] for more details). These methods are different in that they focus on improving the reconstruction of sharp features and edges in the implicit function, not on more accurate isosurface interpolation.

In Lempitsky's work [14] a smooth implicit function is reconstructed from a binary volume by solving a constrained optimization problem minimizing the function curvature. This differs from our approach in that we do not use binary input and perform more accurate interpolation "on the fly" without having to solve a global optimization problem. As in Lempitsky's work, we also guarantee correctness in that we only place an isovertex along an edge whose endpoints have opposite signs.

3. Hermite Interpolation

The interpolation problem in primal contouring methods is one-dimensional because we are only interested in the root of the implicit function F along an edge $\mathbf{e}$. We call the restriction of F to this one-dimensional subspace $f = F\big|_{\mathbf{e}}$. To perform Hermite contouring, the values F and the gradient ∇F must be available at the sampling positions. For interpolation, however, we are only interested in the derivative $f' = \nabla F\big|_{\mathbf{e}}$ at the sampling positions along the direction of the edge $\mathbf{e}$. If the edges are axis-aligned, the derivative f' is just the corresponding component of the gradient ∇F. Otherwise, the directional gradient along $\mathbf{e}$ can be obtained with the dot product: $f' = \langle \nabla F, \mathbf{e} \rangle / \|\mathbf{e}\|^2$.

An edge $\mathbf{e}$ is represented by its two endpoints $\mathbf{x}_0$, $\mathbf{x}_1$, which are the sampling positions of F. To formulate the interpolation in a uniform setting, we scale the interval between $\mathbf{x}_0$, $\mathbf{x}_1$ to $[0, 1]$. This requires scaling the derivatives ∇F by a factor of $\|\mathbf{x}_0 - \mathbf{x}_1\|$. Given the function values and derivatives

$$\begin{aligned} f(0) &= v_0 \qquad f'(0) = d_0 \\ f(1) &= v_1 \qquad f'(1) = d_1 \end{aligned} \tag{1}$$

we describe several ways for using Hermite interpolation to obtain a more accurate isovertex position along the edge $\mathbf{e}$. In particular, we investigate cubic interpolation as well as two different types of quadratic interpolation.

3.1. Third Order Polynomial

A cubic function has four degrees of freedom, so it seems natural to use the two value and two derivative constraints to obtain a unique solution for the polynomial coefficients

$$\begin{aligned} p(x) &= a_0 + a_1 x + a_2 x^2 + a_3 x^3 \\ p'(x) &= a_1 + 2a_2 x + 3a_3 x^2. \end{aligned} \tag{2}$$

Substituting the constraints from (1) into (2) leads to the linear system of equations (see, for example [13])

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} v_0 \\ v_1 \\ d_0 \\ d_1 \end{pmatrix} \tag{3}$$

with the unique solution

$$\begin{aligned} a_0 &= v_0 \qquad a_2 = 3v_1 - 3v_0 - 2d_0 - d_1 \\ a_1 &= d_0 \qquad a_3 = 2v_0 - 2v_1 + d_0 + d_1. \end{aligned} \tag{4}$$

Although straightforward, a cubic polynomial can have up to three real roots whose location and count may be sensitive to small perturbations of the function coefficients, see Figure 4. To uniquely define the position of an isovertex, we observe that there must be an odd number of roots along a zero-crossing edge, and we always use the "middle" root.

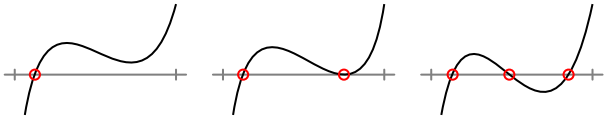


Figure 4. Different cases for the roots of a cubic function: One root (left), three roots counting multiplicity (middle), and three distinct roots (right).

This corresponds to using the single root in the case of linear and quadratic interpolation, and is also well-defined for higher-order interpolants.

3.2. Second Order Polynomials

Using a second order interpolant is the correct choice if the implicit function is known to be quadratic. Examples include PSR, where the implicit function is represented as a linear combination of second-order B-splines (or if the implicit function is regularized to have small third derivative). Since a quadratic function has three degrees of freedom, substituting the four constraints from (1) into (2) with $a_3 = 0$ yields an overdetermined linear system of equations of form $\mathtt{A}\mathbf{x} = \mathbf{b}$

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} v_0 \\ v_1 \\ d_0 \\ d_1 \end{pmatrix}. \tag{5}$$

The coefficient matrix $\mathtt{A}$ has full rank, so there is no exact solution in general.

Least-Squares Solution: We can solve the linear system in (5) in a least-squares fashion using the normal equation, multiplying with $\mathtt{A}^\intercal$ on the left to get:

$$\mathtt{A}^\intercal\mathtt{A}\mathbf{x} = \mathtt{A}^\intercal\mathbf{b} \quad \Rightarrow \quad \mathbf{x} = (\mathtt{A}^\intercal\mathtt{A})^{-1}\mathtt{A}^\intercal\mathbf{b}. \tag{6}$$

The matrix $(\mathtt{A}^\intercal\mathtt{A})^{-1}\mathtt{A}^\intercal$ can be precomputed and the coefficients can be hard-coded as in (4). However, the least-squares solution produces a polynomial $p(x)$ where the constraints on the values are not exactly met. This can lead to the situation where, although $f(x)$ has a zero-crossing along the edge, $p(x)$ does not. For this reason we will not further consider this solution.

Least-Squares Derivatives: Instead, to guarantee that $p(x)$ always has a root along the edge if $f(x)$ also has a root, the quadratic function must interpolate the values of the implicit function $p(0) = v_0$ and $p(1) = v_1$. We now discuss a solution that is least-squares optimal for the derivatives, and interpolates the function values. For the value constraints, we have according to (5):

$$\begin{aligned} a_0 &= v_0 \\ a_0 + a_1 + a_2 &= v_1. \end{aligned} \tag{7}$$

The derivatives give rise to the constraints which must be met in a least-squares sense

$$\begin{aligned} a_1 &= d_0 \\ a_1 + 2a_2 &= d_1 \end{aligned} \tag{8}$$

which leads to the minimization problem

$$\arg\min_{a_1,a_2} : (a_1 - d_0)^2 + (a_1 + 2a_2 - d_1)^2. \tag{9}$$

From (7) we know that $a_2 = v_1 - a_1 - v_0$, and substituting a_2 in (9) yields a least-squares problem in a single variable:

$$\arg\min_{a_1} : (a_1 - d_0)^2 + (a_1 - 2v_1 + 2v_0 + d_1)^2 \tag{10}$$

Setting the derivative of (10) to zero and solving for a_1 leads to the polynomial coefficients

$$\begin{aligned} a_0 &= v_0 \\ a_1 &= \frac{d_0 - d_1}{2} + v_1 - v_0 \\ a_2 &= \frac{d_1 - d_0}{2}. \end{aligned} \tag{11}$$

Examining the coefficient a_2, we see that the second order term vanishes if the implicit function is locally linear, i.e., if $d_0 = d_1$.

Third Order Elimination: Finally, we discuss two possible second-order solutions that can be obtained from the third order solution in (4) by eliminating the third order coefficient a_3. This can be achieved by introducing an additional degree of freedom for the derivatives.

Scaling the derivatives by s and setting a_3 to zero gives

$$\begin{aligned} 2v_0 - 2v_1 + s \cdot d_0 + s \cdot d_1 &= 0 \\ s &= \frac{2v_1 - 2v_0}{d_0 + d_1}. \end{aligned} \tag{12}$$

If the derivatives d_0 and d_1 in (4) are scaled by s, the cubic term vanishes. However, the solution in (12) becomes unstable if the derivatives cancel each other out, i.e., $d_0 + d_1 \approx 0$. Whether this instability causes actual problems depends on the properties of the implicit function. For example, this is the method implemented in the PSR code for interpolating the indicator function, which has a steep gradient in the vicinity of the isosurface and is unlikely to have partial derivatives with opposite signs.

An alternative approach to scaling is to introduce an additive degree of freedom o

$$\begin{aligned} 2v_0 - 2v_1 + (d_0 + o) + (d_1 + o) &= 0 \\ o &= \frac{1}{2}(2v_1 - 2v_0 - d_0 - d_1). \end{aligned} \tag{13}$$

This solution has an interesting property. When adding the offset o to the derivatives d_0 and d_1 in (4), it can be shown that this solution is equivalent to the solution in (11).

4. Algebraic Surfaces

We now compare the interpolation methods on synthesized, non-linear implicit functions. A "ground truth" isosurface is generated by sampling a $512 \times 512 \times 512$ voxel grid and using linear interpolation to define the isovertex positions on zero-crossing edges. The test meshes are then extracted from a $64 \times 64 \times 64$ voxel grid and compared to the ground truth. The following interpolation methods are evaluated:

- `LINEAR`: Linear interpolation without derivatives
- `SCALING`: The quadratic method in (12) that scales the derivatives to eliminate the third order term
- `LSDERIV`: The quadratic method in (11) and (13) that interpolates the function values and least-squares fits the derivatives
- `CUBIC`: The cubic polynomial fit in (4)

For comparison to the ground truth, we use *Metro* [3], a tool for measuring distances between triangle meshes. Color-coding is used to visualize the distance between the ground truth and the test mesh directly on the surface, with red indicating larger distances. We also compare the impact of using the analytically computed gradient ∇F with a central differences approximation of ∇F. Note that, when using finite differences in conjunction with cubic interpolation, one obtains the standard Catmull-Rom interpolant [2].

Smooth Box: The implicit function of the *Smooth Box* dataset is given by

$$F(\mathbf{x}) = x^4 + y^4 + z^4 - 1. \tag{14}$$

This is a fourth-order function and cannot be exactly reconstructed with any of the interpolation methods in Section 3. Figure 5 visualizes the reconstruction error for all interpolation methods. Table 1 lists the maximum, mean and root-mean-square (RMS) distances to the ground truth mesh for the analytic and finite differences gradient.

Genus-2: The implicit function for the *Genus-2* dataset is a fifth-order polynomial with mixed terms

$$\begin{aligned} F(\mathbf{x}) = {} & 2y(y^2 - 3x^2)(1 - z^2) \\ & + (x^2 + y^2)^2 - (9z^2 - 1)(1 - z^2). \end{aligned} \tag{15}$$

The error is visualized in Figure 6 and distances to the ground truth are given in Table 2. For both datasets the reconstruction errors of the non-linear interpolants are barely distinguishable and give nearly identical results regardless of whether analytic gradients or finite-differences are used.

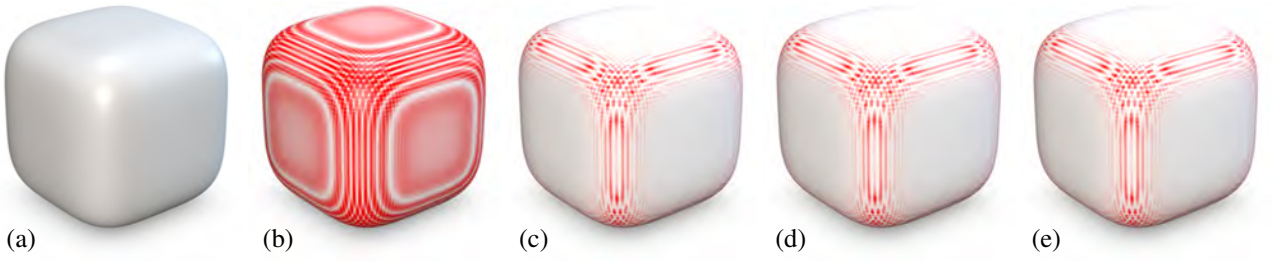


Figure 5. *Smooth Box*: (a) Ground truth, (b) `LINEAR` interpolation with color-coded reconstruction errors, (c) `SCALING` interpolation, (d) `LSDERIV` interpolation, and (e) `CUBIC` interpolation. All gradients have been computed analytically.

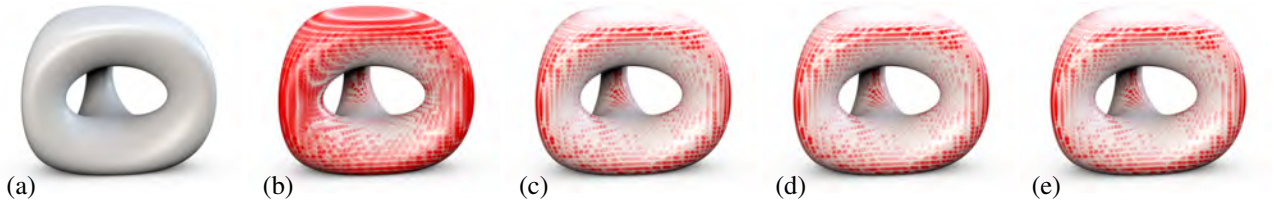


Figure 6. *Genus-2*: (a) Ground truth, (b) `LINEAR` interpolation, (c) `SCALING` interpolation with analytic gradients, (d) `LSDERIV` interpolation with analytic gradients, and (e) `LSDERIV` interpolation with approximate finite differences gradients.

Analytic ∇F	max	mean	RMS
`LINEAR`	5.200640	2.527520	2.655661
`SCALING`	4.576201	0.914086	1.225579
`LSDERIV`	4.576010	0.912602	1.224503
`CUBIC`	4.577743	0.912709	1.225184
Approx. ∇F	max	mean	RMS
`SCALING`	4.576851	0.916040	1.227008
`LSDERIV`	4.575684	0.911629	1.223793
`CUBIC`	4.581211	0.912659	1.225978

Table 1. *Smooth Box*: Distances to the ground truth mesh with analytic ∇F (top) and finite differences approximation (bottom). The distances are scaled for readability (factor 10^4).

Analytic ∇F	max	mean	RMS
`LINEAR`	2.0846366	0.4342808	0.5301991
`SCALING`	2.1002761	0.1964645	0.2915423
`LSDERIV`	2.0975643	0.1957446	0.2908659
`CUBIC`	2.0964227	0.1959128	0.2908642
Approx. ∇F	max	mean	RMS
`SCALING`	2.0997145	0.1974943	0.2925005
`LSDERIV`	2.0943636	0.1953266	0.2904807
`CUBIC`	2.0925705	0.1961537	0.2905572

Table 2. *Genus-2*: Distances to the ground truth mesh with analytic ∇F (top) and finite differences approximation (bottom). The distances are scaled for readability (factor 10^3).

5. Analytic and Discrete Surfaces

We implemented the interpolation methods in *Poisson Surface Reconstruction* (PSR) [9, 10] and the more recent *Floating Scale Surface Reconstruction* (FSSR) [5] to analyze the impact of Hermite interpolation on real surface reconstruction algorithms. Note that the `SCALING` method in (12) is already implemented in the PSR code [18]. In both, PSR and FSSR, the gradient of the implicit function can be computed analytically. Because the original weighting function in FSSR is not $\mathcal{C}^1$-continuous, we replace it with the weighting function $w(r) = \frac{1}{3^{12}}(r-3)^{12} \cdot (r+1)^4$.

We first consider synthetic data for which the ground truth is available and the reconstruction errors are easier to measure and visualize. Then, we demonstrate Hermite interpolation on real-word data from 3D scanners and Multi-View Stereo. Finally, we show an application to isosurface extraction from medical images.

5.1. Synthetic Data

We first evaluate the interpolation methods on two synthetic datasets, namely the *Sphere* and the *Blob*. To this end, we obtained high-resolution triangle meshes for both datasets and use these as ground truth. A point set is generated by computing per-vertex normals and, in the case of FSSR, also per-vertex scale values. The connectivity information is then discarded and the resulting point sets are used for reconstruction with PSR and FSSR.

Sphere Dataset: Because both PSR and FSSR use non-linear basis functions, estimation of isovertex positions using linear interpolation leads to artifacts, see Figure 7. Table 3 gives the distances of the reconstructed meshes from the ground truth. PSR produces essentially the same quality result for all non-linear interpolants while the FSSR error improves for most metrics with cubic interpolation.

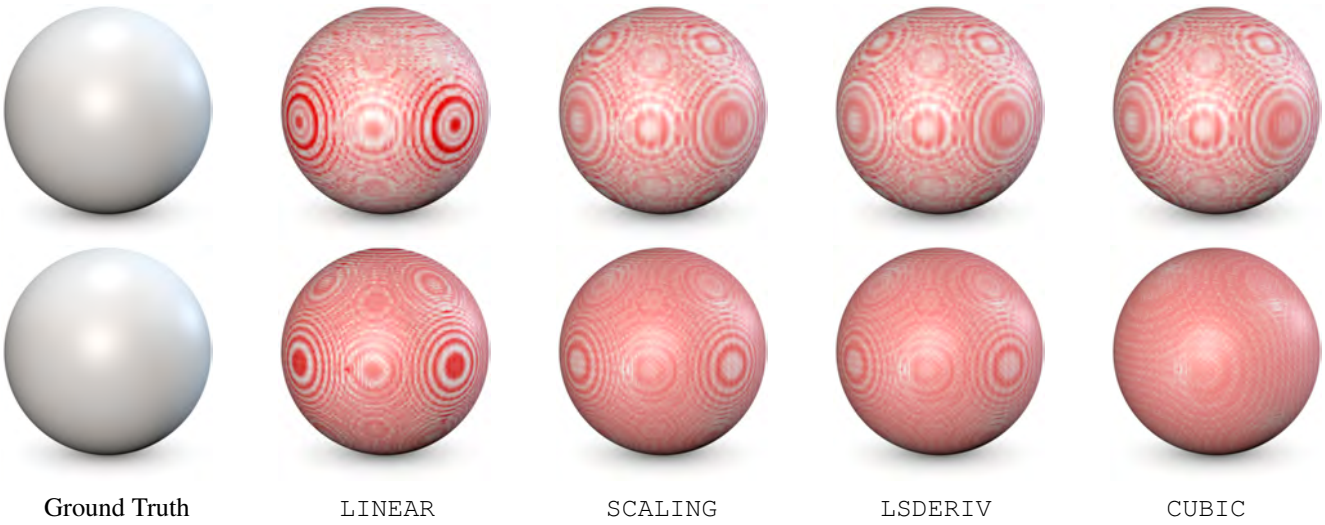


Figure 7. *Sphere:* Visualization of the reconstruction error with PSR (top) and FSSR (bottom).

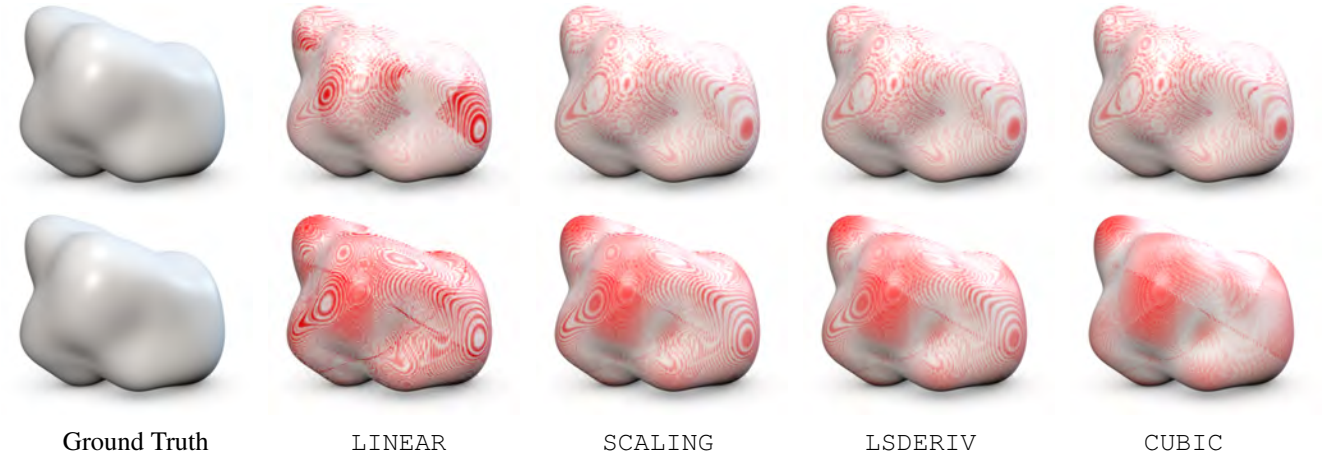


Figure 8. *Blob:* Visualization of the reconstruction error with PSR (top) and FSSR (bottom).

PSR [9]	max	mean	RMS
LINEAR	1.4475699	0.2142690	0.2911505
SCALING	0.8729671	0.1757755	0.2155248
LSDERIV	0.8729671	0.1757931	0.2155422
CUBIC	0.8729671	0.1758032	0.2155564
FSSR [5]	max	mean	RMS
LINEAR	1.5472957	0.3554895	0.3957082
SCALING	0.7770098	0.3520047	0.3692734
LSDERIV	0.7552927	0.3527912	0.3668787
CUBIC	0.6887877	0.3547093	0.3637461

Table 3. Reconstruction error on the *Sphere* dataset. The statistic shows the error between ground truth and the reconstruction using PSR (top) and FSSR (bottom). The distances have been obtained with *Metro* [3] and scaled for readability (factor 10^3).

PSR [9]	max	mean	RMS
LINEAR	3.837804	0.321320	0.467301
SCALING	3.137207	0.282015	0.369597
LSDERIV	3.137207	0.282022	0.369603
CUBIC	3.137207	0.281913	0.369466
FSSR [5]	max	mean	RMS
LINEAR	6.066898	0.659474	0.861424
SCALING	5.913879	0.485656	0.622890
LSDERIV	5.921924	0.434055	0.561907
CUBIC	6.012503	0.391485	0.513969

Table 4. Reconstruction error on the *Blob* dataset. The statistic shows the error between ground truth and the reconstruction using PSR (top) and FSSR (bottom). The distances have been obtained with *Metro* [3] and scaled for readability (factor 10^4).

Figure 9. *Stanford Bunny:* Geometric difference between LINEAR and CUBIC interpolation with PSR (left) and FSSR (right).

Blob Dataset: We also evaluate the different interpolation approaches on the *Blob* dataset, which exhibits more interesting curvature changes. The reconstruction errors are visualized in Figure 8. Similar to the *Sphere* dataset, linear interpolation leads to strong ringing artifacts and larger errors, see Table 4.

It is noteworthy that, with PSR, the quality improvement from LINEAR to non-linear interpolation is substantial. However, which higher-order interpolant is used barely makes a difference. This is because PSR represents the implicit function as the sum of second-order B-splines, so all interpolants reproduce the quadratic function along the edge. For FSSR, the CUBIC interpolation improves mean and RMS error as well as the visual appearance, although the maximum error can remain large.

5.2. Scanner and MVS Data

Next, we evaluate the interpolation methods on real-world scanner and MVS data. Because a ground truth model is not available for this data, we focus on a visual comparison between the LINEAR and the CUBIC interpolation. Note that visually, all higher-order methods produce results that are almost indistinguishable.

Stanford Bunny: We reconstructed the Stanford Bunny using PSR and FSSR with both, LINEAR and the CUBIC interpolation. The geometric difference between the two methods is visualized in Figure 9. This difference is presumably caused by the improved fitting with the CUBIC interpolant, and the ringing artifacts of the linear interpolation method become clearly visible.

Miniature City: The miniature city is a Multi-View Stereo dataset with 76 input images and has been reconstructed with the publicly available *Multi-View Environment* [6]. The resulting point cloud with 4,627,606 samples was then used as input for PSR and FSSR with the LINEAR and CUBIC interpolation method. The geometric improvement is visualized in Figure 10 and clearly visible even without color coding.

Figure 10. *Miniature City:* 2 out of 76 input images (top). Geometric difference between LINEAR (middle) and CUBIC (bottom) interpolation with PSR (left) and FSSR (right).

5.3. MRI Data

Brain: We compared the LINEAR and the CUBIC interpolation on an MRI scan of a brain obtained from the OASIS MRI database [21] (resolution $182 \times 218 \times 182$). Since the dataset comes without gradients, finite differences are used to estimate them. In Figure 11 we provide results including contrast-enhanced renderings to highlight the artifacts caused by the linear method, which are otherwise hard to visualize.

6. Conclusion

We presented Hermite interpolation for Marching Cubes-like algorithms to eliminate the majority of the artifacts that occur when contouring non-linear implicit functions with traditional linear interpolation. The extracted triangle meshes are guaranteed to have the same connectivity as the meshes extracted with traditional Marching-Cubes, but the accuracy of the isovertex positions is improved.

The proposed interpolation methods, particularly the quadratic ones, are simple to implement and can be applied to a wide range of surface extraction algorithms. The computational overhead of the quadratic methods is insignificant and in fact barely measurable. The cubic interpolation increases the total surface extraction time by about 3% with our implementation.

We have demonstrated the applicability of Hermite interpolation on PSR and FSSR and show that when gradients cannot be computed analytically, the finite differences approximation is still successful in removing the artifacts. Any of the non-linear interpolation methods substantially

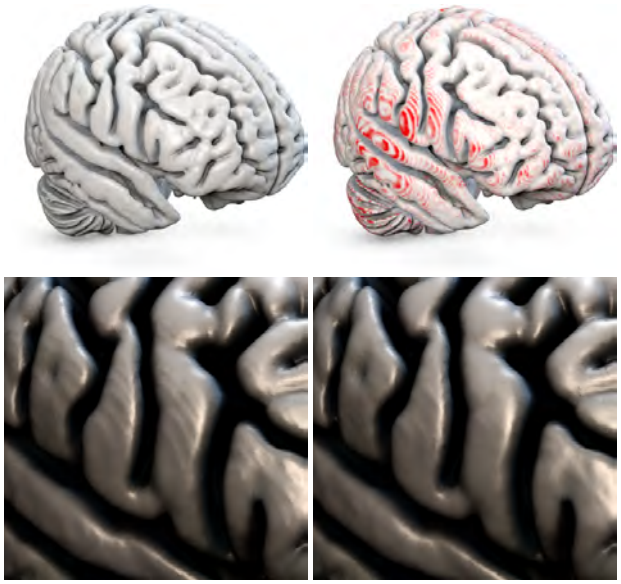


Figure 11. *Brain*: The brain isosurface with `CUBIC` interpolation (top left) and the error distance compared to linear interpolation (top right). High-contrast close-ups (bottom) of the linear method (left) and the cubic method (right) show the ringing caused by linear method.

increases surface accuracy, but "the right" method depends on the application: For example, in PSR, the quadratic methods provide sufficient accuracy while in FSSR, cubic interpolation leads to further improvement.

Acknowledgements

Part of the research leading to these results has received funding from the European Commission's FP7 Framework Programme under grant agreements ICT-323567 (HARVEST4D) and ICT-611089 (CR-PLAY).

References

[1] J. Bloomenthal. Polygonization of Implicit Surfaces. *Computer Aided Geometric Design*, 5(4):341–355, 1988. 1, 2

[2] E. Catmull and R. Rom. A Class of Local Interpolating Splines. In *Computer Aided Geometric Design*, pages 317 – 326. 1974. 4

[3] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: Measuring Error on Simplfied Surfaces. *Compututer Graphics Forum*, 17(2):167–174, 1998. 4, 6

[4] A. Doi and A. Koide. An Efficient Method of Triangulating Equi-Valued Surfaces by using Tetrahedral Cells. *IEICE Transactions on Information and Systems*, 74(1):214–224, 1991. 1, 2

[5] S. Fuhrmann and M. Goesele. Floating Scale Surface Reconstruction. In *Proceedings of ACM SIGGRAPH*, 2014. 2, 5, 6

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

[7] R. S. Gallagher and J. C. Nagtegaal. An Efficient 3-D Visualization Technique for Finite Element Models and Other Coarse Volumes. In *Proceedings of ACM SIGGRAPH*, pages 185–194, 1989. 2

[8] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual Contouring of Hermite Data. *ACM Transactions on Graphics*, 21(3), July 2002. 2

[9] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson Surface Reconstruction. In *Eurographics Symposium on Geometry Processing*, pages 61–70, New York, New York, USA, 2006. 2, 5, 6

[10] M. Kazhdan and H. Hoppe. Screened Poisson Surface Reconstruction. *ACM Transactions on Graphics*, 32(3):1–13, June 2013. 2, 5

[11] M. Kazhdan, A. Klein, K. Dalal, and H. Hoppe. Unconstrained Isosurface Extraction on Arbitrary Octrees. In *Eurographics Symposium on Geometry Processing*, pages 125–133, 2007. 1, 2

[12] L. P. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Seidel. Feature Sensitive Surface Extraction from Volume Data. *Proceedings of ACM SIGGRAPH*, D:57–66, 2001. 2

[13] D. H. U. Kochanek and R. H. Bartels. Interpolating Splines with Local Tension, Continuity, and Bias Control. In *Proceedings of ACM SIGGRAPH*, pages 33–41, 1984. 3

[14] V. Lempitsky. Surface Extraction from Binary Volumes with Higher-Order Smoothness. *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages 1197–1204, 2010. 2

[15] W. E. Lorensen and H. E. Cline. Marching Cubes: A high resolution 3D surface construction algorithm. In *Proceedings of ACM SIGGRAPH*, pages 163–169, 1987. 1, 2

[16] J. Manson and S. Schaefer. Isosurfaces Over Simplicial Partitions of Multiresolution Grids. *Computer Graphics Forum*, 29(2):377–385, 2010. 2

[17] A. Marinc, T. Kalbe, M. Rhein, and M. Goesele. Interactive Isosurfaces with quadratic C^1 Splines on Truncated Octahedral Partitions. *Information Visualization*, 11(1):60–70, 2012. 2

[18] Michael Kazhdan. Poisson Surface Reconstruction Code. http://www.cs.jhu.edu/ misha/Code/PoissonRecon/. 2, 5

[19] H. Mueller and M. Stark. Adaptive generation of surfaces in volume data. Technical report, 1991. 1, 2

[20] T. S. Newman and H. Yi. A Survey of the Marching Cubes Algorithm. *Computers and Graphics*, 30(5):854–879, 2006. 2

[21] OASIS. Open Access Series of Imaging Studies. http://www.oasis-brains.org/. 7

[22] S. Schaefer, T. Ju, and J. Warren. Manifold dual contouring. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):610–619, May 2007. 2

[23] S. Schaefer and J. Warren. Dual Marching Cubes: Primal Contouring of Dual Grids. *Computer Graphics Forum*, 24(2):195–201, June 2005. 2

[24] G. Varadhan, S. Krishnan, T. Sriram, and D. Manocha. Topology Preserving Surface Extraction using Adaptive Subdivision. *Proceedings of Symposium on Geometry Processing*, page 235, 2004. 1, 2