Eurographics Symposium on Geometry Processing 2018
T. Ju and A. Vaxman
(Guest Editors)

Volume 37 (2018), Number 5

Hierarchical Quad Meshing of 3D Scanned Surfaces

Dennis R. Bukenberger, Hendrik P. A. Lensch

Department of Computer Graphics, Eberhard Karls University, Tübingen, Germany

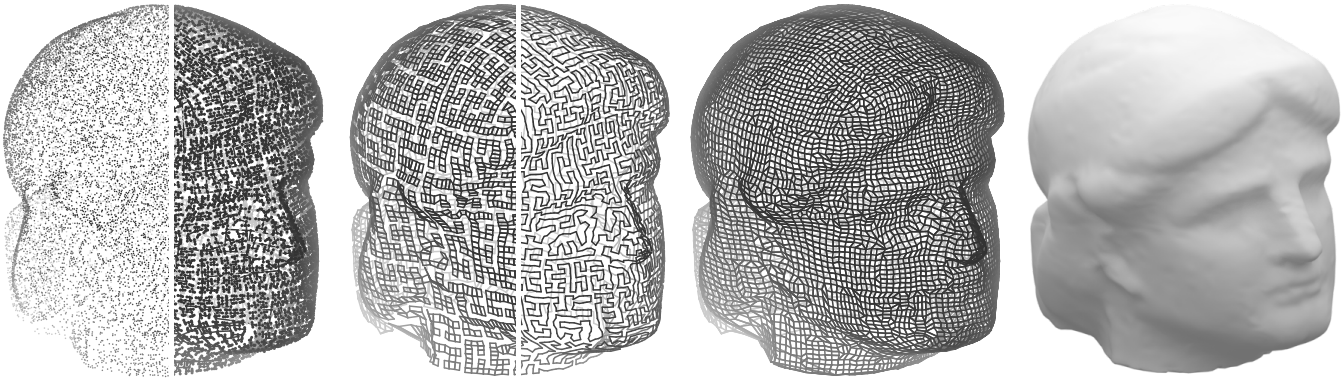


Figure 1: *Snapshots of our quad meshing process - from left to right: Minerva 3D Scan Point Cloud [Bol09] / kd-tree with subsampled vertices, interconnected tile maze / open edge path around the maze, the final quad mesh and a rendering of the reconstructed surface. The initial point cloud may be sampled from an existing surface for remeshing or originate from a real 3D scan as demonstrated here.*

Abstract

In this paper we present a novel method to reconstruct watertight quad meshes on scanned 3D geometry. There exist many different approaches to acquire 3D information from real world objects and sceneries. Resulting point clouds depict scanned surfaces as sparse sets of positional information. A common downside is the lack of normals, connectivity or topological adjacency data which makes it difficult to actually recover a meaningful surface. The concept described in this paper is designed to reconstruct a surface mesh despite all this missing information. Even when facing varying sample density, our algorithm is still guaranteed to produce watertight manifold meshes featuring quad faces only. The topology can be set-up to follow superimposed regular structures or align naturally to the point cloud's shape. Our proposed approach is based on an initial divide and conquer subsampling procedure: Surface samples are clustered in meaningful neighborhoods as leafs of a kd-tree. A representative sample of the surface neighborhood is determined for each leaf using a spherical surface approximation. The hierarchical structure of the binary tree is utilized to construct a basic set of loose tiles and to interconnect them. As a final step, missing parts of the now coherent tile structure are filled up with an incremental algorithm for locally optimal gap closure. Disfigured or concave faces in the resulting mesh can be removed with a constrained smoothing operator.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Hierarchy and geometric transformations, Modeling packages, Object hierarchies

1. Introduction

3D reconstruction is a well studied subject in computer graphics but also gained great research interests in other fields as manufacturing, robotics or the automotive industry. Many popular acquisition techniques for 3D scans reproduce the scanned scenery as loose sets of 3D points. However, many applications require surfaces or meshes to make further use of an object. Many popular meshing algorithms are based on surfaces or oriented point cloud samples as in field-alignment or Poisson-based solutions. Our proposed approach can utilize the unoriented point cloud of an object's surface to directly reconstruct a watertight mesh. By construction, the output of our algorithm is a pure quad mesh. Thus, it does not require any additional remeshing for producing quads, while it is trivial to generate triangular meshes from our output by inserting a diagonal to each face. Our algorithm relies on a binary kd-tree for spatial 3D clustering: This hierarchical data structure is utilized to generate 2^n meaningful vertices on the point cloud. Furthermore, the kd-tree captures valuable neighborhood affiliations during its construction, which is then used in the meshing step.

1.1. Motivation

Most common 3D scanning techniques natively produce point clouds as representation of the scanned surfaces. The scanning procedures either sample surfaces pointwise by construction or accu-

mulate correlations of multiple inputs in single points. Either way, scanned surfaces will be represented with sparsely distributed singularities of three-dimensional position data. Some techniques are able to recover color information or estimate normal directions per sample. Rendering point clouds is challenging and not trivially possible with common ray-tracing or rasterization approaches, detecting collisions or intersections is not directly possible and their actual shape is hard to grasp for humans and machines without explicit neighborhood information.

There exist many algorithms and mathematical approaches to recover surfaces from point clouds by fitting primitives or solving Poisson equations. However, often additional work is required to cleanup and prepare meshes for downstream applications. The goal of this approach is to produce clean and watertight quad-meshes. We achieve that with a mesh structure that is adaptive to the input as the result of kd-tree-based subsampling, exploiting the trade-off between resolution and regularity. While common meshing techniques produce triangulations or quad-dominant meshes [BLP*13], our algorithm is designed to produce quad-only meshes from point clouds directly without further remeshing or subdivision steps.

1.2. Overview

The main strengths of our proposed meshing algorithm can be summarized as: **Points only:** In contrast to other meshing algorithms, ours does not require any further information as normals, faces, color or motion. **Quad mesh:** Whereas other approaches may only promise quad dominant meshes, ours is guaranteed to feature only quads without remeshing. **Arbitrary input:** Point clouds can be of any given size, the resulting mesh resolution can be chosen independently. **Watertight:** Results produced by our algorithm are closed continuous manifold surfaces and do not need additional care to close holes, e.g. as for 3D printing or volume rendering. **Adaptability:** Tiles vary in size, adaptive to the point cloud resolution instead of leaving holes as other procedures do. **Alignment:** The mesh topology may follow superimposed axis-aligned regularity for organic shapes or align adaptively to point cloud features.

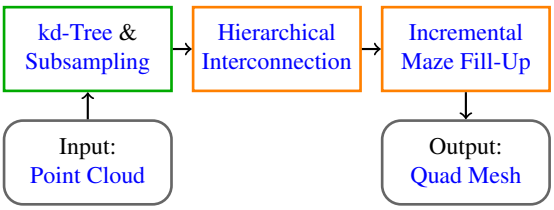


Figure 2: *Schematic flowchart of the proposed procedure. Blue labels are clickable links to the associated section in the paper. The stages can be categorized as geometry acquisition and meshing.*

The three main steps of our proposed procedure are described in the following and are illustrated as schematic pipeline in Figure 2.

• **kd-Tree & Subsampling:** A balanced binary kd-tree is constructed by recursively subdividing the input point cloud with axis-aligned splits. A sphere is fitted to all samples in a leaf and a suitable proxy sample on the sphere's surface is selected as representative vertex. Based on the kd-tree, four vertices create a basic quad tile. Section 3 elaborates on this in more detail.

• **Hierarchical Interconnection:** Dictated by the tree-node's relationships, special tiles are added to interconnect two related parts of the tree on each level. As the algorithm progresses from the leaf's down to the root, all tiles will be interconnected. The resulting structure is one coherent set of quad tiles, resembling a maze. By construction it is guaranteed that after this step, all connected quads are enclosed by one coherent path of open edges. This algorithm is discussed in Section 4.1.

• **Incremental Maze Fill-Up:** All dead ends of the coherent maze-like structure are closed in this last step. A priority queue is processed incrementally until all open edges, and therefore, the mesh is closed. The resulting surface is one watertight manifold, featuring quad faces only. This algorithm is introduced in Section 4.2 and by default converges to objects with genus 0. Higher genera are possible but require manual intervention as shown in Section 6.3.

2. Related Work

Scans, Point Cloud Generation and Meshing: Many popular 3D objects used in computer graphics (e.g. Stanford Bunny, Happy Budda, Armadillo) originate from 3D scans [Sta14]. One of the most ambitious scans is the *digital Michelangelo project* [LPC*00] which features a scan of the 5.17m tall David statue. A 3D model of this statue was published in 2009, featuring about 1 billion polygons. While there are many different forms and approaches of surface reconstruction [BTS*14], we will mainly keep the scope on point cloud to mesh techniques. Algorithms for automated meshing often optimize for some kind of mathematical regularity or global parametization [LLZ*11]. A popular approach to actually recover surfaces from point clouds is the *Poisson surface reconstruction* [KBH06] which, however, requires all points to be oriented. Concepts to actually incorporate some form of mesh-topology feature alignment are based on field-alignment [JTPSH15, STJ*17]. These concepts all create meshes based on point clouds, by exploiting additional information as principal directions [LLZ*11] or vertex normals [KBH06, JTPSH15, STJ*17]. The reconstruction approaches of Kalogerakis et al. [KNSS07] and Zhang et al. [ZLGH10] incorporate their own methods to determine normals in one framework.

Many concepts rely on the assumption of rather regularly sampled point clouds, irregular input often requires specialized approaches for surface reconstruction [HDD*92]. For moving-least-squares methods [LCOL07, SSW09] the sample density has great impact on reconstruction accuracy. The technique proposed by Ohtake et al. [OBA*03] approaches the problem as a hiarachical fitting problem and is therefore more robust to non-uniform sampling. Methods that are specialized on the reconstruction from non-uniform point clouds are so far able to generate resampled point sets [LCOLTE07] or triangulated surfaces [HK06]. Our procedure is adaptive to sample density and always results in a watertight quad-only mesh. Results of the listed field-aligned and quad-dominant approaches tend to be quite regular but may also feature holes or fail completely if the sample density is varying too much (Figures 13, 19, 20).

Quad Remeshing: Once a surface is recovered from a 3D scan it may not yet fulfill the requirements for further processing: Many algorithms only produce triangulated meshes and further processing steps are required to produce quads. Subdivision surfaces [CC78]

result in quad meshes but also artificially increase the mesh resolution which is not always tolerable. Another approach to produce quad meshes from given surfaces was formulated by Knupp [Knu95] using a vector-field based parametrization. Further publications [KNP07, RLL*06, BZK09] elaborate on more details of this method. The approach for remeshing by Daniels et al. [DSC09] introduced a novel hierarchical mapping method to reconstruct arbitrary polygonal meshes with quads from simplified base domains. Mapping a grid of integer iso-lines from $\mathbb{R}^2$ to surfaces creating quad meshes was later introduced as *integer grid map* by Bommes et al. [BCE*13]. In the technique for quad remeshing published by Ebke et al. [ESCK16] the user has interactive control to modify topology and mesh flow. However, all quad remeshing approaches require a surface (parameterized or meshed) which is not given when starting from a point cloud.

Relevant work on geometry: Mousa et al. [MCAG07] employ a kd-tree in a similar way as we do, however theirs is aimed at efficient spherical harmonics representations of 3D objects rather than meshes. The process of *deep points consolidation* proposed by Wu et al. [WHG*15] associates each surface point with a deep point which form a meso-skeleton. The reconstructed surface is an improvement to naive solutions but involves a Poisson solver nevertheless. Resulting meshes of our procedure sometimes appear faceted, cosmetic improvements, e.g., for rendering purposes can be achieved with *bilateral mesh denoising* as proposed by Fleishman [FDCO03]. Needless to say, our results are also suitable to smoothing algorithms as Taubin's technique for *surface smoothing without shrinkage* [Tau95], subdivision surfaces or quad mesh simplification [TPC*10].

3. kd-Tree: Divide and Conquer on arbitrary 3D Point Clouds

Point clouds of different sources bear challenges as varying sample-count, -density or -distribution. We propose an algorithm to face these challenges using abstraction: A specialized divide and conquer scheme allows us to subsample given information and recursively construct a hierarchical data structure modeling spatial coherence. Sole assumption in our procedure is that input samples originate from surfaces and not volumes. Nevertheless, the algorithm is tolerant to noise and can handle (single) outliers.

Our reconstruction is based on a balanced kd-tree. Benefits of this tree are that it is easy to construct and traverse and has low computation and memory overhead. Furthermore, an important feature for our followup meshing strategy is that the kd-tree can be constructed to be balanced, which is not very suitable for more general binary-space-partitioning (BSP) approaches or non-binary Octrees. The grouped samples contained in our tree's leaf nodes are used to construct a new representative vertex. Four of these new samples will later be connected to one quadrilateral face, in the following also called a tile. New tiles are added on each tree level to interconnect two sets of tiles each, obeying the tree hierarchy. This is why the tree's internal structure is essential for the upcoming interconnection and meshing operations in Section 4.

As listed in the introduction, the first step in our procedure is to establish a kd-tree. This section is dedicated to its construction. Section 5.1 introduces a kd-tree extension for feature alignment where splits are no longer axis-aligned but derived from input features.

3.1. Split Characteristics

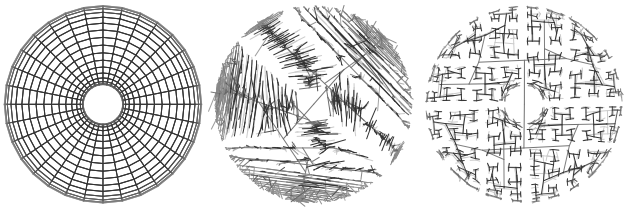


Figure 3: *Seen from above from left to right: Source mesh of a torus; BSP-tree using mean split position and eigenvector split direction; our balanced kd-tree with axis aligned splits at the median.*

The construction of the tree has impact on the distribution of our proxy vertices and therefore on the resulting mesh structure. Figure 3 compares results of two different split approaches: Recursively splitting the point sets along the plane that is orthogonal to the largest eigenvector creates a very far-flung and dispersed tree. We achieved a more regular and therefore more suitable structure with splits along the three world space axes x, y, z. The largest variance among these axes determines which dimension is used for the split. If not stated otherwise, we used $k = 100$ as criteria to split a node with more than k samples, which yields at least 50 samples per leaf node. Figure 15 in the results Section 6.1 compares outcomes using four different ks on the same point cloud. The subsampling technique in Section 3.2 approximates all leaf samples with a least-squares-fitted sphere. Therefore, at least 4 samples per leaf are advisable and $8 \leq k$ should be satisfied.

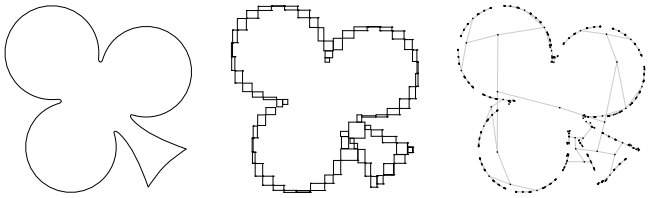


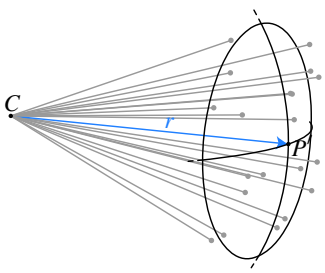
Figure 4: *2D example of an asymmetric and not axis aligned shape with rounded and sharp features. The spread of the kd-tree already approximates the surfaces after only a few levels.*

As illustrated in Figure 4, the impact of axis aligned splits is only minor and the recursive decomposition is adaptive enough to approximate organic shapes and structures without axis aligned symmetry. Nevertheless, in Section 5.1 we demonstrate the possibility to use split planes adaptively aligned to the point cloud. Furthermore, the example on the right in Figure 4 visualizes the importance of the assumption that the point cloud samples model surfaces and not volumes. Only the first few nodes close to the root actually populate the interior of the sampled space. Nodes on or close to the leaf level are already distributed over the sampled surface itself.

Balance: To ensure the binary tree is balanced and all leafs end up on the same level, the median is employed as split position. Furthermore, a leaf count of 2^i with $2 \leq i$ is always divisible by 4 which is crucial to construct quadrilateral faces. For the following, let the root node be level 1 and leaf nodes on level n. Four leaf nodes are required to construct at least one quadrilateral face, therefore all leaf nodes must be on a level $3 \leq n$.

3.2. Subsampling: Spherical Surface Fits

One representative vertex for all samples in a leaf is created by subsampling the points inside a node. The synthetic vertices created in this step are the sole geometric base for the generated surface mesh. No further points are added in any other stage of our procedure.



$$\begin{bmatrix} |s_0|^2 \\ \vdots \\ |s_m|^2 \end{bmatrix} = \begin{bmatrix} 2s_0 & 1 \\ \vdots & \vdots \\ 2s_m & 1 \end{bmatrix} \begin{bmatrix} C \\ t \end{bmatrix} \quad (1)$$

$$r = \sqrt{|C|^2 + t} \quad (2)$$

$$P = \frac{1}{m+1} \sum_{i=0}^{m} s_i \quad (3)$$

$$P' = C + r \frac{P - C}{|P - C|} \quad (4)$$

Figure 5: *Gray points visualize samples $s \in \mathbf{S}$ (with $|\mathbf{S}| = m+1$) of a single leaf node. The representative vertex P' is found on the surface of a least-squares fitted sphere with center C and radius r.*

A leaf node may contain up to k samples of the point cloud. Due to the spatial decomposition in the kd-tree, the samples $s \in \mathbf{S}$ in a leaf are in some sense a nearest neighborhood. A single point P' is determined to represent all samples in a leaf node. The arithmetic mean P (Equation 3) of all samples' 3D positions is not suitable to model bent or curved surfaces properly and is subject to noise. Therefore, a sphere is fitted to the set of samples: The overdetermined system of Equations 1 is solved in a least-squares sense and yields the sphere's center C and radius r (Equation 2). The mean of all samples P is then projected onto the sphere's surface as point P' to better represent the properties of the actually sampled surface. This is illustrated in Figure 5 and formulated in Equation 4.

3.3. From Leafs to Quad Tiles

All vertices of the final quad mesh are essentially the kd-tree's leaf node's P' points. To join vertices to become faces, one has to determine how to connect them: Our algorithm starts with nodes on level $n-2$. Each of those nodes has two children on level $n-1$ which again have two leaf nodes each. This makes four leaf nodes (each has a vertex P') for a node on level $n-2$ to form a tile. Besides their affiliation, there is no information on how these vertices should be arranged as a quad face. In a triangular case there is only one option but in a quad face there are three possibilities, as illustrated in Figure 6. For our goal of a regular mesh, faces are favorably convex and without twists. The best solution out of the three possibilities is found as the one with the smallest perimeter. P' points are usually inside their leaf's bounding box, otherwise they may be clipped.

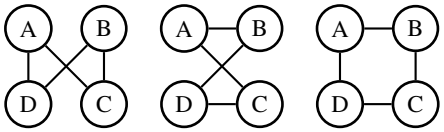


Figure 6: *Three possible constellations to form a quad face with four vertices A, B, C, D. Our algorithm will select the solution on the right, which has the smallest perimeter.*

Since bounding boxes do not intersect each other and a tile is constructed from four nearest leaf nodes we can guarantee that created tiles will also not intersect each other.

3.4. Subsequent Split Alignment

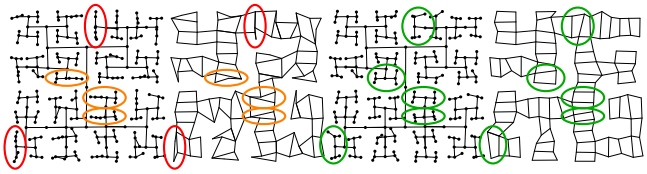


Figure 7: *kd-Tree and tiles resulting from samples of a simple 2D plane. Left: When nodes on level $n-2$ and $n-1$ split in the same dimension. Right: Results with our split alignment correction.*

It is possible that nodes on level $n-2$ and $n-1$ split along the same dimension. As illustrated in red on the left in Figure 7, this causes tiles to become very narrow. If only one level $n-1$ node has the same split dimension as its parent, tiles become disfigured as pointed out in orange. Either of these cases might be the best local split according to the applied condition, however, there is no point in keeping those tiles if their shape is counterproductive for further use. Therefore, we extended our tree construction algorithm to detect such constellations and make use of the second best option for the split dimension. Improved results of realigned level $n-1$ nodes are illustrated in green on the right in Figure 7.

4. Meshing

This section elaborates further details on our main meshing method which proceeds in two steps: First, the basic set of loose quad tiles from the kd-tree is interconnected to form one coherent set of tiles. Second, dead ends of the resulting maze-like structure are filled up incrementally to finalize the meshing.

4.1. Hierarchical Interconnection

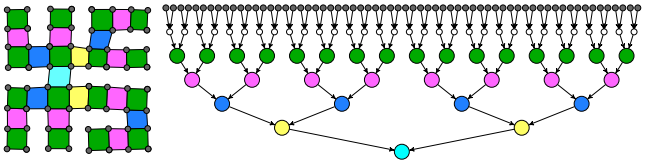


Figure 8: *A color coded example of geometry on the left with its corresponding tree structure on the right. Levels n and $n-1$ are illustrated in gray and white respectively, green nodes are level $n-2$.*

The following algorithm is illustrated in Figure 8. As described in Section 3.3, the initial set of tiles (■) is created from descendants of level $n-2$ nodes (⁘). By progressing through the tree from leaf to root level, the algorithm determines which sets of tiles are about to be connected next: A level $n-3$ node (●) specifies which two tiles from level $n-2$ (■) should be connected by inserting a new one (■). A level $n-4$ node (●) specifies which two sets of three tiles (■■■) should be connected with a new one (■), and so on. Although there are usually more possible ways, only one tile is created per node which yields $2^{i-1}-1$ tiles per level $1 \leq i$.

This constraint is crucial: The created structure will always feature only one closed path of coherent open edges around the geometry. The yet unoccupied tile positions are resolved with the fill-up algorithm detailed in Section 4.2.

Finding a Connection: Each node has two children and each child comes with a coherent set of tiles, starting on level $n-2$ with only one tile per node. It has to be determined for each node, how the tile sets of its children can be connected by inserting one new tile. Therefore, the algorithm has to select one edge of each child's tile set, create two new edges and add them as the connecting tile.

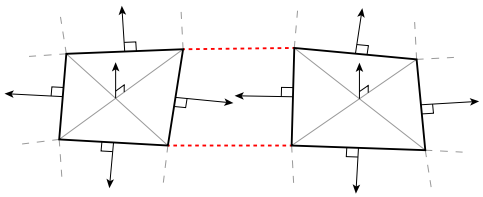


Figure 9: *Only edges with their normals facing other geometry are considered for new faces. Combined length of the dotted red lines (edges to be created) indicates where to create a new face.*

Similar to the situation illustrated in Figure 6, the algorithm has to determine an order for four vertices in the new tile. Since two edges are already given, say AD and BC, the center case in Figure 6 is not possible and there are only two options left. As one can easily comprehend, one constellation of edge pairs is usually more suitable to be connected than the other. It has to be determined where the connection can be established: We chose the combined length of the two new edges as the objective. These are illustrated in red in Figure 9. The tile, that requires the shortest two new edges will be added. However, the number of possible edge pairs grows very quickly: For a level $n-2-i$ with $1 \leq i \leq n-3$ the number of possible edge pairs to chose from is 2^{2+2i}.

To keep this computation feasible we filter out the majority of edges beforehand, based on the criteria that their edge normals have to point in to the direction of connectable geometry. The four vertices of a tile are not necessarily coplanar, so the tile normal is computed as the cross product of the tile's diagonals. An edge normal is perpendicular to the edge and computed as the cross product of edge direction and tile normal. As illustrated in Figure 9, from 16 possible edge pairs there is now only one pair of edges left where both edge normals point towards the other set of tiles. However, if the number of edge pairs is still too large, we can more rigorously limit the set of pairs to a fixed number k. Then only the first k edge pairs with the smallest distance between their midpoints will be considered for the creation of a new tile. We also achieved great filter results with a maximum threshold λm on the midpoint to midpoint distance of an edge pair. We chose m as the mean length of all existing edges from the basic tile set and a scaling factor $1 \leq \lambda \leq 10$ dependent on the desired mesh regularity.

4.2. Fill-Up: A-Maze-ing Surface Reconstruction

The interconnected tiles from the tree traversal create a tile structure resembling a maze spanning through the whole point cloud. As commonly known, the distinction between a maze and a labyrinth is that a labyrinth has only one way through and no dead ends whereas a maze can have multiple ways and also dead ends. In our mesh there is actually only one path connecting two tiles but it still features dead ends, and therefore, technically classifies as a maze.

The maze outline is one coherent path of all open edges, *topologically equivalent to closed loop*. By construction the number of open edges is dividable by 4 so that it is always possible to close this circle in an iterative fashion with quads. The following algorithm only considers the maze's negative so that empty space is actually interpreted as closable. To fill the *loop* hole, the algorithm inserts one quad at a time by closing the surface that is spanned by three connected open edges. A priority queue ensures to process the most well-formed quad candidate first. In general this will lead to the effect that the maze hole is filled up from the dead ends to the more open space.

(A,B,C)	(I,J,K)
(B,C,D)	(J,K,L)
(C,D,E)	(K,L,M)
(D,E,F)	(L,M,N)
(E,F,G)	(M,N,O)
(F,G,H)	(N,O,P)
(G,H,I)	(O,P,A)
(H,I,J)	(P,A,B)

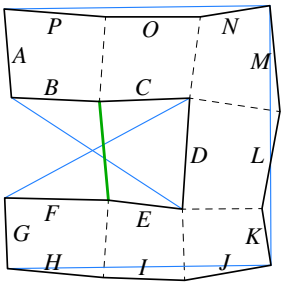


Figure 10: *Adjacent open edges $A,\ldots,P$ ordered in triples, inner edges (dashed) are ignored. Only orange triples feature edges of three distinct tiles, possible closing edges are drawn in on the right. The green edge is the best, so (C,D,E) is closed up.*

Three edges are considered as a dead end if they are all open (only one face attached) and form one coherent edge path. Therefore, all open edges are assigned to groups of three adjacent edges each, as illustrated by the example in Figure 10. However, not all triples should be considered: Negative dead-ends like (F,G,H), (P,A,B) and corner-cases like (A,B,C), (E,F,G), (G,H,I), (I,J,K), (J,K,L), (L,M,N), (M,N,O) and (O,P,A) should be discarded. We can filter out these cases with the additional constraint that all three edges in a triple must originate from three distinct tiles. This is also illustrated in Figure 10. The remaining six triples are highlighted in orange, their corresponding close-up edges are drawn in on the right (blue, green).

Our algorithm maintains a priority queue featuring all considerable triples sorted by the objective function $f(\ldots)$ in ascending order. Objective $f(e_L,e_C,e_R,e_N)$ measures the well-formedness of a potential quad with new edge e_N in context of a left-center-right edge triple (e_L,e_C,e_R). It is formulated in Equation 5, with $1 = \lambda_\varphi + \lambda_\theta$. Components $\varphi(\ldots)$ and $\theta(\ldots)$ are illustrated in Figure 11.

$$f(e_L,e_C,e_R,e_N) = \lambda_\varphi \cdot \varphi(e_L,e_C,e_R,e_N) + \lambda_\theta \cdot \theta(e_L,e_R) \qquad (5)$$

The first term of the objective function is expressed in Equation 6 and determines the ratio of the new edge compared to the perimeter of the whole face. In the worst case the new edge e_N has the same length as e_L,e_C and e_R combined and the quad would have no surface area at all.

$$\varphi(e_L,e_C,e_R,e_N) = 4\left|\frac{|e_N|}{\sum_{i\in[L,C,R,N]}|e_i|} - \frac{1}{4}\right| \qquad (6)$$

Equation 7 expresses the second part of the objective function, which considers the angle between opposite edges e_L and e_R. Here the worst case is defined as e_L and e_R pointing away from each other. The most desirable scenario is that both are parallel and point into the same direction.

$$\theta(e_L, e_R) = \frac{\frac{e_L}{|e_L|} \cdot \frac{e_R}{|e_R|} - 1}{-2} \tag{7}$$

The triple with the smallest value for $f(\ldots)$ is popped off the queue and instantiated as new face. In Figure 10 the best possible edge corresponds to the triple (C, D, E) and is marked in green. This step is repeated incrementally on all open edges until the queue is empty and therefore all dead ends are closed up. Figure 12 shows exemplary progress of this algorithm based on the mesh from Figure 8. An animated progress of the whole procedure is featured in our supplemental video.

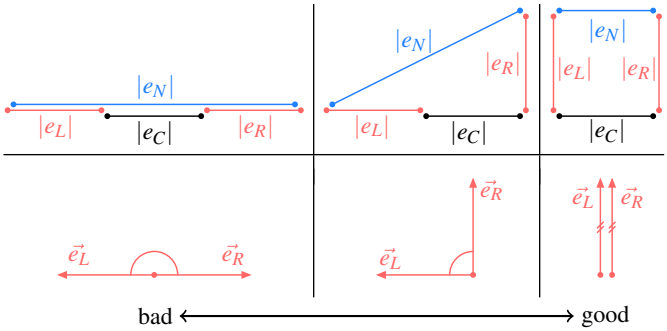


Figure 11: *Good and bad edge constellations according to Eq. 5. Edge-length ratios in the upper row are evaluated by the φ-term (Eq. 6), enclosed angles in the bottom row by the θ-term (Eq. 7).*

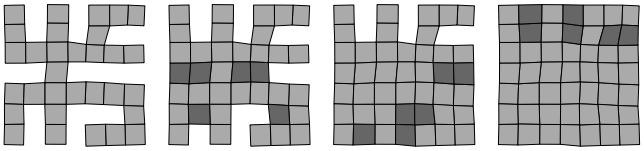


Figure 12: *Progress steps of filling up the maze structure with new quad faces. From left to right after 0, 6, 12 and 18 iterations.*

5. Extensions

This section introduces two techniques to advance the algorithms described in Sections 3 and 4 with the ability to produce feature-aligned and more regular mesh topology.

5.1. Feature Alignment with Robust Principal Axes

The orientation and layout of our final mesh is founded in the underlying kd-tree and its axis-aligned splits. While organic shapes as the busts in Figure 16 often benefit from this superimposed orientation, an adaptive alignment is suitable in other scenarios e.g. as shown for the finger in Figure 13.

Mesh flow alignment to the scanned object's shape is an important feature in many quad (re)meshing applications. Our default axis-aligned setup is easily extendable to incorporate feature alignment as well. Therefore, we introduce alignment layers in our kd-tree: Instead of axis-aligned splits, each node determines a unique orientation for its subset of the point cloud. Four nodes on level 3 are illustrated with their orientation axes in Figure 13e. We employ the method introduced by Liu et al. [LR09] for determining robust principal axes along which the kd-tree splits are performed. Decedents of alignment nodes may inherit the orientation of their parent nodes for subsequent recursive splits along the same axes or determine their own orientations if necessary.

Figure 13 compares reconstructions with different algorithms based on the same point cloud. The field-alignment algorithms (a) and (b) natively align mesh topology to principal directions of the oriented point cloud samples. The official implementations of Jakob et al. [JTPSH15] and Schertler et al. [STJ*17] crashed if the input point cloud did not feature faces or normals. Therefore, artificial normals were created using Meshlab's neighborhood-based algorithm with default settings. For comparability, the algorithms were tuned to produce a target resolution of 2k faces. Due to the low and varying point cloud resolution the algorithms of Jakob et al. (13a) *degraded gracefully* with a gap close to the finger tip and Schertler et al. (13b) produced okay-ish results only around the knuckle joint. With default settings, the application of Schertler et al. is able to recover the finger but fails if tuned to match this mesh resolution.

Our default solution with axis-aligned splits (13c) is a watertight quad mesh but the mesh flow is not yet aligned to the fingers structure. A feature aligned result of our method using the automatically oriented nodes from (e) is shown in Figure 13d where our quad mesh follows the fingers shape.

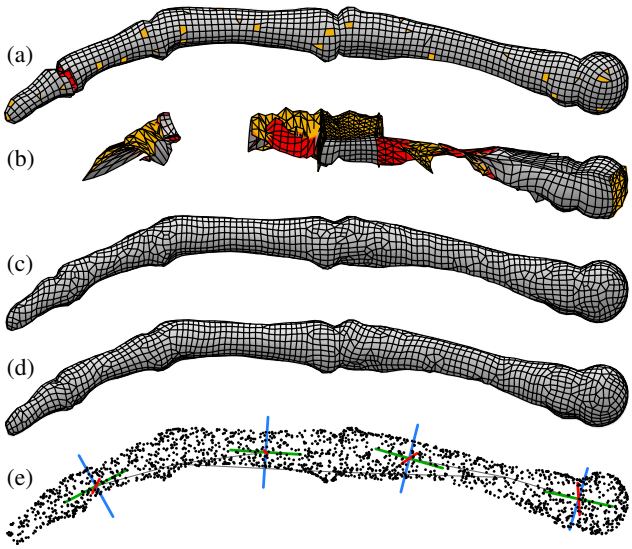


Figure 13: *(a) Bones of a finger reconstructed with [JTPSH15], (b) [STJ*17], (c) Ours w/ axis-aligned splits, (d) Ours w/ adaptive splits, (e) Point cloud and automatically aligned nodes used in (d). Legend: ■ normal quad, ■ backfacing, ■ non-quad.*

5.2. Additional Mesh Optimization

To improve mesh quality we apply optimizations in two stages: The first step is a rectification operation applied locally on each quad

tile before the tile interconnection in Section 4.1. Therefore, vertices of a tile are repositioned around its initial center of gravity by leveling out the diagonals of the tile. With increasing number of iterations, tiles more and more approximate a square shape. Due to more regularly shaped tiles, this step has actual influence on the interconnection result afterwards.

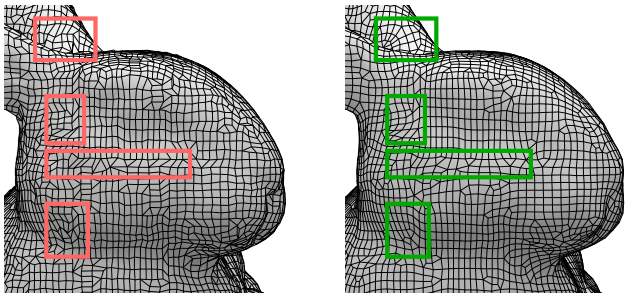


Figure 14: *Our raw output mesh on the left and after regularization and constrained smoothing on the right. Stanford Bunny [Sta14].*

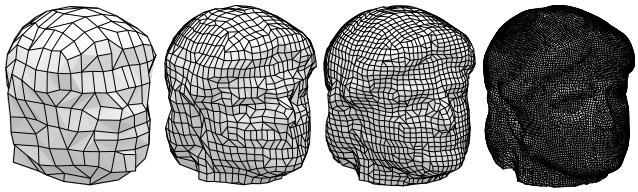
After the final step, when the mesh is filled up and watertight we can apply a constrained smoothing operator. Vertices are moved with respect to their mesh neighborhood but constrained by their affiliated point samples. This relaxation has no impact on the mesh topology. A comparison of a raw output mesh and an optimized result is shown in Figure 14. Colored boxes point out improvements due to the first regularization step, with improved mesh flow.

6. Discussion

In this section we discuss the performance and capabilities of our proposed concept. Results on real and synthetic data are featured in Section 6.1, comparisons to state-of-the-art quad-meshing procedures are documented in Section 6.2.

6.1. Results

Examples for meshing with different resolutions are illustrated in Figure 15. With a fixed number of real 3D scan samples, the split condition k was lowered from left to right. This has direct influence on the number of tree levels and therefore the number of vertices and tiles of the mesh. As one can observe, the rough shape and key features are captured in all resolutions.



k=500, n=8 k=100, n=10 k=50, n=11 k=10, n=14

Figure 15: *Lower split conditions k increase the tree depths n and resulting mesh resolution. Minerva Scan [Bol09], 98.5k samples.*

The images shown in Figure 16 illustrate results of simulated 3D scan meshing. An arbitrary number of surface samples is distributed randomly over given 3D models. To approximate 3D scan results with noise, all samples were perturbed with random jitter vectors. Our resulting meshes form a complete watertight manifold of genus zero. Although axis-aligned kd-tree splits were used, the organic and rounded nature of the surfaces is captured nonetheless. Insufficient sample density on fine details may cause mesh artifacts as pointed out with the red box. The green box shows a flawless reconstruction of the same region when sampled with higher density.

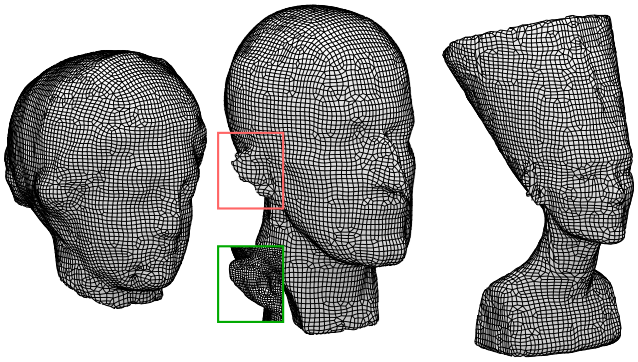


Figure 16: *Meshing results of the Igea Artifact, Max Planck [DFRS03] and Nefertiti [NAB] busts. Sampling artifacts (red) on fine structures are resolved with higher sample density (green).*

Our results on a real 3D scan are shown in Figure 17 Since the angel is captured only from the front, the point cloud's resolution decreases on surface regions which are close to parallel to the view direction as on the side of the model. Our result mesh adapts this varying resolution with quads of different size. The scan features blind spots under the chin and arms without any samples. Despite this depth discontinuity and lack of data, our algorithm recovers a continuous surface. The object's back-plane was removed manually for illustration purposes.

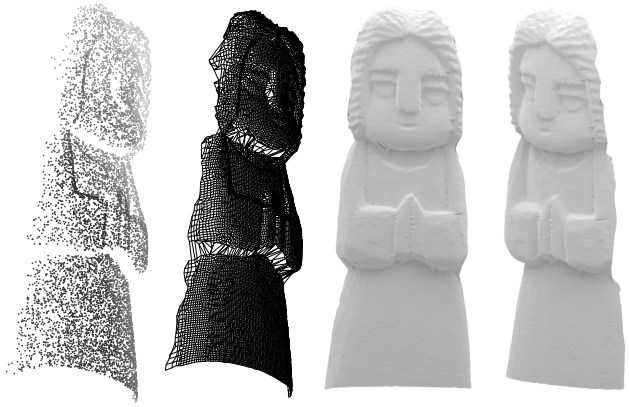


Figure 17: *Reconstruction of a 3D scan [GRL17] with 1.4M samples (only 10k visualized). Our mesh is adaptive to the scan's resolution and covers even the empty areas under the chin and arms.*

The examples shown in Figure 18 demonstrate the impact of the input's orientation on our result meshes. Whereas a) was sampled as-is from a given axis-aligned model, b) was rotated by 45° on the vertical axis prior to the following axis-aligned kd-tree subdivision. The resulting meshes differ, but neither drastically suffered from the assumed axis-alignment. c) and d) show examples derived

from the same oriented input as a) and b) respectively but with an alignment-node, as introduced in Section 5.1, on tree level 0. This allows the subdivision to start on an approximately identically oriented input and c) ≈ d). Small differences may still occur due to the RANSAC-based algorithm [LR09] for determining the orientation.

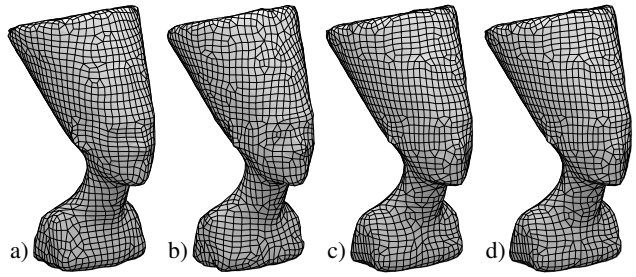


Figure 18: *Influence of orientation: a) Object sampled as-is, b) rotated vertically by 45°, c) same as a), d) same as b). But c) and d) both include an alignment node (Section 5.1) on level 0. Therefore our results become invariant to the input orientation and c) ≈ d).*

6.2. Comparison to state-of-the-art quad-meshing procedures

A comparison of our work against the state-of-the-art for quad-meshing from point clouds is illustrated in Figure 19. All algorithms used the same input point cloud and were tuned to produce the same output resolution. As required for the available implementations of Jakob et al. and Schertler et al., normals were again estimated from the basic point cloud in a preceding step. Our procedure is adaptive to the underlying point cloud density and therefore, the resulting mesh might appear not as homogeneous as the competitor's. This can be observed in the neck region, where our result features bigger tiles to compensate for lower sample density and the others just produced holes. The same effect is illustrated in Figure 20 which explicitly compares behavior under varying sample density. Zoom-in boxes on the meshes in Figure 19 point out regions where sample density was insufficient for this level of detail or high curvature. This results as holes in the mesh of Jakob et al. and an increased number of non-quads in the mesh of Schertler et al., whereas ours adapts to the input and returns a watertight quad-only mesh. As the curvature comparison in the bottom rows show, our result models coarse and fine details with comparable quality.

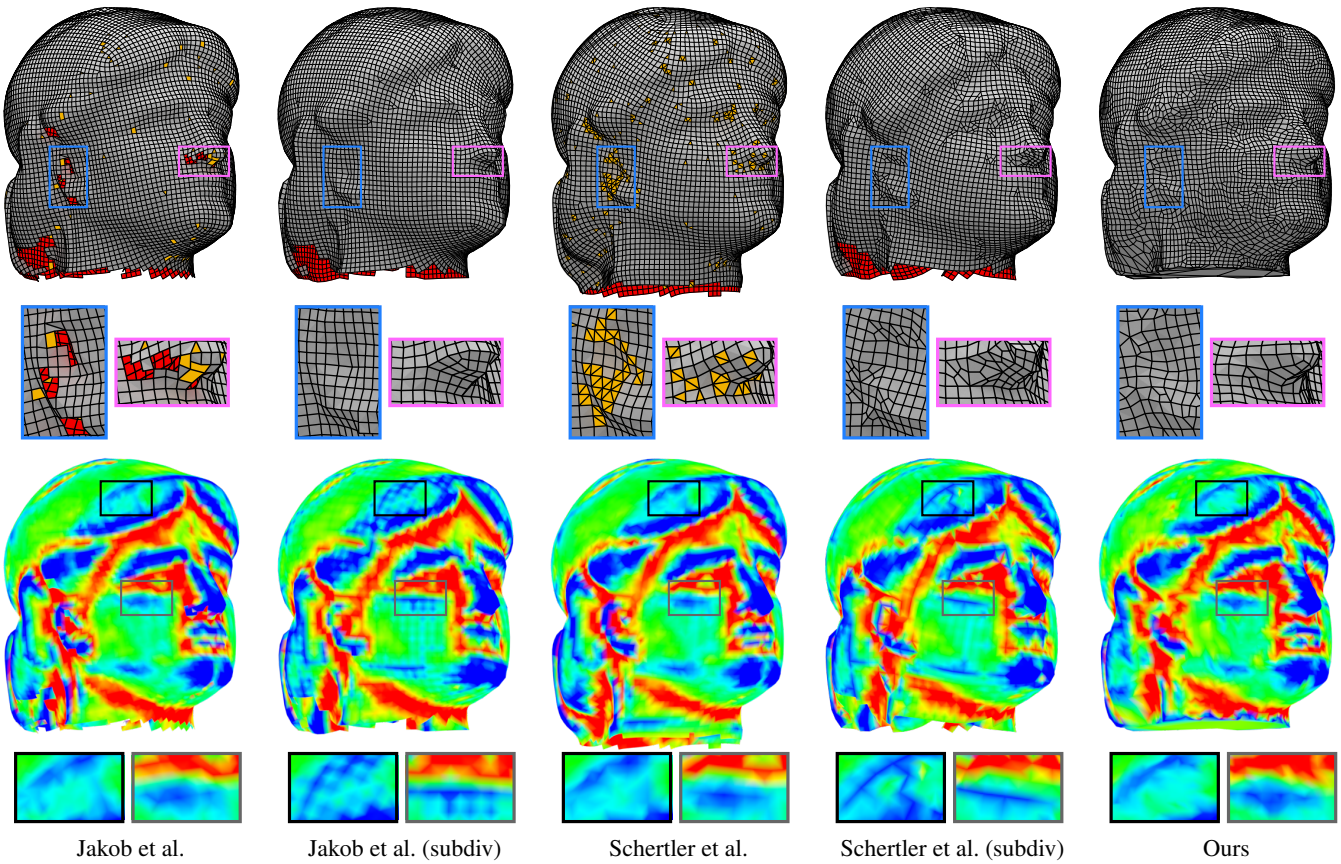


Figure 19: *Reconstructions on the same point cloud (with artificial normals for Jakob et al. and Schertler et al.) for a target resolution of ∼8k faces. Backfacing tiles are shaded red to show holes, non-quads in orange. The bottom row shows color-coded curvature. subdiv columns show results from coarser meshes, which were then subdivided to also feature quad faces only and to match the target resolution. As pointed out in the bottom row, subdivision may introduce unwanted curvature artifacts. Table 1 lists a statistical comparison of these meshes.*

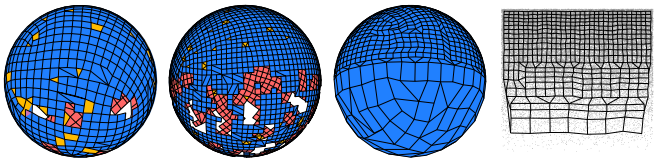
The *subdiv* columns show results from coarser mesh results which were subdivided. The result of Jakob et al. was subdivided using their official implementation, the result of Schertler et al. with Blender. This additional step also yields quad-only meshes but on the other hand introduces curvature artifacts, which do not occur in results that natively meet the target resolution.

	Vertices Valence			Vertices per Face		
	≤ 3	4	$5 \leq$	≤ 3	4	$5 \leq$
Jakob et al.	3.3%	95.1%	1.6%	1.3%	98%	0.7%
Schertler et al.	1.5%	91.5%	7%	7.2%	92.8%	
Ours	8.2%	84.3%	7.5%		**100%**	

Table 1: *Statistical evaluation of result meshes shown in Figure 19, accounting only native target-resolution meshes.*

A quantitative comparison of the results from Figure 19 is listed in Table 1. On one hand, the mesh of Jakob et al. counts the least amount of singularities but on the other hand left some open regions in the mesh. The mesh of Schertler et al. was able to cover such regions but features an increased amount of non-quads in these areas. Besides that our mesh natively features quad tiles only, it is also the only actually closed manifold and therefore perfectly suitable for 3D printing or volume rendering without further steps required.

Adaptivity: Two extreme examples with varying sample density are illustrated in Figure 20: The input point cloud sphere for the examples on the left features 90% of all samples on the upper and 10% on the lower hemisphere. Whereas field-aligned approaches for quad-dominant meshes [JTPSH15, STJ*17] failed to come up with useable results in this case, our algorithm was able to handle a density gradient even as steep as at this equator line. Our adaptive mesh tends to become a bit more inharmonious but recovers a closed manifold nevertheless. Examples in Figure 13 show another comparison focused on feature alignment where competing results also contain holes due to the same reason.



Jakob et al. Schertler et al. Ours Ours

Figure 20: *Meshing results under mixed sample density. Left: 100k samples with 90% on the upper and 10% on the lower hemisphere. Right: A unit plane with inverse quadratic decreasing density.*

The 2D plane on the right of Figure 20 was sampled uniformly in x and with inverse quadratic decreasing density in y. Nevertheless, our mesh adapts to this change in sample density flawlessly.

6.3. Limitations

The first part of our procedure, which is the kd-tree decomposition, subsampling and hierarchical tile interconnection succeeds to produce a coherent maze of quad tiles on any given input. However, the default reconstruction algorithm introduced in Section 4.2, converges only on object structures of genus zero properly. Per definition, the algorithm incrementally fills up dead ends of the maze structure until the surface is closed. For objects with holes (genus larger than zero) the mesh will eventually run out of dead ends to fill up. In Figure 21 b) the dead ends are actually the cross sections through the torus. The algorithm will start to close up these cross sections with caps and converge to the c-shape of genus zero shown in c). However, after insertion of the two quads pointed out in d), the algorithm is able to finish the mesh with genus one.

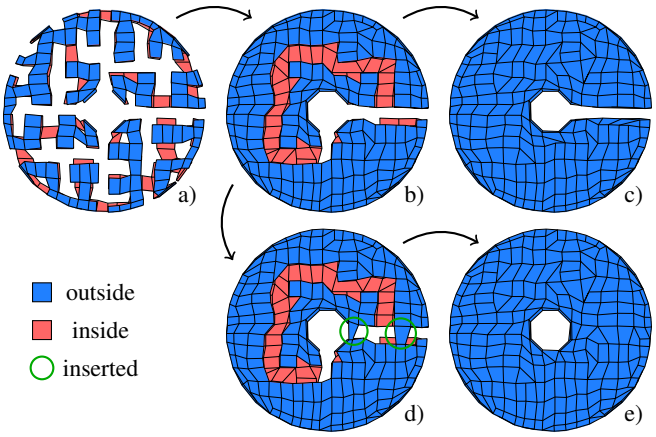
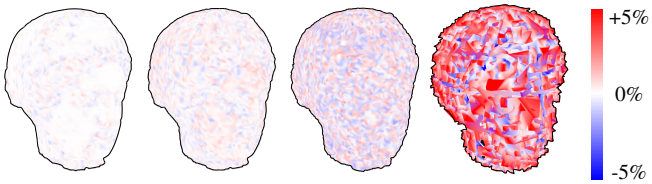


Figure 21: *With our default algorithm the torus in the upper row is resolved with genus zero. After insertion of the two quads pointed out in green, the torus is resolved with genus one.*

A critical point in every signal reconstruction is the issue of sample density versus highest feature frequency. As one can easily comprehend, samples that are placed too sparse can not contribute to reconstruct fine details. In our case, the sampling artifacts show up as dents or wrinkles in the final mesh, e.g. at very small features. Examples of artifacts in meshes can be observed at the ears of the two busts on the right in Figure 16. The thickness of the ears and therefore the distance of vertices on these surfaces is less than the regular tile size. These unfavorably sampled tiles are interconnected nevertheless, causing the edgy outer ear shapes. Since these artifacts can be justified as sampling issues, increasing the sample density and therefore mesh resolution can resolve these situations. Other small features as dents or bulges are captured and reconstructed with the available resolution: See the small dent on the chin of Igea or the characteristic applications on Nefertiti's crown.



0% 0.1% 1% 10% in
0.1522% 0.3292% 0.7197% 2.8108% out

Figure 22: *Mesh results: 500k samples perturbed with different magnitudes of input noise (in). Absolute mean errors of the reconstruction compared against ground truth are listed in the out-row.*

Results under varying levels of input noise are illustrated in Figure 22. For this test, 500k samples from the Igea artifact (Figure 16) are perturbed with random jitter noise vectors of varying amplitude. The noise scale amplitudes are given below the plots in the *in*-row. Values listed in the *out*-row are the means of absolute errors against ground truth, compared to the object bounding box's diagonal serving as 100%. With such heavy noise as in the rightmost example, it becomes unclear if samples belong to the same surface and how it can be subsampled. The fill-up algorithm will do its best on the ill-placed tiles but can not guarantee to avoid self intersections.

6.4. Performance and Complexity

The algorithm for hierarchical interconnection in Section 4.1 derives its complexity from the size of the binary kd-tree. However, connectable edges are prefiltered and sorted based on normal directions and reasonable distance to each other. For filtering, the algorithm once has to process possible edge pairs, but due to the simple geometric nature of this operation, the theoretical $\mathcal{O}(n^2)$ complexity is neglectable, nevertheless. The main benefit is gained by limiting computation time and memory consumption to a fixed maximum for the actual matching. The default fill-up approach from Section 4.2 maintains a priority queue of edge triples. Each iteration introduces one new edge by closing and removing three open ones. Therefore, this algorithm operates in linear time by effectively removing two edges each step. There is no overhead due to global optimization, since the maze structure already provides a global basis and further only locally optimal solutions are required.

The result shown in Figure 17 is one of our largest reconstructions with more than 1.4 million surface samples. Nevertheless, the tree decomposition, subsampling and tile interconnection was resolved in less than one minute with unoptimized Python code on a single CPU. With the same conditions, the fill-up routine consumed less than 10 seconds to resolve 2^{14} open edges.

6.5. Outlook

Results in this paper where our procedure is applied as remeshing tool are based on samples randomly distributed over the surface. However, the limitations in Section 6.3 caused by sampling issues can be resolved very easily with a more sophisticated sampling strategy as demonstrated in Figure 16. Samples placed in correlation to surface curvature would cover small features with more samples, so they could be recovered with appropriate mesh resolution. Furthermore, in a remeshing application an input mesh is given and can be used to further optimize our remeshed vertices.

Figure 21 illustrates a method to overcome the topology issue for reconstruction of objects with a genus larger than zero. This could be automated using our algorithm for hierarchical interconnection, introduced in Section 4.1. The algorithm determines reasonable pairs of edges based on their normal orientation and distance to each other. Our incremental fill-up algorithm from Section 4.2 could run interleaved with one of these steps every nth iteration in order to handle objects of even higher genus.

Many strengths of our technique can be derived from features of the underlying kd-tree. Our created maze structure acts as a very powerful abstraction of a point cloud. It is one coherent mesh of $2^{i-1}-1$ quad tiles, guaranteed not to intersect itself and is surrounded by one coherent path of 2^i open edges. The kd-tree furthermore represents meaningful neighborhood affiliations and adjacency information. This data could be valuable input for a diversity of machine learning approaches featuring convolutional neural networks, e.g., for classification, mapping or segmentation.

7. Conclusion

This publication presents a novel and innovative way to deal with the task of point cloud reconstruction. Our framework creates watertight quad meshed on noisy point clouds of a scanned surface without any topology information or estimated normal orientations. As demonstrated in exemplary scenarios with respect to varying sample density, our method recovers watertight manifold quad-only surfaces where state-of-the-art field-aligned approaches tend to fail. However, when compared to these methods supplied with sufficiently uniform samples, there is still potential to improve our results in terms of mesh regularity and number of singularities. Robustness to noise, complex topology and minimizing approximation errors are important topics that lie beyond the scope of this paper and have to be explored in future work. Nevertheless, since our resulting meshes are natively free of holes, they are trivially suitable for 3D printing or volume rendering where closed meshes are crucial. Our novel approach to construct tile vertices is not only suitable to approximate rounded organic structures but can also robustly recover coplanar regions. Independent from the approximated mesh, axis-aligned kd-tree splits favor regularly distributed vertices. However, special tree nodes also allow to automatically determine the orientation of point cloud segments and adapt the split direction for feature-aligned mesh topology.

References

[BCE*13] BOMMES D., CAMPEN M., EBKE H.-C., ALLIEZ P., KOBBELT L.: Integer-grid maps for reliable quad meshing. *ACM Trans. Graph. 32*, 4 (July 2013), 98:1–98:12. URL: http://doi.acm.org/10.1145/2461912.2462014, doi:10.1145/2461912.2462014. 3

[BLP*13] BOMMES D., LÉVY B., PIETRONI N., PUPPO E., SILVA C., TARINI M., ZORIN D.: Quad-mesh generation and processing: A survey. *Computer Graphics Forum 32*, 6 (2013), 51–76. Article first published online: 4 MAR 2013, DOI: 10.1111/cgf.12014. URL: http://vcg.isti.cnr.it/Publications/2013/BLPPSTZ13a. 2

[Bol09] Dept. of Math. Bologna University Scan Repository, 2009. Online; accessed Oktober-2017, http://www.dm.unibo.it/~morigi/homepage_file/research_file/scan_db/res_scan.html. 1, 7

[BTS*14] BERGER M., TAGLIASACCHI A., SEVERSKY L. M., ALLIEZ P., LEVINE J. A., SHARF A., SILVA C. T.: State of the Art in Surface Reconstruction from Point Clouds. In *Eurographics 2014 - State of the Art Reports* (2014), Lefebvre S., Spagnuolo M., (Eds.), The Eurographics Association. doi:10.2312/egst.20141040. 2

[BZK09] BOMMES D., ZIMMER H., KOBBELT L.: Mixed-integer quadrangulation. *ACM Trans. Graph. 28*, 3 (July 2009), 77:1–77:10. URL: http://doi.acm.org/10.1145/1531326.1531383, doi:10.1145/1531326.1531383. 3

[CC78] CATMULL E., CLARK J.: Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-aided design 10*, 6 (1978), 350–355. 2

[DFRS03] DeCarlo D., Finkelstein A., Rusinkiewicz S., Santella A.: Suggestive Contour Gallery, 2003. URL: http://gfx.cs.princeton.edu/proj/sugcon/models/. 7

[DSC09] Daniels J., Silva C. T., Cohen E.: Semi-regular quadrilateral-only remeshing from simplified base domains. In *Computer Graphics Forum* (2009), vol. 28/5, Wiley Online Library, pp. 1427–1435. 3

[ESCK16] Ebke H.-C., Schmidt P., Campen M., Kobbelt L.: Interactively controlled quad remeshing of high resolution 3d models. *ACM Trans. Graph. 35*, 6 (Nov. 2016), 218:1–218:13. URL: http://doi.acm.org/10.1145/2980179.2982413, doi:10.1145/2980179.2982413. 3

[FDCO03] Fleishman S., Drori I., Cohen-Or D.: Bilateral mesh denoising. In *ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), SIGGRAPH '03, ACM, pp. 950–953. URL: http://doi.acm.org/10.1145/1201775.882368, doi:10.1145/1201775.882368. 3

[GRL17] Groh F., Resch B., Lensch H. P. A.: *Multi-view Continuous Structured Light Scanning*. Springer International Publishing, Cham, 2017, pp. 377–388. URL: https://doi.org/10.1007/978-3-319-66709-6_30, doi:10.1007/978-3-319-66709-6_30. 7

[HDD*92] Hoppe H., DeRose T., Duchamp T., McDonald J., Stuetzle W.: Surface reconstruction from unorganized points. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1992), SIGGRAPH '92, ACM, pp. 71–78. URL: http://doi.acm.org/10.1145/133994.134011, doi:10.1145/133994.134011. 2

[HK06] Hornung A., Kobbelt L.: Robust Reconstruction of Watertight 3D Models from Non-uniformly Sampled Point Clouds Without Normal Information. In *Symposium on Geometry Processing* (2006), Sheffer A., Polthier K., (Eds.), The Eurographics Association. doi:10.2312/SGP/SGP06/041-050. 2

[JTPSH15] Jakob W., Tarini M., Panozzo D., Sorkine-Hornung O.: Instant field-aligned meshes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH ASIA) 34*, 6 (Nov. 2015). doi:10.1145/2816795.2818078. 2, 6, 9

[KBH06] Kazhdan M., Bolitho M., Hoppe H.: Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2006), SGP '06, Eurographics Association, pp. 61–70. URL: http://dl.acm.org/citation.cfm?id=1281957.1281965. 2

[KNP07] Kælberer F., Nieser M., Polthier K.: Quadcover - surface parameterization using branched coverings. *Computer Graphics Forum 26*, 3 (2007), 375–384. URL: http://dx.doi.org/10.1111/j.1467-8659.2007.01060.x, doi:10.1111/j.1467-8659.2007.01060.x. 3

[KNSS07] Kalogerakis E., Nowrouzezahrai D., Simari P., Singh K.: Construction of curvature-aligned meshes from point clouds. *Technical Report CSRG-569* (2007). 2

[Knu95] Knupp P.: Mesh generation using vector-fields. *J. Comput. Phys. 119*, 1 (June 1995), 142–148. URL: http://dx.doi.org/10.1006/jcph.1995.1122, doi:10.1006/jcph.1995.1122. 3

[LCOL07] Lipman Y., Cohen-Or D., Levin D.: Data-Dependent MLS for Faithful Surface Approximation. In *Geometry Processing* (2007), Belyaev A., Garland M., (Eds.), The Eurographics Association. doi:10.2312/SGP/SGP07/059-067. 2

[LCOLTE07] Lipman Y., Cohen-Or D., Levin D., Tal-Ezer H.: Parameterization-free projection for geometry reconstruction. In *ACM SIGGRAPH 2007 Papers* (New York, NY, USA, 2007), SIGGRAPH '07, ACM. URL: http://doi.acm.org/10.1145/1275808.1276405, doi:10.1145/1275808.1276405. 2

[LLZ*11] Li E., Lévy B., Zhang X., Che W., Dong W., Paul J.-C.: Meshless quadrangulation by global parameterization. *Computers & Graphics 35*, 5 (2011), 992 – 1000. doi:https://doi.org/10.1016/j.cag.2011.05.003. 2

[LPC*00] Levoy M., Pulli K., Curless B., Rusinkiewicz S., Koller D., Pereira L., Ginzton M., Anderson S., Davis J., Ginsberg J., Shade J., Fulk D.: The digital michelangelo project: 3d scanning of large statues. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2000), SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., pp. 131–144. URL: http://dx.doi.org/10.1145/344779.344849, doi:10.1145/344779.344849. 2

[LR09] Liu Y.-S., Ramani K.: Robust principal axes determination for point-based shapes using least median of squares. *Comput. Aided Des. 41*, 4 (Apr. 2009), 293–305. URL: http://dx.doi.org/10.1016/j.cad.2008.10.012, doi:10.1016/j.cad.2008.10.012. 6, 8

[MCAG07] Mousa M.-H., Chaine R., Akkouche S., Galin E.: Efficient spherical harmonics representation of 3d objects. In *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications* (Washington, DC, USA, 2007), PG '07, IEEE Computer Society, pp. 248–255. URL: http://dx.doi.org/10.1109/PG.2007.19, doi:10.1109/PG.2007.19. 3

[NAB] Nelles J. N., Al-Badri N.: Nefertiti Head Model. URL: http://nefertitihack.alloversky.com/. 7

[OBA*03] Ohtake Y., Belyaev A., Alexa M., Turk G., Seidel H.-P.: Multi-level partition of unity implicits. In *ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), SIGGRAPH '03, ACM, pp. 463–470. URL: http://doi.acm.org/10.1145/1201775.882293, doi:10.1145/1201775.882293. 2

[RLL*06] Ray N., Li W. C., Lévy B., Sheffer A., Alliez P.: Periodic global parameterization. *ACM Trans. Graph. 25*, 4 (Oct. 2006), 1460–1485. URL: http://doi.acm.org/10.1145/1183287.1183297, doi:10.1145/1183287.1183297. 3

[SSW09] Silva C. T., Scheidegger C. E., Wang H.: Bandwidth selection and reconstruction quality in point-based surfaces. *IEEE Transactions on Visualization & Computer Graphics 15* (2009), 572–582. doi:doi.ieeecomputersociety.org/10.1109/TVCG.2009.13. 2

[Sta14] The Stanford 3D Scanning Repository, 2014. Online; accessed Oktober-2017, http://graphics.stanford.edu/data/3Dscanrep/. 2, 7

[STJ*17] Schertler N., Tarini M., Jakob W., Kazhdan M., Gumhold S., Panozzo D.: Field-aligned online surface reconstruction. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 36*, 4 (July 2017). doi:10.1145/3072959.3073635. 2, 6, 9

[Tau95] Taubin G.: Curve and surface smoothing without shrinkage. In *Computer Vision, 1995. Proceedings., Fifth International Conference on* (1995), IEEE, pp. 852–857. 3

[TPC*10] Tarini M., Pietroni N., Cignoni P., Panozzo D., Puppo E.: Practical quad mesh simplification. *Computer Graphics Forum (Special Issue of Eurographics 2010 Conference) 29*, 2 (2010), 407–418. URL: http://vcg.isti.cnr.it/Publications/2010/TPCPP10. 3

[WHG*15] Wu S., Huang H., Gong M., Zwicker M., Cohen-Or D.: Deep points consolidation. *ACM Trans. Graph. 34*, 6 (Oct. 2015), 176:1–176:13. URL: http://doi.acm.org/10.1145/2816795.2818073, doi:10.1145/2816795.2818073. 3

[ZLGH10] Zhang L., Liu L., Gotsman C., Huang H.: Mesh reconstruction by meshless denoising and parameterization. *Computers & Graphics 34*, 3 (2010), 198 – 208. Shape Modelling International (SMI) Conference 2010. doi:https://doi.org/10.1016/j.cag.2010.03.006. 2