The Visual Computer (2021) 37:2725–2739
https://doi.org/10.1007/s00371-021-02183-6

ORIGINAL ARTICLE

Be water my friend: mesh assimilation

Dennis R. Bukenberger[1] · **Hendrik P. A. Lensch**[1]

Accepted: 28 May 2021 / Published online: 2 July 2021

Abstract
Inspired by the ability of water to assimilate any shape, if being poured into it, regardless if flat, round, sharp, or pointy, we present a novel, high-quality meshing method. Our algorithm creates a triangulated mesh, which automatically refines where necessary and accurately aligns to any target, given as mesh, point cloud, or volumetric function. Our core optimization iterates over steps for mesh uniformity, point cloud projection, and mesh topology corrections, always guaranteeing mesh integrity and ϵ-close surface reconstructions. In contrast with similar approaches, our simple algorithm operates on an individual vertex basis. This allows for automated and seamless transitions between the optimization phases for rough shape approximation and fine detail reconstruction. Therefore, our proposed algorithm equals established techniques in terms of accuracy and robustness but supersedes them in terms of simplicity and better feature reconstruction, all controlled by a single parameter, the intended edge length. Due to the overall increased versatility of input scenarios and robustness of the assimilation, our technique furthermore generalizes multiple established approaches such as ballooning or shrink wrapping.

Keywords Point cloud reconstruction · 3D scan · Remeshing · Shape assimilation · Meshing

1 Introduction

Without the need for mappings, parameterizations, or specifically trained machine learning models, our approach presents a simple yet powerful extension in the field of geometric surface reconstruction techniques. We demonstrate how our algorithm can top similar state-of-the-art approaches, e.g., when reconstructing fine details and sharp feature edges, in terms of mesh uniformity or versatility of supported input data. Our method is based on the concept of water being quite formless and able to assimilate any shape if being poured into it. Starting from a small initial triangle mesh, the object can grow, shrink, and locally refine to assimilate a target shape with controllable precision. Vertices of the initial mesh extend rather autonomously along surface normals until they reach the target hull space. The expanding surface mesh is supplemented with suitably interpolated vertices where it is stretched the most. In contrast with the surface tension of water, the inter-vertex energy optimization promotes mesh uniformity but without enforcing smoothness priors, often found in other ballooning concepts. Once the vertices are close enough to the target hull, they individually transition from mesh growth mode to a projection scheme. Due to the bilateral weighting concept of the projection, vertices are effectively pulled into intersecting tangent planes of the hull, thus allowing for accurate reconstructions of small detail and sharp edges. The optimization of vertices is an iteration process, but individual for each vertex and can therefore be executed on the GPU as a massively parallelized operation. Per design, the algorithm assimilates to shape rather than a strict type of input data, which can be given as mesh, point cloud, signed distance, or other volumetric function. With its frugal input requirements and a minimal set of parameters to be adjusted, our method simplifies the rather complex nature of other established techniques; it is situated somewhere between the exhaustively studied fields of shape approximation [12], surface remeshing [1], and point cloud reconstruction [5].

1.1 Contributions

Our approach fulfills all criteria which one would ask of a state-of-the-art reconstruction technique:
Guarantee for closed, watertight manifold meshes, the ability to cope with varying input density, robustness to noise and outliers as well as to missing data and control over higher genus topology. Moreover, we can summarize our

✉ Dennis R. Bukenberger
dennis.bukenberger@uni-tuebingen.de

[1] University Tübingen, Tübingen, Germany

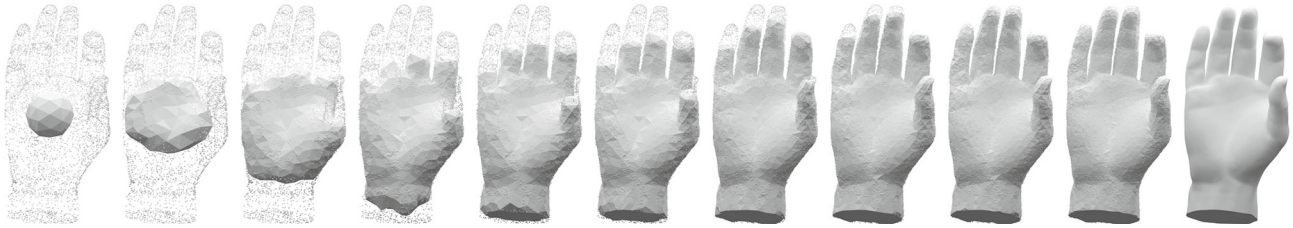


Fig. 1 Progress of our method: A low-poly initialization mesh grows within the target point cloud. The mesh first approximates the rough shape and further progresses by assimilating and refining to reconstruct fine detail

main contributions: • **Mesh uniformity** is achieved as our mesh unravels with an inter-vertex energy, striving to equalize distances between vertices. • **Adaptive resolution** can be realized by locally adapting the specified target edge length to the provided sample density. • **Sharp features** are reconstructed, as the second vertex optimization step is designed to minimize projected distances to surface tangents; thus, the vertices snap on close-by edges and corners. This is where most state-of-the-art methods still lack precision. • **Arbitrary initialization** meshes are possible, as demonstrated with various examples. • **Versatile input** data are supported, as our method is not bound to point clouds and can easily handle meshes as well as volume data and signed distance fields, currently explored in neural surface representations [32,33].

1.2 Related work

For decades, shape reconstruction has been and is still an actively researched topic.

While the ideas of ballooning or shrink-wrapping are not exactly new [17], we are yet to see its full potential. In some aspects, our method follows similar principles like the concept of *Competing Fronts* [35], which is also a coarse-to-fine point cloud reconstruction technique, using growing mesh geometry. But instead of whole mesh fronts, which get frozen at some point, our method operates on an individual vertex basis so that the mesh remains flexible during the optimization. This allows for smoothing out irregularities, without the need for remeshing in every other iteration. Our hull projection scheme furthermore resolves the issue of reconstructing creases and sharp feature edges, an open problem for *Competing Fronts*. Unfortunately, a direct comparison was not possible as there is no reference implementation or result data available. Nevertheless, the concept gained some recent attention with the concept of *Cooperative Evolutions* [31], combining two mesh fronts enclosing on the point cloud from the in- and outside in parallel. A first draft utilizing this idea in a learning based approach was proposed with *Point2Mesh* [20]. A thorough analysis of the actual distinctions of our approach to other ballooning concepts is featured in our discussion in Sect. 3.2.

Conceptually different approaches include the popular Screened Poisson Surface Reconstruction (SPSR) [24–27], which is probably one of the most commonly used point cloud reconstruction methods. In direct comparison, however, SPSR tends to produce rather smoothed out results, whereas ours generally appear sharper and capture finer details, even at lower resolution. Due to the octree nature of the SPSR approach, the resulting triangulation is quite inhomogeneous, while ours approaches both a uniform distribution and Delaunay-like connectivity. Scale Space Meshing (SSM) [16] is an approach that really takes advantage of modern high precision 3D scanners and is able to faithfully reconstruct even very fine details. However, as our comparison shows, sharp edges are not captured very well. The marching cubes algorithm [30] is often used in a medical context to reconstruct surfaces from volumetric data or to remesh existing surfaces. Our algorithm is able to handle both as it can be used on signed distance fields (SDF) as well as on existing meshes with the advantage of producing more evenly distributed vertices and triangle faces. Both Poisson and Marching Cubes are able to recover surfaces but fall short in terms of triangle mesh homogeneity, a key contribution of our approach. Similar deficiencies can be attributed to α-shape- [18], Voronoi-based [2–4,7], or the ball pivoting algorithm (BPA) [6]. For these algorithms, the achievable mesh geometry-, and sometimes topology-, quality crucially depends on the density and uniformity of the given input point set and sometimes requires prior outlier filtering or noise smoothing steps. Our approach is quite invariant to these artifacts, as it is able to cope with noise and can either ignore or incorporate varying sample densities. Impressing developments in quad-mesh reconstructions such as Instant Field Aligned Meshes (IFAM) [23] or Online Surface Reconstruction (OSR) [34] produce nice feature-aligned mesh structures, but often struggle with low or varying sample density as demonstrated in comparison with our results. Most recently, learning-based approaches also joined the reconstruction game [32,33] but often require specifically trained models on an explicit object class. This is no longer a requirement with P2M, which reconstructs point clouds via shrink wrapping and basically learns from the input itself. But

their results also lack precision in corners and sharp object features.

2 Methods

As illustrated in Fig. 1, the goal of our algorithm is to assimilate to a target shape, starting from a simple initialization mesh. Therefore, the basis mesh first approximates the rough shape of the input; finer details are recovered as the optimization continues. This assimilation process follows a simple optimization scheme, iterating over the following five steps, performed in one iteration cycle:

I. Subdivide the mesh where necessary
II. Equalize inter-vertex distances
III. Mesh expansion and hull projection
IV. Fix mesh irregularities
V. Add tunnels to increase genus (optional)

Termination criteria for the optimization can be specified, either as a target edge length l_t for the final mesh or an ϵ-closeness threshold to the target shape.
Each operation is self-contained, meaning the surface remains a closed manifold and valid mesh at all times. Related coarse-to-fine approaches [20,31,35] separate the *mesh development* and *final fitting* phase in their procedure or interleave the assimilation progress with remeshing operations. However, the strength of our method lies in the mixed design of the assimilation (**III.**): Simple mesh growth seamlessly transitions into a projection scheme on an individual vertex basis.

Terminology Our method is first introduced with the focus on point cloud reconstruction; the adaptation to other input scenarios is later addressed in Sect. 2.1. In the following, we associate *vertices* v_i with the growing mesh structure and *samples* $\dot{s}$ with the input point cloud. Variables related to samples are denoted with an over-set dot. Further, we observe local 1-ring neighborhoods around vertices: For a vertex v_i, adjacent mesh neighbors are grouped in N_i where $k_i = |N_i|$ gives the number of neighbors. The average edge length around vertex v_i is therefore given as $l_i = \frac{1}{k_i} \sum_{j \in N_i} |v_i - v_j|$. Vertex normals n_i are derived from the face normals in this 1-ring neighborhood: The normalized portions of corner angles in the triangle fan are used to weigh their normals' contribution, respectively. Further, each vertex v_i is associated with a set S_i of 10 closest point cloud samples. This is cheap to initialize with a brute-forced search as the initial mesh only has very few vertices. When starting from a larger base mesh, a suitable data structure may speed up the first nearest neighbor associations. However, once established, this structure can be maintained cheaply using the mesh as a graph: To update the existing sets or create new ones for added vertices (**I.**), we then only query the sample sets in the vertices' 1-ring neighborhoods for potential new nearest neighbors. Further, each sample is interpreted as a splat disk on the implicit hull, represented by a normal $\dot{n}$ and a radius $\dot{r}$. We compute the radius of a sample as half the median distance to its 10 closest point cloud neighbors. While all our results are generated using this simple approach, more advanced splat methods [11,39] may be suitable as well.

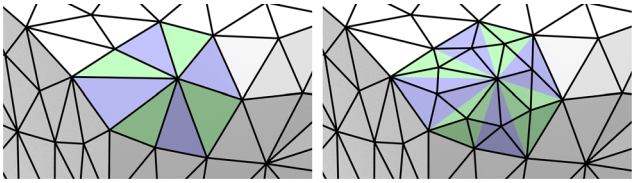


Fig. 2 Mesh refinement: The 1-ring fan of a vertex is $\sqrt{3}$-subdivided. New vertex positions are interpolated as PN triangle centers [38] to maintain local curvature

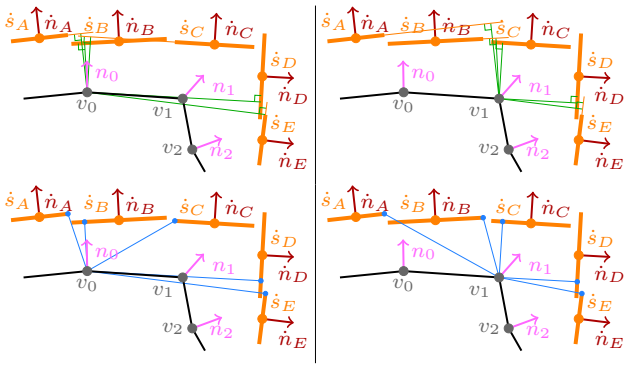


Fig. 3 Point cloud samples $\dot{s}_i$ are shown with their splat disks and normals $\dot{n}_i$. The example shows projections for vertices v_0 (left) and v_1 (right) as well as closest points on splat disks, respectively. Vertices move along the projection direction, weighted by distance and the angle between vertex normal and sample normal, and are pulled into the intersection of tangent planes and thus allow for better edge preservation

I. Refinement The triangle mesh is constantly refined with targeted $\sqrt{3}$-subdivision operations [29]. A vertex qualifies for refinement when its average length l_i of incident edges is larger than the specified target edge length l_t. Or, if ϵ-closeness is specified as the optimization criteria, simply the vertices with the largest l_i get subsequently refined. Therefore, faces around a vertex are replaced as shown in Fig. 2: For each triangle in this 1-ring fan, a new vertex is created along with three new triangles. The new vertices are the interpolated center points of the fan triangles, interpreted as curved PN triangles [38] using vertex normals. This allows us to maintain local curvature and the subdivided geometry smoothly integrates with the surrounding mesh as the outer 1-ring loop edges are not split. This operation increases the valence of the 1-ring neighbor vertices by 1 and may create very obtuse

or pointy triangles. Nevertheless, both issues are fixed within the same cycle by upcoming steps **II.** and **IV.**

II. Equalization In order to obtain the most regular mesh structures with evenly distributed vertices, we employ a simple geometric optimization: Every vertex v_i strives to individually equalize the distances to its direct 1-ring neighbors $v_j \in N_i$, eventually resembling a Delaunay-like triangulation. Equation 1 formulates the update vectors, where $\vec{v}_{ij} = v_j - v_i$ and l_i is the average length of edges incident on v_i.

$$\vec{e}_i = \left(\frac{1}{k_i} \sum_{v_j \in N_i} v_j - l_i \frac{\vec{v}_{ij}}{|\vec{v}_{ij}|} \right) - v_i \tag{1}$$

These geometric increments mainly smooth out dense clusters created by step **I.** and can be applied with a weighted update on vertex positions as $v_i := v_i + \lambda_e \vec{e}_i$. Thus, λ_e allows adjusting the progression speed for mesh homogenization. We achieved the best results with a soft update, using $\lambda_e = 0.1$.

III. Assimilation The goal of this step is to pull vertices to the surface and snap vertices on edges and corners. Once our mesh is close enough to the point cloud hull, a suitable projection technique is applied. However, when starting from a generic base mesh, e.g., a simple sphere inside of, or enclosing, the point cloud, it first has to grow or shrink into a rough approximation of the target shape, so that the projection may take over later. Whereas this is how it is usually done, we propose a mixed procedure, adaptive to each individual vertex.

This step modifies the mesh's geometry by moving vertices with individual update vectors. These vectors $\vec{a}_i$ may embody surface projection (**III.A**) or mesh growth (**III.B**). Equation 2 anticipates which one is to choose, dependent on an individual vertex v_i and its weights $\dot{w}_{ij}$, summed up in $\omega_i = \sum_{j \in S_i} \dot{w}_{ij}$. Other variables are elaborated in the following.

$$\vec{a}_i = \begin{cases} \frac{1}{\omega_i} \sum_{j \in S_i} \dot{w}_{ij} \dot{\vec{v}}_{ij} & \textbf{III.A)} \text{ if } \omega_i > \epsilon \\ n_i \cdot \frac{o_i l_i}{4} & \textbf{III.B)} \text{ else} \end{cases} \tag{2}$$

III.A) Projection For valid weights $\dot{w}_{ij}$, gathered from the point cloud samples associated with vertex v_i, the update vector $\vec{a}_i$ is the weighted sum of v_i's projection vectors $\dot{\vec{v}}_{ij}$ into the samples' tangent planes (Fig. 3, green). Projected points compute as $\dot{v}_{ij} = v_i + \dot{\vec{v}}_{ij}$ with vectors $\dot{\vec{v}}_{ij} = \dot{n}_j \cdot (\dot{s}_j - v_i)\dot{n}_j$ where $\dot{n}_j$ is the normal of point cloud sample $\dot{s}_j$. The bilateral weights $\dot{w}_{ij}$ combine a distance and an angular component. This gives us the advantage of only pulling qualified vertices into edges and corners, while their direct neighbors reside in flat regions nearby. As meaningful

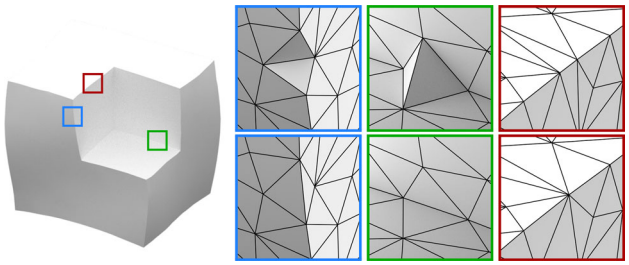


Fig. 4 Mesh improvement operations: The concave (blue) and convex (green) edges (standing out in contrast with their neighbors) are fixed by rotation. A short (red) edge is collapsed to one vertex

distance measures, we use the closest points (Fig. 3, blue) on the individual splat disks, respectively, as formulated in Eq. 3, where $\dot{r}_j$ is the radius of sample $\dot{s}_j$. Further, the actual distance weight is the result of a Gaussian, centered on $\mu = 0$ with $\sigma = \frac{l_i}{3}$. The angular component computes as the dot-product of vertex and sample normal, clipped to values ≥ 0. As formulated in Eq. 4, $\dot{w}_{ij}$ is simply the product of both components.

$$\dot{d}_{ij} = \begin{cases} \left| v_i - \dot{s}_j + \frac{\dot{v}_{ij} - \dot{s}_j}{|\dot{v}_{ij} - \dot{s}_j|} \cdot \dot{r}_j \right| & \text{if } |\dot{v}_{ij} - \dot{s}_j| > \dot{r}_j \\ \left| \dot{\vec{v}}_{ij} \right| & \text{else} \end{cases} \tag{3}$$

$$\dot{w}_{ij} = \exp\left(-\frac{9\dot{d}_{ij}^2}{2l_i^2} \right) \cdot \max\left(0, n_i \cdot \dot{n}_j\right) \tag{4}$$

Figure 3 visualizes the strength of our proposed concept. The contribution of a force that pulls vertices into tangent planes is scaled down with increasing distance from its originator. However, the multiplication with the angular component is what really makes the difference when it comes to recovering more surface detail: Vertex v_0 lies on a flat part of the mesh, the contribution of samples *just around the corner* ($\dot{s}_D, \dot{s}_E$) is decreased significantly or even canceled out completely. For a vertex residing in a curved neighborhood or on an edge like v_1, the attraction of samples from both sides of the implicit edge ($\dot{s}_B$, $\dot{s}_C$ & $\dot{s}_D$, $\dot{s}_E$) effectively pulls it evenly into the intersection of their tangent planes. Even if the splat disks themselves do not intersect, this is guaranteed because only the projection directions contribute to the update vectors.

III.B) Growth If the weights summed up in ω_i are too small, the vertex v_i is not yet sufficiently close to the point cloud hull. Therefore, the displacement vector $\vec{v}_i$ is derived from its normal n_i, scaled by $\frac{l_i}{4}$, This very robustly ties the geometric increment of a vertex to the scale of its local mesh neighborhood. Further, the normal is oriented with a value $o_i = \pm 1$, which computes as the combined majority sign of all dot-products between normal n_i and projection vectors $\dot{\vec{v}}_{ij}$. This allows for mesh expansion as shown in Fig. 1, when the mesh is within the target shape, but also for shrinkage as

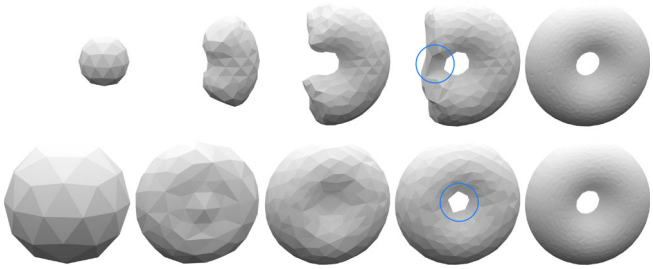


Fig. 5 Adapting the topology. Tunnels are invariant of the surface orientation and automatically apply for ballooning (top) as well as shrink wrapping (bottom)

shown in Fig. 6 (top), with the base mesh enclosing the target shape. Further robustness is exemplified in Fig. 23 (top) with an out-of-target initialization mesh, which first gets drawn into the target and then starts to grow.

Again, vertices are displaced using a differential update $v_i := v_i + \lambda_a \vec{a}_i$, where λ_a allows to adjust the mesh's assimilation speed. However, as the differential increment is by design also linked to dimensions of the local mesh neighborhood, a large λ_a could provoke single vertices to grow out too fast. A moderate growth rate of $\lambda_a = 0.1$ allows step **II.** to level out length discrepancies before the next assimilation update and thus promotes a more even and homogeneous mesh expansion.

IV. Improvements To retain a high mesh quality, unfavorable edge constellations are detected and fixed [17] on the fly: Concave or convex edges, which stand out in contrast with their direct triangle opposites, are fixed with a simple rotation. Figure 4 shows examples of these cases and how they are resolved. Edges shorter than a certain length (e.g., $\frac{l_t}{10}$) are collapsed, merging two vertices into one. This might occur after the edge was rotated or if two neighboring vertices are drawn onto the same implicit edge and get too close to each other. As this step proactively prevents local foldovers, our results do not feature manifold violation artifacts.

V. Object topology adaptation In the most basic scenario, the mesh assimilation is initialized on a simple spherical mesh with a target of the same genus 0. However, to support reconstructions of objects with higher genus, our algorithm supports a common ballooning technique: Tunnels are inserted between opposing mesh fronts to automatically adapt the mesh's topology. The approach of *Cooperative Evolutions* [31] is to detect self-intersections after they have occurred, delete the affected regions, stitch them together, refine the topology, and smooth the geometry. Our solution is more similar to the pro-active approach of *Competing Fronts* [35], using proximity tests on moving vertices to trigger the insertion of tunnels *before* the mesh self-intersects. Therefore, the 1-ring neighborhoods around close vertices or triangles are removed and replaced by tunnels, connecting the opposing edge rings. Tunnels are triangulated analogously to *Freestyle* [36] using the Bresenham algorithm [9] and can therefore be ensured to be watertight. Dedicated topology

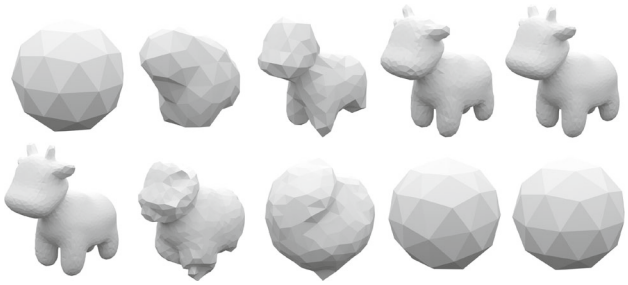


Fig. 6 Top: Our algorithm assimilates the coarse icosphere to the target shape; given as quad mesh [28]. Bottom: Swapped input and target lead to a coarsening; with a larger target edge length, short edges are incrementally collapsed. Progress is left to right

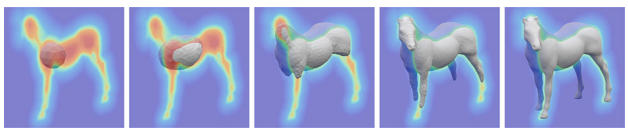


Fig. 7 Mesh construction from a signed distance field, visualized as a color coded cross-section slice: outside, close to 0, inside. As there are no splat disks (point cloud tangents), the hull projection step is omitted

refinement or geometry smoothing is not required as both are automatically taken care of within the next iterations cycles. However, as the initial mesh may be quite arbitrary, tunnels can also resolve inter-mesh situations as shown in Fig. 19 (center) and Fig. 23 (bottom), when initialized with separated geometry. As shown in Fig. 5, the operation is not bound to surface orientation and may also form inverted tunnels, resulting in the correct genus nevertheless.

This step is optional and may be omitted if the target genus is known to be 0 or the initialization mesh already has the correct genus.

2.1 Generalized input

To be able to mesh more than just point clouds by assimilation, we actually only have to adapt step **III.** of the optimization. This is demonstrated with meshed input, signed distance fields, and volumetric data.

Remeshing of existing meshes, e.g., to ensure hole-free reconstruction or uniform mesh quality, is quite trivial to handle, as we can simply use actual surface triangles instead

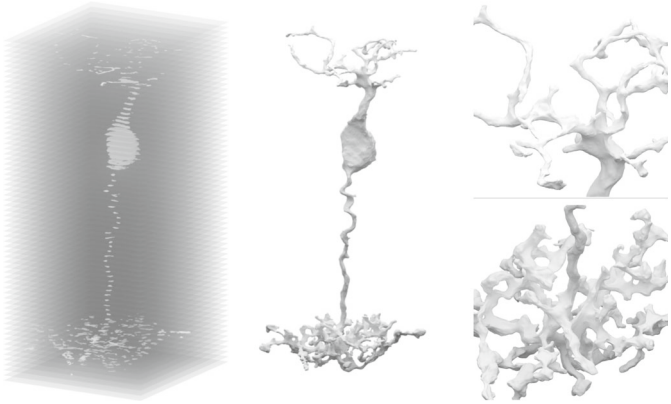


Fig. 8 Tomography data [21]: A neuron, given as 1325 binary slices (not all shown) with 661×595 pixels each, reconstructed as surface mesh

of splat disks. Distances, directions, and weighting follow straightforward. Two examples using a meshed input as the target are shown in Fig. 6. On top, the sphere shrink wraps around the target shape and refines for a smooth and curved surface. For the bottom case, the target edge length is increased, which leads to an incremental collapse of short edges and the mesh assimilates the coarse icosphere again.

Signed distance fields are supported by our mesh construction method as well. Therefore, one only has to replace the orientation o_i in Eq. 2 with a signed-distance. It is sufficient to sample the field once for each vertex position, acquiring single scalar values, respectively, as the vertex normal is used as movement direction. The update vectors $\vec{a}_i$ are always in mesh growth mode (**III.B**), since there is no trivial tangent space given, as with splat disks. This basically comes down to an analogous concept as *Cooperative Evolutions* [31] and, therefore, has the same limitations: As there is no surface projection, vertices are not automatically drawn toward edges or corners; thus, sharp features are not recovered very well. Nevertheless, smooth and curved surfaces are still reconstructed accurately: Update vectors (Eq. 2) are automatically scaled down to 0 length as the (absolute) signed distance decreases, and therefore, the mesh closely approaches the implicit target hull. Figure 7 illustrates surface reconstructions of a target shape, given as a signed distance field.

General volumetric 3D data, e.g., as acquired from computational tomography, require a bit more effort but can be reconstructed as well. Figure 8 shows the reconstruction of a CT scanned neuron, given as sliced binary volume. Analogous to the signed distance field, there is again no trivial tangent space for hull projection. Orientations are not so easily determined as with the SDF, but nevertheless can be easily sampled and trilinearly interpolated from the volume at mesh vertex positions, with binary values giving the sign. Update vectors are scaled down to zero length, once the mesh vertex is situated between both in- and outside labeled voxels, indicating that the mesh reached a surface. By construction, the resulting mesh is watertight. Segments of dendrites separated by noise in the scan data are, however, not connected by our approach.

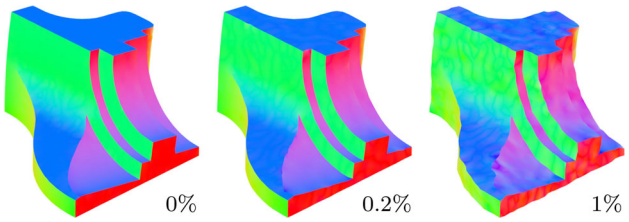


Fig. 9 Mesh construction on noisy input. Besides some kinks, strong edges remain sharp even with increasing noise of randomly jittered samples. The noise magnitude is given in percent of the bounding box diagonal

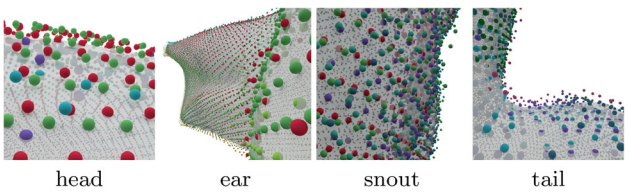


Fig. 10 Reconstruction on 3D scan data [37]. When aligned, the 10 individual partial scans (colored) create noisy overlap regions. Since every mesh vertex is attracted by multiple sample tangents, the noise is leveled out and the reconstruction is smooth nevertheless

3 Experiments and discussion

In this section, we experiment with the achievable mesh quality of our method, challenged with the five most common point cloud artifacts, as described by Berger et al. [5]. A numerical comparison follows in Sect. 3.1 as well as a qualitative discussion in Sect. 3.3.

Noise and outliers are prominent artifacts in 3D scan data and can be robustly dealt with by our algorithm. Figure 9 shows results of the challenge designed for *robust moving least-squares fitting* [19]. A point cloud, sampled reasonably dense but randomly, is perturbed with increasing magni-

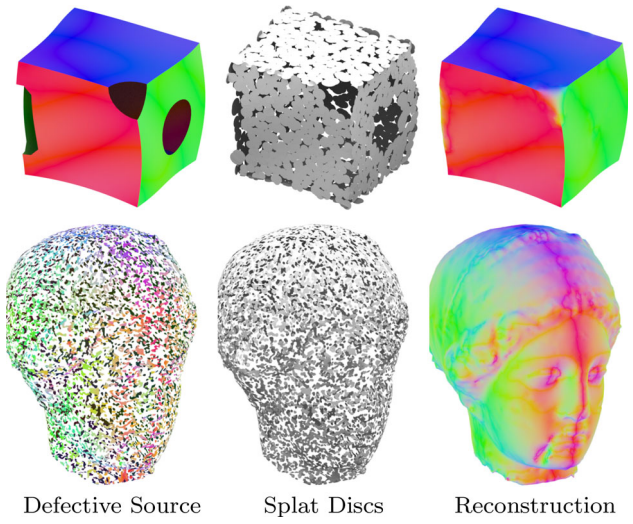


Fig. 11 Filling up holes: Results on defected, than sampled input. The twistcube is missing portions on a curved side region, an edge and a corner. 75% of Igea's surface was removed by cutting out random patches

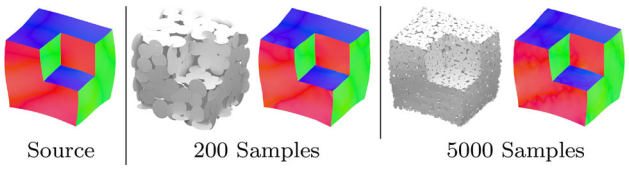


Fig. 12 Sample density has little influence on the reconstruction as the disk radii automatically adapt

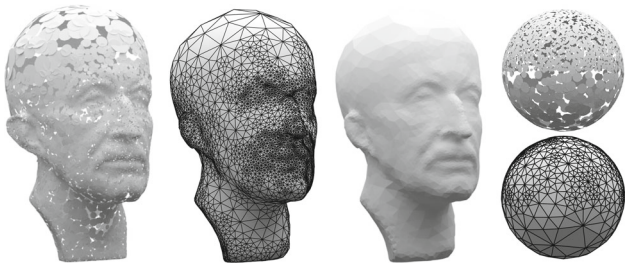


Fig. 13 Adaptive meshing results: The bust was importance-sampled based on local surface curvature. The reconstructed mesh adapts accordingly. The sphere features a very steep density gradient with 10k samples located on the upper and 1k on the lower hemisphere

tudes of random spherical jitter noise, scaled by the fandisk's bounding box diagonal. Without noise, ground truth-like results are achieved. At 0.2% noise scale [19], flat areas are still flat, curves smooth, edges sharp and only a few kinks become noticeable on very obtuse edges. With noise even further increased by factor 5, our reconstruction still robustly captures all major features of the object. Figure 10 shows details of the noisy overlap regions in a real 3D scan, along with the smoothly reconstructed surface. When not extracted in a prefiltering step, outliers do not cause any difficulties in our approach either: Mesh vertices are associated with a fixed number of nearest point cloud neighbors. Therefore, far-out outliers are usually not part of any neighborhood, but if so anyway, their influence is still minimal as their contribution is weighted inversely by their distance.

Missing data in the input geometry is recovered by the reconstruction. The first example in Figure 11 shows the *twistcube* with cutout geometry on a curved side area, an edge, and on a corner. The reconstructed mesh smoothly interpolates the missing regions, stays within the implicit hull, and does not grow through the holes. The second example shows Igea, where 75% of the source mesh was cut out and removed in randomly shaped patches. Our algorithm still reconstructs a closed mesh and recovers fine details.

Sample density is not a decisive factor for successful reconstructions. While modern 3D scanners usually provide quite dense point clouds and more information always allows for more precise results, our algorithms splat disks also automatically adapt to fewer and sparser samples. Figure 12 shows the same object with different sampling but accurate results with smooth curves and sharp edges in both cases.

One might argue that equidistantly distributed vertices—the default outcome of our algorithm—are not always a favorable choice for meshed objects. Therefore, we can easily switch to a sample-density adaptive mode, as exemplified in Fig. 13. To simulate varying density, the ground truth mesh was importance-sampled, based on surface curvature. This leads to a higher sample density on the facial features and fewer, sparsely distributed samples on the head or neck region. Now we can locally scale the target edge length for each vertex individually, based on the radius of the closest splat disks. Figure 13 (right) shows a more challenging example with a very steep density gradient: 10 times more samples are placed on the upper than on the lower hemisphere. The result adopts this change in density with coarser geometry on the bottom and fine geometry on the top.

3.1 Comparisons

Besides validation against common point cloud challenges or the qualitative discussion in the upcoming section, we provide a numerical comparison to a variety of state-of-the-art reconstruction techniques, evaluated on over 2000 randomly selected objects from the Thingi10K data set [40]. A comparison to the most closely related technique, *Competing Fronts* [35], is not included, since there is no implementation available and the authors could not provide reference results. Ground truth objects are scaled-down and centered to fit in a $[-1:1]$ cube. Oriented point clouds are sampled uniformly from the source objects, where the number of samples

Table 1 Results of multiple reconstruction techniques, compared to the ground truth of over 2000 objects from the Thingi10K data set [40], as mean and median

		HD		RMSE		SAD		TVD		NAD		WT
[6,13]	BPA	0.057155	0.052247	0.002710	0.002693	−0.093504	−0.050309	0.065322	−0.014808	0.151215	0.000000	
[14,16]	SSM	0.089543	0.090939	0.004422	0.004029	−0.470502	−0.425178	0.028261	−0.447994	0.220108	0.001329	
[23]	IFAM	0.139415	0.129359	0.004021	0.004036	−0.413034	−0.371409	0.127780	−0.379604	0.197236	0.000000	
[34]	OSR	0.124078	0.101473	0.007759	0.006079	−0.133871	−0.016991	0.025842	−0.001302	0.287676	0.517084	
[10]	SSDR	0.029466	0.025658	0.001922	0.001747	−0.031290	−0.022540	0.115445	−0.003310	0.092074	0.931677	
[24,26]	SPSR	0.034983	0.033962	0.002623	0.002557	−0.043226	−0.031528	0.010974	0.000109	0.128020	0.998227	
	Ours	0.040563	0.019009	0.001269	0.000773	−0.015890	0.000910	0.003979	0.000570	0.040147	1.000000	

The errors (best close to 0, except WT best close to 1), include the symmetric Hausdorff distance (HD) measured with 200k samples, the root-mean-squared-error (RMSE) measuring the closest point distance of result faces to the ground truth, the surface area deviation (SAD) and total volume deviation (TVD) between reconstruction and ground truth, as well as the amount of watertight (WT) results, all given in percent, except for the normal angle deviation (NAD) which is given in radians

computes as object surface area $\times 1000$. Where possible, we specified $l_t \approx 0.025$, except for *IFAM* where we set the target vertex count to half the amount of the input point cloud samples, to actually produce a decent result. Other parameters were left on auto- or the default settings, respectively. Total failure cases (<5%), where the applications crashed and produced no output, could not be taken into account.

Results of our numerical comparison are listed in Table 1 with means and medians of different error measures, respectively. In the average HD, our method is only outranked by two other approaches but supersedes (or is close to) the competition in all other measures. This small drawback can be attributed to a limitation of our method, namely very, very thin geometry. Such a case is included in Fig. 20 with object 1489590: With geometry, becoming thinner than the target edge length, the mesh growth comes to a halt, as there are no further refinements. Portions of the point cloud remain unexplored and thus result in rather high HD errors.

A heuristic, indicating how many point cloud samples are actually in use for hull projections, could potentially improve on this matter. The other models in Figs. 20 and 21 exemplify, why some procedures fall behind in our numerical comparison: While flat or curved surfaces are often reconstructed quite well, edges on feature lines or sharp geometry are smoothed out. This especially shows on CAD-like models with sharp edges, e.g., for object 200080, our result is the only one that accurately reconstructed the threads on the screw.

3.2 Other ballooning methods

Whereas the numerical comparisons are mostly focused on established state-of-the-art meshing procedures, we also want to point out clear conceptional distinctions between our and related ballooning approaches.

• **Competing Fronts** (CF) [35] also use a deformable mesh complex, iteratively approaching the target shape. However, CF strictly grows from within the target as vertices only move along positive normal directions and get frozen once close enough to the point cloud. Our mesh remains flexible throughout the optimization as vertices move and align individually. Further, our method generalizes ballooning and shrink wrapping, as the mesh may grow from within the target, but also enclose on it from the outside. As shown in Fig. 23, our mesh's growth direction is robust enough to even overcome a misaligned initialization. In order to maintain high-quality mesh structures, CF employs remeshing operations [8] every 6-18 iteration cycles. In our iteration, corrective operations are performed in every cycle but targeted at the affected mesh regions. The increase in an object's genus is approached in a similar defensive manner, as both methods replace opposing geometry with a triangulated tun-

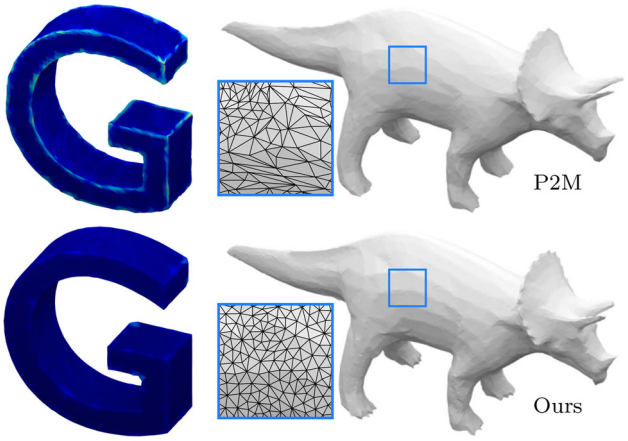


Fig. 14 Comparing results of the learning-based approach Point2Mesh (P2M) [20] to ours

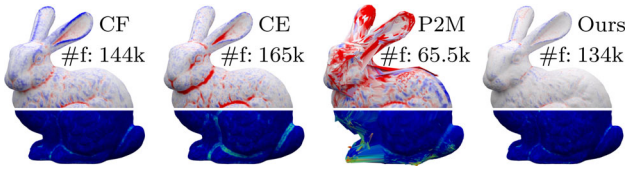


Fig. 15 Error comparisons with related ballooning approaches, visualizing distance errors (top), normal deviations (bottom), and listing the number of faces (#f)

nel before the mesh self-intersects. See step **V.** of Sect. 2 for more details. CF finalizes the optimization with a dedicated projection step to guarantee ϵ-close reconstruction results, which, however, is invariant of the point cloud tangent space and thus does not recover any sharp features. Our optimization is designed to automatically transition from mesh growth to a projection scheme on an individual vertex basis. Further, in our projection, the vertices are explicitly drawn toward intersections of tangent planes, thus faithfully reconstruct sharp corners and edges

• **Cooperative Evolutions** (CE) [31] is based on *two* mesh structures, approaching the target shape from the interior *and* exterior simultaneously. Once both hulls are close enough, individual vertices from both hulls have to be matched to each other and the final geometry is extracted from the space between both meshes. Objects of a higher genus are also possible: Self-intersections are identified; the affected geometry is removed, again stitched together, smoothed, and refined. The mesh evolution in CE is based on a signed distance field, recomputed every iteration cycle, and mixed with a normal distance field. In contrast, our mesh vertices are linked to the individual, fixed-size sets of nearest samples, easily updated with sets from their 1-ring neighbor vertices, respectively. Mesh integrity in CE is, as in CF, maintained with dedicated global remeshing [8]. Further, CE is invariant of the point cloud orientations and thus solely relies on the distance fields. As shown in our SDF experiments (Sect. 2.1), our method can also easily operate on distance fields. However, this also comes with the same drawbacks and limitations of CE, as our method is then also no longer able to recover sharp edges.

• **Point2Mesh** (P2M) [20] is a state-of-the-art learning-based approach, iteratively shrink wrapping a convex hull mesh around a target point cloud. The individual vertex displacements are the results of a learning self-prior network, which does not penalize manifold violations. Therefore, P2M requires heavy manifold reconstructions [22] every few iteration cycles to actually maintain a valid surface. Adapting the genus within the optimization is not possible. However, P2M may also start on a given mesh with the correct genus, i.e., a coarse alpha shape [18] or Poisson [26] mesh. This is also natively supported by our method; moreover, our algorithm may even start on separated geometry as shown in Figs. 19 and 23. Comparisons featured in Fig. 14 again exemplify the strengths of our approach, i.e., generating a uniform mesh structure and reconstructing sharp features and fine details. This can be seen, especially on the edges of the **G** as well as on the toes and epoccipitals on the triceratops' frill. The target edge lengths for our results were set to the average edge length of the P2M meshes, respectively. On the same hardware (GTX 1080 Ti) and starting from the same convex hull mesh, P2M finished the given 6000 iterations for the triceratops within 70 minutes, whereas ours performed 1500 iterations and terminated after 108 seconds.

Ballooning results Figure 15 provides a qualitative comparison against results of CF, CE, and P2M. The visualization shows distance errors and normal angle deviations between reconstruction and input point cloud. Although the CF and CE results have a higher mesh resolution than ours, they still struggle to capture the rounded ridges and valleys of the bunny. With the original point cloud (362k samples) and a memory-limited size of 65.5k faces, the official P2M code totally maxed out the 24 GB of memory on a Titan RTX card, performed 30k iterations, ran for 237 h, but the result is still riddled with artifacts. This is the best result we were able to generate. Even without special attention to sharp features, our method still provides the most accurate reconstruction of round and organic shapes.

3.3 Discussion

With our method, we achieve the same ϵ-closeness to the target shape as *Competing Fronts* [35], which is a guaranteed result of our projection scheme (Fig. 16). Figures 17 and 20 visualize magnified deviation errors form the sampled ground truth meshes. An effect that becomes visible here is the mesh's tendency to exaggerate curved regions.

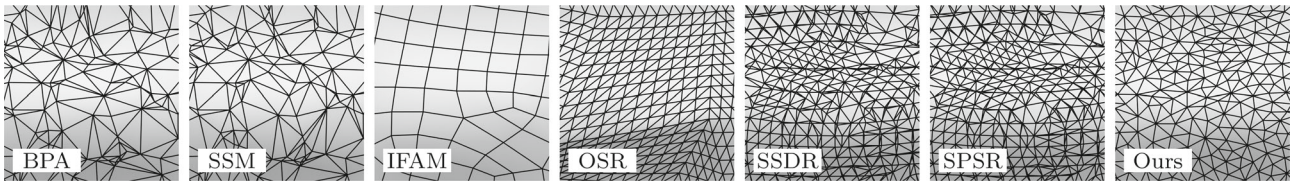
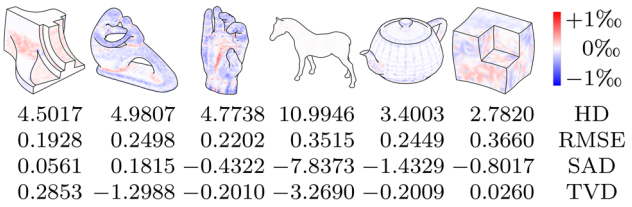


Fig. 16 Comparison on mesh uniformity with a zoom-in on one of the nobs of object 101634 from Fig. 20



4.5017	4.9807	4.7738	10.9946	3.4003	2.7820	HD
0.1928	0.2498	0.2202	0.3515	0.2449	0.3660	RMSE
0.0561	0.1815	−0.4322	−7.8373	−1.4329	−0.8017	SAD
0.2853	−1.2988	−0.2010	−3.2690	−0.2009	0.0260	TVD

Fig. 17 Errors analogous to Table 1 but multiplied by 100 for better readability. Hull projections on tangent planes cause vertices to exaggerate on concave (vertex inside) and convex (vertex outside) curved regions. This effect is not as prominent on the horse as it is reconstructed from an SDF without hull projections

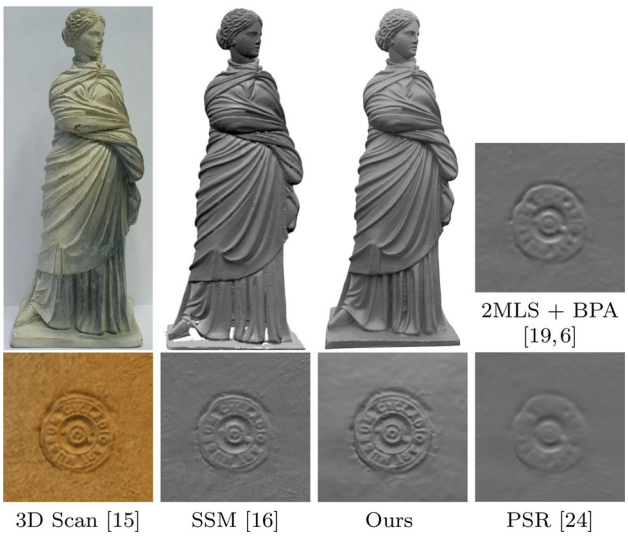


Fig. 18 Reconstructions of the Tanagra 3D scan with PSR, MSL & BPA, SSM, and ours. The reconstructed mesh is significantly smoother in appearance along the folds of the toga while the inscription (on the back of the statue) is much better readable. Image Source [16]

This is caused by the tangent projection, which, on concave regions, intersect inside and for convex regions intersects outside of the mesh. By design, mesh vertices are drawn toward intersections of tangent planes, thus resulting in vertices slightly in or outside of the actual hull. However, this is a small trade-off in our bilateral sample-weighting scheme, which generally allows for a robust and accurate reconstruction of smooth regions and sharp edge features, respectively, even under the presence of noise. The listed numerical values in Fig. 17 and Table 1 for HD and RMSE play in the same order of magnitude as compared state-of-the-art methods.

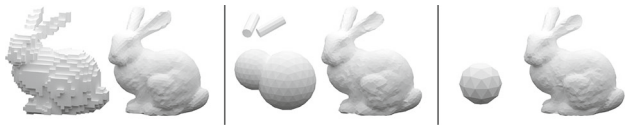


Fig. 19 Reconstructions of the Bunny [37] from three different initialization meshes: Starting from a coarse octree-voxelization, separate geometric objects, and the default icosphere, respectively

Figure 18 compares reconstructions of the Tanagra high precision 3D scan [15]. While the PSR [24] result is again quite smoothed out, the MLS [19] filtered BPA [6] result features better details. Nevertheless, our mesh equals or even supersedes the sharpness and fine-detail quality of the SSM [16] result. The zoom-ins shown in Fig. 16 demonstrate the achievable mesh quality. Whereas other methods' mesh structure either directly depends on the input points (BPA, SSM), smooth out details for the sake of regularity (IFAM, OSR), or triangulate octree-cells (SSDR, SPSR), our results natively approximate Delaunay-like triangulations.

Section 2.1 introduced the ability to adapt our method with little effort to various other input scenarios, giving examples for high-quality point cloud reconstruction, remeshing, volume- and CT-scan meshing, or surface reconstruction from signed distance fields which are currently popular in neural surface representations. Further examples for the versatility of our algorithm are shown in Fig. 6, where *Spot* is given as a quad-meshed target, enclosed by a coarse initial mesh, which automatically assimilates its shape by *shrink wrapping*. Swapping target and init mesh leads to the inverse operation: The only parameter to be adapted is a larger target edge length, automatically coarsening the mesh as small edges incrementally collapse.

Examples in Figure 19 show reconstructions of the Bunny given as point cloud but starting from different initialization meshes. The initial mesh on the left is a coarse voxelization of the point cloud volume. As this initial hull already fills out most parts of the target shape, it is quite easy for our algorithm to project vertices into the point cloud while equalizing their distances. In the second case, the algorithm starts from separate objects which individually grow and connect via inter-mesh tunnels as they approach each other. The rightmost example is the default mesh, which automatically grows into the desired shape from within the target point cloud. However, while the same initialization meshes lead

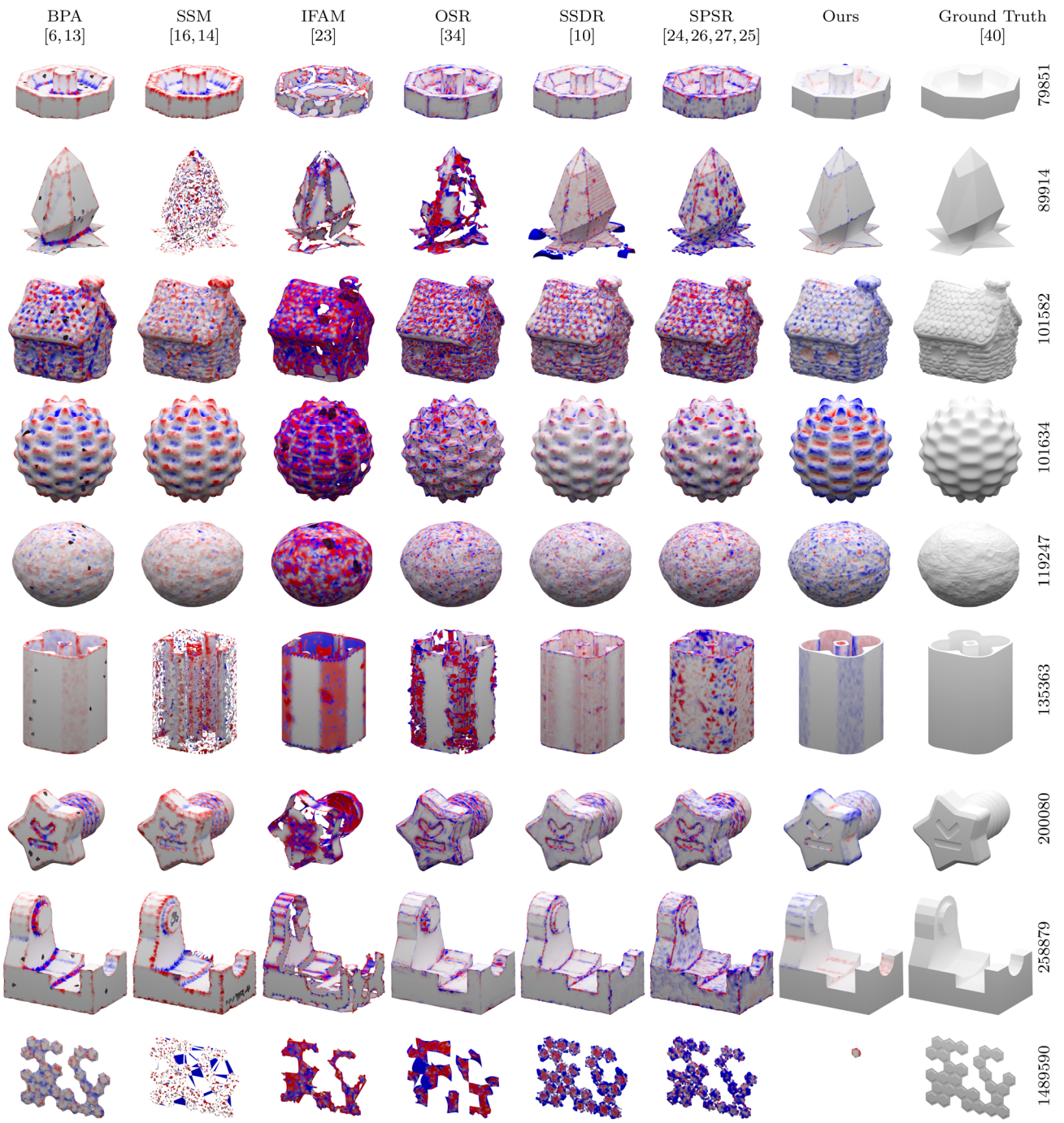


Fig. 20 Randomly selected objects (IDs on the right) from the Thingi10K [40] data set serve as ground truth. All objects are scaled down to fit in a $[-1{:}1]$ cube and are uniformly sampled. The number of samples computes as surface area $\times$ 1000. 89914 is a thin hollow shell and therefore not as simple as it may appear. 1489590 is a failure case, discussed in Sect. 3.1, of our approach where the geometry is too thin for the mesh to grow

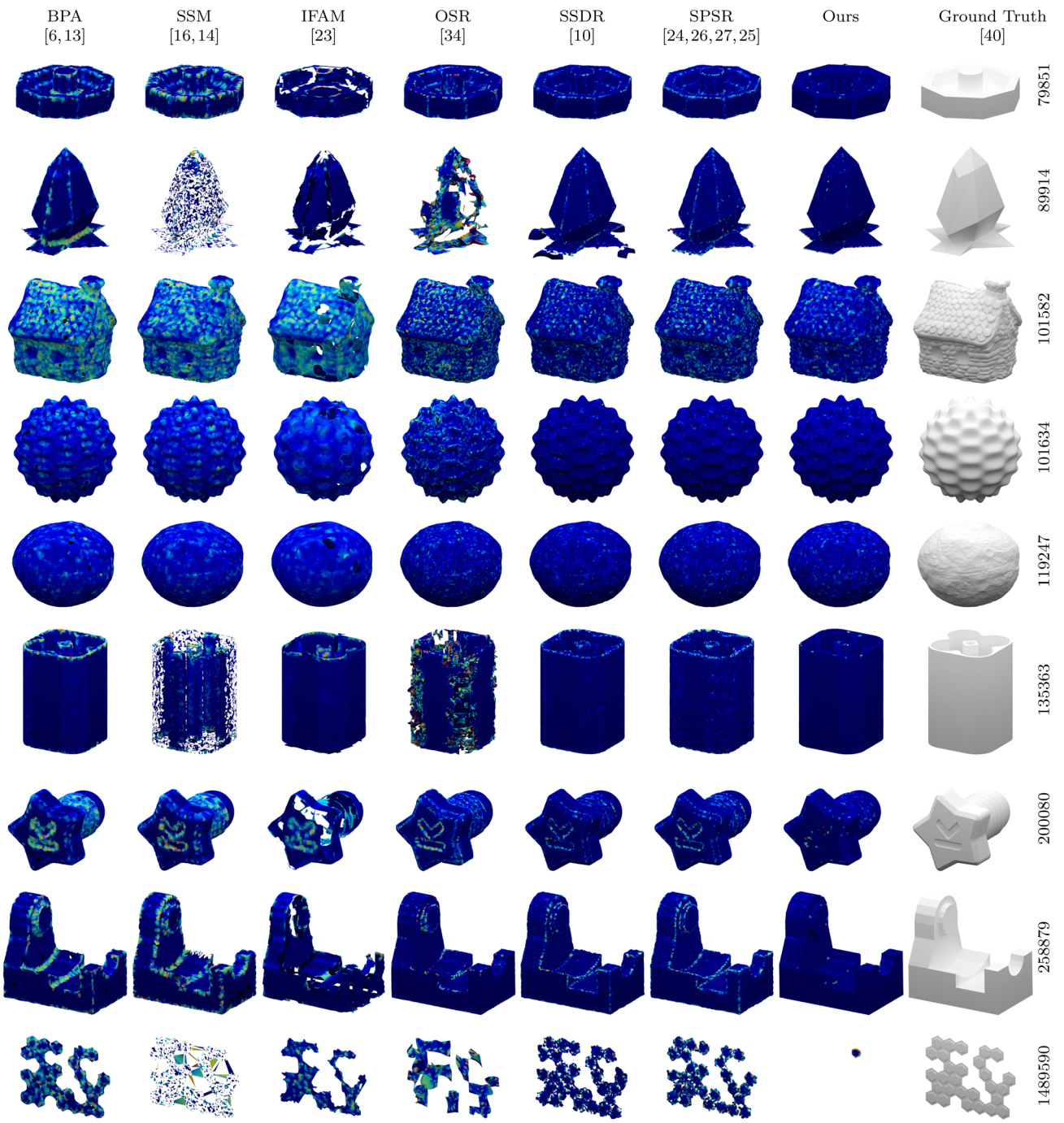


Fig. 21 Same results as from Fig. 20 with visualized surface normal errors. Many methods perform well on curved and flat regions, but sharp edges as on the CAD-like objects are often not recovered very well

to identical results, the algorithm will not produce canonical results from different init meshes. As shown with the Igea example in Fig. 23 (top), the init. mesh does not necessarily have to be placed fully within the target shape. The normal-based orientation for mesh growth direction is robust enough, to cope with miss-aligned starting conditions so that the initial mesh will get *sucked in* and then starts to grow.

Performance comparisons to other related work might yet not be very meaningful, as they are often single-core CPU implementations and ours a mixture of Python and Cuda code. All vertex-based operations in our pipeline are easily parallelized and thus executed on the GPU. Single-threaded

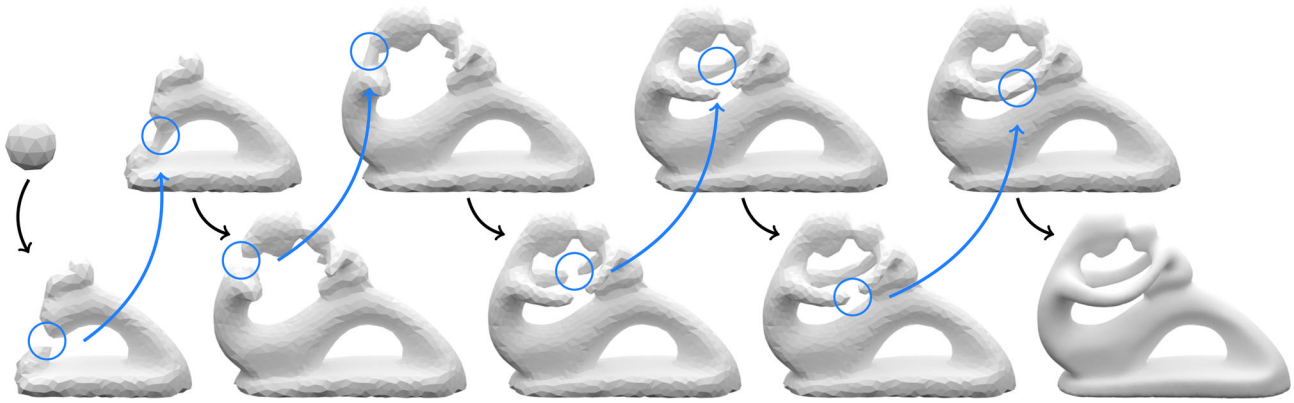


Fig. 22 Objects of higher genus are reconstructed, using intra-mesh tunnels. Black arrows indicate normal growth process. Proximity checks trigger tunneling operations on close (but not directly adjacent) geometry, shown in blue. The iteration continues normally afterward

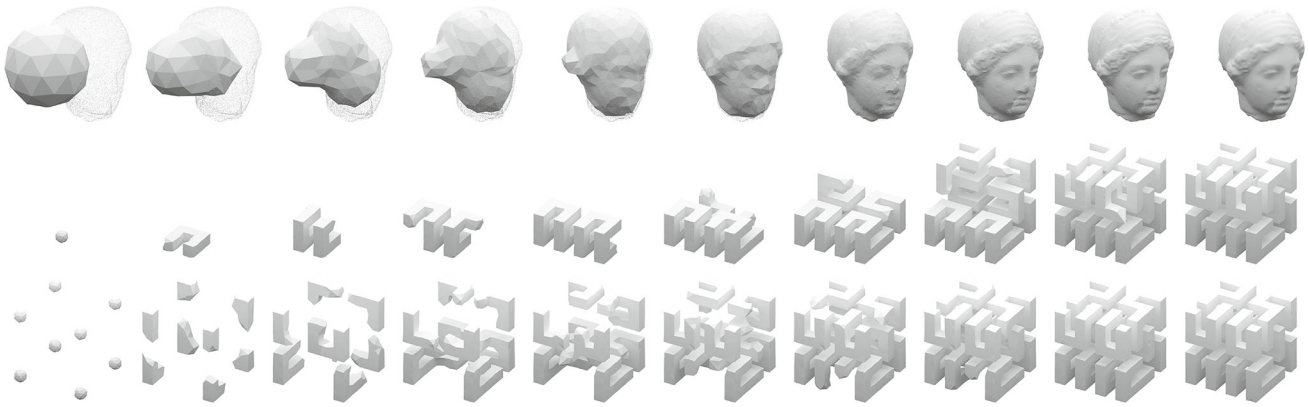


Fig. 23 Top: mesh growth orientation is robust enough to overcome miss-aligned initialization meshes. The Hilbert curve: assimilated from one (mid row) starting position and from eight (bottom) individual positions, connected via inter-mesh tunnels

operations on the mesh data structure only interleave the fast optimization steps. Therefore, our algorithm is able to successfully finish hundreds of iterations within seconds. Full reconstructions of larger results, however, play in the same order of magnitude as competing techniques, ranging from seconds to a few minutes. A speedup can be gained, e.g., from tailored input like coarse voxelizations, as shown in Fig. 19 (left), or well-placed initial geometry, as shown in Fig. 23 (bottom), where separate initial meshes grow in parallel until fused to one data structure via inter-mesh tunnels.

4 Conclusion

In this work, we present a mesh construction/remeshing algorithm that is easy to implement but still checks all the important boxes of the state of the art in this well-researched field: ϵ-close surface reconstructions of point clouds and other (re)meshing tasks, e.g., volumetric functions or CT-scans, robustness to noise or outliers, and support for higher genus object topology with a guaranteed closed, watertight manifold surface. Moreover, we can note several improvements over similar approaches: Instead of global subdivisions, our targeted mesh refinements avoid unnecessary increases in mesh resolution or repeated remeshing steps. The mesh resolution may adapt to the given sample density or distribute vertices uniformly. With a minimal parameter space (target edge length or closeness threshold) to be specified, the process robustly completes fully automated, without the need for different reconstruction modes, finalizing projection, or smoothing passes. This can be attributed to our concept of a seamless transition between mesh growth and surface assimilation on an individual vertex basis. Our bilateral weighing scheme for surface projections allows for perfectly captured sharp object features as well as smooth areas. This is substantiated with a bulk error-benchmark on a large dataset comparing state-of-the-art reconstruction methods as well as various qualitative comparisons, including

other ballooning concepts. Furthermore, we demonstrate the flexibility of our approach to cope with arbitrary input and initialization situations. As our method is still able to adapt to any given target shape, it exemplifies a powerful advancement and generalization of related *ballooning* and *shrink wrapping* approaches.

Funding Open Access funding enabled and organized by Projekt DEAL.

Declarations

Conflict of interest The authors declare that they have no conflict of interest. No animals were harmed during the development of this publication.

References

1. Alliez, P., Ucelli, G., Gotsman, C., Attene, M.: Recent advances in remeshing of surfaces. Shape Analysis and Structuring, pp. 53–82. Springer, New York (2008)
2. Amenta, N., Bern, M.: Surface reconstruction by voronoi filtering. Discrete Comput Geom **22**(4), 481–504 (1999)
3. Amenta, N., Bern, M., Kamvysselis, M.: A new voronoi-based surface reconstruction algorithm. In: Proceedings of the 25th annual conference on Computer graphics and interactive techniques, pp. 415–421 (1998)
4. Amenta, N., Choi, S., Kolluri, R.K.: The power crust. In: Proceedings of the sixth ACM symposium on Solid modeling and applications, pp. 249–266 (2001)
5. Berger, M., Tagliasacchi, A., Seversky, L.M., Alliez, P., Guennebaud, G., Levine, J.A., Sharf, A., Silva, C.T.: A survey of surface reconstruction from point clouds. Computer Graphics Forum, pp. 301–329. Wiley Online Library, Hoboken (2017)
6. Bernardini, F., Mittleman, J., Rushmeier, H., Silva, C., Taubin, G.: The ball-pivoting algorithm for surface reconstruction. IEEE Trans. Vis. Comput. Graph. **5**(4), 349–359 (1999)
7. Boltcheva, D., Lévy, B.: Surface reconstruction by computing restricted voronoi cells in parallel. Comput. Aided Des. **90**, 123–134 (2017)
8. Botsch, M., Kobbelt, L.: A remeshing approach to multiresolution modeling. In: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, pp. 185–192 (2004)
9. Bresenham, J.E.: Algorithm for computer control of a digital plotter. IBM Syst. J. **4**(1), 25–30 (1965)
10. Calakli, F., Taubin, G.: Ssd-c: Smooth signed distance colored surface reconstruction. In: Dill, J., Earnshaw, R., Kasik, D., Vince, J., Wong, P.C. (eds.) Expanding the Frontiers of Visual Analytics and Visualization, pp. 323–338. Springer, London (2012)
11. Campos, R., Garcia, R., Alliez, P., Yvinec, M.: Splat-based surface reconstruction from defect-laden point sets. Graph. Models **75**(6), 346–361 (2013)
12. Cohen-Steiner, D., Alliez, P., Desbrun, M.: Variational shape approximation. In: ACM SIGGRAPH 2004 Papers, pp. 905–914 (2004)
13. Digne, J.: An analysis and implementation of a parallel ball pivoting algorithm. Image Process. On Line **4**, 149–168 (2014). https://doi.org/10.5201/ipol.2014.81
14. Digne, J.: An implementation and parallelization of the scale space meshing algorithm. Image Process. On Line **5**, 282–295 (2015). https://doi.org/10.5201/ipol.2015.102
15. Digne, J., Audfray, N., Lartigue, C., Mehdi-Souzani, C., Morel, J.M.: Farman institute 3D point sets - high precision 3D data sets. Image Process. On Line **1**, 281–291 (2011). https://doi.org/10.5201/ipol.2011.dalmm_ps
16. Digne, J., Morel, J.M., Souzani, C.M., Lartigue, C.: Scale space meshing of raw data point sets. Computer Graphics Forum, pp. 1630–1642. Wiley Online Library, Hoboken (2011)
17. Duan, Y., Qin, H.: Intelligent balloon: a subdivision-based deformable model for surface reconstruction of arbitrary topology. In: Proceedings of the sixth ACM symposium on Solid modeling and applications, pp. 47–58 (2001)
18. Edelsbrunner, H., Mücke, E.P.: Three-dimensional alpha shapes. ACM Trans. Graph. (TOG) **13**(1), 43–72 (1994)
19. Fleishman, S., Cohen-Or, D., Silva, C.T.: Robust moving least-squares fitting with sharp features. ACM Trans. Graph. (TOG) **24**(3), 544–552 (2005)
20. Hanocka, R., Metzer, G., Giryes, R., Cohen-Or, D.: Point2mesh: a self-prior for deformable meshes. ACM Trans. Graph. (2020). https://doi.org/10.1145/3386569.3392415
21. Helmstaedter, M., Briggman, K.L., Turaga, S.C., Jain, V., Seung, H.S., Denk, W.: Connectomic reconstruction of the inner plexiform layer in the mouse retina. Nature **500**(7461), 168–174 (2013)
22. Huang, J., Su, H., Guibas, L.: Robust watertight manifold surface generation method for shapenet models. arXiv preprint arXiv:1802.01698 (2018)
23. Jakob, W., Tarini, M., Panozzo, D., Sorkine-Hornung, O.: Instant field-aligned meshes. ACM Trans. Graph. **34**(6), 189 (2015)
24. Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: Proceedings of the Fourth Eurographics Symposium on Geometry Processing, SGP '06, pp. 61–70. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2006)
25. Kazhdan, M., Chuang, M., Rusinkiewicz, S., Hoppe, H.: Poisson surface reconstruction with envelope constraints. Computer Graphics Forum, pp. 173–182. Wiley Online Library, Hoboken (2020)
26. Kazhdan, M., Hoppe, H.: Screened poisson surface reconstruction. ACM Trans. Graph. (ToG) **32**(3), 1–13 (2013)
27. Kazhdan, M., Hoppe, H.: An adaptive multi-grid solver for applications in computer graphics. Computer Graphics Forum, pp. 138–150. Wiley Online Library, Hoboken (2019)
28. Keenan's 3D Model Repository (2020). Online; Accessed from Oct-2020, http://www.cs.cmu.edu/~kmcrane/Projects/ModelRepository/
29. Kobbelt, L.: $\sqrt{3}$-subdivision. In: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pp. 103–112 (2000)
30. Lorensen, W.E., Cline, H.E.: Marching cubes: a high resolution 3d surface construction algorithm. ACM Siggraph Comput. Graph. **21**(4), 163–169 (1987)
31. Lu, W., Liu, L.: Surface reconstruction via cooperative evolutions. Comput. Aided Geom. Des. **77**, 101–831 (2020)

32. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (2019)
33. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 165–174 (2019)
34. Schertler, N., Tarini, M., Jakob, W., Kazhdan, M., Gumhold, S., Panozzo, D.: Field-aligned online surface reconstruction. ACM Trans. Graph. (TOG) **36**(4), 77 (2017)
35. Sharf, A., Lewiner, T., Shamir, A., Kobbelt, L., Cohen-Or, D.: Competing fronts for coarse-to-fine surface reconstruction. Comput. Graph. Forum, pp. 389–398. Wiley Online Library, Hoboken (2006)
36. Stanculescu, L., Chaine, R., Cani, M.P.: Freestyle: sculpting meshes with self-adaptive topology. Comput. Graph. **35**(3), 614–622 (2011)
37. The Stanford 3D Scanning Repository (2014). Online; Accessed from Oct-2017, http://graphics.stanford.edu/data/3Dscanrep/
38. Vlachos, A., Peters, J., Boyd, C., Mitchell, J.L.: Curved pn triangles. In: Proceedings of the 2001 symposium on Interactive 3D graphics, pp. 159–166. ACM (2001)
39. Wu, J., Kobbelt, L.: Optimized sub-sampling of point sets for surface splatting. Computer Graphics Forum, pp. 643–652. Wiley Online Library, Hoboken (2004)
40. Zhou, Q., Jacobson, A.: Thingi10k: a dataset of 10,000 3d-printing models. arXiv preprint arXiv:1605.04797 (2016)

Dennis R. Bukenberger received his MSc in computer science from the University of Tübingen in 2016. He is currently working as a PhD student in the Computer Graphics Department at the University of Tübingen. His research interests include geometry processing, automated abstraction, 3D reconstruction, and meshing.

Hendrik P. A. Lensch holds the chair for computer graphics at the University of Tübingen. He received his diploma in computer science from the University of Erlangen in 1999 and his PhD from Saarland University in 2003. He was a visiting assistant professor at Stanford University, USA, and head of an independent research group at the MPI Informatik. From 2009 to 2011, he was a full professor at the Institute for Media Informatics at Ulm University, Germany. His research interests include 3D appearance acquisition, computational photography, machine learning, global illumination and image-based rendering, and massively parallel programming.