Volume xx (200y), Number z, pp. 1–12

Robust Segmentation of Multiple Intersecting Manifolds from Unoriented Noisy Point Clouds

Abstract

We present a method for extracting complex manifolds with an arbitrary number of (self) intersections from unoriented point clouds containing large amounts of noise. Manifolds are formed in a three step process. First, small flat neighborhoods of all possible orientations are created around all points. Next, neighborhoods are assembled into larger quasi-flat patches, whose overlaps give the global connectivity structure of the point cloud. Finally, curved manifolds are extracted from the patch connectivity graph via a multiple-source flood fill. The manifolds can be reconstructed into meshed surfaces using standard existing surface reconstruction methods. We demonstrate the speed and robustness of our method on several point clouds, with applications in point cloud segmentation, denoising, and medial surface reconstruction.

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

1. Introduction

Point cloud models of 3D shapes are created by many applications such as surface scanning [PNF*08], stereo reconstruction [RH07], shape processing [RT08b,JR07], and medical imaging [SHG06]. Extracting 3D surfaces from such point clouds is an important task. Many methods exist for reconstructing an approximation of a continuous 2D surface $\mathcal{S} \subset \mathbf{R}^3$ from a point cloud S. However, most such methods pose constraints on the *manifold* structure of $\mathcal{S}$ and/or structure of the point cloud S, such as the sampling density of S, availability of normals, and presence of watertight manifolds. A more restrictive constraint is that $\mathcal{S}$ is a single non-self-intersecting manifold $\mathcal{M} \subset \mathbf{R}^3$. Point clouds created by 3D laser scans typically meet this constraint. Other applications however can create clouds which sample intersecting manifolds. Examples hereof are medial shape processing [MBC12], where 3D skeleton manifolds naturally intersect; CAD reverse-engineering, where one aims to separate several intersecting shapes from a single point cloud, *e.g.* when topology information was lost; and dimensionality reduction [SP05].

Extracting multiple (self) intersecting manifold surfaces from unoriented noisy point clouds is very challenging, since this process is ill-posed without prior assumptions on the manifold structure [CLK09]. To address this, we propose to first segment the cloud into subsets which have manifold properties. Our contribution focuses on point cloud *segmentation* (as opposed to surface *reconstruction*): Given an unoriented point cloud, we first extract locally quasi-flat point groups. Next, we merge these groups into larger point sets using a global smoothness criterion. No assumption is made on the manifolds' shapes, sizes, or (self) intersections. Finally, we reconstruct meshed manifolds from each such point set using existing standard surface reconstruction techniques. Manifold smoothness is controlled by two parameters. Our method is robust against geometric or topological noise, *i.e.* extracts smooth, clean, manifolds embedded into noisy clouds.

The structure of this paper is as follows. Section 2 reviews related work. Section 3 details the three steps of our method: local point classification (Sec. 3.1), global classification (Sec. 3.2), and manifold reconstruction (Sec. 3.3). Section 4 presents applications in manifold cloud clustering, denoising, and medial surface reconstruction. Section 5 discusses our method. Section 6 concludes the paper and outlines future work directions.

2. Related Work

Extracting manifold surfaces from noisy point clouds has been researched in several fields, as follows.

Data clustering: If the cloud is a mix of smooth (intersecting) manifolds with noise, extraction can be seen as a data clustering task. Clustering methods use assumptions on the underlying data, *e.g.* linear manifolds [VMS05, HH05], minimal manifold separation [CH06], knowing the manifold count [SP05], or normal data [RVV06]. Spectral clustering handles curved manifolds, but is slow for intersecting manifolds [Gol09]. K-manifolds [SP05] estimates inter-point geodesic distances via dimensionality reduction [TdSL00] and uses this distance to cluster points via expectation-maximization. K-manifolds extracts curved manifolds, but needs to know the cluster count, and each point can belong to just one cluster. Since a geodesic distance is used to separate clusters, this fails for non-intersecting clusters (see further Sec. 4.1). Kushnir *et al.* [KGB06] cluster manifolds by finding a minimal normalized-cut in a weighted graph. This method is not suitable for manifolds with varying sampling density. Difference-of-normals (DoN) [ITHG12] estimates point normals at two spatial scales r_1 and r_2 by fitting a plane on point neighborhoods with radii r_1 and r_2 respectively. Large values of the difference of the two normals indicate shapes in the cloud at a scale between r_1 and r_2, which can next be segmented from the surrounding cloud.

Outlier removal: Points with statistical properties diverging

from the desired smooth manifolds are removed [KNT00,JKN98, Haw80, RMB*08, BKNS00]. Such methods do not address point clouds specifically. However, data statistics (distribution, distances, and density) can have a too wide variance for non-uniformly sampled clouds with many noisy outliers, *e.g.* tens of percents of the cloud size. Sotoodech *et al.* [Sot07] remove outliers recursively, in decreasing distance from the cluster center. However, this approach does not work well for datasets where noise and data overlap or which contain non-intersecting manifolds far away from the largest data density (see also Sec. 4.2). For such datasets, tensor voting provides an alternative solution [MM07]. Local geometric structure, or features, are represented using *e.g.* second-order symmetric tensors of the covariance matrix created from the neighbors of each point. Each point lying on a (potentially-noisy) manifold propagates its local feature in a neighborhood of user-specified radius σ, by casting a vote to all nearby points. The neighborhood is determined by a dense 'voting field', aligned with the local point. The vote is a tensor generated according to the local tensor and position in the voting field. Accumulating all votes generates the new local tensor or feature. 3D surfaces are finally extracted by isosurfacing the extremal directions of the tensor field [MM07, MT00].

Surface reconstruction: Point-cloud surface reconstruction often uses global priors, *e.g.* surface smoothness, water-tightness, viewpoint-invariance, and topology [BMR*99, AC01, Flö09, DG03,CSD04,JR07,DLRW09,TOZ*11,SSZCO10,KBH06]. For noisy clouds, we distinguish between denoising [UH07] and surface extraction methods [DG04, SW09, MDD*10]. Chang *et al.* present a comprehensive comparison of the strengths and limitations of 16 surface reconstruction methods [CLK09]. They show that, apart of their own method, all other reviewed methods cannot handle non-manifold surfaces and/or intersecting manifolds. However, the method of Chang *et al.* is much slower, and more complicated, than our proposal (see Sec. 5).

Our contribution: Our goal is to separate manifold points from an input noisy cloud. Our priors are a set of (self-) intersecting manifolds with or without boundaries. This is more general than typical priors in data clustering, outlier removal, and surface reconstruction: Unlike manifold clustering, we allow points to belong to several, one, or no manifold (*i.e.*, noise). Unlike outlier removal, we allow a much higher amount and spread of noise. Unlike surface reconstruction, we segment the input cloud into separate manifolds, so we can directly use standard surface reconstruction tools for each such manifold.

3. Method

Our input is an unoriented point cloud $S = \{\mathbf{x}_i\} \subset \mathbf{R}^3$ which samples a surface $\mathcal{S} \subset \mathbf{R}^3$ consisting of one or several possibly (self-) intersecting manifolds $\mathcal{M}_i$ embedded into noise. We first classify, or label, S into a set of point clouds $M_i \subset S$, such that each M_i is the sampled representation of $\mathcal{M}_i$. The classification handles intersecting manifolds, *i.e.* $M_i \cap M_j$ can be non-void for $i \neq j$. To handle noisy clouds, we allow certain points from S which cannot be classified as belonging to any smooth manifold, to be labeled as noise, and further ignored, *i.e.* $\bigcup_i M_i \subseteq S$.

Classification has two phases (Fig. 1). First, for each small spatial neighborhood $\nu(\mathbf{x}_i) \subset S$ of a point $\mathbf{x}_i \in S$, we compute all point sets $c_i \subset \nu(\mathbf{x}_i)$ which describe quasi-flat manifolds embedded in ν. We call this *local classification* (Sec. 3.1). Next, we group the point sets c_i into sets M_k which describe large, compact, manifolds $\mathcal{M}_k$. We call this *global classification* (Sec. 3.2). Finally, we reconstruct piecewise-linear (meshed) representations of the manifolds $\mathcal{M}_k$ from the sets M_k (Sec. 3.3).

3.1. Local classification

A key property of a 2D manifold is its *local flatness*. To reason about local flatness, we consider each neighborhood $\nu(\mathbf{x}_i)$, $\mathbf{x}_i \in S$. Such neighborhoods can be defined *e.g.* using the k nearest neighbors of $\mathbf{x}_i$ (Fig. 1 a), as detailed further in Sec. 5.

3.1.1. Local Surface Estimation

To find all manifolds which a neighborhood $\nu(\mathbf{x}_i)$ admits, we first construct all possible flat surfaces which pass through $\mathbf{x}_i$ and two other points $\mathbf{x}_m, \mathbf{x}_n \in \nu(\mathbf{x}_i), i \neq m \neq n$, *i.e.* all triangles $T_\nu = \{(\mathbf{x}_i, \mathbf{x}_m, \mathbf{x}_n)\}$. We exclude degenerate triangles which have near-collinear or near-identical vertices. The maximum triangle count is the number of possible 2-permutations without repetition *i.e.* $\frac{k!}{(k-2)!}$ where $k = |\nu(\mathbf{x}_i)|$ is the neighborhood size. For each neighborhood ν, we gather the unoriented normals $\mathbf{n}_i$ of all triangles $t_i \in T_\nu$ into a normal-set N_ν (Fig. 1 b).

Consider now the map $\gamma : N_\nu \to \mathbf{S}^2$ where $\mathbf{S}^2$ is the unit sphere, which maps from each normal $\mathbf{n} \in N_\nu$ to the sphere point indicated by $\mathbf{n}$, seen as a vector starting at the sphere center. γ is the *Gauss map* of a surface formed by all triangles in T_ν [GP05]. The peaks of γ coincide with the normal directions of the most salient manifolds, since these are the directions along which many triangles, packed in the small spatial area ν, are oriented. Hence, we can use γ to extract our desired quasi-flat manifolds within ν. We distinguish three cases:

a. If ν samples a quasi-flat surface, all triangles $t_i \in T_\nu$ have similar normals $\mathbf{n}$, so γ has two clear peaks (at $\mathbf{n}$ and $-\mathbf{n}$);
b. If ν samples the intersection of two or more manifolds, γ has two or more such peak-pairs;
c. If γ does not have clearly separated peaks, but is a rather flat signal, ν samples a volumetric point distribution rather than a few intersecting quasi-flat surfaces, so we cannot reliably extract clear quasi-flat manifolds from ν.

All points whose normals fall under a peak-pair in γ are thus points in ν which belong to the same quasi-flat surface. Hence, if we can reliably find well-separated peaks in γ (which we discuss next in Sec. 3.1.2), we find the desired separate quasi-flat surfaces in ν. If such peaks are far apart in γ, their corresponding quasi-flat surfaces should *self-intersect* within ν, since ν is small and these surfaces are not parallel. For example, the Gauss map for seven triangles in Fig. 1 c captures the intersection of two quasi-flat surfaces within the considered neighborhood, one with 3 triangles, the other one with 4 triangles.

3.1.2. Segmenting the Gauss Map

To separate the different quasi-flat surfaces in a neighborhood ν, we cluster the neighborhood's Gauss map γ using the geodesic distance on the sphere $\mathbf{S}^2$ between the map's normals, defined as

$$d(\mathbf{n}_i \in N_\nu, \mathbf{n}_j \in N_\nu) = \arccos |\mathbf{n}_i \cdot \mathbf{n}_j| \in \mathbf{R}^+ \qquad (1)$$

where $\cdot$ denotes vector dot product. As we do not know *a priori* the number of peaks in γ, we use a hierarchical bottom-up, full-linkage, agglomerative clustering on the distance matrix induced by d on N_ν [dHINM04]. This yields a dendrogram $\mathcal{D}$ whose nodes $n_i = \{t_j \in T_\nu\}$ are sets of triangles having similar normals.

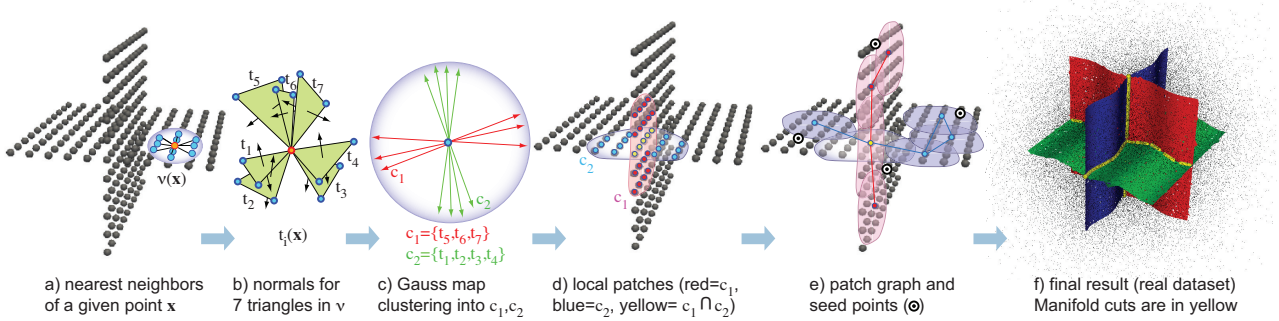

Figure 1: *(a-e) Algorithm steps (see Secs. 3.1-3.3 for details). (f) Three manifolds are extracted from a noisy point cloud.*

We next cut $\mathcal{D}$ at a user-given normal dissimilarity level d_{max}. For each node n_i in the cut, we collect all vertices of triangles in n_i to create a quasi-flat surface or *patch* $c_i \subset \nu$. We also store in each patch c_i all normals of its triangles. We denote this normal set $\mathbf{n}(c_i)$. The threshold d_{max} gives the amount of curvature we allow for our quasi-flat surfaces over ν. Small d_{max} values yield more, and flatter, patches c_i. Larger values yield fewer, and more curved, patches. Setting d_{max} is discussed further in Sec. 5.

From the above, we obtain, for each input point $\mathbf{x}_i$, a set $\pi_i = \{c_j\} \subset \nu(\mathbf{x}_i)$ of quasi-flat patches c_j which pass through $\mathbf{x}_i$. The size of the set π_i tells what type of point $\mathbf{x}_i$ is: If $|\pi_i| = 1$, one patch passes through $\mathbf{x}_i$, so we say that $\mathbf{x}_i$ is a *flat* point. If $|\pi_i| > 1$, more patches pass through $\mathbf{x}_i$, so we consider that $\mathbf{x}_i$ is an *intersection* point of several manifolds or a flat patch surrounded by noise. For example, a point on the intersection line of the two planes in Fig. 1 d has two patches $\pi = \{c_1, c_2\}$. Of course, the notions of flatness and intersection used here are subject to the flatness threshold d_{max} (see also Sec. 5). We do not further distinguish between several intersecting manifolds and noise (both occurring when $|\pi_i| > 1$) explicitly by counting the size of the set π_i, but implicitly, by the global process which joins overlapping similar-orientation patches, as described further in Sec. 3.2.

Our approach is similar to [WHH10] where, for each point, all local triangulations containing the current point are projected on a Gauss map, which is next clustered. However, [WHH10] use this only to find sharp edges. We use our local patches to reconstruct manifolds *past* such implicit edges, as described next in Sec. 3.2.

Conceptually, our patches c_j resemble the splats used in point-based rendering (PBR) [WK04,BSK04]: We also want to approximate the surface (or intersecting surfaces) around each point $\mathbf{x}_i$ by a local quasi-flat structure, using the k nearest neighbors of $\mathbf{x}_i$. Like splats, our patches overlap their neighbors, to produce a coverage of the approximated surface. However, splats are typically circular or elliptic in shape. Our patches do not have shape constraints apart from their quasi-planarity. Secondly, a patch explicitly stores all normals of its points (used next to merge patches into manifolds, see Sec. 3.2). Splats typically use normals from an analytic surface fit to their sample points. During our merging process, we only enforce *local* quasi-flatness constraints on the resulting manifolds. A splat has typically a *global* smoothness constraint, due to the above-mentioned surface fit. As such, we use here the term *patches* to distinguish these from PBR splats.

3.2. Global classification

To find the desired manifolds M_k from the local patch-sets $\pi_i = \{c_j\}$, we assume that each c_j is part of exactly one M_k. We justify this as follows. First, patches are quasi-flat or lightly curved, by construction, so it makes little sense to assign the same patch to two different manifolds. Secondly, any two such patches from the same small neighborhood ν are oriented at strongly different angles, since our clustering threshold d_{max} finds strong, separated, peaks in the Gauss map (see also Sec. 5). Thus, when such situations occur, we have two manifolds intersecting in ν, each of the two patches belonging to a different manifold. Thirdly, we assume that all points in any M_k belong to some patch, *i.e.* that our manifolds are a union of patches $\bigcup_k M_k = \bigcup_i \pi_i$.

Assembling patch sets π_i into manifolds M_k is described next.

3.2.1. Patch connectivity graph

First, we determine how patches in the patch-sets π_i around each point $\mathbf{x}_i$ relate to each other. For this, we construct a patch connectivity graph $G = (V, E = V \times V)$, as follows (Fig. 1 e). For each patch c_i, we add a graph vertex $u(c_i)$ to V. Given two patches c_α and c_β, we define their surface dissimilarity δ as

$$\delta(c_\alpha, c_\beta) = \min_{\mathbf{n}_A \in \mathbf{n}(c_\alpha),\ \mathbf{n}_B \in \mathbf{n}(c_\beta)} d(\mathbf{n}_A, \mathbf{n}_B) \qquad (2)$$

with d given by Eqn. 1. If two points $\mathbf{x}_i \in S, \mathbf{x}_j \in S$, which belong to the same patch-set, have patches $c_\alpha \in \pi_i, c_\beta \in \pi_j$ whose dissimilarity δ is below a given value δ_{max}, we add an edge to E between $u(c_\alpha)$ and $u(c_\beta)$. Since we compare only points $\mathbf{x}_i, \mathbf{x}_j$ from the same patch-set, δ is evaluated only for close, overlapping patches. Although Eqn. 2 is equivalent to a full-linkage between all normals in $\mathbf{n}(c_\alpha)$, $\mathbf{n}(c_\beta)$, we speed up its computation by adding an edge between two graph nodes as soon as we find two normals $\mathbf{n}_A, \mathbf{n}_B$ which are closer than δ_{max}. We store G as a binary adjacency matrix where each entry defines if two patches are connected or not. Setting δ_{max} is discussed further in Sec. 5.

G could be directly used to find points of the different manifolds M_k *e.g.* by finding its connected components via flood filling. However, this only works if manifolds intersect at non-acute angles, which is not always the case (see examples in Sec. 4.2). If manifolds intersect at acute angles, a connected component of G can cover more than a single manifold. Indeed, patches close to manifold intersections and which belong to different manifolds differ too little in terms of normals, and thus get connected when building G. We solve this issue and robustly detect intersecting manifolds from the patch graph G in two steps: seed point detection (Sec. 3.2.2) and manifold point labeling (Sec. 3.2.3).

3.2.2. Seed Point Detection

Seed points $\Omega \subset S$ are cloud points located in quasi-flat areas and far from potential manifold intersections. They are starting points for a flood fill process over G which ultimately delivers our manifold points M_i (Sec. 3.2.3). We compute Ω as follows.

1. Mark all cloud points $\mathbf{x}_i \in S$ as unvisited, and set Ω to $\varnothing$.
2. Find an unvisited *flat* point $\mathbf{x}_i$, *i.e.* with $|\pi_i| = 1$ (Sec. 3.1.2).
3. Add $\mathbf{x}_i$ to Ω.
4. Mark all unvisited flat neighbors $\{\mathbf{x}_j \in \nu(\mathbf{x}_i) \big| |\pi_j| = 1\}$ of $\mathbf{x}_i$ as visited, using a flood fill.
5. Repeat from step 2 until all points in S are visited.

Ω contains seed points far away from manifold intersections. The point flatness condition prevents the fill to leak from one manifold to another; intersections act as flood barriers, since they have non-flat points. So, we get as many seeds as manifold segments delimited by manifold intersection curves and manifold boundaries, *e.g.* four seeds for the cross in Fig. 1 a, one for each arm.

3.2.3. Manifold labeling

The seed set Ω (Sec. 3.2.2) could be directly used to find manifold connected-components separated by intersection curves. Although such results are useful [UH03], we aim to find *entire* manifolds *past* intersection curves. For instance, for the shape in Fig. 1 f, we want to find three intersecting surfaces (red, green and blue in Fig. 1 f) rather than twelve quarter-surfaces. Since seed points $\mathbf{s}_i \in \Omega$ are flat, their patch-sets have a single patch, *i.e.* $\pi(\mathbf{s}_i) = \{c(\mathbf{s}_i)\}$. We use this observation as follows (see also Fig. 2 for a 2D sketch): For each $\mathbf{s}_i \in \Omega$, we assign a unique ID to $c(\mathbf{s}_i)$ (step A). Next, we do a flood-fill over patches (step B). For this flood fill, we use the patch connectivity (stored in G) instead of the point neighbors (given by ν) used to find seeds (Sec. 3.2.2). Hence, the patch-level flood fill can cross manifold intersections but stays confined to the surface of a *single* manifold.

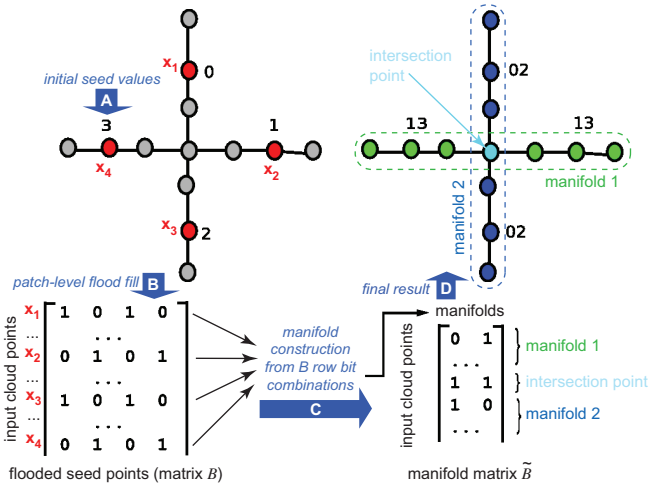

Figure 2: *Manifold labeling: (A) assignment of IDs* 0..3 *to seed points* $\mathbf{x}_1..\mathbf{x}_4$*; (B) patch-level flood fill yielding matrix B, (C) construction of manifolds, and (D) final manifolds (Sec. 3.2.3). From the four seed points (marked red), we extract two manifolds with labels 02 and 13, and one intersection point with label 0123.*

The patch-level flood fill adds to every patch the IDs of the seeds which flooded through that patch. However, we need this information at point level, *i.e.*, know which seeds flooded through each cloud point. We store this in a compact and fast way by using a binary matrix $B = \{b_{ij}\}$, where each row $1 \le i \le |S|$ is a bit-vector whose non-zero values encode the seeds $1 \le j \le |\Omega|$ which flooded through point $\mathbf{x}_i$. We finally find the manifolds M_i as all points which share the same row bit-patterns in B. Hence, we find as many manifolds as different row bit-patterns we have in B.

Points on manifold intersections contain seed-point information from all manifolds intersecting at that location. To add such points to their intersecting manifolds, we visit the k-neighbors of a given non-intersection point $\mathbf{x}_i$, and add to the manifolds of $\mathbf{x}_i$ those which contain the same seed vertices as $\mathbf{x}_i$. The result is a manifold binary matrix $\tilde{B} = \{\tilde{b}_{ij}\}$, where rows $1 \le i \le |S|$ correspond to the input points $\mathbf{x}_i \in S$ and columns to manifolds M_j, and $\tilde{b}_{ij} = 1$ indicates that $\mathbf{x}_i \in M_j$ (Fig. 2, step C). A point can belong to none, one or more manifolds. Points which were not reached from any seed point, and are not seed points themselves, belong to no manifold, and are labeled as noise. Points which belong to more manifolds directly give us the manifold intersection curves, see *e.g.* the light-blue point at the intersection in Fig. 2 (step D), or the three yellow intersection curves in Fig. 1 f.

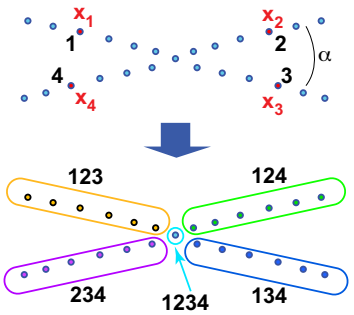

Figure 3: *Labeling of manifolds intersecting at acute angles.*

Figure 3 sketches two sampled manifolds that intersect at a small acute angle α. If the terminal 'branches' of the two manifolds are far enough from each other, phase A finds four seeds ($\mathbf{x}_1..\mathbf{x}_4$, labeled with IDs 1..4, Fig. 3 top). Next, the patch-level flood fill propagates these seeds. If α is very small around the intersection point, each ID will 'leak' from its manifold-segment to two other segments past the intersection point, *e.g.* the top-left ID 1 will flood the top-right and bottom-right branches. In phase C, we thus find *four* different manifold segments, having the unique labels 123, 124, 134, and 234, and one intersection point with label 1234. If α is larger, we have the situation in Fig. 2, where we find only two manifolds. When α tends to zero, then a single seed point is found, thus a single manifold is extracted.

3.3. Manifold reconstruction

We have now classified the input cloud S into a set of (intersecting) point-sets M_k, each one sampling a separate manifold. We can now use several existing methods to reconstruct the desired manifold surfaces $\mathcal{M}_k$. A good candidate is the ball pivoting method [BMR*99], which can reconstruct approximating triangle mesh surfaces from manifold point clouds. Figure 1 f shows the final result on our running example. The three surfaces are correctly extracted from the input cloud. Manifold intersections are marked in yellow. Non-manifold noise points, found by our extraction, are black.

Point classification into separate manifolds is essential to reconstruction quality: Feeding an entire, unclassified, cloud to most existing point cloud reconstruction methods [DG04, DG03, KBH06, AC01], create various artifacts, as shown further in Sec. 4.4, since such methods are not designed to handle noisy (self) intersecting clouds.

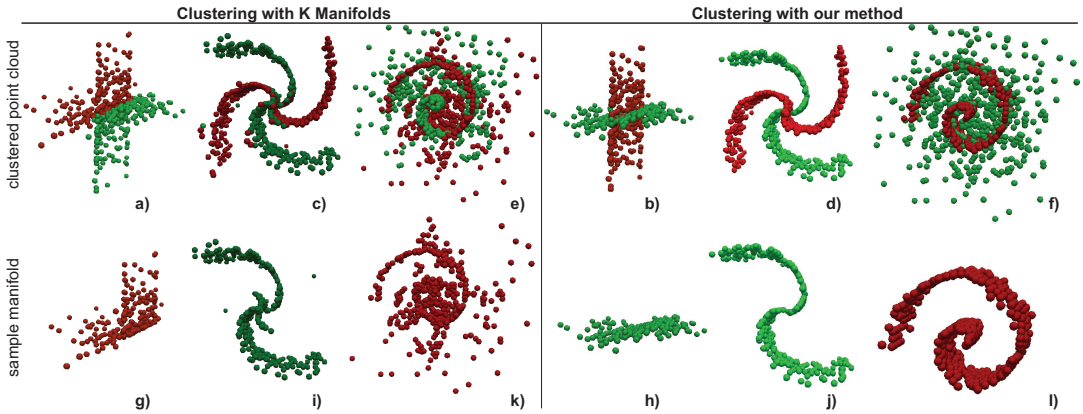

Figure 4: *Manifold clustering: Our algorithm (right)* vs *K-manifolds (left). Top row: clustering results. Bottom row: One selected manifold from the clustering, displayed separately for illustration purposes. Input clouds are similar to examples in [SP05].*

4. Applications

4.1. Manifold clustering

Manifold clustering of point clouds aims at labeling each point as belonging to a manifold. The labeled cloud can be further applied in manifold learning algorithms [SP05], shape segmentation, or as an input for surface reconstruction. We compare next our method with the well-known K-manifolds algorithm [SP05]. As input, we use non-uniformly sampled manifolds embedded in 3D, see Fig. 4: Two intersecting curved surfaces (a,b), a four-branch spiral (c,d), and a densely-sampled spiral cut by a sparsely-sampled plane (e,f). Since K-manifolds is very slow (several minutes for a few thousand points), we limited our tests to small clouds only. As Fig. 4 shows, our method finds the several manifolds present in the input robustly and eliminates the surrounding noise. In contrast, K-manifolds does handle well intersecting manifolds (Fig. 4 a *vs* 4 b, 4 c *vs* 4 d), but separates manifolds from noise less well (Fig. 4 k *vs* 4 l).

4.2. Noise removal from (intersecting) manifold clouds

Noise in point clouds can occlude the underlying surfaces and limit the success of surface reconstruction, clustering, and registration [CLK09]. We evaluated the noise-removal ability of our method for clouds sampling various surfaces. For a fraction ϕ of the points in each cloud, we add an outlier close to each cloud point $\mathbf{x}$ in a random direction and at a random distance from $\mathbf{x}$ ranging from 0 to 40% of the cloud's size. We next use our classification method (Sec. 3) to find noisy points, *i.e.*, which do not belong to any manifold. When removing these points, we should recover the initial noiseless cloud (ground truth).

We compared our method with the statistical outlier removal (SOR) [RMB*08] and tensor voting (TV) [MM07, MT00]. Figure 5 shows the results. We remove considerably more outliers (noise points) than SOR, and produce surfaces very close to the original. Our method works well even in spatially complex areas, see *e.g.* the gun handle. TV also effectively removes most noise points. However, TV tends to create spurious surfaces that connect the original surface with nearby noise points (Fig. 5, green markers: bunny years, mouse tail connected to body, and thickening of shuttle wings), or it removes details altogether (Fig. 5, red markers: shuttle tailwing and gun handle). The ratio of incorrectly kept noise *vs* incorrectly removed details is strongly influenced by TV's volume sampling resolution [MM07, MT00]: Higher resolutions remove less non-noise points, but also keep too many noise points. Lower resolutions sub-sample the TV tensor field. As such, fewer manifold structures can be traced through noisy areas (so more noise is removed), but also more original detail points are classified as noise and thus removed.

Figure 6 shows the difference between added and removed points, as percentage of the original input point-count, for values of ϕ ranging from 0 (no noise added) to 40% extra noise points added. Values are averaged for all models in Fig. 5. We did not include TV in this comparison, since its behavior with respect to keeping incorrect noise points *vs* removing correct original points strongly varies with parameter settings (as explained earlier) and also with the actual input cloud. For $\phi = 0$, both our method and SOR remove some non-noise points (5% and 15% respectively). Although, ideally, this value should be zero for a cloud consisting only of manifolds, we remove less non-noise points than SOR, which is desirable. As more noise is added, the ratio of added *vs* removed points stays stable with our method, *i.e.*, we remove only the added noise. In contrast, SOR removes increasingly more points than the added noise – up to almost 20% of the original non-noise points.

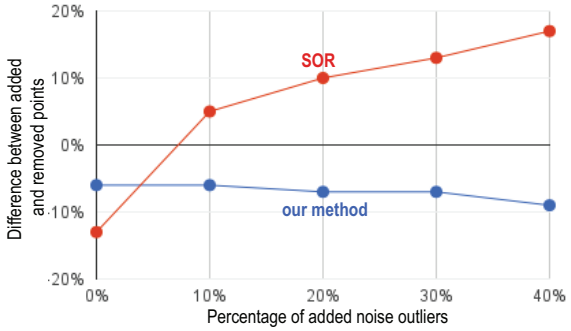

Figure 6: *Percentage of removed points in excess of added noise points for different noise amounts, averaged for the models in Fig. 5 Blue line: our method; red line: SOR method [RMB*08])*

4.3. Medial surface segmentation and reconstruction

Medial surfaces, or skeletons, contain the loci of maximally inscribed balls within a given shape [PSS*03, SP99]. Such surfaces consist of tens of manifolds of various sizes which meet along a set of Y-intersection curves [Dam06, LK07, CLK09]. Each manifold corresponds to a separate edge-set on the initial surface $\mathcal{S}$. Separating medial surfaces into their corresponding manifolds

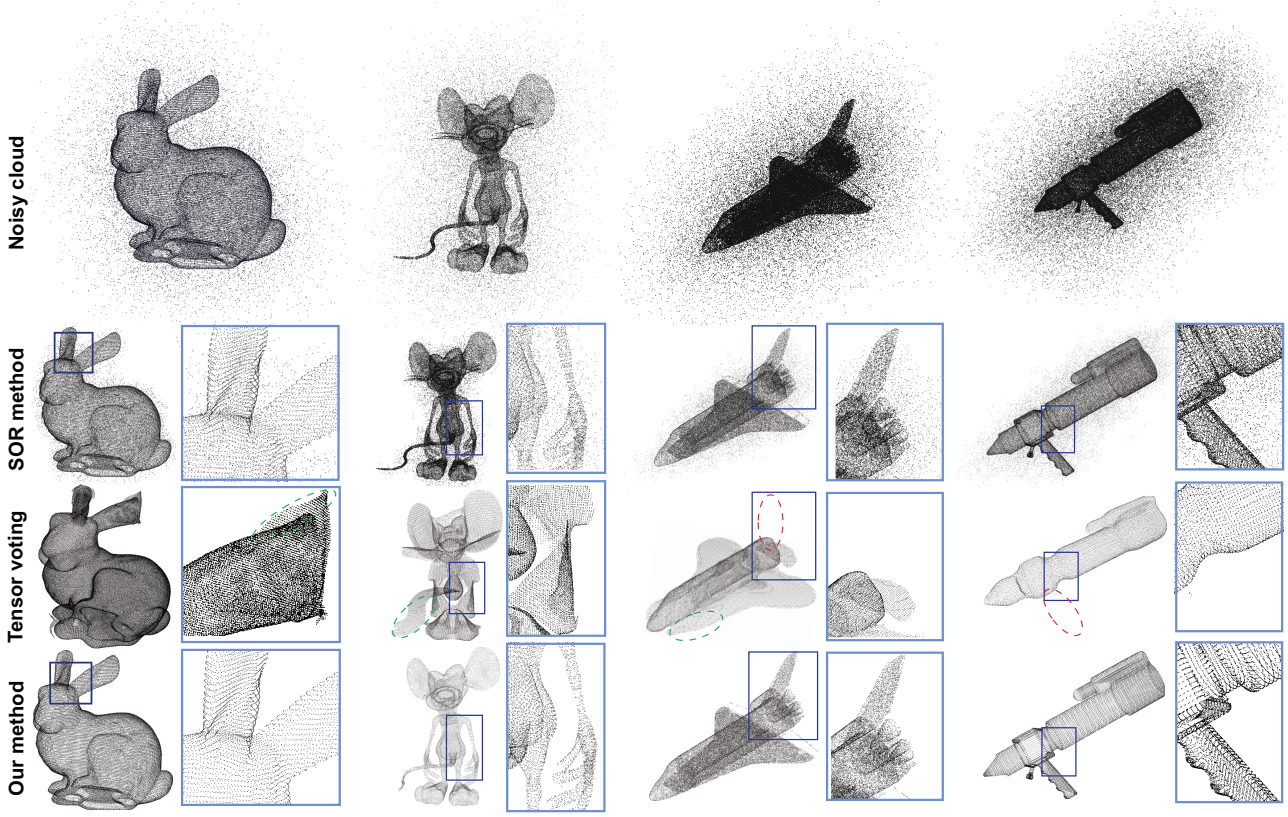

Figure 5: *Noise removal from manifold clouds. Noisy shapes (top) are denoised with SOR [RMB*08], TV [MM07], and our method. Zoom-ins show details marked in blue. TV: red markers show incorrectly removed points; green markers show not removed noise.*

is useful for applications such as shape classification [CLK09], shape matching [CC00], and segmentation [RT08a, RT08c].

Recently, Ma *et al.* proposed a very efficient GPU method to extract highly accurate medial point clouds of complex 3D shapes [MBC12], thereby making medial surfaces practical for applications on very large models. However, they stress that medial point *clouds* are of limited use since, for typical applications, one needs mesh-based skeletons. Creating such representations from medial clouds is *highly* challenging, since typical medial surfaces contain numerous (self-) intersections of very closely spaced, non-watertight, manifolds. Also, small-scale noise on the input shape creates spurious medial sheets, which show up as outlier points in the medial point cloud [SP99, MBC12, MGP10].

Using our method on the medial clouds computed by [MBC12], we can thus simultaneously

A: eliminate outlier (noise) medial points;
B: obtain a separate mesh for each medial manifold.

Figure 7 shows this on two medial clouds. Images (e-l) show the medial manifolds extracted by our method, colored differently for display purposes. As visible, our method captures well the complex medial topology and also robustly finds and eliminates the quite many noise points – compare images (i-l) where noise is drawn in black with images (e-h) where we eliminated the noise. Since such noise is unavoidable when extracting medial axes of discrete objects, our method has the added value of acting as a filter that generates clean medial surfaces.

For comparison ground-truth, we next used the high-accuracy medial surface reconstruction method in [TJ12] which works as follows: Given an input *mesh* shape, the medial cloud is computed following Ma *et al.*. Next, the medial cloud is simplified, or regularized, by eliminating noise points that correspond to small-scale surface details, using the medial importance metric in [RvWT08]. Finally, the input mesh is collapsed onto the simplified (clean) medial cloud to yield the medial mesh, using the so-called feature transform [RvWT08].

Our method produces nearly identical medial surfaces with [TJ12] (Fig. 7 a-d *vs* Fig. 7 e-h). Differences consist in small-scale holes in our manifolds, which do not exist in the reconstruction of [TJ12]. Upon closer inspection, we see that these holes are due to limitations of the ball pivoting method used following our point classification, and not due to the fact that our method incorrectly classifies manifold points as noise. Hence, we argue that our main goals **A** and **B** are reached. [TJ12] does not produce such small holes in the medial surface reconstruction. This is expected, since this method requires the input to be provided as an oriented *mesh* rather than an unoriented cloud. Moreover, [TJ12] needs the feature transform linking this mesh with its medial cloud, the relatively expensive and complicated importance computation for denoising [RvWT08] (which adversely affects goal **A**), and delivers a single unstructured medial surface (thus does not satisfy goal **B**). Although we require far less information (meshless unoriented medial clouds), *i.e.* use no knowledge that these points encode a medial surface, we can still extract separate and clean medial manifolds.

4.4. Surface segmentation and reconstruction

Segmentation: Given a point cloud S which samples a 3D surface $\mathcal{S}$, our method can segment $\mathcal{S}$ into smooth regions separated

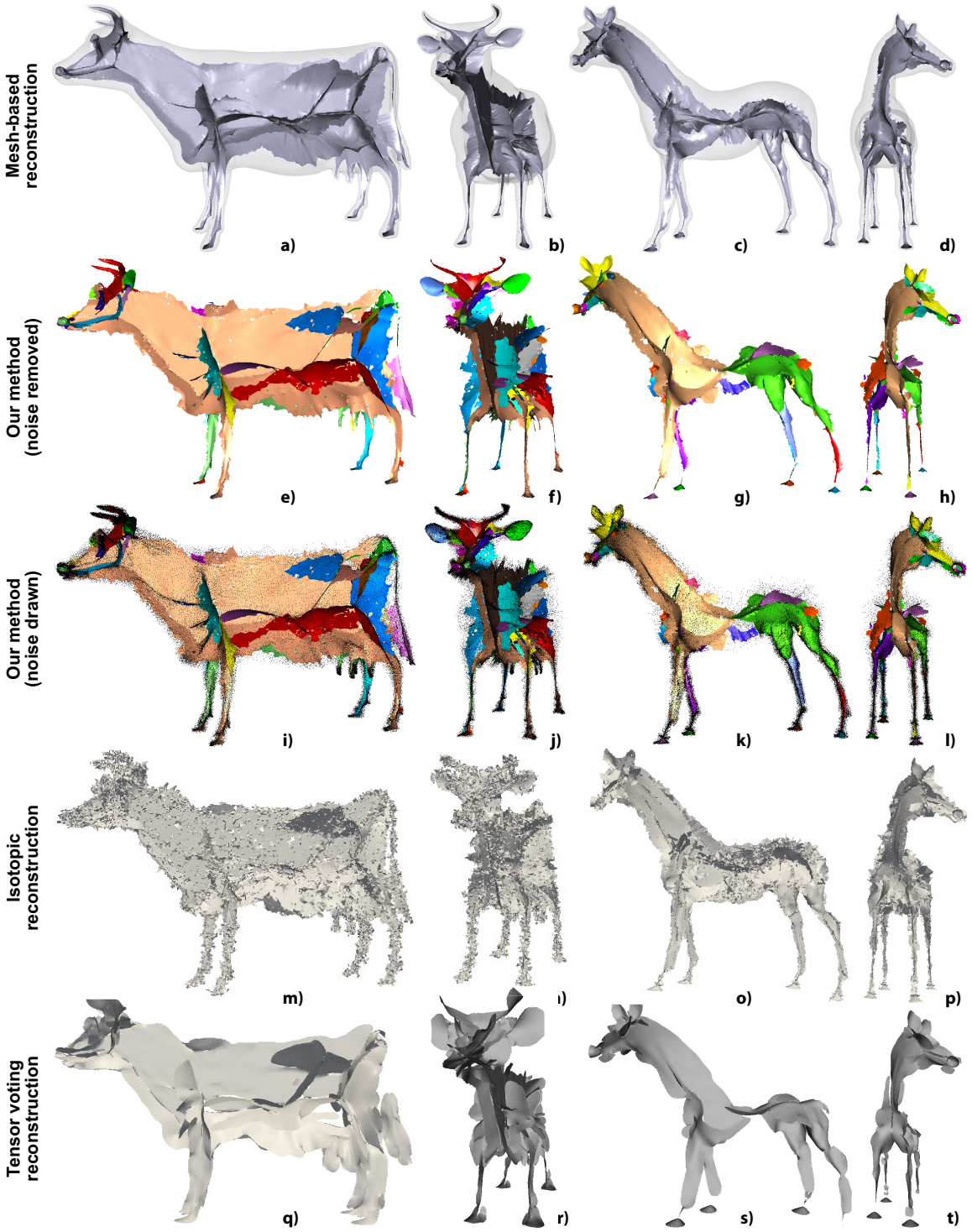

Figure 7: *Medial surface reconstruction: (a-d) High-accuracy ground truth [TJ12]. (e-h) Our method (noise points removed). (i-l) Our method, (detected noise rendered in black). (m-p) Isotopic reconstruction [DLRW09]. (q-t) Tensor voting reconstruction [MM07].*

by sharp edges. In contrast to many surface segmentation techniques, we do not require $\mathcal{S}$ to be closed, non-intersecting, be a single manifold, have normals, or be provided as a mesh. Figures 10 (d-l) illustrate our segmentation. Zoom-ins show details to provide insight into the point samples' distribution in various areas. To better visualize the segmented point-sets, we show their reconstructions by ball pivoting. Image (d) shows the segmentation of a cloud from a 3D structured-light scan of a room by a Kinect device. The point sampling is quite uniform, but noisy. The objects in the room are segmented correctly from each other and from the floor. The back wall (red) is only partially segmented since points in that range, far from the camera, are highly noisy, so they do not create a smooth manifold. In images (e-f), the various parts of the rabbit statue (body, plinth faces, inner ear surfaces, and heart detail) are correctly found. Since the cloud describes a *hollow* shape, we also get manifolds for the inner surfaces – see the head, body and plinth cavities in the half-opaque rendering in Fig. 10 f. Very thin, highly-curved, details like the

screw connecting the head to the heart shape are labeled as noise, as they have a higher local curvature than the imposed flatness d_{max} (Sec. 3.1.2). Images (i-j) show the segmentation of a cloud having several tens of intersecting shapes (sails, mast parts, and hull parts). In image (i), we used a low-resolution cloud (38K points). As seen in the zoom-in, fine details such as the masts have extremely few points. Like for the screw in the rabbit model, few or no patches are found along the mast, so no manifolds are found there (case **c** of local surface estimation, Sec. 3.1.1). Using a higher-resolution cloud (image (j), 74K points) finds the detail manifolds along the masts. Image (k) shows a cloud (125K points) of a CAD model of a car engine with over 100 self-intersecting surfaces and highly non-uniform sampling. Image (l) shows the extracted manifolds. As for the ship, too thin and/or sparsely sampled details are classified as outliers and no manifolds are extracted there.

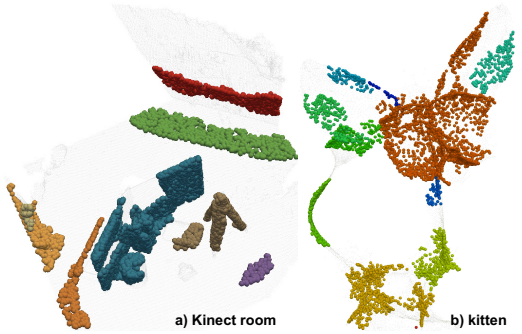

Figure 8: *Difference-of-normals (DoN) segmentation [ITHG12] applied to two point clouds (compare with results in Fig. 10 b,d).*

Figure 8 shows the results of the DoN method [ITHG12], as implemented in the PCL library [Ioa13], on to two of our clouds. As the DoN method requires, we first specified the scale range $[r_1, r_2]$ within which features are sought. We did this, by trial and error, so as to obtain the desired manifolds: For the room scene, we want to segment the various objects placed on the floor; for the kitten model, we want to find its medial manifolds. The DoN threshold $\Delta\mathbf{n}$ was set to various values in the range $[0.1, 0.5]$, similar to [ITHG12]. The resulting point clusters from DoN are shown as colored balls. Input points not segmented by DoN are shown light gray. We see that DoN can separate the small-scale objects from the room floor. Also, a large part of the kitten's medial manifolds are found and separated from each other. However, several issues are visible. First and foremost, the large manifolds present in the room (floor, walls) are not found – since these are not within the user-selected scale range $[r_1, r_2]$. If we carefully tune *both* r_1 and r_2, we can find parts of these manifolds. However, in the same time, we loose the smaller-scale object segments. For the kitten medial cloud, segmenting is even more challenging, since its manifolds are less well separated from each other. In contrast, our method better separates both large and small manifolds, for point clouds (Fig. 10 b,d). This is explained by the fact that our method does not search for manifolds at a *given* scale range, but tries to construct the largest possible manifolds allowed by the d_{min} and δ_{min} constraints.

Reconstruction and denoising: We next use our method to reconstruct surfaces formed by several intersecting manifolds embedded into noise, and compare our results (Figure 9) with several surface reconstruction methods, which are well known in the literature, easy to use, and their authors provided their implementations: isotopic reconstruction [DLRW09], ball pivoting [BMR*99], Poisson reconstruction [KBH06], and tensor voting [MM07, MT00]. Our method recovers best the various manifolds embedded into the noisy clouds. Isotopic reconstruction yields the next best results, as it can handle surfaces with boundaries, but still creates many small-scale spurious, non-manifold, surface fragments. Ball pivoting, as expected, cannot handle well dense noise and has problems for highly non-uniform clouds, like the rhino model whose rump has a much lower sampling density than the rest of the model (see zoom-ins in Fig. 9). However, if ball pivoting is executed *after* our clustering method, most noise points are discarded, given the built-in denoising of our method. This drastically improves the effectiveness of ball pivoting (compare Fig. 9, first and third rows). Also, since each segmented manifold is smooth, ball-pivoting can be used with a larger rolling-ball radius. This increases the ball pivoting robustness with respect to non-uniform sampling. Comparing the results of ball pivoting with isotopic reconstruction, we see that the latter suffers far less from noise and also does not produce undesired holes. Hence, isotopic reconstruction is a very good candidate to replace ball pivoting in our per-manifold reconstruction following the proposed denoising and classification. Poisson reconstruction, using an octree depth and solver divide value of 10, produces smooth surfaces, but cannot handle well (intersecting) manifolds with boundaries. Finally, tensor voting, used with a volume sampling resolution of 500^3 and point-neighborhood size $\sigma = 15$ (for details, see [MM07, MT00]) yields smooth surfaces, but fails in thin areas where parallel surfaces are close to each other, like the thin muzzle and horns of the elk model.

We also used the isotopic and tensor voting methods to extract manifolds from medial clouds (see Fig. 7, two bottom rows). We notice here similar issues as in Fig. 9. The isotopic method tends to create small-scale non-manifold noisy details. In contrast, tensor voting creates very smooth surfaces and handles manifold intersection regions very well. However, tensor voting has the tendency to extend the reconstructed manifolds far into the noisy regions in an anisotropic way, *i.e.* retains noise points which allow a smooth continuation of the medial manifolds but in the same time eliminates noise points from the same areas if these are not aligned with the reconstructed manifolds.

5. Discussion

Generality: We use a local feature detection for each point-cloud spatial neighborhood, followed by a global flood fill to find individual manifolds. Our approach has two main contributions. First, we extract manifold clouds from large amounts of embedding noise. Next, we segment manifold clouds from a single input cloud. This allows a *direct* reuse of existing surface reconstruction or shape analysis methods for point clouds on complex, multi-manifold, noisy clouds, even when such methods were designed to work only on smooth manifold clouds.

Robustness: Our method is robust to outlier noise (Figs. 5, 7). This feature is due to the hierarchical clustering of the local normal maps (Sec. 3.1.2) and the global clustering of local quasi-flat patches (Sec. 3.2). The first clustering separates locally relevant patches from noisy outliers, *i.e.* acts as a fine-grained noise filter. The second clustering ensures that only similar-orientation patches get grouped into smooth manifolds, *i.e.* acts as a coarse-grained noise filter. We pose no constraints on the cloud sampling density, as we detect local flatness using k nearest neighbors, rather than range-search with a user-prescribed radius,

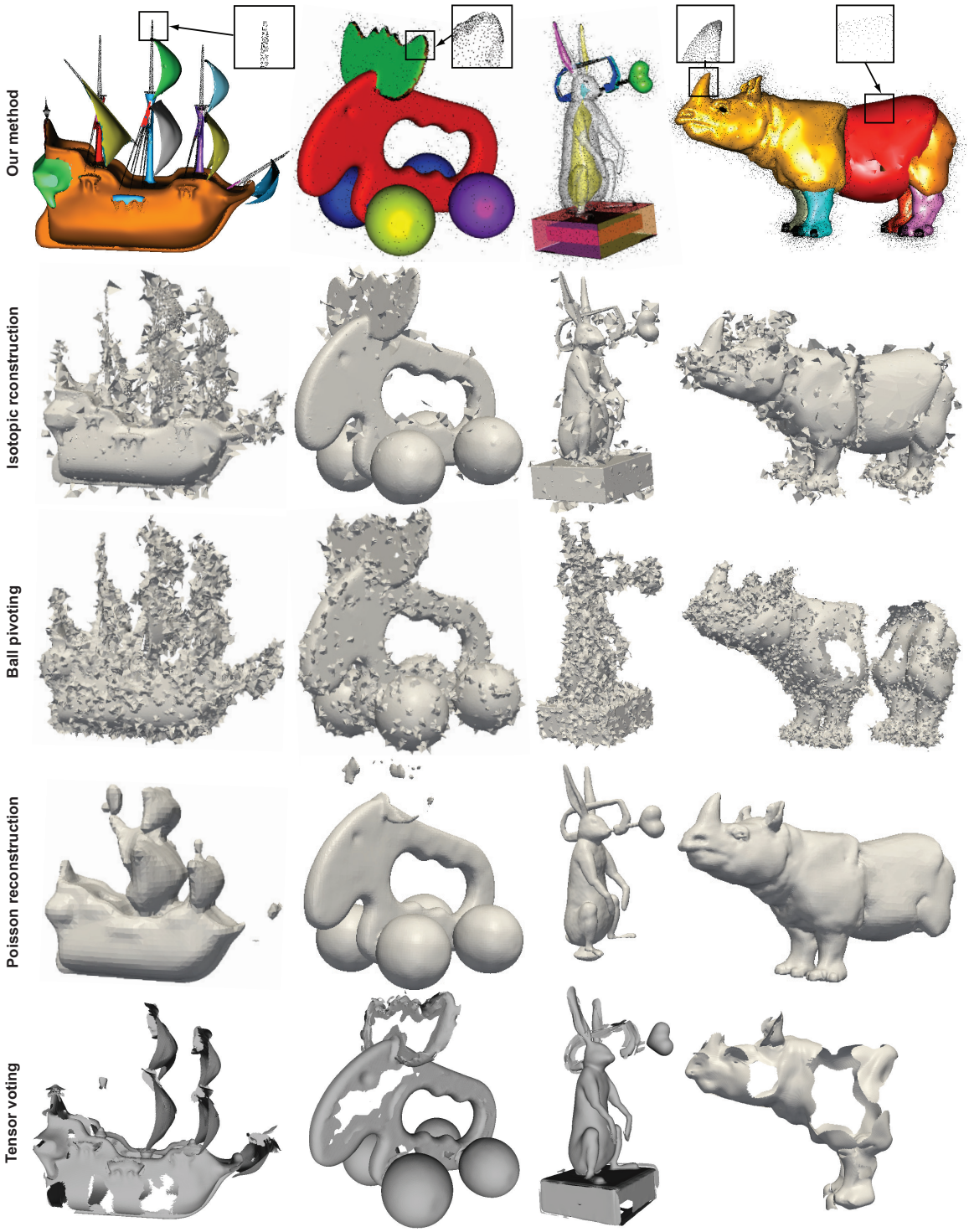

Figure 9: *Shape reconstruction comparison for noisy point clouds with intersecting manifolds. From top to bottom, rows: our method, isotopic reconstruction [DLRW09], ball pivoting [BMR*99], Poisson reconstruction [KBH06], and tensor voting [MM07]. Zoom-ins show point cloud details of selected model areas, for getting insight into the sampling distribution.*

such as *e.g.* [ITHG12]. The examples shown here indicate that we can handle clouds with a large amount of sampling-density variation (see Figs. 9 and 10). However, we acknowledge our limits: Highly non-uniformly sampled clouds (*e.g.* Figs. 10 i,k) will yield many points classified as noise.

Parameters: Our method has three parameters, as follows. The *neighborhood size* (k nearest neighbours) should be large enough to create triangles around a given point $\mathbf{x}$ to represent all possible surfaces crossing $\mathbf{x}$, but not too large so that meaningless surfaces are created. On all our models, regardless of sampling density, $k \in [7..10]$ provided good results. Setting k too large creates, along with the desired patches (that is, oriented along the sampled manifolds), several spurious patches at various random orientations. However, these are typically much fewer than the desired patches, so their effect gets filtered out by the Gauss map

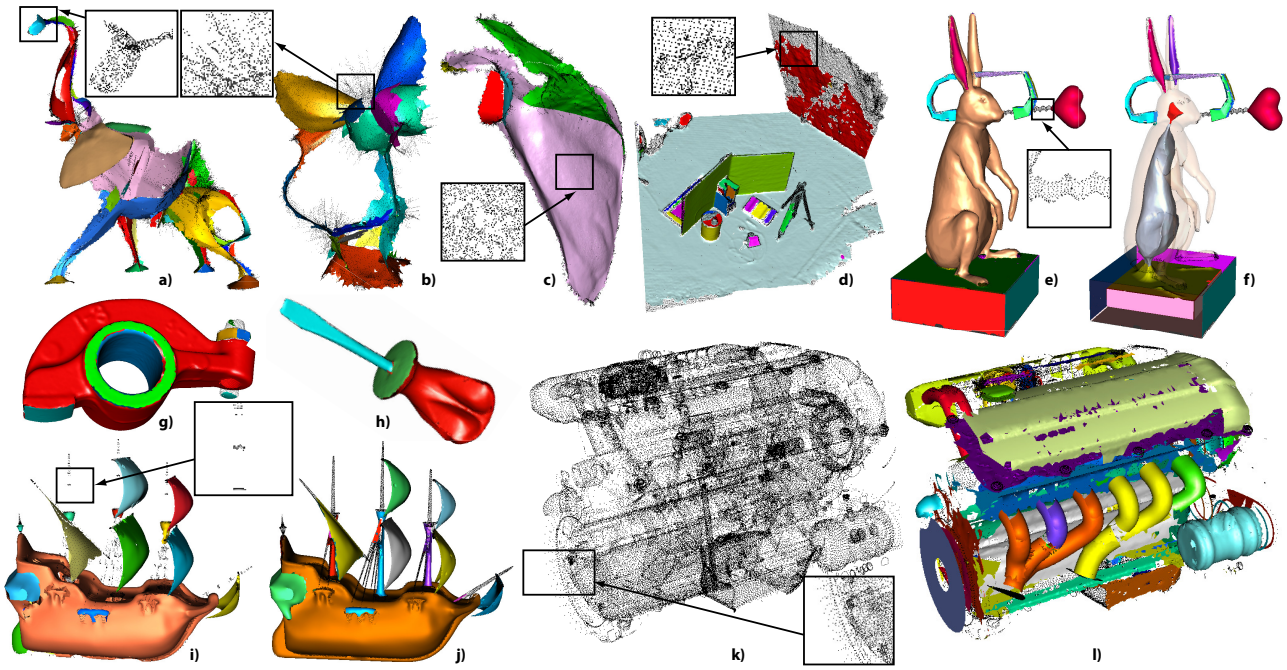

Figure 10: *Shape segmentation examples. (a-c) Medial surfaces. (d) Structured light acquisition. (e,f) Shape with inner surfaces. (g,h) Mechanical shapes. (i,j,k,l) Mix of different structures with varying sampling density.*

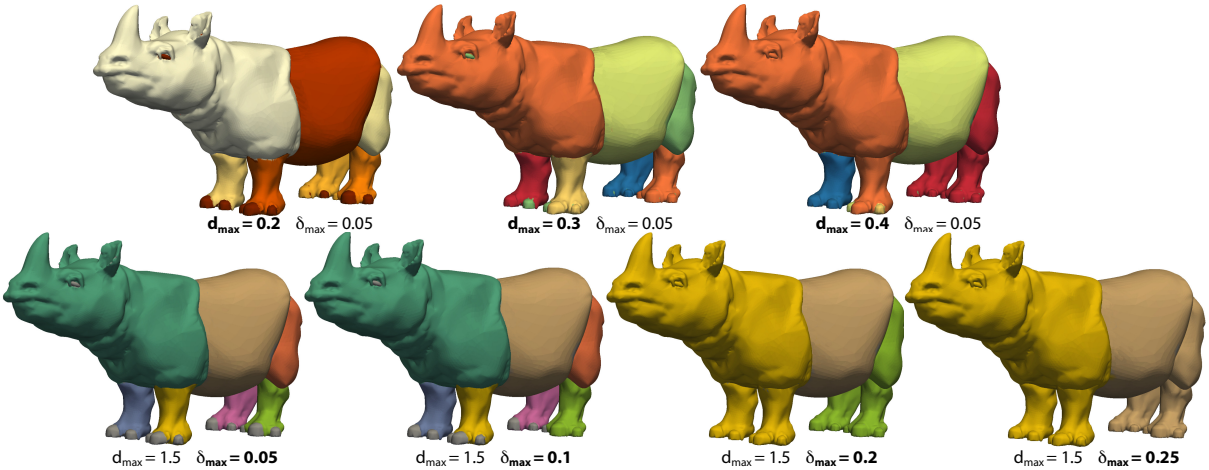

Figure 11: *Effect of parameters d_{max} and δ_{max} on manifold extraction results.*

segmentation and subsequent patch-level flood fill steps. The *local flatness* $d_{max} \in [0, \pi/2]$ (Sec. 3.1.2), set in this paper to 0.15, models the compromise between the extracted manifold smoothness, robustness to point-displacement noise, and manifold separation accuracy. Figure 11 (top row) illustrates this. Lower d_{max} yield more manifolds, since we allow normals to locally vary less within a manifold. This, for example, separates the toes and eyes detail of the rhino. Larger d_{max} values yield fewer, but potentially more curved, manifolds – the toes and eye get merged with the surrounding points. Increasing d_{max} even further merges the lower and upper leg fragments. For noisy datasets, larger d_{max} values also have the effect of classifying more points as noise, since less curvature is allowed within a manifold.Classifying points as noise prevents them from being treated by the ball pivoting reconstruction which, in turn, creates the small-scale holes mentioned in Sec. 4.3. However, we argue that such holes are not a classification problem, but a limitation of the postprocessing surface reconstruction method being used: If the aim is to remove such holes *and* still extract smooth manifolds, as controlled by d_{max}, then one should use a surface reconstruction tool that can handle non-uniformly sampled *manifold* clouds. If, however, the aim is to extract less smooth manifolds, then d_{max} should be increased. Another approach would be to set d_{max} adaptively as a function of the neighborhood's point distribution. However, how to do this and still guarantee the desired noise removal and manifold intersection detection is a topic of future research.

The patch similarity $\delta_{max} \in [0, \pi/2]$ (Sec. 3.2.1) acts similarly to d_{max}, but at a coarser scale (Fig. 11, bottom row). Small δ_{max} values yield relatively flat manifolds, *i.e.* split large manifolds along their crease lines. Large δ_{max} values yield less, and more curved, manifolds. For the rhino model, increasing δ_{max} progressively merges all toe details with the legs, and further merges legs with the rump. For the figures in this paper, we used $\delta_{max} \in [0.05, 0.1]$.

Performance: We implemented our method in C++ using *kd*-trees for nearest-neighbor searches [MA12]. We easily added

CPU parallelization for local classification (Sec. 3.1) and patch graph building (Sec. 3.2.1), since points and patches are treated independently. Table 1 shows timings for a single-threaded *vs* a 4-core 2.8 GHz MacBook 4 GB RAM laptop. Our method scales well with the number of available cores. If desired, a GPU (*e.g.* CUDA) port could be easily done for further speed-ups. Our CPU method takes roughly half the time of the GPU surface reconstruction from noisy clouds in [SW09]. For the same inputs, the tensor voting surface extraction in [MLT13] takes tens of minutes (at a volume resolution 500^3 and $\sigma = 15$). This is not surprising, since tensor voting uses several convolution passes of a large 3D tensor volume with a filter of kernel size σ. Decreasing the volume resolution speeds up tensor voting, but makes it loose small-scale manifold details. The DoN method [ITHG12] has similar costs to our method – 4.9 and 15.96 seconds for the kitten and room clouds in Fig. 8 respectively (for our timings, see Tab. 1).

Model	Points	Manifolds	Unclustered points	Time (sec.) (1-core CPU)	Time (sec.) (4-core CPU)
Mouse (N)	54829	61	21288	27.33	7.52
Space Shuttle (N)	106580	36	35902	34.68	12.58
Glue gun (N)	128887	124	46379	26.32	7.31
Stanford bunny (N)	42322	4	5976	20.97	4.86
Intersecting planes	30486	3	0	4.89	1.38
Rabbit	124998	40	4012	7.89	2.15
Engine	124481	426	34235	52.33	14.29
Ship	38240	67	9420	13.56	3.72
Ship 2	74573	117	18232	28.02	7.63
Screwdriver	27152	5	673	7.43	1.96
Rockerarm	43213	4	1322	12.12	3.59
Kinect room	135402	19	23733	36.87	9.75
Elephant (MS)	173012	28	16232	56.32	15.51
Cow (MS)	252180	51	22309	89.74	24.04
Scapula (MS)	116930	4	5022	37.28	10.25
Pig (MS)	225281	76	15471	73.38	20.23
Horse (MS)	120442	42	10503	39.65	11.10
Kitten (MS)	43510	23	9503	12.38	3.53

Table 1: *Timings for models shown in this paper (N=model with added noise (see Fig. 5); MS=medial surface (see Fig. 7))*

Computing the patch dissimilarity δ with full-linkage (Eqn. 2) is $O(N^2)$ worst-case for a patch with N normals on average. N is a few tens for all tested models. The early termination criterion ($\delta < \delta_{max}$, Sec. 3.2.1) makes this cost much lower in practice, roughly $O(N)$ (see also below). We also tested an average-linkage patch dissimilarity, *i.e.*, using the distance between patch average normals, which is $O(N)$. For the models in this paper, this gave a speed-up of about 20%, with a slight quality decrease – a few small-sized manifolds appear, since averages of two patch normal sets usually differ more than the closest normals of such sets. Given this, we chose to pay the small extra cost of full-linkage for increased manifold quality.

Limitations: If a neighborhood ν has no apparent 2D manifold structure, but a volumetric or one-dimensional point density, the local Gauss map has no clearly separated peaks. In that case, the neighborhood is labeled as noise (see *e.g.* ship's thinnest masts and its ropes, and the rabbit screw in Fig. 10). If this happens for most neighborhoods, *e.g.* in the case of a surface sampled overall by a thick point cloud, our manifold extraction will fail. This is expected, as our method is designed to find 2D manifolds only. In this respect, tensor voting is more general, as it can extract 2D surfaces, 1D curves, and junction points where several surfaces or curves intersect. However, for the manifold extraction case, both our method and tensor voting share the same limitation: Given a neighborhood where n manifolds intersect, if the sampling rate of these manifolds is too low with respect to n, neither method will be able to reliably separate these manifolds, and both methods will classify the neighborhood as noise. Examples of such configurations are visible for the car engine cloud segmentation (Fig. 10 k. When the sampling rate is high enough, our method can reliably extract several intersecting manifolds, as illustrated by the medial examples in Fig. 7.

Comparison: Many surface reconstruction methods exist, so the comparisons in Sec. 4 are clearly not exhaustive. Also, our method does aim to replace all general-purpose surface *reconstruction* methods from point clouds for all input clouds. Specifically, our method should be used when one needs to extract (self) *intersecting* manifolds with *boundaries* from *noisy* clouds. For this context, the only two comparable methods we are aware of are [CLK09] and [MM07]. For instance, [BMR*99] and [DLRW09] can handle boundaries, but are challenged by noise and intersections; [KBH06] can handle noise well, but not manifold boundaries and intersections. [DG04, SW09, MDD*10] can handle noise well, but cannot handle intersections. In the class of methods that explicitly handle intersections, we are around 5 times faster than [CLK09] (Tab. 1 *vs* Fig. 27 in [CLK09]). Note that [CLK09] does not appear to include the medial surface computation cost. If one added that cost, our method is over 15 times faster. Also, [CLK09] is considerably more complex to implement, as it requires a separate robust extraction of 3D medial surfaces from point clouds [LK07]. Compared to [MM07], we are over two orders of magnitude faster, and handle better manifolds which are close to each other.

6. Conclusions

We have presented a method to robustly segment unoriented point clouds into smooth manifolds. We handle clouds with complex combinations of an unknown number of potentially (self) intersecting, open or closed, manifolds embedded into noise. Using a clustering approach, we find the most probable local quasi-flat surface patches passing through each point, and merge these patches to classify the input points into manifolds or noise. Classified per-manifold points are reconstructed into a mesh using out-of-the-box cloud reconstruction methods. Compared to other methods, we allow input points to be classified as belonging to no manifold (*e.g.* noise), one manifold, or being on the intersection of several manifolds. This allows handling highly noisy clouds or point clouds having complex structures. The method allows for an easy parallelization, is simple to use, and has robust default parameter values. We demonstrate our method on several point clouds with use-cases in manifold extraction from embedding noise, regularized medial-surface reconstruction, and point-cloud surface segmentation, and point-cloud reconstruction.

One application area for further study is CAD reverse engineering, *i.e.*, the recovery of separate 3D parts, or components, from a point cloud where topology information has been lost. A different direction is to consider other data than point clouds. By replacing the definitions of spatial neighborhood and orientation similarity, we could extract smooth manifold-like structures embedded in other spaces, *e.g.* find bundles or sheets of fibers in tractography datasets or segment multivariate spatial data.

References

[AC01] AMENTA N., CHOI S.: The power crust. In *Proc. SMA* (2001), ACM, pp. 65–73. 2, 4

[BKNS00] BREUNIG M., KRIEGEL H.-P., NG R., SANDER J.: LOF: Identifying density-based local outliers. In *Proc. SIGMOD* (2000), ACM, pp. 93–104. 2

[BMR*99] BERNARDINI F., MITTLEMAN J., RUSHMEIER H., SILVA C., TAUBIN G.: The ball-pivoting algorithm for surface reconstruction. *IEEE TVCG 5*, 4 (1999), 349–359. 2, 4, 8, 9, 11

[BSK04] BOTSCH M., SPERNAT M., KOBBELT L.: Phong splatting. In *Proc. PBG* (2004), pp. 25–32. 3

[CC00] COSTA L., CESAR R.: *Shape analysis and classification: Theory and practice*. CRC Press, 2000. 6

[CH06] CAO W., HARALICK R.: Nonlinear manifold clustering by dimensionality. In *Proc. ICPR* (2006), IEEE, pp. 920–924. 1

[CLK09] CHANG M.-C., LEYMARIE F. F., KIMIA B. B.: Surface reconstruction from point clouds by transforming the medial scaffold. *CVIU 113*, 11 (2009), 1130–1146. 1, 2, 5, 6, 11

[CSD04] COHEN-STEINER D., DA F.: A greedy Delaunay-based surface reconstruction algorithm. *Visual Comput. 20*, 1 (2004), 4–16. 2

[Dam06] DAMON J.: Global medial structure of regions in $\mathbf{R}^3$. *Geometry and Topology 10* (2006), 2385–2429. 5

[DG03] DEY T., GOSWAMI S.: Tight Cocone: A water-tight surface reconstructor. In *Prog. SMA* (2003), pp. 127–134. 2, 4

[DG04] DEY T., GOSWAMI S.: Provable surface reconstruction from noisy samples. In *Proc. Ann. SCG* (2004), pp. 428–438. 2, 4, 11

[dHINM04] DE HOON M., IMOTO S., NOLAN J., MYIANO S.: Open source clustering software. *Bioinformatics 20*, 9 (2004), 1453–1454. 2

[DLRW09] DEY T., LI K., RAMOS E., WENGER R.: Isotopic reconstruction of surfaces with boundaries. *CGF 28*, 5 (2009), 1371–1382. 2, 7, 8, 9, 11

[Flö09] FLÖRY S.: Fitting curves and surfaces to point clouds in the presence of obstacles. *CAGD 26*, 2 (2009), 693–707. 2

[Gol09] GOLDBERG A. B.: Multi-manifold semi-supervised learning. *Proc. AISTATS* (2009), 169–176. 1

[GP05] GAUSS C. F., PESIC P.: *General investigations of curved surfaces*. Dover Books on Mathematics. Dover Publ., 2005. 2

[Haw80] HAWKINS D. M.: *Identification of Outliers*. Chapman and Hall, London, 1980. 2

[HH05] HARALICK R., HARPAZ R.: Linear manifold clustering. *Machine Learning and Data Mining in Pattern* (2005), 132–141. 1

[Ioa13] IOANNOU Y.: DoN implementation in the PCL library, 2013. URL: pointclouds.org/documentation/tutorials/don_segmentation.php. 8

[ITHG12] IOANNOU Y., TAATI B., HARRAP R., GREENSPAN M.: Difference of normals as a multi-scale operator in unorganized point clouds. *ArXiv e-prints* (2012). arXiv:1209.1759. 1, 8, 9, 11

[JKN98] JOHNSON T., KWOK I., NG R.: Fast computation of 2-dimensional depth contours. In *Proc. KDDM* (1998), AAAI Press, pp. 224–228. 2

[JR07] JALBA A., ROERDINK J.: Efficient surface reconstruction using Coulomb potentials. *IEEE TVCG 13*, 6 (2007), 1512–1519. 1, 2

[KBH06] KAZHDAN M., BOLITHO M., HOPPE H.: Poisson surface reconstruction. In *Proc. SGP* (2006), pp. 61–70. 2, 4, 8, 9, 11

[KGB06] KUSHNIR D., GALUN M., BRANDT A.: Fast multiscale clustering and manifold identification. *Patt. Recog. 39*, 10 (2006), 1876–1891. 1

[KNT00] KNORR E. M., NG R. T., TUCAKOV V.: Distance-Based Outliers: Algorithms and Applications. *J. VLDB 8*, 3-4 (2000). 2

[LK07] LEYMARIE F., KIMIA B.: The medial scaffold of 3D unorganized point clouds. *IEEE TVCG 29*, 2 (2007), 313–330. 5, 11

[MA12] MOUNT D., ARYA S.: Approximate nearest neighbor search software, 2012. URL: www.cs.umd.edu/~mount/ANN. 10

[MBC12] MA J., BAE S. W., CHOI S.: 3D medial axis point approximation using nearest neighbors and the normal field. *Visual Comput. 28*, 1 (2012), 7–19. 1, 6

[MDD*10] MULLEN P., DEGOES F., DESBRUN M., COHEN D., ALLIEZ P.: Signing the unsigned: robust surface reconstruction from raw pointsets. *CGF 29*, 5 (2010), 1733–1741. 2, 11

[MGP10] MIKLOS B., GIESEN J., PAULY M.: Discrete scale axis representations for 3D geometry. In *Proc. ACM SIGGRAPH* (2010), pp. 394–493. 6

[MLT13] MEDIONI G., LEE M.-S., TANG C.-K.: TV3D 3D tensor voting library, 2013. URL: www.cse.ust.hk/~cstws/research/TensorVoting3D. 11

[MM07] MORDOHAI P., MEDIONI G.: *Tensor voting: a perceptual organization approach to computer vision and machine learning*. Morgan & Claypool, 2007. 2, 5, 6, 7, 8, 9, 11

[MT00] MEDIONI G., TANG C.-K.: Tensor voting: Theory and applications. In *Proc. RFIA* (2000). 2, 5, 8

[PNF*08] POLLEFEYS M., NISTÉR D., FRAHM J. M., AKBARZADEH A., MORDOHAI P., CLIPP B., ENGELS C., GALLUP D., KIM S. J., MERRELL P., SALMI C., SINHA S., TALTON B., WANG L., YANG Q., STEWÉNIUS H., YANG R., WELCH G., TOWLES H.: Detailed real-time urban 3D reconstruction from video. *Int. J. Comput. Vision 78*, 2-3 (2008), 143–167. 1

[PSS*03] PIZER S., SIDDIQI K., SZEKELY G., DAMON J., ZUCKER S.: Multiscale medial loci and their properties. *IJCV 55*, 2-3 (2003), 155–179. 5

[RH07] RINGBECK T., HAGEBEUKER B.: A 3D Time of Flight Camera for Object Detection. *Measurement 9* (2007), 867–879. 1

[RMB*08] RUSU R. B., MARTON Z. C., BLODOW N., DOLHA M., BEETZ M.: Towards 3D point cloud based object maps for household environments. *Robotics and Autonomous Systems 56*, 11 (2008), 927–941. 2, 5, 6

[RT08a] RENIERS D., TELEA A.: Patch-type segmentation of voxel shapes using simplified surface skeletons. *CGF 27*, 7 (2008), 1837–1844. 6

[RT08b] RENIERS D., TELEA A.: Robust segmentation of voxel shapes using medial surfaces. In *Proc. SMI* (2008). 1

[RT08c] RENIERS D., TELEA A.: Segmenting simplified surface skeletons. In *Proc. DGCI* (2008), Springer, pp. 132–145. 6

[RVV06] RABBANI T., VAN DEN HEUVEL F., VOSSELMANN G.: Segmentation of point clouds using smoothness constraint. *Intl. Archives of Photogrammetry Remote Sensing and Spatial Information Sciences 36*, 5 (2006), 1–6. 1

[RvWT08] RENIERS D., VAN WIJK J. J., TELEA A.: Computing multiscale skeletons of genus 0 objects using a global importance measure. *IEEE TVCG 14*, 2 (2008), 355–368. 6

[SHG06] SITEK A., HUESMAN R. H., GULLBERG G. T.: Tomographic reconstruction using an adaptive tetrahedral mesh defined by a point cloud. *IEEE Trans. Med. Imag. 25*, 9 (2006), 1172–1179. 1

[Sot07] SOTOODEH S.: Hierarchical clustered outlier detection in laser scanner point clouds. In *Intl. Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* (2007), vol. 36, ISPRS, pp. 383–388. 2

[SP99] SIDDIQI K., PIZER S.: *Medial Representations: Mathematics, Algorithms and Applications*. Springer, 1999. 5, 6

[SP05] SOUVENIR R., PLESS R.: Manifold Clustering. In *Proc. ICCV* (Beijing, China, 2005), pp. 648–653. 1, 5

[SSZCO10] SHALOM S., SHAMIR A., ZHANG H., COHEN-OR D.: Cone carving for surface reconstruction. *ACM TOG 29*, 6 (2010), 547–555. 2

[SW09] SHEUNG H., WANG C.: Robust mesh reconstruction from unoriented noisy points. In *Proc. SPM* (2009), ACM, pp. 13–24. 2, 11

[TdSL00] TENENBAUM J. B., DE SILVA V., LANGFORD J. C.: A global geometric framework for nonlinear dimensionality reduction. *Science 290*, 5500 (2000), 2319–2323. 1

[TJ12] TELEA A., JALBA A.: Computing curve skeletons from medial surfaces of 3D shapes. In *Proc. Theory & Practice of Computer Graphics* (2012), Eurographics, pp. 132–139. 6, 7

[TOZ*11] TAGLIASACCHI A., OLSON M., ZHANG H., HAMARNEH G., COHEN-OR D.: VASE: Volume-aware surface evolution for surface reconstruction from incomplete point clouds. *CGF 30*, 5 (2011), 1563–1571. 2

[UH03] UNNIKRISHNAN R., HEBERT M.: Robust extraction of multiple structures from non-uniformly sampled data. In *Proc. IROS* (2003), pp. 1322–1329. 4

[UH07] UNNIKRISHNAN R., HEBERT M.: Denoising manifold and non-manifold point clouds. In *Proc. BMVC* (2007). 2

[VMS05] VIDAL R., MA Y., SASTRY S.: Generalized Principal Component Analysis. *IEEE TPAMI 27*, 12 (2005), 1945–1959. 1

[WHH10] WEBER C., HAHMANN S., HAGEN H.: Sharp feature detection in point clouds. In *Proc. SMA* (2010), pp. 175–186. 3

[WK04] WU J., KOBBELT L.: Optimized sub-sampling of point sets for surface splatting. *CGF 23*, 3 (2004), 643–652. 3