Graph-based representations of point clouds

Mattia Natali, Silvia Biasotti, Giuseppe Patanè, Bianca Falcidieno

Istituto di Matematica Applicata e Tecnologie Informatiche
Consiglio Nazionale delle Ricerche
Via De Marini 6
16149 Genova, Italy

Abstract

This paper introduces a skeletal representation, called Point Cloud Graph*, that generalizes the definition of the Reeb graph to arbitrary point clouds sampled from m-dimensional manifolds embedded in the d-dimensional space. The proposed algorithm is easy to implement and the graph representation yields to an effective abstraction of the data. Finally, we present experimental results on point-sampled surfaces and volumetric data that show the robustness of the Point Cloud Graph to non-uniform point distributions and its usefulness for shape comparison.*

Keywords: Graph-based representations, point clouds, shape abstraction, shape comparison.

1. Introduction

Shape representation from data samples is a well known problem in many fields of science and engineering. In most of cases, the data samples are assumed to approximate a manifold embedded in a higher-dimensional space. A typical example is shape reconstruction from range images obtained by scanning real 3D objects: here, the dimension of the embedding space is three (i.e., the dimension of the Euclidean space) while the intrinsic dimensionality of the surface is two.

Generally, point sets are supposed to densely sample the boundary of a smooth surface, which is reconstructed through moving least-squares [4, 6, 41], implicit [1] and Voronoi/ Delaunay [5, 29] approximations. Since point sets are able to represent arbitrarily complex 3D shapes without needing the explicit storage of the manifold connectivity, they have become a surface representation alternative to polygonal meshes and have been widely used for several applications. Among them, we mention ray tracing [2], surface reconstruction [43, 68], sampling [4], simplification [52], segmentation [8], spectral analysis [51], machine learning [12, 62], progressive rendering and streaming [33]

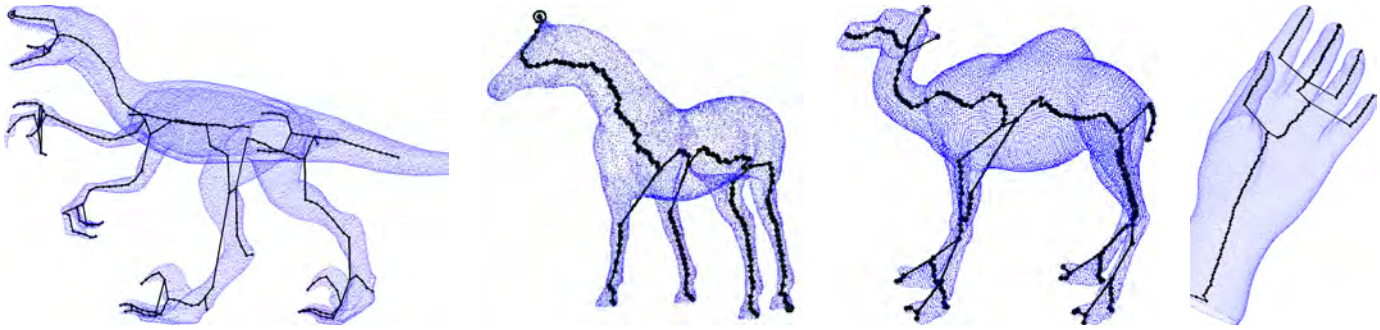
Since only few works address the problem of computing high-level representations [14] of point clouds, this paper tackles the problem of defining graph-based representations of point clouds. By *skeletal representation* we mean an explicit graph-like coding of the essential structure of the shape underlying the input point cloud and the way the shape components glue together to form the whole.

In general, a skeletal representation yields a compact and expressive shape abstraction, which attempts to reflect the human intuition. The use of sufficiently concise, informative, and easily computable skeletal representations, instead of the whole models, may facilitate the comparison process. In fact, the search in a database for an object similar to a query can be nearly impossible if approached by simply comparing point clouds or bulks of thousand triangles. Important aspects that drive the definition of a skeletal representation are the invariance to translations, rotations, and scalings; the identification and abstraction of shape features; the independence of the representation with respect to the shape embedding and discretization; the property of being medial with respect to the shape.

From a general perspective, two main philosophies drive the definition of skeletal representations on triangulated surfaces: (i) defining a *medial* structure representation that always falls inside the shape and is equidistant from the shape boundary at each point or (ii) explicitly representing how the basic components of the shape are glued together to form the whole. We highlight that in the latter case, the skeletal representation is not necessarily medial with respect to the shape.

Main examples of medial representations are: the Medial Axis [19, 20, 58], which in 3D may contain both curve segments and sheets with non-manifold connections; the medial curves computed through segmentation [22, 25, 39]; the medial geodesic skeleton [30]; the mesh contraction based on Laplacian smoothing [9] and surface-based operations [3, 26, 60].

As a representative of the second class of skeletal representations, the *Reeb graph* [54] codes the evolution and arrangement of the level sets of a real function $f : \mathcal{M} \to \mathbb{R}$, defined over a manifold $\mathcal{M}$. The Reeb graph has been proven to be always a 1-dimensional complex and in its original definition provides a description that is not invertible. This means that the input shape cannot be exactly recovered from the Reeb graph and the geometric information stored in its nodes and arcs. Since the Reeb graph is parametric with respect to the input map, changing f induces different descriptions of the same surface, which can be tackled to shape comparison [38], segmentation [13], and visualization [63]. Examples of functions effectively used



(a) (b) (c) (d)

Figure 1: (a-d) Graph representations of several point-sampled surfaces with different features and sampling densities. The original triangle mesh representing the model in (a) has 11 components.

in applications are geodesic distances, harmonic and Laplacian eigenfunctions. Efficient algorithms for the computation of the Reeb graph exist for polyhedral surfaces [24, 50], volume models [63], and higher dimensional data [49, 32, 37].

A main limitation of the aforementioned approaches is that they assume a manifold connectivity for the representation of the input shape, thus making skeletons unavailable for non-manifold models, such as triangle soups and point sets in arbitrary dimension. Concerning point sets, the main approaches for skeletal representations are based on medial-like concepts and exploit the identification of a rotational symmetry axis [61] through symmetry detection; the Voronoi diagrams [47]; a thinning process based on the 1D moving least-squares construction [40]; a Laplacian-based contraction [21]; and the maximal spheres inscribed inside the input point set [56]. Methods that approximate the Medial Axis generally assume that the point cloud densely samples the external surface of a solid [5, 28]. A few methods generalize the Reeb graph to point clouds, either using the level sets of geodesic distance functions and a discrete Reeb graph coding [67, 65] for human body scans, or introducing a cluster-based multi-resolution structure, which may admit input functions of co-dimension higher than one [59].

Overview and contribution. This paper introduces a skeletal representation, called *Point Cloud Graph*, which generalizes the definition of the Reeb graph to arbitrary point clouds sampled from m-dimensional manifolds embedded in the d-dimensional space. The input point sets represent single shapes, scenes with several objects, and volumetric data, without assumptions on the quality of the input point sets in terms of noise, missing data, and low sampling densities.

The proposed approach computes the Point Cloud Graph of the point set $\mathcal{P} := \{\mathbf{p}_i\}_{i=1}^{n} \subseteq \mathbb{R}^d$ by joining the connected components of strips of a real function $f : \mathcal{P} \to \mathbb{R}$. To extract this skeletal representation, we exploit the local connectivity of the k-nearest neighbor graph of $\mathcal{P}$, which is also used to identify the connected components of $\mathcal{P}$ ($\mathcal{P}$ may represent a set of shapes) and of the strips of $\mathcal{P}$ induced by f. Intuitively, the Point Cloud Graph codes the points according to their nearness but it might distort large scale distances. This is a desirable property in those applications where large scale distances carry a little meaning. Moreover, the flexibility of the choice of the function f makes the Point Cloud Graph suitable for several applications (e.g., shape abstraction, sketching, comparison).

Replacing the level sets with strips leads to a robust computation of the skeletal curve when $\mathcal{P}$ has deficiencies in terms of noise, missed data, and multiple components. Additionally, this choice avoids the need of computing the moving least-squares surface underlying $\mathcal{P}$ and allows us to extract the skeleton of an arbitrary set of points in $\mathbb{R}^d$, without requiring a local smoothness or connectivity of the underlying shape. Finally, in $\mathbb{R}^3$ the Point Cloud Graph reduces to the Reeb graph of the underlying manifold as the point cloud becomes denser. Fig. 1 shows the results of the proposed algorithm on point-sampled surfaces.

The main contribution of the proposed approach relies on its generality with respect to $(\mathcal{P}, f)$ and the capability of handling point sets in any dimension. Concerning the first contribution, our computation of the Point Cloud Graph handles shapes that are not necessarily described as an assembly of cylindrical patches and joints as in [61]; is restricted neither to point sets representing 0-genus surfaces nor to a specific scalar function; does not use any template to drive the graph extraction, as in [65] for human scans. Finally, we directly compute the skeletal representation without a graph post-processing, which is generally required by the Laplacian-based contraction [21].

According to the definition of the Reeb Graph [54], the Point Cloud Graph codes a point set $\mathcal{P}$ in a 1D representation, whose properties depend on those ones of f and the shape underlying $\mathcal{P}$. Note that reducing the width of the partition of the interval containing the image of f forces the strips to converge to the corresponding level sets. Even though the Point Cloud Graph cannot be used to exactly recover the input data (*invertibility property*), the graph is useful to compute an approximation of the input point set through an implicit representation $\Sigma := \{\mathbf{p} : F(\mathbf{p}) = 0\}$ with radial basis functions [61].

Concerning the computation of the Point Cloud Graph for higher dimensional data, the proposed approach remains unchanged by substituting surface with volume strips. The Point Cloud Graph, as well as the Reeb graph, does not distinguish all the features in higher dimensions [16]; in fact, in case of volumetric data it may not code cavities. Since the Point Cloud Graph is intended to generalize the Extended Reeb graph to point clouds embedded in $\mathbb{R}^d$, differently from [59] we consider

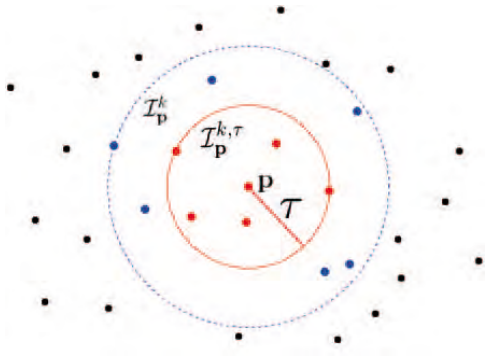


Figure 2: Example of *k-nearest neighbor* $\mathcal{I}^k_{\mathbf{p}}$, which includes the k points of $\mathcal{P}$ closest to $\mathbf{p}$; $\mathcal{I}^{k,r}_{\mathbf{p}}$ is the set of points in $\mathcal{I}^k_{\mathbf{p}}$ whose distances from $\mathbf{p}$ are lower than r.

only $\mathbb{R}$ as co-domain of the function f and we do not admit the overlap between clusters of points. Admitting overlapping domains would not be meaningful for the equivalence relation in Definition 2.2. Furthermore, in $\mathbb{R}^3$ the number of loops of the corresponding Reeb graph would be no more equal to the genus of the input surface.

The paper is organized as follows. In Section 2, we provide formal definitions of the point cloud connectivity, introduce the notion of Point Cloud Graph, and detail our graph extraction technique. In Section 3, we present our experimental settings, discuss the robustness of the method with respect to noise and parameters, and show shape matching as a possible application. Conclusions and future developments are provided in Section 4.

2. The Point Cloud Graph

Our graph representation broadens to point sets concepts related to the Reeb graph; in particular, it generalizes the Extended Reeb graph (ERG) originally defined on triangle meshes [15] and the Discrete Reeb graph [67]. We name this new representation *Point Cloud Graph* (PCG). Similarly to the ERG, the aim of the method is to extract the PCG of the pair $(\mathcal{P}, f)$, where $\mathcal{P} := \{\mathbf{p}_i\}_{i=1}^n \subseteq \mathbb{R}^d$ is a set of points in $\mathbb{R}^d$ and $f : \mathcal{P} \to \mathbb{R}$ is a scalar function defined on $\mathcal{P}$, i.e., $f(\mathbf{p}_i)$ is known for each point $\mathbf{p}_i \in \mathcal{P}$. The idea behind our approach is to organize the data into a family of strips of points of $\mathcal{P}$, to associate a node to each connected component of the strip, and to insert an arc between two nodes if their distance is less than a user given threshold.

In the following, we introduce the connectivity among the points of $\mathcal{P}$ (Section 2.1), the definition of the Point Cloud Graph of $(\mathcal{P}, f)$ (Section 2.2), and its computation (Section 2.3).

2.1. Connectivity and connected sets of point clouds

Within the *k-nearest neighbor graph* $\mathcal{T}$ of $\mathcal{P}$, each point $\mathbf{p}_i \in \mathcal{P}$ is associated to its k nearest points of $\mathcal{P}$, which identify the *neighbor* $\mathcal{I}^k_{\mathbf{p}_i} := \{\mathbf{p}_{j_s}\}_{s=1}^k$ of $\mathbf{p}_i$. In a similar way, the σ-nearest neighbor of $\mathbf{p}_i$ is defined as the set of points of $\mathcal{P}$ that fall inside the sphere of center $\mathbf{p}_i$ and radius σ. Finally, we introduce the neighbor

$$\mathcal{I}^{k,\tau}_{\mathbf{p}_i} := \{\mathbf{p}_{j_s} \in \mathcal{I}^k_{\mathbf{p}_i} : \|\mathbf{p}_i - \mathbf{p}_{j_s}\|_2 \leq \tau\},$$

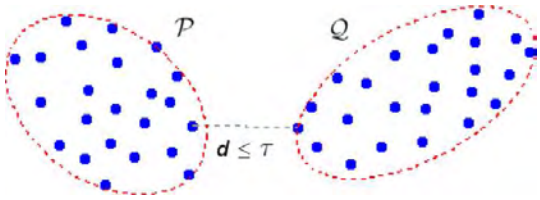


Figure 3: Connectivity between two point sets $\mathcal{P}$ and $\mathcal{Q}$.

which contains the elements of $\mathcal{I}^k_{\mathbf{p}_i}$ whose distance from $\mathbf{p}_i$ is equal to or lower than τ (Fig. 2). These different types of neighbors will be used to extract the connected components of the strips and join the corresponding nodes of the graph without meshing the point set (Fig. 3). To this end, we adopt the following notions of connectivity and connected components of point sets.

Definition 2.1. *Let $\mathcal{P} := \{\mathbf{p}_i\}_{i=1}^n$ and $\mathcal{Q} := \{\mathbf{q}_j\}_{j=1}^m$ be two point sets. Given a positive threshold τ, $\mathcal{P}$ and $\mathcal{Q}$ are* τ-connected *if exist two points $\mathbf{p}_i \in \mathcal{P}$ and $\mathbf{q}_j \in \mathcal{Q}$ such that $\|\mathbf{p}_i - \mathbf{q}_j\|_2 \leq \tau$.*

In particular, a point set $\mathcal{P}$ is said *τ-connected* if each non-empty subset Ω of $\mathcal{P}$ and its complementary set Ω^C in $\mathcal{P}$ are themselves *τ-connected*. Then, a *connected component* of a point set $\mathcal{P}$ is a τ-connected set of points in $\mathcal{P}$. Finally, given two τ-connected components C_1 and C_2 we define their distance $d(C_1, C_2)$ as

$$d(C_1, C_2) = \min_{\substack{\mathbf{p}_1 \in C_1 \\ \mathbf{p}_2 \in C_2}} \|\mathbf{p}_1 - \mathbf{p}_2\|_2. \tag{1}$$

2.2. Graph definition

We now generalize the Reeb graph definition to scalar functions defined on point sets. Given the scalar function $f : \mathcal{P} \to \mathbb{R}$, we denote its minimum and maximum with

$$v_m := \min_{i=1,\dots,n} \{f(\mathbf{p}_i), \mathbf{p}_i \in \mathcal{P}\}, \qquad v_M := \max_{i=1,\dots,n} \{f(\mathbf{p}_i), \mathbf{p}_i \in \mathcal{P}\},$$

and $\overline{\mathrm{Im}(f)} = [v_m, v_M]$ is the interval of $\mathbb{R}$ that contains the discrete image $\mathrm{Im}(f) := \{f(\mathbf{p}_i), \mathbf{p}_i \in \mathcal{P}\}$ of f. For any interval $[a, b]$, $a < b$, contained in $\overline{\mathrm{Im}(f)}$, the *discrete strip* related to $[a, b]$ (Fig. 4(a)) is defined as the set $\mathfrak{S}_{[a,b]} = \{\mathbf{p}_i \in \mathcal{P} : a \leq f(\mathbf{p}_i) \leq b\}$. Then, we replace the role of contours in the definition of the Reeb graph [54] with the concept of strips.

Definition 2.2. *Let $f : \mathcal{P} \to \mathbb{R}$ be a real-valued function defined on a point cloud $\mathcal{P}$ and $\mathcal{J} = \{\mathcal{J}_1, \dots, \mathcal{J}_m\}$ be a partition of $\overline{\mathrm{Im}(f)}$ by non-empty intervals, i.e. $\overline{\mathrm{Im}(f)} = \bigcup_{k=1}^m \mathcal{J}_k$, $\mathcal{J}_i \bigcap \mathcal{J}_j = \emptyset$, $i \neq j$. Then, the* Point Cloud Graph *of $\mathcal{P}$ with respect to f and $\mathcal{J}$ is the quotient space of $\mathcal{P} \times \mathbb{R}$ defined from the equivalence relation "$\sim$": $(\mathbf{p}, f(\mathbf{p})) \sim (\mathbf{q}, f(\mathbf{q}))$ if and only if $\exists \mathcal{J}_k \in \mathcal{J}$ such that:*

1. *$f(\mathbf{p}), f(\mathbf{q}) \in \mathcal{J}_k$;*
2. *$\mathbf{p}, \mathbf{q} \in \mathcal{P}$ belong to the same connected component of $\mathfrak{S}_{\mathcal{J}_k}$.*

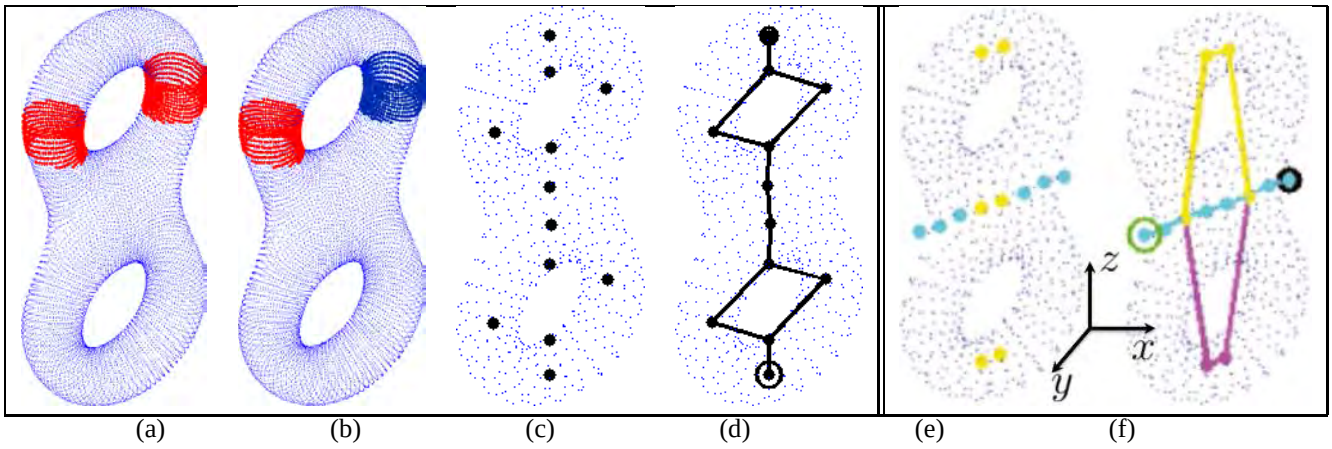


Figure 4: (a-d) Main steps of the proposed approach; as f, we consider a harmonic function with one maximum and one minimum. (a) A strip (red) and (b) its connected components are identified with different colors. (c) The nodes of the *Point Cloud Graph* are computed as centroids of each connected component and (d) linked to form the arcs of the PCG. (e,f) PCG with respect to the height function oriented according to the y-axis. The same axis frame is used in the rest of the paper. In (e), the green nodes represent strips with more than one component and (f) the corresponding arcs are identified with three colors.

The connected components of a strip (Fig. 4(b)) correspond to its τ-connected sets defined in Section 2.1. In particular, we notice that Definition 2.2 introduces the PCG through an equivalence relation that requires the set $\mathcal{J}$ to be a partition. Indeed, the elements of $\mathcal{J}$ cannot intersect and the PCG cannot be implemented through the clustering strategy introduced in [59].

2.3. Graph extraction

To describe how our algorithm extracts the Point Cloud Graph as a couple $\mathcal{G} = (V, E)$, where V and E are respectively the set of the graph nodes and edges, we distinguish four fundamental steps:

1. choice of the scalar function f;
2. extraction of the strips of f (Fig. 4(a));
3. identification of the connected components of each strip (Fig. 4(b)) and creation of the set V of nodes (Fig. 4(c));
4. generation of the set E of arcs (Fig. 4(d)).

Choice of the scalar function f. The graph extraction scheme can be applied to any map f defined on $\mathcal{P}$, thus providing a set of characterizations and different descriptions of the shape underlying $\mathcal{P}$. The properties of the corresponding skeleton will reflect those of f, thus yielding to a multi-view shape description. Coding the PCG as an attributed graph, the choice of the function f influences the geometric and topological information stored in its nodes and arcs. Scalar functions may be either induced by the application context or intrinsically defined by the manifold $\mathcal{M}$ underlying $\mathcal{P}$. In the following, we briefly review the computation of the geodesic, harmonic, and Laplacian functions.

Recent works [44, 55] on the computation of geodesics on a point set $\mathcal{P}$ have enriched the class of scalar functions on $\mathcal{P}$ with geodesics-based maps, previously defined on triangle meshes [38] and used for shape comparison [27, 45]. For instance, in [55] piecewise linear approximations of geodesic paths on point-sampled surfaces are computed by minimizing an energy function, which takes into account both the geodesic path length and its closeness to the underlying surface. An alternative is to trace the shortest path among the nodes of an extended sphere-of-influence graph. In this case, the Point Cloud Graph associated to the averaged geodesic distance from a set of source points is useful for the computation of bending invariant shape signatures [55, 65].

To define harmonic scalar functions on a point set $\mathcal{P}$, the Laplace-Beltrami operator is discretized by the Laplacian matrix $\mathbf{L} := (L_{ij})_{i,j=1}^{n}$ as [10, 11, 23]

$$L_{ij} := \begin{cases} -1 & i = j, \\ a_{ij}/\alpha_i & \mathbf{p}_j \in \mathcal{N}_{\mathbf{p}_i}, \\ 0 & \text{else}, \end{cases} \quad \begin{cases} a_{ij} := \exp\left(-\frac{\|\mathbf{p}_i - \mathbf{p}_j\|_2^2}{h^2}\right), \\ \alpha_i := \sum_{j \in \mathcal{N}_{\mathbf{p}_i}} a_{ij}. \end{cases}$$

We briefly remind that the vector $\mathbf{h}$, $\mathbf{h} \neq \mathbf{0}$, is an *eigenvector* of $\mathbf{L}$ related to the *eigenvalue* λ if and only if $\mathbf{Lh} = \lambda\mathbf{h}$. Once $\mathbf{L}$ has been built, the computation of the harmonic scalar function resembles the case of triangle meshes [31, 34, 46]. Choosing a set of boundary conditions $\mathcal{B} := \{f(\mathbf{p}_i) = a_i\}_{i \in \mathcal{I}}$, $\mathcal{I} \subseteq \{1, \ldots, n\}$, we solve the linear system $\mathbf{L}^\star\mathbf{f}^\star = \mathbf{b}$, where $\mathbf{f}^\star := (f(\mathbf{p}_i))_{i \in \mathcal{I}^C}$ is the vector of unknowns, $\mathcal{I}^C$ is the complementary set of $\mathcal{I}$, $\mathbf{b}$ is a constant vector, and $\mathbf{L}^\star$ is achieved by removing the i^{th}-row and i^{th}-column of $\mathbf{L}$, $i \in \mathcal{I}$.

The Laplace-Beltrami eigenfunctions [12], or the heat kernel [42], provide a family of maps whose Reeb graphs code the features of $\mathcal{P}$ in a multi-scale manner; i.e., from global to local levels of detail. Even though a general choice of f does not guarantee that the corresponding Reeb graph is inside $\mathcal{P}$, specific choices of the input maps such as the Laplace-Beltrami eigenfunctions provide representations that are centered and well aligned with generalized cylinders of $\mathcal{P}$ (if any). Furthermore, non-cylindrical joints are represented as graph edges without self-intersections. For shape analysis, we mainly focus on functions that are intrinsically defined by the point cloud, such as the Laplacian eigenfunctions.

Extraction of the strips of f. According to Definition 2.2, the strips are extracted with respect to a partition $\{\mathcal{J}_k\}_{k=1}^{m}$ of the in-

Table 1: Computational cost of the main steps of the proposed approach, where s is the number of source points used for the computation of the geodesic distance.

Task	**Comput. cost**
Load	$O(n)$
Function	$O(n)$-$O(sn \log n)$
k-nearest neigh. graph	$O(n \log n)$
Conn. components	$O(kn)$
Arc constructions	$O(n)$

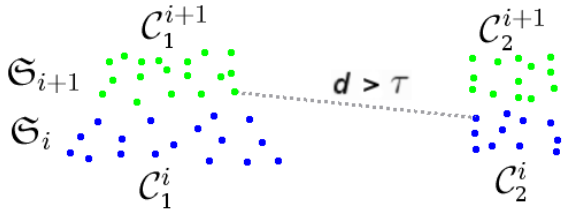


Figure 5: The node $\mathbf{n}_1^i$ is linked to node $\mathbf{n}_1^{i+1}$. Similarly, there is an arc between nodes $\mathbf{n}_2^i$ and $\mathbf{n}_2^{i+1}$. Since $d(C_2^i, C_1^{i+1}) > \tau$, we have not a linking edge between $\mathbf{n}_2^i$ and $\mathbf{n}_1^{i+1}$.

terval $\overline{\mathrm{Im}(f)}$ (Fig. 4(a)). The easiest way to partition $\overline{\mathrm{Im}(f)}$ into n_s sub-intervals is to select $n_s + 1$ values $v_0 := v_m, v_1, \ldots, v_{n_s} := v_M$, $v_i \leq v_{i+1}$, and define each strip as $\mathfrak{S}_{\mathcal{J}_i} := f^{-1}(\mathcal{J}_i)$, $\mathcal{J}_i = [v_i, v_{i+1})$, $i = 0, \ldots, n_s - 1$. This slicing strategy is quite common in the extraction of discrete approximations of the Reeb graph because uniform interval subdivisions of $\overline{\mathrm{Im}(f)}$ allow us to approximate the size and the relevance of a feature in terms of the length of the arcs of the graph; i.e., the longer the arc the more important the feature coded by the graph. Furthermore, it is possible to define an iterative sequence in the interval subdivision that makes the graph multi-resolutive. For more details on the slicing strategy, we refer the reader to [16, 38].

Connected components of strips and creation of the set V of nodes. Once we have identified the strip $\mathfrak{S}_{\mathcal{J}_i}$, we detect its τ-connected components with respect to Definition 2.1. To extract a τ-connected component C_j^i of the strip $\mathfrak{S}_{\mathcal{J}_i}$, we select a point $\mathbf{p}_{j_1} \in \mathfrak{S}_{\mathcal{J}_i}$ that has not been marked as belonging to any τ-connected component. Then, all the points of $\mathfrak{S}_{\mathcal{J}_i} \cap \mathcal{I}_{\mathbf{p}_{j_1}}^{k,\tau}$ are marked as belonging to C_j^i and we recursively repeat this expansion on all the points of C_j^i. If at the end of this process there are points of $\mathfrak{S}_{\mathcal{J}_i}$ that are still unmarked, then we select one of them, identify a new connected component, and continue until all the points of $\mathfrak{S}_{\mathcal{J}_i}$ have been labeled as visited. The whole process is applied to all the strips and ends when each point of $\mathcal{P}$ has been assigned to some τ-connected component. Finally, we associate a node $\mathbf{n}_j^i$ to every connected component C_j^i. As spatial representative of $\mathbf{n}_j^i$, we choose the centroid of C_j^i, (Fig. 4(c)).

Creation of the set E of arcs. According to Definition 2.2, the node $\mathbf{n}_j^i$, which codes the connected component C_j^i, must be linked to the nodes that correspond to the connected components of the strips $\mathfrak{S}_{\mathcal{J}_{i-1}}$ and $\mathfrak{S}_{\mathcal{J}_{i+1}}$. Note that these strips $\{\mathfrak{S}_{\mathcal{J}_i}\}_i$ are visited sequentially with respect to the increasing ordering of the corresponding intervals $\mathcal{J}_i$. According to Equation (1), two connected components C_s^i and C_r^{i+1} are linked if $d(C_s^i, C_r^{i+1}) \leq \tau$; in this case, we add the arc $(\mathbf{n}_s^i, \mathbf{n}_r^{i+1})$ to $\mathcal{G}$ (Fig. 4). The extraction of the set E of arcs ends when all the possible links among the connected components of two consecutive strips have been processed (Fig. 5). The construction of the arcs allows us to easily recognize branching parts. Figs. 4(d,e) show the Point Cloud Graph of the same point cloud with respect to two different functions. Differently from [57] and in order to extract the graph of scenes, which are typically composed of several components (Fig. 6), we do not automatically connect adjacent strips that have only one connected component. Finally, Figs. 7, 8, and 9 show the Point Cloud Graph of the same shape with different functions.

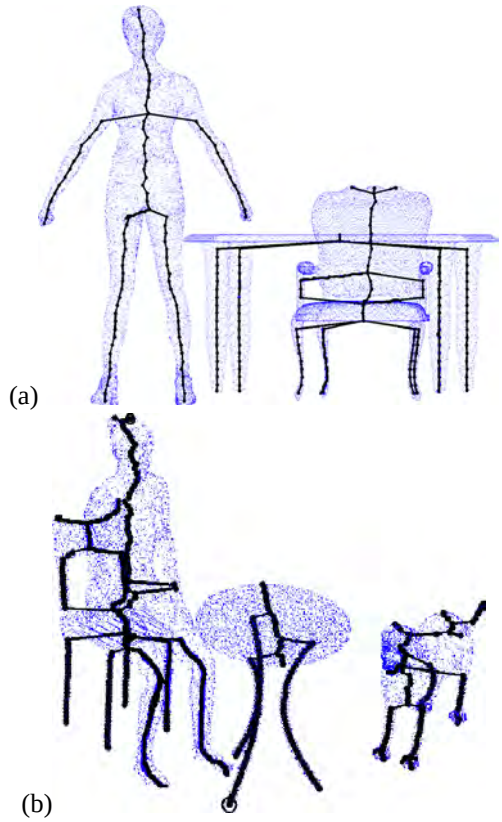


Figure 6: Point Cloud Graph of a scene with (a) three and (b) four components. As scalar function, we have chosen the height function with respect to the z-axis.

Computational cost. Our algorithm is computationally efficient and handles point clouds with hundred thousands of points and more than one object. Analyzing the single steps of the algorithm (Table 1), the data loading requires $O(n)$ operations, where n is the number of points of $\mathcal{P}$. The computation of the input scalar function f varies from $O(n)$ to $O(n \log n)$. For instance, the evaluation of the height function and the distance from the center of mass is linear in the number of points; the computation of the geodesic distance from s source points is $O(sn \log n)$ using the Dijkstra's algorithm; and the solution of the Laplace-Beltrami eigenvalue problem is super-linear in n

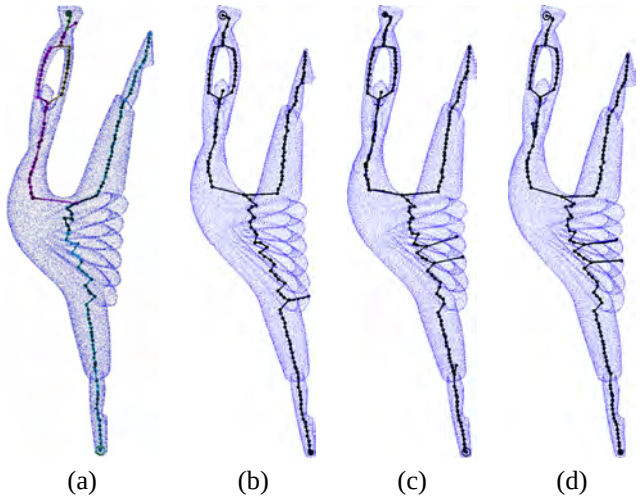


Figure 7: Point Cloud Graph of the same point cloud with respect to different scalar functions: (a) height function with respect to the z-axis; (b) $f(\mathbf{p}) := \log(\|\mathbf{p}\|_2 + 1)$; (c) $f(x,y,z) := x^2 - y^2$, $\mathbf{p} := (x,y,z)$; (d) $f(\mathbf{p}) := \|\mathbf{p}\|_2$.

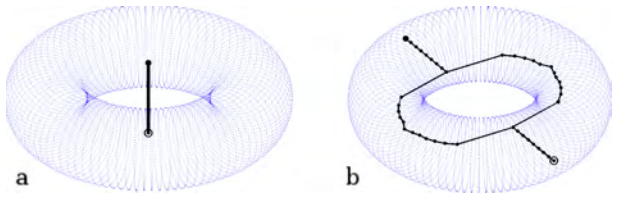


Figure 8: In (a), the graph is represented as two overlapping arcs. (b) Changing the scalar function, the arcs of the Point Cloud Graph are explicitly coded. Here, f is the height function with respect to (a) the z-axis and (b) y-axis.

and the number of eigenfunctions computed. The computation of the k-nearest neighbor graph $\mathcal{T}$ takes $O(n \log n)$ operations [7] and the creation of the strips is linear in n. For each strip, the computation of the connected components takes $O(kn)$ operations. The extraction of the arcs of $\mathcal{G}$ requires to traverse two consecutive strips and visit their points at most twice; indeed, this step runs in $O(n)$ operations. Finally, the overall computational cost is $O(n)+O(n \log n)+O(kn) = O(\max\{kn, n \log n\})$, which does not considerably differ from the $O(n \log n)$ time required to compute the Reeb graph over triangle meshes [24, 16] and an efficient implementation of the clustering strategy in [59].

3. Discussion and results

Once our experimental settings have been introduced (Section 3.1), we discuss the main properties and degrees of freedom in the computation of the Point Cloud Graph; namely, the choice of the parameters (Section 3.2), its robustness (Section 3.3), the generalization to volume data and non-orientable surfaces (Section 3.4), and the application to shape comparison (Section 3.5).

3.1. Experimental settings

To analyze the behavior of the Point Cloud graph with respect to the size of the neighbor of each point through the connectivity parameters introduced in Section 2.1, we have applied

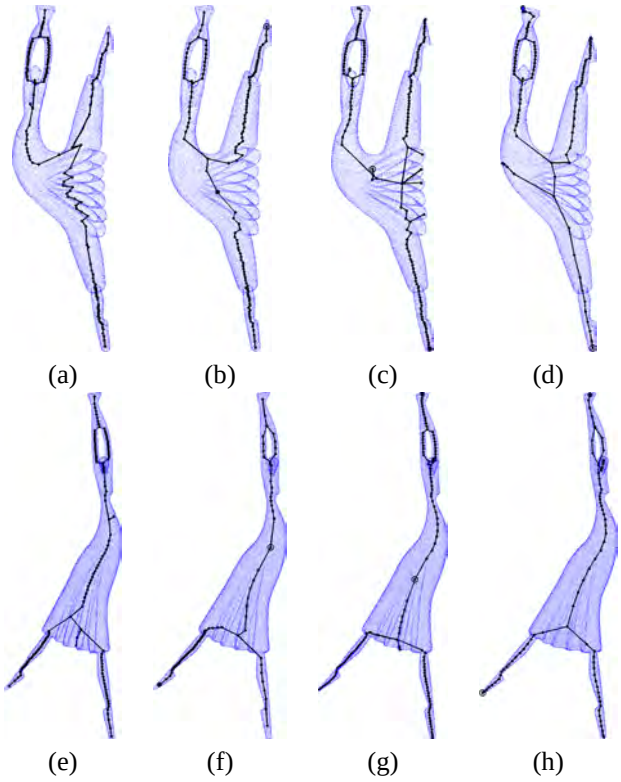
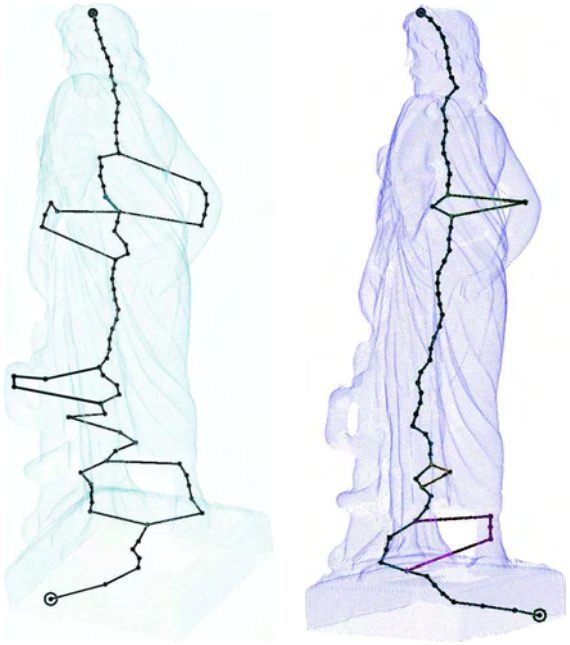


Figure 9: Point Cloud Graphs of the scans of two statues with respect to (a,b,e,f) Laplacian-Beltrami eigenfunctions and (c,d,g,h) harmonic functions.

our method to point sets with different sampling density, noise level, distribution of shape features, and missed parts. Most of point clouds corresponds to 3D scans of real models such as small statues and human bodies in different poses. We also consider point samples of volumetric data, i.e., where the underlying manifold is a 3−manifold with boundary embedded in $\mathbb{R}^3$. For our tests, we have considered 50 points clouds from the AIM@SHAPE repository [1], 30 body scans from the CAESAR Data Samples[2], and the 400 models of the SHREC 2007 benchmark[3]. The non-orientable models have been obtained from parametric samples of the Moebius surface and Klein's bottle and the two scenes have been composed from objects of the data sets. To evaluate how the Point Cloud Graph depends on the data quality, the point clouds have been perturbed with geometric noise, by modifying the point coordinates. To show the scalability and the efficiency of the method (Table 2), the size of the data set ranges from a few thousand points to over one million. All the experiments have been performed over a mini-laptop equipped with Linux operative system, *Mobile Intel Celeron 900MHz*, and *2048MByte Ram*.

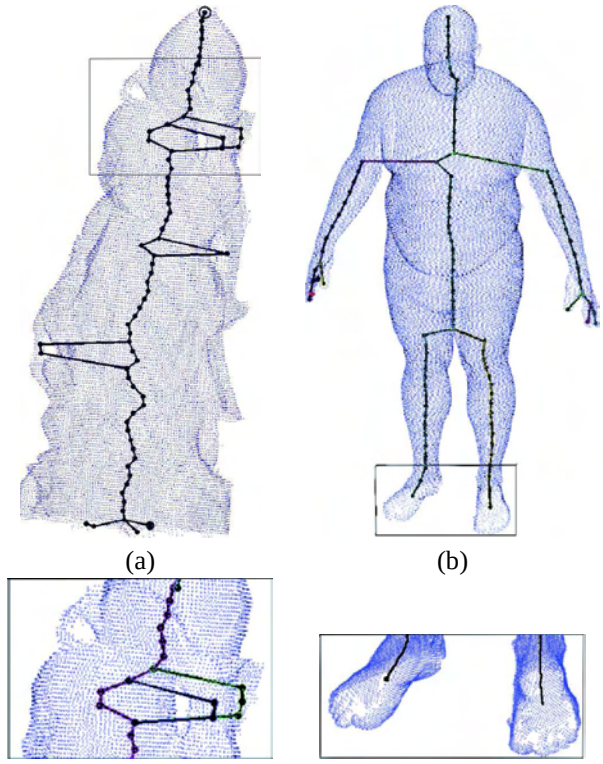
Beside the choice of the input function f, the arcs and nodes of the Point Cloud Graph are determined by the point cloud connectivity stored in the k-nearest neighbor graph. For point sets that represent 3D surfaces, we have experimentally verified that when the sampling of $\mathcal{P}$ is sufficiently dense the number of

[1] `http://shapes.aim-at-shape.net`
[2] `http://www.hec.afrl.af.mil/HECP/Card1b.shtml#caesarsamples`
[3] `http://watertight.ge.imati.cnr.it/`



(a) (b)

Figure 10: (a) The adaptive selection of the threshold τ_i allows us to better identify through holes in the point clouds as loops of the Point Cloud Graph. (b) The choice of a constant τ provides a skeleton that codes a lower number of local details. Here, f is the first non-trivial Laplacian eigenfunction.



(a) (b) (c) (d)

Figure 11: Point Cloud Graph of a point set with (a,b) a low and irregular sampling density with missed parts (shoulder and feet in (b)), which are occluded during the acquisition process. (c,d) Zoom-in. For both examples, the extracted skeletal representations capture the main features of the underlying surface. In both cases, we have selected the first non-trivial Laplacian eigenfunction.

loops of the PCG is equal to the genus of the surface underlying $\mathcal{P}$. Since the sampling density varies from model to model, it is crucial to automatically select a threshold that identifies the connected components of both the shapes of a scene and the strips of each building shape. To this end, we assume that the 3D shapes are *coherently sampled*, i.e., the local sampling density $\sigma_{\mathcal{P}}$ of $\mathcal{P}$ [53] is equal to or lower than the distance used to identify the connected components of the strips, the size of the through holes and the connected components.

3.2. Choice of the parameters

Since the choice of the parameters τ and k is crucial to obtain an effective representation of the shape characteristics, we analyze how their choice influences the Point Cloud Graph and how to automatically determine them. To deal with non-uniform point samples or partially missing data, we introduce an adaptive definition of τ, which is iteratively tuned according to the local density of the point cloud $\mathcal{P}$. To guarantee the coherence of the point cloud, we fix the value of τ_1 as a multiple of $\sigma_{\mathcal{P}}$ and initialize the connected components of $\mathfrak{S}_{\mathcal{J}_i}$. Then, during the expansion process and in a neighbor of a point $\mathbf{p}_{j_i}$ of the $i-$th strip $\mathfrak{S}_{\mathcal{J}_i}$ we iteratively refine the constant τ_{j+1}, $j \geq 1$, as follows:

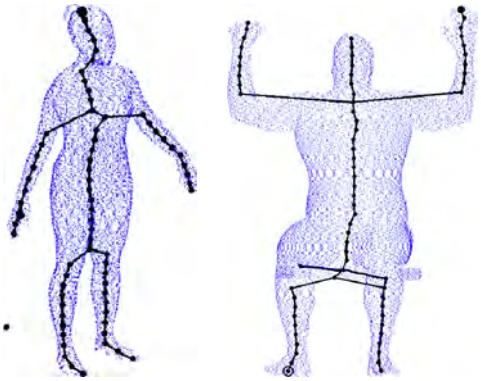
$$\tau_{j+1} := \begin{cases} \frac{\alpha_{j+1}+j\tau_j}{j+1}, & |\mathcal{I}^{k,\tau_j}_{\mathbf{p}_{j_i}}| = k, \\ \frac{2\alpha_{j+1}+j\tau_j}{j+1}, & |\mathcal{I}^{k,\tau_j}_{\mathbf{p}_{j_i}}| < k, \end{cases}$$

where $|\mathcal{I}|$ is the number of points of the set $\mathcal{I}$ and $\alpha_{j+1} = \max_{\mathbf{p}\in\mathcal{I}^{k,\tau_j}_{\mathbf{p}_{j_i}}} \|\mathbf{p} - \mathbf{p}_{j_i}\|_2$. In those shape regions where an irregular variation of the sampling density occurs, the choice of τ_j might provide problems for the identification of topological handles whose size is approximately τ_j (Fig. 10).

A crucial part for the extraction of $\mathcal{G}$ is related to the computation of the connected components of each strip and the generation of the arcs of the graph, where multiple components occur. These two steps of the algorithm are guided by the expansion radius τ_j of the neighbor of each point of $\mathfrak{S}_{\mathcal{J}_i}$. The tolerance τ that identifies the connected component of the strip $\mathfrak{S}_{\mathcal{J}_i}$ will be also defined as a multiple of τ_j.

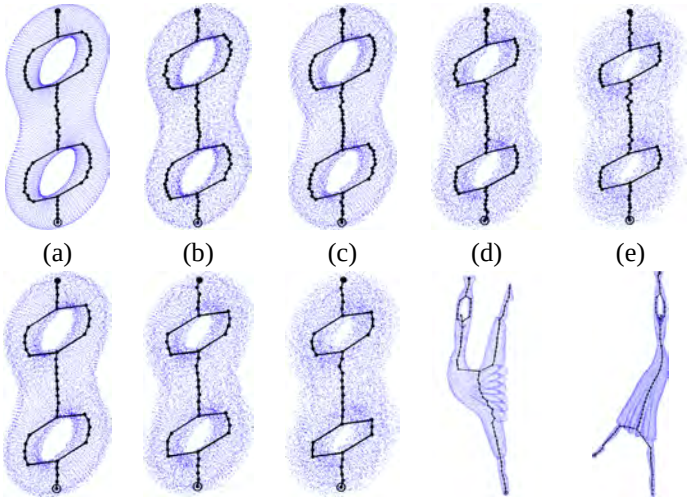
The adaptive choice of the parameters is also crucial when we deal with scenes that include components with a different sampling density. For instance, in Fig. 6(b) the table model is denser than the other ones and the dog surface is not uniformly sampled. In case of a non-uniform distribution of points, the Point Cloud Graph could present more/less connections than expected or many connected components (Figs. 11 and 12). Fig. 12(a) shows how our representation automatically distinguishes spurious data, such as regions of the platform on which the human is standing during the body acquisition, from body parts partially occluded. Additionally, the rear part of the head is correctly connected to the main body.

Table 2 summarizes the characteristics of the PCG in terms of number of elements, loops, and connected components with respect to different choices of k and τ. For the computation



(a) (b)

Figure 12: (a,b) Point Cloud Graph extracted from partially occluded body scans and in different postures. In these examples, we have selected the first non-trivial Laplacian eigenfunction.



(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

Figure 13: If the original point cloud (a) is perturbed with a Gaussian noise (b-e), then the number of nodes and arcs of the PCGs does not change. The PCGs in (f-h) correspond to the ones in (c-e) selecting $n_s = 30$ instead of $n_s = 50$. For (i,j), the reference Point Cloud Graphs are shown in Figs. 9(b,g), respectively. In these examples, the chosen scalar function is the height function in the z-axis direction.

of these graphs, the number of strips has been fixed to 30 for the bi-torus model, 100 for the hand model, and 50 for the 3D scene. Our tests have shown that if the parameters k and τ are arbitrarily chosen the number of loops of the graph may vary (e.g., Fig. 13). Moreover, the value τ_j affects the connectivity of the graph: this is not surprising because when τ_j increases the $\tau-$connected sets become larger and the corresponding nodes are connected.

In our data set, we have experimentally verified that a good compromise between computational complexity and efficacy of the description is to choose k smaller than 12; to initialize τ_1 from 5 to 10 times $\sigma_{\mathcal{P}}$; and to set τ as $2\tau_j$, where τ_j the is the adaptive threshold previously discussed. If not differently specified in the text, then we set $k = 10$, $\tau_1 = 5\sigma_{\mathcal{P}}$, and $\tau = 2\tau_j$.

3.3. Robustness

We now discuss the robustness of the graph to noise, local deformations, and missed data by experimentally verifying how these factors affect the corresponding structure. To this end, we simulate a geometric perturbation of the point cloud modifying the coordinates of the points through random Gaussian perturbations. The variance of noise perturbation of the models in Figs. 13(b-e) is 2%, 5%, 10%, and 15% of the maximum diameter of $\mathcal{P}$, respectively. All these graphs have been obtained using $n_s = 50$ strips. The overall structure of the graph is the same even if the number n_s of strips varies: Figs. 13(f-h) show the PCGs extracted setting $n_s = 30$ and the noise variance is equal to 5%, 10%, and 15%. Moreover, we notice that when the bitorus model is perturbed with a noise variation higher than 5% the corresponding triangle mesh is no more manifold and small self-intersections appear; indeed, we are not able to extract the Reeb graph from the mesh while this is possible with our PCG. Additional examples are depicted in Figs. 13(i,j): these models correspond to 2% noise perturbations of the ones in Figs. 9(b,g), respectively. In all cases, the extraction of the skeletal structure remains stable; i.e., the number and position of nodes and arcs do not significantly change. This property is mainly due to the fact that we code the evolution of the strips instead of the contours, which are more sensitive to local perturbations of both $\mathcal{P}$ and f.

As shown in Figs. 12 and 14, the Point Cloud graph handles either irregularly or partially sampled data, due to occlusions during the acquisition process. For instance, Fig. 12(a) shows the behavior of the graph with respect to shape outliers. In fact, this body scan presents a few points (low-left) that can be considered as noise. With our standard choice of the parameters k and τ_j, the human model and few isolated points (left part) are abstracted as distinct graphs. The smallest component disappears only when the chosen parameter τ allows us to glue this small component to the body. Moreover, Fig. 14 depicts that, differently from [61], the flexibility in the choice

Table 2: Point Cloud Graph complexity. The variation of τ and k influences the connectivity of the graph in terms of number of connected components CC, vertices $|V|$, edges $|E|$, and loops. The Bi-torus, the Hand, and the Scene point clouds are respectively shown in Figs. 13(a), 1(d), and 6(a), respectively.

Model	n	τ	k	$\|V\|$	$\|E\|$	loops	CC
Bi-torus	12K	$2\tau_j$	10	42	43	2	1
Bi-torus	12K	$2\tau_j$	7	70	97	18	1
Bi-torus	12K	$10\tau_j$	7	42	43	2	1
Bi-torus	12K	$2\tau_j$	100	31	31	1	1
Bi-torus	12K	$2\tau_j$	102	30	29	0	1
Hand	37K	$2\tau_j$	10	155	154	0	1
Hand	37K	$2\tau_j$	7	161	160	0	1
Hand	37K	$2\tau_j$	4	302	407	106	1
Scene	330K	$2\tau_j$	10	250	250	3	3
Scene	330K	$4\tau_j$	8	253	252	3	4
Scene	330K	$10\tau_j$	12	232	233	3	2

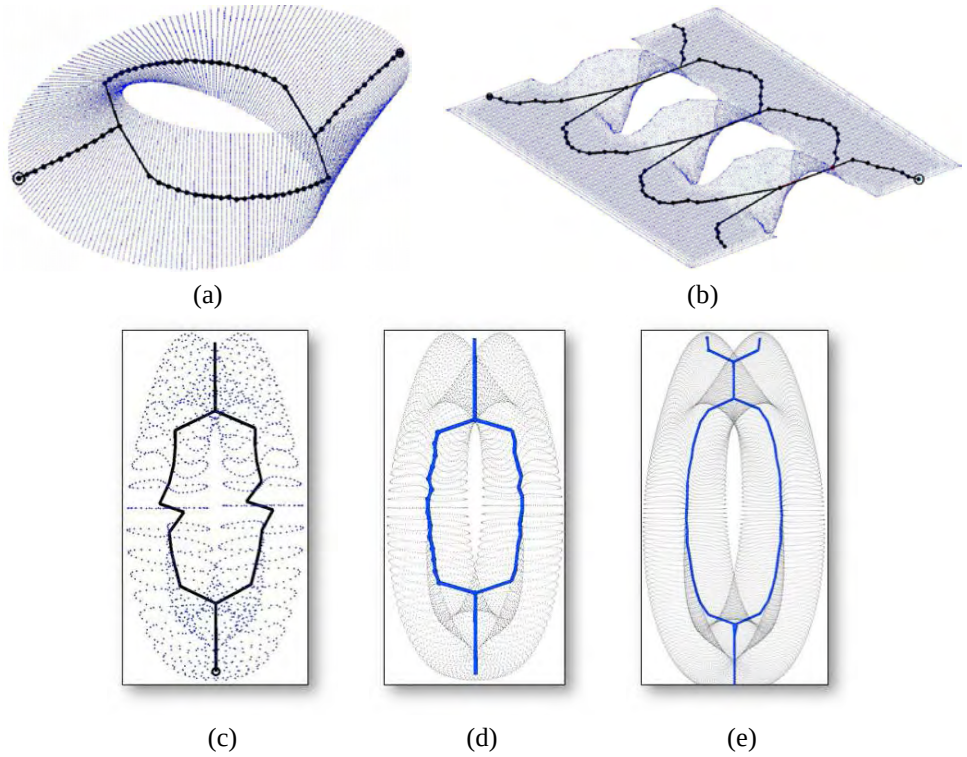


Figure 15: Point Cloud Graph of point sets sampled from (a) a Mobius surface, (b) a plane with three twists, and (c-e) a Klein bottle at different resolutions. In (a,b), the input map is the height function with respect to the y-axis. In (c-e), we have selected the first non-trivial Laplacian eigenfunction.

of the parameter τ automatically provides an estimation of the entity of the missed part. In fact, these examples represent a sequence of different samples of the same statue, whose resolution increases from (a) to (d). To compute the PCG, we have considered the distance from the center of mass and set the parameters k, τ_1, and τ with the default values discussed in Section 3.2. The graph in Fig. 12(a) highlights that the bust and the bottom of the statue is completely missing: this implies that a loop of the graph is broken and an additional loop appears in the bottom. The two intermediate PCGs in Fig. 12(b,c) are qualitatively and qualitatively equivalent while the PCG of a finer sample (Fig. 12(d)) of the statue correctly recognizes the two hands and has an additional loop.

3.4. Non-orientable surfaces and volume data

In the following, we show that our approach is able to describe a class of data larger than the Reeb graph, including point clouds originated by surfaces, volume data, or $m-$dimensional manifolds embedded in $\mathbb{R}^d$ with multiple components. For instance, since the original triangle mesh representing the model in Fig. 1(a) contains 11 components, it is not possible to compute the Reeb graph directly on the triangle mesh while our algorithm effectively runs also on this example. The same remark holds for the representation of sets of objects as in case of scenes (Fig. 6).

Our graph representation also handles point sets representing non-orientable surfaces (Fig. 15) and volume data (Fig. 16), without building a manifold representation of the model. In all these examples, the values of k and τ are the default ones except for the models in Fig 15(c-e) ($\tau = 10\tau_j$) and Figs. 15(c-e) ($k = 25$, $k = 15$, $k = 30$). Volume data are quite common in medical and FEM applications. Until now, the extraction of the Reeb graph description for these data required the generation of a tetrahedral representation of $\mathcal{P}$ [49] and the identification of hole cuts [63] to have computational efficiency. In its original definition, the Reeb graph is not able to fully represent cavities, similarly the PCG has the same limitation. For instance, in Figure 16(c) the skull cavity is simply represented with a sequence of nodes and arcs. Table 3 reports statistics on the Point Cloud Graph computation for 3D shapes with different sampling den-

Table 3: Statistics on the Point Cloud Graph extraction for some of our test models, the last four rows refer to point clouds of volumetric data: n number of points of $\mathcal{P}$, n_S number of strips used to extract the description, $|V|$ cardinality of the set of nodes, $|E|$ number of arcs of $\mathcal{G}$. Time is expressed in seconds.

Model	n	n_S	$\|V\|$	$\|E\|$	Time
Monk - 11(a)	30K	120	137	141	1,2
Camel - 1(c)	35K	140	241	242	1,7
Hand - 1(d)	53K	100	221	220	4,1
Ippocrates - 10(a)	102K	200	222	226	11,9
Human - 11(b)	190K	120	249	248	41,8
Scene - 6(a)	330K	200	1003	1003	370
Raptor - 1(a)	1M	400	1164	1142	435
Hand - 16(a)	29K	30	49	48	5
Vertebrae - 16(b)	17K	20	20	19	3
Skull - 16(c)	38K	50	65	65	5
Ear - 16(d)	153K	20	60	92	221

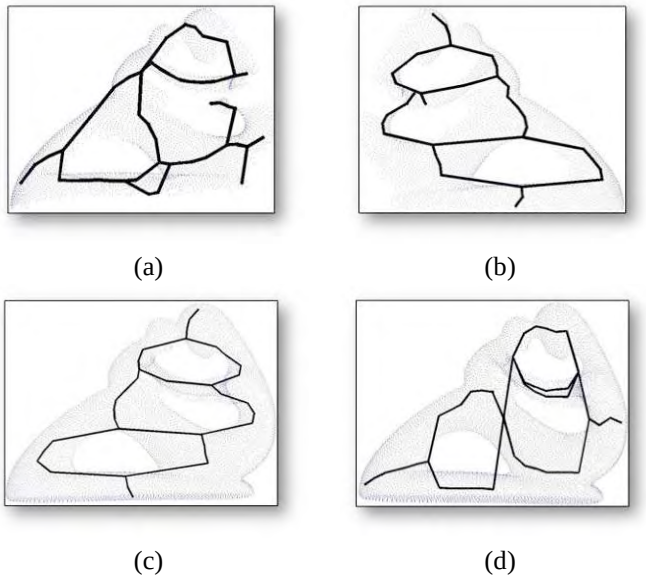


Figure 14: (a-d) Point Cloud Graph with respect to irregular sampling density and missed part. Slightly increasing the shape resolution improves the quality of the Point Cloud Graph, in terms of a lower number of terminal arcs and a better alignment with the underlying shape. Here, the height function is in the direction of the z-axis.

sities and resolution.

3.5. Shape comparison

A current limitation of the use of the Reeb graph for matching purposes is that the existing algorithms [17, 64, 18] and applications to relevance feedback [36] requires the models to be watertight and without topological artifacts (e.g., dangling edges, multiple components). Indeed, the use of the Reeb graph is limited to a narrow number of data sets. In this context, we outline how the graph matching techniques used for Reeb graph comparison is easily adapted to the Point Cloud Graph, thus broadening the use of this description to quite a number of data sets. In our experiments, we compare the Point Cloud Graphs using the graph distance [48], which is an extension to set of graphs of Laplace-based metric [66], and match both single or sets of graphs. More formally, we consider the set of elementary symmetric polynomials:

$$S_j(v_1, \ldots, v_n) = \sum_{i_1 < \cdots < i_j} v_{i_1} v_{i_2} \cdots v_{i_j}, \qquad j = 1, \ldots, n.$$

Then, the *feature vector* of the Point Cloud Graph G is defined as the matrix

$$\mathbf{B} = (f_{1,1}, \ldots, f_{1,n}, \ldots, f_{n,1}, \ldots, f_{n,n})^T \ ,$$

where $f_{i,j} = sign(S_j(\Phi_{1,i}, \ldots, \Phi_{n,i})) \ln(1 + |S_j(\Phi_{1,i}, \ldots, \Phi_{n,i})|)$ and $\Phi_{i,j}$ denotes the entry (i, j) of the matrix Φ that decomposes the Laplacian matrix $\mathbf{L}$ with constant weights of the graph as $\mathbf{L} = \Phi\Phi^T$. The distance between two Point Cloud Graphs G_1, G_2 whose feature vectors $\mathbf{B}_1, \mathbf{B}_2$ are known, is defined by:

$$D(G_1, G_2) := \left| \|\mathbf{B}_1\|_2 - \|\mathbf{B}_2\|_2 \right| \ .$$

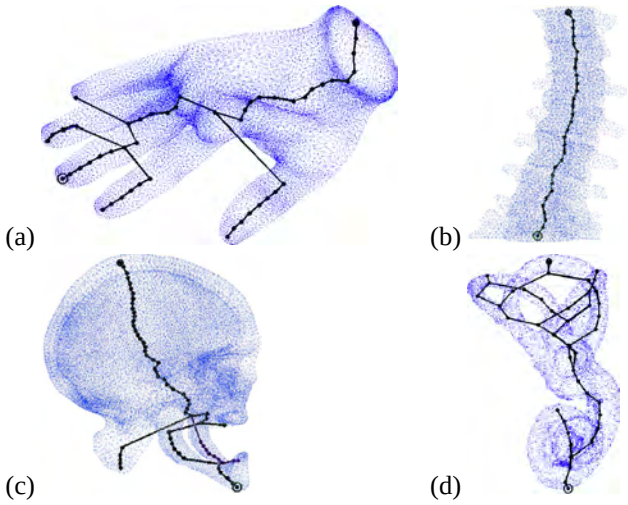


Figure 16: (a-d) Point Cloud Graph of volume data. Here, the map is the height function with respect to the main direction provided by the Principal Component Analysis on the input data set. According to the definition of Reeb graph, (a) highlights a situation for which the choice of f generates a PCG which is not medial to the shape.

D is a pseudo-metric, which satisfies positivity, symmetry and triangle inequality; identity is not verified (i.e., $D(G_1, G_2) = 0 \not\Rightarrow G_1 \simeq G_2$). More details can be found in [48].

In our tests, we have selected seven classes (human, cup, table, glass, octopus, plier, and bird models) from the SHREC 2007 benchmark on watertight models [35] and tested the retrieval performance of the Point Cloud Graph using the first and second non-trivial Laplace-Beltrami eigenfunctions, either singularly or in combination. Table 4 quantitatively compares the distances between couples of models (three humans, two cups, and two tables) computed either using the PCG (bottom value) or the Reeb graph (top value) as shape signatures. Despite the relative relevance of the numerical scores, the values of the distances are nearly comparable and in both cases well discriminate among objects belonging to different classes. A qualitative comparison of the two descriptors is shown in Fig. 17, where the precision-recall diagrams of the PCG and the Reeb graph (RG) [48] are depicted over the benchmark and the human model class. Again, these diagrams confirm that the performance of the two descriptors is substantially the same; in fact, in our feeling the relevance of the PCG graph is in the extension of the application domain (more objects, even disconnected and polygon soups) rather than one more method that slightly improves graph matching using Reeb graphs.

4. Future work

Differently from the usual Reeb graph description, our Point Cloud Graph is able to deal with point sets and multiple connected components, without requiring any pre-processing step. Therefore, we approach a larger scenario of applications, which spans from medicine to robotics to ambient intelligence. Further investigations are needed to identify which class of functions is the most suitable for a given task and to analyze how

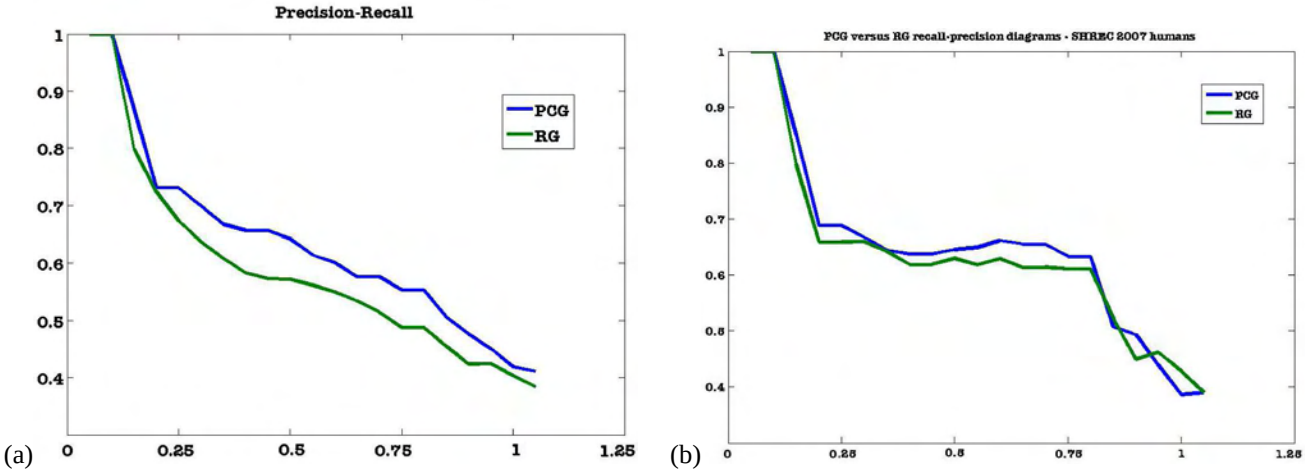


Figure 17: (a) Precision (vertical axis) versus recall (horizontal axis) diagrams over three classes of the data set [35] and (b) focus on the class of the human models.

many geometric attributes must be stored for effectively addressing shape retrieval and recognition issues. In general, our framework effectively codes shape features independently on the dimension of the underlying manifold and the embedding space. Moving from these considerations, the Point Cloud Graph significantly extends the class of shapes to which graph-based descriptors may be applied, for instance triangle soups, data scans, and X-ray crystallography.

Acknowledgements

Special thanks are given to the anonymous reviewers for their valuable comments, which helped us to improve the content of the paper. This paper has been partially supported by the EC-FP7 Coordination Action FOCUS K3D. Models are courtesy of J. Cao, A. Tagliasacchi, the AIM@SHAPE and CAESAR data set.

References

[1] A. Adamson and M. Alexa. Approximating and intersecting surfaces from points. In *Symp. on Geometry Processing*, pages 230–239, 2003.

[2] A. Adamson and M. Alexa. Ray tracing point set surfaces. In *IEEE Shape Modeling International*, pages 272–282, 2003.

[3] O. Aichholzer, D. Alberts, F. Aurenhammer, and B. Gartner. A novel type of skeleton for polygons. *Journal of Universal Computer Science*, 1:752–761, 1995.

[4] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Point set surfaces. In *IEEE Visualization*, pages 21–28, 2001.

[5] N. Amenta, S. Choi, and R. K. Kolluri. The power crust, unions of balls, and the medial axis transform. *Computational Geometry: Theory and Applications*, 19:127–153, 2000.

[6] N. Amenta and Y. Joo Kil. Defining point-set surfaces. In *ACM Siggraph*, pages 264–270, 2004.

[7] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM*, 45(6):891–923, 1998.

[8] M. Attene and G. Patanè. Hierarchical structure recovery of point-sampled surfaces. *Computer Graphics Forum*, Vol. 29, Num. 6, pages 1905-1920, 2010.

[9] O. K.-C. Au, C.-L. Tai, H.-K. Chu, D. Cohen-Or, and T.-Y. Lee. Skeleton extraction by mesh contraction. In *ACM Siggraph*, pages 1–10, 2008.

[10] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[11] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

[12] M. Belkin, J. Sun, and Y. Wang. Constructing Laplace operator from point clouds in $\mathbb{R}^d$. In *Proc. of the Sympos. on Discrete Algorithms*, pages 1031–1040, 2009.

[13] S. Berretti, A. Del Bimbo, and P. Pala. 3D Mesh Decomposition using Reeb Graphs. *Image and Vision Computing*, 27(10):1540–1554, 2009.

[14] S. Biasotti, D. Attali, J.-D. Boissonnat, H. Edelsbrunner, G. Elber, M. Mortara, G. Sanniti di Baja, M. Spagnuolo, and M. Tanase. Skeletal structures. In L. De Floriani and M. Spagnuolo, editors, *Shape Analysis and Structuring*, pages 145–183. Springer, 2007.

[15] S. Biasotti, B. Falcidieno, and M Spagnuolo. Extended Reeb Graphs for surface understanding and description. In G. Borgefors and G. Sanniti di Baja, editors, *Discrete Geometry for Computer Imagery Conference*, volume 1953 of *LNCS*, pages 185–197. Springer, 2000.

[16] S. Biasotti, D. Giorgi, M. Spagnuolo, and B. Falcidieno. Reeb graphs for shape analysis and applications. *Theoretical Computer Science*, 392(1–3):5–22, 2008.

[17] S. Biasotti, D. Giorgi, M. Spagnuolo, and B. Falcidieno. Size functions for comparing 3D models. *Pattern Recognition*, 2008.

[18] S. Biasotti, S. Marini, M. Spagnuolo, and B. Falcidieno. Sub-part correspondence by structural descriptors of 3D shapes. *Computer-Aided Design*, 2006.

[19] H. Blum. A transformation for extracting new descriptors of shape. In *Models for the Perception of Speech and Visual Form*, pages 362–380. MIT Press, 1967.

[20] S. Bouix, K. Siddiqi, A. Tannenbaum, and S. W. Zucker. *Statistics and Analysis of Shapes*. Springer, 2007.

[21] J. Cao, A. Tagliasacchi, M. Olson, Z. Su, and H. Zhang. Point cloud skeletons via laplacian-based contraction. In *IEEE Proc. of Shape Modeling International*, pp. 187-197, 2010.

[22] J.-H. Chuang, N. Ahuja, C.-C. Lin, C.-H. Tsai, and C.-H. Chen. A potential-based generalized cylinder representation. *Computers & Graphics*, 28(6):907 – 918, 2004.

[23] R. R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, 2006.

[24] K. Cole-McLaughlin, H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Loops in Reeb graphs of 2-manifolds. In *Proc. of the Symposium on Computational Geometry*, pages 344–350, 2003.

[25] N. D. Cornea, M. F. Demirci, D. Silver, A. Shokoufandeh, S. J. Dickinson, and P. B. Kantor. 3D object retrieval using many-to-many matching of curve skeletons. In *Proc. of IEEE Shape Modeling and Applications*, pages 368–373, 2005.

[26] N. D. Cornea, D. Silver, D. Yuan, and R. Balasubramanian. Comput-

Table 4: Distances between models: RG descriptor (top value) versus the PCG (bottom).

							
	0.0000 0.0000	0.0033 0.0027	0.0034 0.0027	0.2277 0.2679	0.6496 0.3653	0.0395 0.0269	0.0146 0.0173
	0.0033 0.0027	0.0000 0.0000	0.0009 0.0006	0.2179 0.2417	0.6496 0.4453	0.0395 0.0277	0.0146 0.0153
	0.0034 0.0027	0.0009 0.0006	0.0000 0.0000	0.2773 0.2417	0.6496 0.4453	0.0395 0.0277	0.0146 0.0153
	0.2277 0.2679	0.2179 0.2417	0.2773 0.2417	0.0000 0.0000	0.0107 0.0079	0.1002 0.1124	0.2476 0.1587
	0.6496 0.3653	0.6496 0.4453	0.6496 0.4453	0.0107 0.0079	0.0000 0.0000	0.3736 0.2404	0.4149 0.3479
	0.0395 0.0269	0.0395 0.0277	0.0395 0.0277	0.1002 0.1124	0.3736 0.2404	0.0000 0.0000	0.0028 0.0038
	0.0146 0.0173	0.0146 0.0153	0.0146 0.0153	0.2476 0.1587	0.4149 0.3479	0.0028 0.0038	0.0000 0.0000

ing hierarchical curve- skeletons of 3D objects. *The Visual Computer*, 21(11):945–955, 2005.

[27] T. Darom, M. R. Ruggeri, D. Saupe, and N. Kiryati. Processing of textured surfaces represented as surfel sets: representation, compression and geodesic paths. In *IEEE Int. Conference on Image Processing*, pages 605–608, 2005.

[28] T. Dey and W. Zhao. Approximate medial axis as a Voronoi subcomplex. In *Proc. of the Symp. on Solid Modeling and Applications*, pages 356–366, 2002.

[29] T. K. Dey and S. Goswami. Provable surface reconstruction from noisy samples. *Comput. Geom. Theory Appl.*, 35(1):124–141, 2006.

[30] T. K. Dey and J. Sun. Defining and computing curve-skeletons with medial geodesic function. In *Proc. of Symposium on Geometry Processing*, pages 143–152, 2006.

[31] S. Dong, S. Kircher, and M. Garland. Harmonic functions for quadrilateral remeshing of arbitrary manifolds. *Computer Aided Geometric Design*, 22(5):392–423, 2005.

[32] H. Doraiswamy and V. Natarajan. Efficient algorithms for computing Reeb graphs. *Computational Geometry: Theory and Applications*, (42):606–616, 2009.

[33] S. Fleishman, D. Cohen-Or, M. Alexa, and C. T. Silva. Progressive point set surfaces. *ACM Transactions on Graphics*, 22(4):997–1011, 2003.

[34] M. S. Floater and K. Hormann. Surface parameterization: a tutorial and survey. In *Advances in Multiresolution for Geometric Modelling*, pages 157–186. 2005.

[35] D. Giorgi, S. Biasotti, and L. Paraboschi. Watertight models track. Technical Report IMATI-CNR-GE 09/07, 2007.

[36] D. Giorgi, P. Frosini, M. Spagnuolo, and B. Falcidieno. 3D relevance feedback via multilevel relevance judgements. *The Visual Computer*, 26:1321–1338, 2010.

[37] W. Harvey, Y. Wang, and R. Wenger. A randomized $O(m \log m)$ time algorithm for computing Reeb graphs of arbitrary simplicial complexes. In *ACM Symp. on Computational Geometry*, pages 267-276, 2010.

[38] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. Topology matching for fully automatic similarity estimation of 3D shapes. In *ACM Siggraph*, pages 203–212, 2001.

[39] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Transactions on Graphics*, 22(3):954–961, 2003.

[40] In-Kwon Lee. Curve reconstruction from unorganized points. *Computer Aided Geometric Design*, 17(2):161–177, 2000.

[41] D. Levin. Mesh-independent surface interpolation. *Geometric Modeling for Scientific Visualization*, 3:37–49, 2003.

[42] C. Luo, I. Safa, and Y. Wang. Approximating gradients for meshes and point clouds via diffusion metric. *Computer Graphics Forum*, 28:1497–1508(12), 2009.

[43] B. Mederos, L. Velho, and L. H. de Figueiredo. Moving least squares multiresolution surface approximation. *SibGrapi*, pages 19–26, 2003.

[44] F. Memoli and G. Sapiro. Distance functions and geodesics on submanifolds of $\mathbb{R}^d$ and point clouds. *SIAM Journal of Applied Mathematics*, 65(4):1227–1260, 2005.

[45] F. Memoli and G. Sapiro. A theoretical and computational framework for isometry invariant recognition of point cloud data. *Foundations of Computational Mathematics*, 5(3):313–347, 2005.

[46] X. Ni, M. Garland, and J. C. Hart. Fair morse functions for extracting the topological structure of a surface mesh. In *ACM Siggraph*, pages 613–622, 2004.

[47] R. Ogniewicz and M. Ilg. Voronoi skeletons: theory and applications. In *Proc. of Computer Vision and Pattern Recognition*, pages 63–69, 1992.

[48] L. Paraboschi, S. Biasotti, and B. Falcidieno. Comparing sets of 3D digital shapes through topological structures. In *Proc. of International Conference on Graph-based Representations in Pattern Recognition*, pages 114–125, 2007.

[49] V. Pascucci, G. Scorzelli, P.-T. Bremer, and A. Mascarenhas. Robust on-line computation of Reeb graphs: simplicity and speed. *ACM Trans. Comp. Graph.*, 26(3):58, 2007.

[50] G. Patanè, M. Spagnuolo, and B. Falcidieno. A minimal contouring approach to the computation of the reeb graph. *IEEE Trans. on Visualization and Computer Graphics*, 2009.

[51] M. Pauly and M. Gross. Spectral processing of point-sampled geometry. In *ACM Siggraph*, pages 379–386, 2001.

[52] M. Pauly, M. Gross, and L. P. Kobbelt. Efficient simplification of point-sampled surfaces. In *Proc. of the conference on Visualization*, pages 163–170, 2002.

[53] M. Pauly, R. Keiser, L. P. Kobbelt, and M. Gross. Shape modeling with point-sampled geometry. *ACM Transactions on Graphics*, 22(3):641–650, 2003.

[54] G. Reeb. Sur les points singuliers d'une forme de Pfaff complètement intègrable ou d'une fonction numèrique. *Comptes Rendus Hebdo-*

madaires des Sèances de l'Acadèmie des Sciences, 222:847–849, 1946.

[55] M. R. Ruggeri, T. Darom, D. Saupe, and N. Kiryati. Approximating geodesics on point set surfaces. In *Symposium on Point-based Graphics 2006*, pages 85–94, 2006.

[56] A. Sharf, T. Lewiner, A. Shamir, and L. Kobbelt. On-the-fly curve-skeleton computation for 3D shapes. *Computer Graphics Forum*, 26, 2007.

[57] Y. Shinagawa, T. L. Kunii, and Y. L. Kergosian. Surface coding based on Morse theory. *IEEE Computer Graphics and Applications*, 11:66–78, 1991.

[58] K. Siddiqi and S. M. Pizer, editors. *Medial Representations: Mathematics, Algorithms and Applications*. Springer, 2007.

[59] G. Singh, F. Memoli, and G. Carlsson. Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition. pages 91–100, Prague, Czech Republic, 2007. Eurographics Association.

[60] S. Svensson, I. Nystr om, and G. Sanniti di Baja. Curve skeletonization of surface-like objects in 3D images guided by voxel classification. *Pattern Recognition Letters*, 23(12):1419–1426, 2002.

[61] A. Tagliasacchi, H. Zhang, and D. Cohen-Or. Curve skeleton extraction from incomplete point cloud. *ACM Transactions on Graphics*, 28(3):1–9, 2009.

[62] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[63] J. Tierny, A. Gyulassy, E. Simon, and V. Pascucci. Loop surgery for volumetric meshes: Reeb graphs reduced to contour trees. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1177–1184, 2009.

[64] T. Tung and F. Schmitt. The Augmented Multiresolution Reeb Graph approach for content-based retrieval of 3D shapes. *Int. J. of Shape Modelling*, 11(1):91–120, June 2005.

[65] N. Werghi, Y. Xiao, and J. P. Siebert. A functional-based segmentation of human body scans in arbitrary postures. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 36(1):153–165, 2006.

[66] R. C. Wilson, E. R. Hancock, and B. Luo. Pattern vectors from algebraic graph theory. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(27):1112–1124, 2005.

[67] Y. Xiao, P. Siebert, and N. Werghi. A discrete Reeb graph approach for the segmentation of human body scans. In *3DIM*, pages 378–385, 2003.

[68] H. Xie, J. Wang, J. Hua, H. Qin, and A. Kaufman. Piecewise C^1 continuous surface reconstruction of noisy point clouds via local implicit quadric regression. In *IEEE Visualization*, page 13, 2003.