Persistence Analysis of Multi-scale Planar Structure Graph in Point Clouds

T. Lejemble[1], C. Mura[2], L. Barthe[1] and N. Mellado[1]

[1]IRIT, Université de Toulouse, CNRS
[2]Department of Informatics, University of Zurich

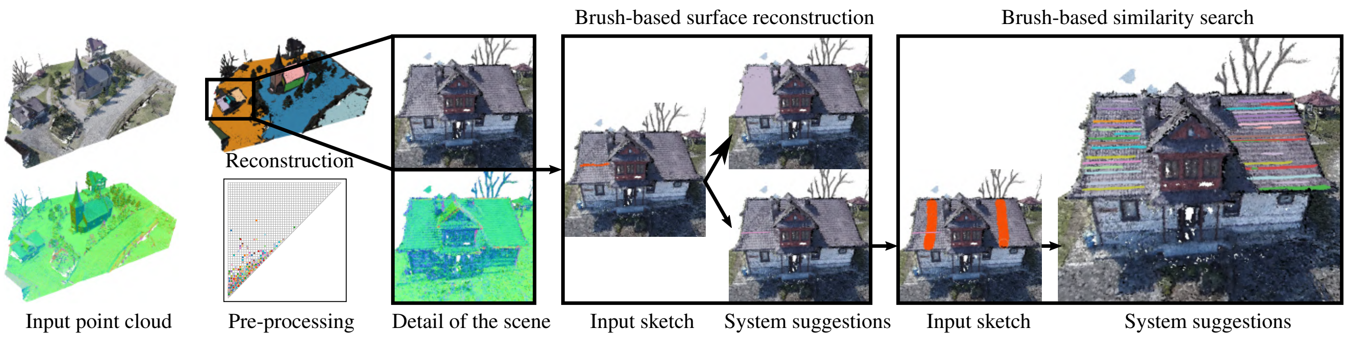


Figure 1: *Starting from an input point cloud equipped with normal vectors, our approach extracts meaningful planar components describing the geometry at multiple scales. Using persistence analysis, we offer to the user several ways to interactively explore, visualize and reconstruct the input data. The user can for instance generate planar reconstructions at arbitrary scales, select planar components by sketching directly on the point clouds, and/or find similar planar components.*

Abstract

Modern acquisition techniques generate detailed point clouds that sample complex geometries. For instance, we are able to produce millimeter-scale acquisition of whole buildings. Processing and exploring geometrical information within such point clouds requires scalability, robustness to acquisition defects and the ability to model shapes at different scales. In this work, we propose a new representation that enriches point clouds with a multi-scale planar structure graph. We define the graph nodes as regions computed with planar segmentations at increasing scales and the graph edges connect regions that are similar across scales. Connected components of the graph define the planar structures present in the point cloud within a scale interval. For instance, with this information, any point is associated to one or several planar structures existing at different scales. We then use topological data analysis to filter the graph and provide the most prominent planar structures.
Our representation naturally encodes a large range of information. We show how to efficiently extract geometrical details (e.g. tiles of a roof), arrangements of simple shapes (e.g. steps and mean ramp of a staircase), and large-scale planar proxies (e.g. walls of a building) and present several interactive tools to visualize, select and reconstruct planar primitives directly from raw point clouds. The effectiveness of our approach is demonstrated by an extensive evaluation on a variety of input data, as well as by comparing against state-of-the-art techniques and by showing applications to polygonal mesh reconstruction.

CCS Concepts
• ***Computing methodologies*** $\rightarrow$ *Point-based models; Shape analysis;*

1. Introduction

Recent years have seen tremendous improvements in the capabilities and the accessibility of 3D point-based acquisition devices. This enables the acquisition of large-scale 3D models of real-world entities – from complex objects to buildings and environments – with unprecedented quality and detail, which generates a massive amount of digital data that need to be processed and interpreted to fully exploit their information.

Interpreting and understanding data content remains a very important, yet challenging and open problem. Several approaches aim at organizing large datasets into semantically meaningful parts. Typically, the basic assumption is that parts of the data that carry a semantic meaning must exhibit coherent geometric properties, such as sheer size or fitness to a certain primitive type [AP10]. Thus, many state-of-the-art pipelines detect structures of interest in an input model according to specific rules that combine geometric properties with specific scales of observation, often defined a priori based on the specific domain considered (e.g. urban landscapes [VLA15]).

When considering general scanned models, one cannot rely on a given definition of the relevant scales of observation, simply because this would require a hierarchical ontology for all the entities of the real-world. Nevertheless, it can be postulated that the meaningful parts of real-world entities vary depending on the scale of observation and that they exhibit a certain degree of geometric consistency. The keys for capturing the informative parts of a generic 3D model is the definition of (a) a geometric consistency criterion that faithfully describes most real-world entities and (b) the scales of observation that highlight the meaningful structures.

Recently, Fang et al. [FLD18] propose the extraction of meaningful planes at different scales by exploration of a 2D parameter space whose representative sub-spaces are evaluated using a learning process. This method, however, requires both a greedy contraction to sample the space and an extensive set of training data to define the set of potentially meaningful scales.

In this paper, we propose a new representation allowing the interactive exploration and extraction of the meaningful parts of a large-scale 3D model at different scales. As it is well established that man-made objects and structures can be faithfully represented in a piecewise-planar manner [MZL*09, ASF*13], we restrict our geometric characterization to planar structures. The key element of our approach is the definition of a graph whose nodes represent planar regions extracted at increasing scales and whose edges connect regions that are similar across scales. By studying the topological persistence of the graph components, we propose a new way of detecting and characterizing geometrical structures at multiple scales, such that a single point can belong to multiple planar structures depending on their scale.

Using simple and intuitive visual tools, we let the user navigate interactively among the planar components of the model, which are highlighted for different sets of scales, as illustrated in Figure 1. Such exploration allows the user to determine the parts of the model that are meaningful and to export them with a geometrically compact format (e.g. fitting rectangles and polygonal boundary approximations). By doing so, we incorporate human knowledge in the selection of the meaningful parts of a model, performing the bulk of the computations automatically and only requiring simple interaction to analyze and filter the results.

We have tested our approach on a number of both synthetic and real-world models represented as 3D point clouds. The results show that our method allows for an effective analysis of generic scanned 3D models and generates compact representations of the underlying entities at different meaningful scales of observation. Compared to previous approaches, our solution is more flexible in the definition of the scales of interest and leads to the extraction of more informative representations, as demonstrated in a dedicated evaluation. We also demonstrate that our approach remains tractable for large-scale datasets composed of dozens of millions of points.

2. Related Work

Our approach spans several different research topics in computer graphics and vision. We review the most relevant approaches grouped into three main categories, with a focus on techniques applying to point-based inputs.

Simple primitives detection. Structuring a raw 3D point cloud into patches corresponding to simple geometric primitives is a basic first step in many 3D processing pipelines [CLP10, MPM*14, MMBM15]. Since real-world entities (in particular, man-made objects) can be approximated in a piece-wise planar way, planes are among the most common primitives considered. Several methods [RvdHV06,PVBP08] extract planar patches using region growing. They expand a patch from a starting point by aggregating neighbors that have low offset and low normal deviation. To increase efficiency, small planar patches can be represented as voxels on which a similar region growing is performed [VTHLB15]. Note that the maximum offset and normal deviation are normally specified as fixed input thresholds, making these approaches effective only in the presence of low or known levels of noise.

Noise and other defects in the input are handled effectively by randomized methods, which are based on randomly generating a large set of primitive hypotheses (not restricted to planes) from the input data. In some pipelines, inspired by the Ransac algorithm, the primitives that best explain the input data are selected [SWK07]. Other approaches, based on the well-known Hough Transform, let each primitive cast a vote in a discretization of the parameter space and select the primitives corresponding to the most voted parameter sets [BELN11]. Though robust and not restricted to planar primitives, such randomized approaches require testing a high number of primitives to ensure that all relevant features are captured.

Some approaches use the output of both region growing and randomized algorithms as starting point for a more global formulation [PERW16, DYHS18, GLCV19]. These approaches are mostly based on minimizing an energy function designed to penalize the fitting error to the underlying data while favoring the use of a reduced number of models to explain the data [YCS11,IB12]. Similar techniques are used when the input is an RGB-D image [SHKF12] or a sparse point cloud [SSS09]. However, this strategy significantly increases the technical and computational complexity of the processing while only partially improving the initial segmentation.

Although effective in many specific use-cases, the success of the primitive detection techniques presented so far is highly dependent on the correct setting of their parameters, which is often unintuitive. In addition, fixing these parameters implicitly defines one single scale at which the detection is performed. In our pipeline, we rather apply a simple region growing approach with fixed parameters and vary the scale at which the underlying features are computed. This leads to results that reflect different scales of observation by simply varying an intuitive parameter like the scale of observation.

For a more in-depth review of primitive extraction approaches we refer the reader to the recent survey by Kaiser et al. [KYZB19].

Structural segmentation. The output of the previous methods can be used to build more structured abstractions of the input data. The detected primitives can be arranged in a topological graph based on spatial proximity [SWWK08]. The connectivity of this graph combines adjacent primitives in a hierarchical way, either starting from unrefined planar regions and merging them based on planarity [FTK14] or by considering individual points as initial primitives and aggregating them into more general shapes [AP10]. The hierarchical nature of this graph makes it amenable for scale-aware reasoning, though this direction is not explored in these works. In contrast, we build a hierarchical graph that represents the planar primitives at several scales of analysis and use their *persistence* inside this graph to discover relevant structures across different scales.

A number of more advanced approaches detect global relationships between simple fitted primitives and use them to guide an optimization-based fitting. Some approaches detect non-planar primitives (e.g. cylinders, spheres, cones) as well as planar parts [LWC*11], while others only focus on planes and extract both the primitive parameters and their inter-relationships in a joint manner, maximizing robustness [MMBM15, OLA16]. These approaches capture global regularities, but do not convey any explicit information on the scale at which the analysis is performed. Rather than focusing on regularity relations, our goal is to discover the relevance of the structures at different scales of observation.

In fact, the extraction of scale-aware representations of raw 3D models has only recently emerged as a meaningful research problem. Using a data-driven approach, Hu et al. [HCL18] learn a patch-based label assignment, extracting the patches at a single scale and using the available labels to constrain their boundaries. During testing, the segmentation into patches is performed at several geometric scales and the scale that best matches the learned patch-based labeling is selected. This approach yields a scale-aware segmentation, but it heavily relies on the availability of labeled input data and it does not consider the problem of how to convert the output segmentation to a suitable representation.

Fang and colleagues [FLD18] observe that the different scales of abstraction of a model can be obtained by exploring the 2D space defined by the size σ and the fitting error ε of its composing parts. Since the pairs (ε, σ) corresponding to meaningful scales are located on the diagonal of this space, they generate a redundant set of abstractions corresponding to samples on this diagonal, arguing that this amounts to repeatedly applying local geometric contractions to the input model. Finally, they select a fixed number of abstractions, optimally matching them to the learned preferences of a group of users. While we move from similar motivations, we do not rely on local simplifications to generate the meaningful abstractions and extract a large set of candidate meaningful parts at different scales of analysis. Moreover, instead of relying on a learnt definition for the meaningful scale, we automatically propose meaningful parts by analyzing their persistence across different scales of analysis and allow the user to explore and refine our proposals interactively using a variety of intuitive tools.

Multi-scale feature estimation. The estimation of local features such as normal vector and curvature is a fundamental preliminary step for a number of different applications. In the context of surface reconstruction, Hoppe et al. [HDD*92] estimate normal vectors in unstructured point clouds by performing the principal component analysis (PCA) of the k-neighborhood of each point. In their work, a fixed neighborhood size is used, resulting in features that correspond to a single scale. However, varying the value of k allows to perform the analysis at different scales. This approach has been used for the extraction of feature lines by Pauly and colleagues [PKG03]. In their work, they also describe how treating the neighborhood size as a discrete parameter allows to translate the concept of scale-space representations [Wit87] from the functional case to the discrete setting of point-sampled geometry. A similar definition of multi-scale neighborhoods has been also used for feature-based semantic classification [HWS16b, TGDM18].

More recently, Mellado et al. [MGB*12] combine varying scales of analysis with the Moving Least Squares (MLS) representation. In particular, they fit algebraic sphere surfaces [GG07] to neighborhoods of continuously increasing size and define a compact geometric descriptor. This descriptor captures the geometric properties of a surface, including normal and curvature, for any continuous choice of scale and location on the surface. In our work, we use this approach to robustly estimate the local geometric features of the input model in a multi-scale manner and use them to guide the extraction of candidate planar components at each scale. Compared to the original formulation, we replace the standard algebraic sphere fitting by a more robust approach [ÖGG09] that helps preserve sharp features and further decrease the influence of outliers.

3. Key observation

Our approach is based on the key observation that smoothing a surface at increasing scale generates *stable* areas on the surface, i.e. surface areas characterized by similar differential properties at different locations and scales. In Figure 2, we illustrate this concept on a 2D parametric curve (Figure 2(a)). The curvature scale-space [Wit87] of this curve is plotted in Figure 2(b), where the color represents the curvature value (blue positive, orange negative and white null), the abscissa is the curve parameter, and the ordinate is the scale of analysis.

Areas of similar curvature values are revealed by this plot and Figure 2(c) illustrates these *stable* curvature areas in space and scale with as color a representative curvature value. Figure 2(d) finally shows the shape components of corresponding curvature value for each of the *stable* areas. As we can see, these shape components, living in a range of scales, are particularly meaningful for interpreting the curve shape, depending on the scale it is observed (large scale in Figure 2(d-top), medium scale in Figure 2(d-middle) and low scale in Figure 2(d-bottom)). Our goal is to extract such structures in 3D point clouds.

The parametrization of the 2D curve used in this example enables an easy computation of the curvature scale-space which

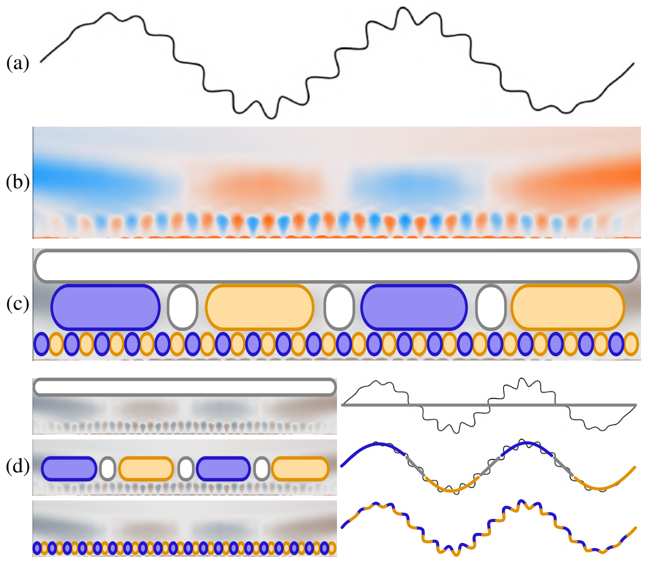


Figure 2: *Illustration of the curvature scale-space of a curve and its application to meaningful components detection. (a) Plot of a parametrized 2D curve. (b) Curvature scale-space of the curve, where the color represents the curvature value (blue-white-orange for positive-null-negative). The abscissa is the curve parametrization, and the ordinate is the scale of analysis. (c) We define* components *as* stable *areas in scale-space. (d) Components represent the geometrical structures at different scales, and any part of the curve can belong to several components at different scales.*

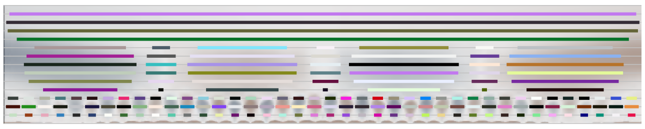
greatly simplifies the detection of *stable* areas. However, the 3D nature of point clouds and their lack of 2D parametrization prevent the direct transfer of this multi-scale analysis. In addition, constructing a robust parametric space from a massive acquired point cloud is still an open challenging problem, and we rather propose a different approach.

4. Overview

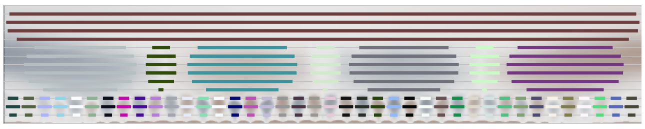
We wish to extract meaningful planar components at different scales from unstructured point clouds following a procedure inspired from the multi-scale analysis presented in the previous section. We propose a two-step process.

First, we compute for each point a set of per-scale normal vectors from local surface reconstructions performed at each different scale (Sections 5.1 and 5.2). This differential information parametrizes a region growing algorithm that groups points in planar *regions* at each scale (Section 5.3). Points in abscissa of Figure 3(a) are grouped in regions (different colors) by scale in ordinate.

Second, we store regions as *nodes* of a graph and create edges between regions extracted at successive scales if they share points. We then define *components* as stable regions linked in the graph, i.e. stable regions over scale (Section 5.4). Figure 3(b) shows with different colors the components obtained from the per-scale region segmentations illustrated in Figure 3(a).



(a) Per-scale segmentation



(b) Component extraction

Figure 3: *(a) Visualization of the segmentation result per scale. The abscissa represents the points of the point cloud, the ordinate is the scale, and colors denote the different segmented regions. (b) Following the concept presented in Section 3, the colors show the resulting components defined by stable regions over scales.*

To summarize, a *component* is defined as a set of *regions* computed at multiple scales and consistently covering the same part of the point cloud. We characterize a component by its scale of *birth*, its scale of *death*, and the geometrical properties of its *regions*. We illustrate the usefulness of our component characterization on several applications in Section 6.

5. Automatic extraction of multi-scale structures

5.1. Surface reconstruction at multiple scales

Our first requirement is the computation of a scale-space in which we can reconstruct a surface at different scales. As suggested by Pauly et al. [PKG03], the smoothing step used to build a scale-space from a parametrized signal can be adapted to point clouds by studying, per-point, the geometrical properties of neighborhood balls of increasing radii. Pauly et al. proposed to fit a plane to these neighborhoods and estimate curvature using PCA. More recently, Mellado et al. [MGB*12] demonstrated that algebraic sphere fitting provides a more stable description at higher scales.

We propose to push this idea further and study the properties of a surface reconstructed at multiple scales. The Moving Least Squares surface reconstruction scheme [Lev98] allows to evaluate continuous surfaces at multiple scales by varying the neighborhood fitting size. In this family of surface definition from point-sets, as suggested by Mellado et al. [MGB*12], we use the Algebraic Point Set Surface (APSS) [GG07,GGG08]. By construction, APSS generates smooth surfaces, which at high scales might lead to over-rounded objects with no planar regions (see Figure 4(a)). To overcome this limitation, we use the APSS formulation with non-linear regression kernel, proposed by Oztireli et al. [ÖGG09] and denoted RIMLS (see Figure 4(b)).

5.2. Scale-space sampling

The computation of segmentations at different scales and the efficient detection of the stable regions require an adequate discretization of the scale-space. Similarly to previous work [PKG03, MGB*12], we consider *m* scale values sampled from an interval

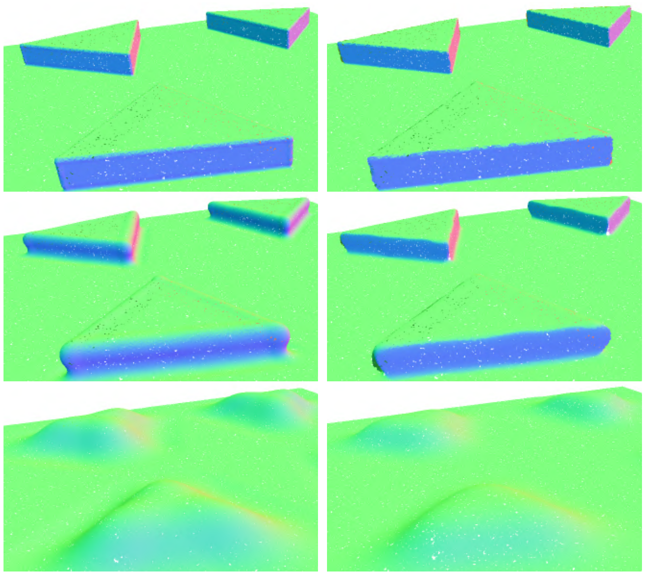


Figure 4: *Comparison between APSS (left) and RIMLS (right) reconstruction at 3 increasing scales with equal scale spacing in logarithmic scale. Surface normal coordinates (x,y,z) are mapped as RGB colors.*

$[t_{\min}, t_{\max}] \subset \mathbb{R}$ of the whole scale-space (we use $m = 50$ for all our experiments). In order to have enough samples to reconstruct the surface at small scales, we define the minimal scale $t_{\min}$ as the average distance of the input points to their k^{th} nearest neighbors. This smallest neighbor size is set empirically to $k = 10$ so that the RIMLS fitting remains sufficiently stable to be representative of the local geometry. Small changes in this value have a negligible effect. The choice of $t_{\max}$ is done conservatively and it is set as the size of the point cloud axis-aligned bounding box, as no planar structure can be bigger than the point cloud itself.

With these choices of $t_{\min}$ and $t_{\max}$, we select m values equally spaced in logarithmic scale, as this sampling strategy has been shown to allow for scale-invariant multi-scale signatures [BK10, MDS15]. Each scale sample t_j is thus computed according to the following equation:

$$t_j = t_{\min} \cdot \left(\frac{t_{\max}}{t_{\min}}\right)^{\frac{j-1}{m-1}} \qquad j = 1, \ldots, m \quad , \tag{1}$$

with $t_1 = t_{min}$ and $t_m = t_{max}$.

5.3. Planar segmentation at multiple scales

We compute the normal vector of a point as the normalized gradient of the scalar field reconstructed with RIMLS and, as commonly done for analyzing point clouds [CLP10, MPM*14], we use normal vectors as main features to steer the segmentation and provide the per-scale planar regions. Figure 5(a) illustrates the reconstructed surface with RIMLS at four different scales. Our expectation is that the segmentation at two successive scales generates similar regions where the underlying scale-space is stable. We thus seek a simple, yet stable region segmentation algorithm.

By construction, seedless region growing offers strong stability as the segmentation result is uniquely defined for a given local threshold criterion: two neighbor points are in the same region if they satisfy the local criterion, e.g. if the angle between their normal vectors is below a given threshold. In practice however, a seedless scheme does not provide control over the global property of a region, and thus the planarity of a region cannot be guaranteed. For instance, a sphere can be detected as a single region, as long as the angular distance between nearby vertices is small enough.

For these reasons, we rather employ a seed-based region growing tailored to enforce stability over scale variation. We define the planarity of a point $\mathbf{p}_i$ at scale t^j as $((\kappa_{1}{}_i^j)^2 + (\kappa_{2}{}_i^j)^2)^{-1}$, where κ_1^j and κ_2^j are the principal curvatures of the RIMLS surface reconstructed at the j^{th} scale. For each scale, we rank the points of the input cloud based on their planarity and start the region extraction from the most planar points. Seeds at different scales are thus points of the same planar region with similar planarity when the region is stable over scales.

In practice, considering the seed with the highest planarity $\mathbf{p}^j_{\text{seed}}$ at j^{th} scale, and with normal vector η^j_{seed}, we define an initial region $\mathcal{R}^j_{\text{seed}}$, associated to the plane that passes through $\mathbf{p}^j_{\text{seed}}$ and has normal vector η^j_{seed}. The region is then expanded by including the spatially close points that belong to its plane and have not yet been assigned to any region, i.e. a point $\mathbf{p}_i$ not already assigned is added to $\mathcal{R}^j_{\text{seed}}$ if (1) it is part of the k nearest neighbors of any $\mathbf{p} \in \mathcal{R}^j_{\text{seed}}$ and (2) the angle between its normal η^j_i and the seed's normal η^j_{seed} is lower than an angle θ. This process is repeated on the non assigned points until all points of the cloud belong to a region. We set $\theta = 5°$ and $k = 10$ for all scales. Note that the same value of k is used to select the minimum scale t_{min} (Section 5.2). This value is also a good compromise as using a smaller value might separate points that are in practice similar to the reconstructed surface and a higher value might lead to unwanted jumps during the region growing, where distant points are assigned to the same region even though they are separated by another thin region.

Each region obtained with this process represents a spatial aggregation of points reconstructed as a planar structure at a given scale t_j. As we use neighborhoods of size t_j, regions can only be representative of structures whose spatial extent is at least comparable to t_j. For this reason, we discard all regions $\mathcal{R}^j_i$ that have a surface area $a^i_j < 2 \cdot t_j$, with a^i_j being the area of the alpha-shape of $\mathcal{R}^j_i$ [EKS83], computed using $\alpha = 2 \cdot t_j$.

For a given scale t_j, the planar segmentation yields a set of regions $\mathcal{S}_j = \{\mathcal{R}^j_1, \ldots, \mathcal{R}^j_{N_j}\}$. These regions form by construction a partition of the input point cloud $\mathcal{P}$ at scale t_j. By repeating the region growing at each t_j, we obtain a set of output segmentations $\mathcal{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_m\}$, which corresponds to collection of regions sampling the scale-space of the input point cloud.

5.4. Multi-scale region graph and components extraction

Our goal is now to extract the planar components from the evolution of regions of $\mathcal{S}$ along scales. To do so, we structure the segmentation set $\mathcal{S}$ in a multi-scale region graph $G = (V, E)$ in which

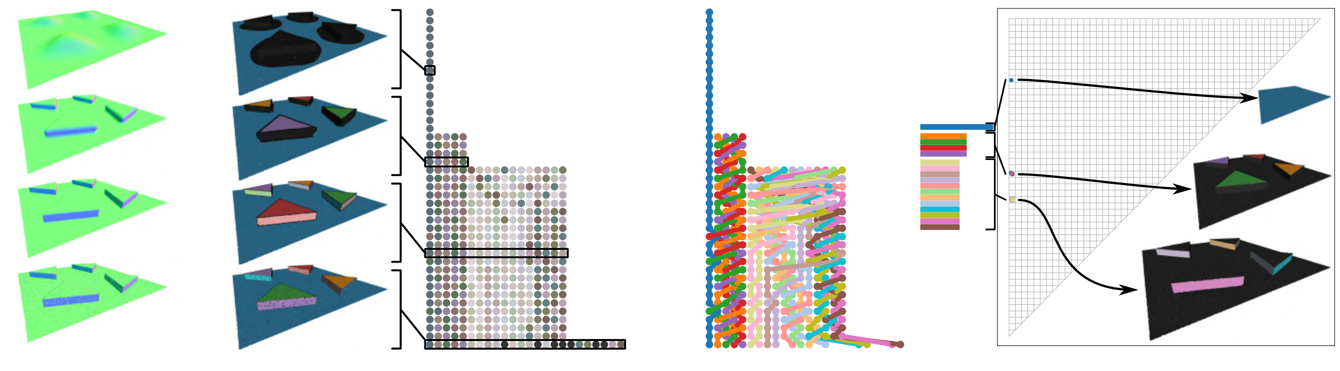

(a) Smoothed input (b) Graph nodes (c) Component extraction (d) Persistence diagram

Figure 5: *Our approach starts by (a) reconstructing the point cloud at different scales with RIMLS. (b) The reconstructed surfaces provide parameters for segmenting the point cloud in planar regions at multiple scales; these regions are stored as nodes in a hierarchical graph here illustrated by line of identical scale, from the lowest scale (bottom) to the higher scale (top). (c) Edges in the graph link the stable regions at successive scales and the colors show the different corresponding* components. *(d) The persistence analysis of the extracted components enables the characterization of planar structures at multiple scales.*

each region $\mathcal{R}_i^j$ is associated to exactly one node in V, denoted v_i^j. As each node v_i^j holds at a specific scale value, we treat G as a hierarchical structure in which all nodes corresponding to the same scale t_j form the j^{th} level of the graph and levels are ordered by increasing scale. Figure 5(b) shows the nodes of a multi-scale region graph sorted by level. Once regions are organized by levels in the graph, we connect by an edge all pairs of similar regions computed at successive scales. Each set of connected nodes then defines a *component* in the topological sense, and represents the stable planar area we are looking for in scale-space.

As we designed our segmentation to give similar results at consecutive scales in stable areas of the scale-space, we expect that regions in stable areas share a sufficient number of points. To measure the overlap between two regions v_i^j and v_k^{j+1}, we use the Jaccard index $J(v_i^j, v_k^{j+1})$ defined as:

$$J(v_i^j, v_k^{j+1}) = \frac{|\mathcal{R}_i^j \cap \mathcal{R}_k^{j+1}|}{|\mathcal{R}_i^j \cup \mathcal{R}_k^{j+1}|}. \tag{2}$$

The Jaccard index $J(v_i^j, v_k^{j+1})$ is symmetric, and equals to 1 when two regions share exactly the same set of points, and drops to 0 for non-overlapping regions. In our settings, two regions at the same level t_{j+1} cannot overlap; as such, one node at level t_j has at most one node at level t_{j+1} for which $J(v_i^j, v_k^{j+1}) > 0.5$.

According to this observation, we connect in the graph all pairs of nodes laying at consecutive scales and having a Jaccard index strictly higher than 0.5 (Figure 5(c)). As illustrated in Figure 6, this simple rule allows the connection of nodes corresponding to regions with similar coverage, while preventing the connection of nodes having an ambiguous relation. One could consider using a stricter threshold, e.g., in range $[0.5:1]$; however, from our experiments this does not improve the component extraction. Each set of connected nodes in the graph finally define a planar component, as illustrated in Figure 5(c). Each component is characterized by a birth level l_b and a death level l_d (respectively, the lowest and highest levels of its regions), as well as the regions it contains. Note that isolated regions have no stability over scales and they are discarded rather than being considered as components.

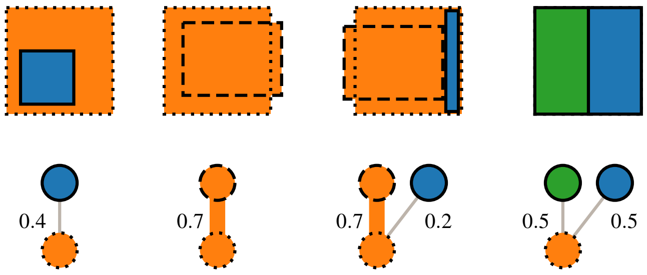
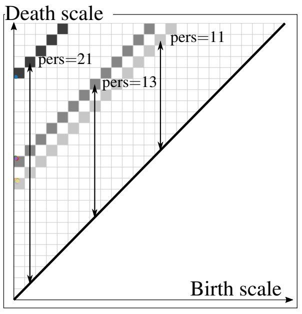


Figure 6: *Examples of relations between nodes. We connect nodes when their Jaccard index is strictly superior to* 0.5*. Connected nodes are colored with the same color, and belong to the same component.*

6. Interactive analysis of multi-scale structures

With this new representation, we propose new interactive tools for user-guided multi-scale exploration of point clouds. As our graph is organized as a collection of components with birth and death scales, we directly benefit from the toolbox developed in the domain of Topological Data Analysis (TDA) [CM17]. More specifically, we consider the concept of *persistence* (inspired by the more formalized notion of *topological persistence* [ELZ02]) and define it for a component $\mathcal{C}$ as the difference between its death and birth scales: $\text{pers}(\mathcal{C}) = l_d - l_b$. The persistence of components can be analyzed using a persistence diagram, i.e. a 2D diagram in which each component is a point with as abscissa its birth scale and as ordinate its death scale. In such diagrams, as illustrated in Figure 7-(a), components of equal persistence $\text{pers}(\mathcal{C})$ are on the same diagonal line, with those having the lowest scale of birth on the left side and those with the highest scale of birth on the right side. The diagonal in the middle of the plot would represent components of null persistence, while the bottom-right part below it would represent those with negative persistence, which do not exist in practice. The components of lower persistence ($\mathrm{pers}(\mathcal{C}) = 11$ in the figure) are those on the first diagonal line above the middle one and those with the highest possible persistence are on the top-left corner.



(a)

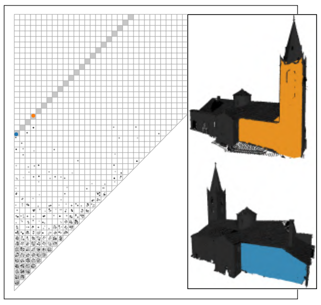

(b)

Figure 7: *(a) Persistence diagram breakdown for the* `tri` *scene (see Figure 5). (b) Comparison between two components with equal persistence on the* `lans` *scene. As shown in supplementary material, the surface covered by the orange component is noisy at small scales, which delays its date of birth.*

Following these observations, we can see in Figure 5-(d) that we have three sets of components grouped by equal persistence. All of them have a birth scale at the lowest scale and those having the lowest death scale are those with the lowest persistence (the closest to diagonal in the diagram). These components have a large enough persistence, meaning that they are stable over scales and they define meaningful planes explaining the data for the scales they cover. The slightly higher ones have a larger scale of death; they are also representative and define planes explaining the data up to higher scales. Finally, a single component exhibits a high scale of death with a very large persistence. These components explain data from small to large scales: at a very high scale, the single component of the overall plane is prominent, while at lower scales, the components of the parts of this plane that have no detail are more representative.

Figure 7-(b) shows two components of identical persistence. The blue one is representative up to the lowest scales because the point cloud is very clean in this part. The orange one has a higher birth scale. This is due to the noise slightly degrading this part of the point cloud.

Thus, according to TDA, components with larger persistence are more likely to represent prominent structures in the graph. In particular, as illustrated in Figure 5-(d), the persistence diagram of the components of a graph G allows to easily discriminate between the relevant geometric structures associated to persistent components. In Figure 8, we show how the persistence diagrams are affected by noise with increasing variance.

In the rest of this section we present a set of intuitive tools that allow to filter components based on their persistence, their scale interval, and/or the properties of their regions. All our tools perform interactively by exploiting the multi-scale region graph computed in pre-processing and let the user rapidly select a richer set

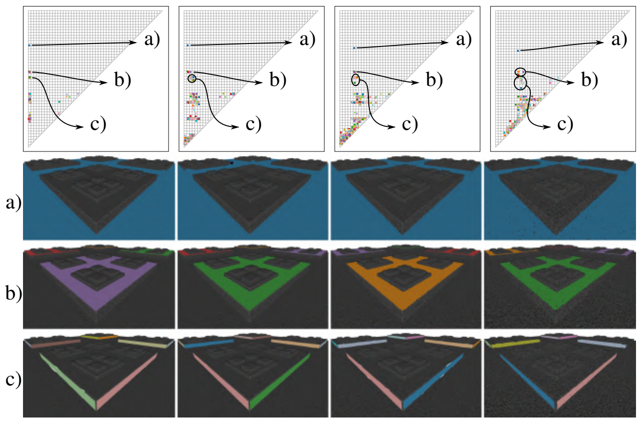


Figure 8: *Impact of positional gaussian noise on the component extraction for the* `cubes` *scene. The standard variation of the noise is a factor of the bounding box diagonal and is set to 1, 5, 10 and 25 times 10^{-5} from left to right.*

of planar parts that facilitates the interpretation of urban, CAD or indoor point clouds, in the same vein as previous well-established approaches for 3D modeling from point clouds [NSZ*10, ASF*13]. In the following, we say that a component holds a point if this point is part of any of the regions included in the component. We denote $\mathcal{P}_\mathcal{C}$ the set of points held by a component $\mathcal{C}$.

During the exploration, we propose several ways to display a component in the 3D space using the points held by the component:

- by coloring points of $\mathcal{P}_\mathcal{C}$ with a given color;
- by projecting the points on a plane approximating the component; we compute this representative plane by fitting the points held by the component in the least squares sense;
- by reconstructing the 2D alpha-shape of the projected point and displaying the resulting mesh or its contours.

6.1. Persistence-based thresholding

In TDA, persistent structures are considered as meaningful, and non-persistent structures as noise. We propose a simple tool where the user selects a minimum persistence value and the system visualizes on the input point cloud the components $\mathcal{C}$ for which the persistence $\mathrm{pers}(\mathcal{C})$ is greater than this minimal value. As shown in Figure 9, this approach is effective for point clouds whose components are distinctly clustered in the persistence diagram.

6.2. Scale-based point cloud segmentation

This tool performs a segmentation of the whole input point cloud with respect to a scale value t given by the user as illustrated in Figures 15 and 16. For every point $\mathbf{p}$, this tool selects the most persistent component among all the components including a region at the given scale t and holding the point $\mathbf{p}$. Points that are not associated to any component are kept unlabeled.

6.3. Interactive brush-based component selection

Using the aforementioned tools, we provide a unique controller acting globally on the point cloud. In order to let the user focus on a sub-part of the point cloud, we propose a brush-based interface to select specific planar components. With this tool, the user defines a set of query points $\mathcal{P}_Q$ by directly painting on the point cloud. Interactively, our system returns a set of components that "best match" the selected points in terms of overlapping and persistence (see Figure 10). More formally, we list all the components that hold any of the selected points, and rank them using the following score:

$$s(\mathcal{C}) = \alpha_{\text{points}}(\mathcal{C}) \cdot \alpha_{\text{pers}}(\mathcal{C}) \quad , \tag{3}$$

with $\alpha_{\text{points}}(\mathcal{C}) = |\mathcal{P}_Q \cap \mathcal{P}_{\mathcal{C}}|$ and $\alpha_{\text{pers}}(\mathcal{C}) = \text{pers}(\mathcal{C})/m$. The score function $s(\mathcal{C})$ accounts both for the fraction of selected points that are held by the component $\mathcal{C}$ and for the persistence of $\mathcal{C}$ normalized over the range of sampled scales. As the persistence score $\alpha_{\text{pers}}(\mathcal{C})$ is normalized, it acts as a penalty factor modulating the overlap ratio between the query and the component. Once a component is selected, the user can either colorize the point cloud accordingly or keep its alpha-shape for further use.

6.4. Interactive similarity search

Another interesting aspect of the components $\mathcal{C}$ is that they offer a high level descriptor that can be used for similarity queries. We propose an interactive search where components are matched with respect to a query and presented to the user as shown in Figure 17.

The tool performs as follows. The user selects a component using the brush-based selection tool. Then, she uses a second brush to select a set of points defining the search area. As for the selection tool, all the components that hold a point belonging to the search area are considered as similarity candidate. In order to verify if a candidate component $\mathcal{C}_c$ matches the query component $\mathcal{C}_q$, we propose to combine multiple criteria depending on the usage case (any criterion can be adjusted interactively):

1. the birth and death levels of $\mathcal{C}_c$ coincide with those of $\mathcal{C}_q$, plus or minus a given threshold.;
2. the planes approximating the components $\mathcal{C}_c$ and $\mathcal{C}_q$ are parallel at an angular threshold;
3. the ratio between the surface area of the alpha-shapes reconstructing the components $\mathcal{C}_c$ and $\mathcal{C}_q$ is lower than a given threshold, expressed as a percentage of the area of the query's alpha-shape.

7. Results

Implementation details. We have developed the prototype of our pipeline in C++, using Eigen [GJ*10] for linear algebra operations and CGAL [The19] for specific computational geometry routines, including alpha-shapes computation.

Datasets. Our test scenes, presented in Table 1, consist of 10 point cloud datasets of varying complexity. In these models, 4 were obtained by Multi-View Stereo (MVS), 3 by LiDAR scans of indoor and outdoor scenes, and 4 are synthetic point clouds generated by sampling hand-modeled meshes. These synthetic models represent relatively simple arrangements of geometric shapes and can be easily corrupted with controlled levels of noise; hence, they are perfect test cases to analyze specific properties of our method. The real-world point clouds represent large-scale building structures (`loudun`, `lans`, `pisa` [MDS15]), groups of buildings and their surroundings (`church`, `munich` [HWS16a]), and indoor environments (`euler`, `room` [ASZ*16]). We applied our different tools to all of our datasets, and we present a subset of our experiments in the following section. The remaining results are given in the supplementary material.

Processing time. Our experiments were run on an Intel(R) Xeon(R) CPU E5-2640 v4 clocked at 2.40GHz with 40 cores and 128G of RAM. The recorded timings for all test models are presented in Table 1, which provides a detailed breakdown of the individual steps. The total processing time ranges from about 110 seconds for our simplest synthetic model `tri` (0.5M points) to about 3.6 hours for `loudun` (35.5M points). It is worth noticing that all these steps need to be completed once only to generate the components, which are stored and simply loaded at the beginning of each interactive exploration stage. The step that requires the most computational power is the surface reconstruction. We recall that we compute, at each scale t_j and for each point of the input point cloud, the differential properties (normal vector and principal curvatures) of an implicit surface obtained using the RIMLS. To speed-up the computations, we sub-sample the neighborhood ball used to reconstruct the RIMLS using poisson-disk sampling, with a disk radius set as $r_j = 0.1t_j$. Note that the method by Fang et al. [FLD18] and Rapter require respectively 12 minutes and a couple of hours to process 1M points. In contrast, our approach requires around 6 minutes for 1M points, and processes a 35M point cloud in 3.6 hours. In addition, more than 90% of the processing time is spent on the RIMLS pre-computations, which could be locally recomputed in case of local editing.

Extraction of prominent structures. The persistence-base thresholding (Section 6.1) and the scale-based segmentation (Section 6.2) tools provide an immediate insight into the most relevant structures of a model at different scales.
We show in Figure 9 the extracted components for all our test models, filtered by increasing minimum persistence. The `pisa` model is a particularly good example to showcase the capabilities of our method. At the lowest persistence level, all the main roof sections and all the alcoves of the facade are captured. When increasing the persistence threshold, only the larger alcoves persist, until reaching the highest level, at which the largest roof sections are the only highlighted structures. A similar, yet even clearer trend is shown by `cubes`, a synthetic model consisting of four nested levels of planar structures. At the lowest persistence value, all planar faces of the boxes appear. As the threshold is increased, the smaller faces progressively disappear, as their features are more and more smoothed out due to the influence of the larger surrounding planes at higher scales of observation.
In Figure 15 and 16, the structures of the test models are highlighted through the segmentation induced by the persistent components that include a given scale. Note in particular how, at low scale, the individual tiles on the roof of `lans` are selected as individual segments, while for larger scales of interest they are

Model	#Points	Data source	Surf. Reconstruction	Segmentation	Filtering	Graph	Components	Total
`tri`	0.5M	Synthetic	77.78	1.59	27.19	3.42	0.00	109.98
`stairs`	1M	Synthetic	194.62	4.22	22.51	5.41	0.00	226.76
`cubes`	10M	Synthetic	1884.62	47.87	922.09	67.85	0.00	2922.43
`lans`	1.2M	LiDAR	310.85	6.31	15.87	4.37	0.01	337.41
`pisa`	2.5M	MVS	719.91	6.84	33.96	6.25	0.02	766.98
`church`	4.3M	MVS	1490.85	15.29	92.02	14.65	0.03	1612.84
`loudun`	35.5M	MVS	12299.40	147.15	606.82	126.43	0.84	13180.64
`room`	1.1M	MVS	372.93	2.31	20.90	5.77	0.00	401.91
`euler`	3.9M	LiDAR	1052.24	10.14	62.48	33.74	0.09	1158.69
`munich`	6.5M	LiDAR	1664.66	33.00	260.93	50.63	0.07	2009.29
`empire`	1M	Synthetic	383.72	2.21	15.87	4.24	0.00	406.04

Table 1: *Description of our test datasets and of the computing time required in preprocess for each subroutine. All timings are given in seconds.*

merged into larger individual roof segments and eventually into a single one. Likewise, the alcoves in the lower part of `pisa`, which appear as single entities at low and medium scales, are fused into a single structure at the highest scale value considered.

Interactive reconstruction and similarity search. Further insights into specific parts of the model can be obtained using the brush-based reconstruction (Section 6.3) and similarity search (Section 6.4) tools. Using the first tool on the `lans` model (Figure 17(a)), one can select individual structures on a facade with very rough sketches and replace them with low-complexity polygonal proxies, built based on their most representative associated components. Thanks to the specific score function used in this process (see Equation 3), the proxies approximate well the geometry of the selected structures. Figure 10(b) presents a similar reconstruction for `loudun`: remarkably, the reconstructed polygons correctly represent the underlying structures despite the high amount of clutter and outliers.

Alternatively, instead of reconstructing a surface, a selection can be used as a template in the search for matching structures. In Figure 17 we show a typical result of this workflow. On the model `lans`, the similarity search tool allows to sketch a first selection on a single roof tile (which becomes the query of our search) and then a second one roughly covering the whole roof segment (which represents the domain of the search). By matching the most persistent component underlying the query with the best matching component of the domain, the tool extracts the components representing all the other tiles in the roof (Figure 17(a)), without requiring that the user engages in a tedious and error-prone selection process. Proceeding in a similar way one can easily extract the individual steps from the staircase of `stairs` (Figure 17(b)) or the individual arches in an arcade of `pisa` (Figure 17(c)).

Point cloud coverage. With our technique, large portions of points may not be labeled. This is due to the segmentation of the RIMLS that tends to shrink at higher scales as the edges get rounded. The regions may be extended to the points that are close to the plane that best fits a region and that has a compatible normal. Figure 11 illustrates the coverage obtained with a distance threshold set to 20% of the bounding box diagonal length and an orientation threshold set to 45 degrees.

	# scale	% aabb	coverage	RMS error	planes
[SWK07]	-	-	0.808	0.034	128
[MMBM15]	-	-	0.817	0.042	163
[FLD18]	1	-	0.816	0.017	239
[FLD18]	2	-	0.816	0.29	40
[FLD18]	3	-	0.816	1.03	9
Ours	1	0.452	0.767	0.0001	128
Ours	2	0.890	0.753	0.0002	92
Ours	3	1.018	0.688	0.0006	49
Ours	4	5.88	0.434	0.0060	4

Table 2: *Comparison of our method with Ransac [SWK07], Rapter [MMBM15] and Fang et al. [FLD18] on the* `empire` *scene. For our method, stable regions are extracted at four scales, set to different percentages of the axis aligned bounding box (aabb) (see Figure 12). The two last columns give the corresponding Root Mean Square (RMS) error and the number of extracted planes. The coverage, RMS error and number of planes values for Ransac, Rapter and Fang et al. come from [FLD18]-Table 3.*

Comparisons. The fundamental difference between our approach and the multi-scale plane fitting proposed by Fang et al. [FLD18], Ransac [SWK07] and Rapter [MMBM15] is that we do not fit the "best" planes considering tolerance, coverage and eventually scale as parameters. We rather find planes faithfully representing the data at different scales. With our approach, one point belongs to zero, one or several planes identified at different range of scales, and at a fixed range of scales, the coverage of our planes significantly varies depending on the data and the considered scales. We ran a comparison on the `empire` scene, as done by Fang et al. [FLD18]-Table 3 with Ransac and Rapter. To avoid a corrupted RIMLS reconstruction at higher scales, we filtered outliers, computing for each point the covariance matrix with a neighborhood ball of radius = 1% of the aabb diagonal and keeping only points with planar neighborhood using standard heuristics (points kept: 79.3%). We have extracted planes (our stable regions) at four different scales: 0.452%, 0.890%, 1.018% and 5.88% of the axis aligned bounding box diagonal. Figure 12, as well as the coverage, root mean square error and number of planes presented in Table 2 at each scale highlight the difference of our approach: it aims at finding planes that explain the surface at different scales, rather than fitting planes that approx-

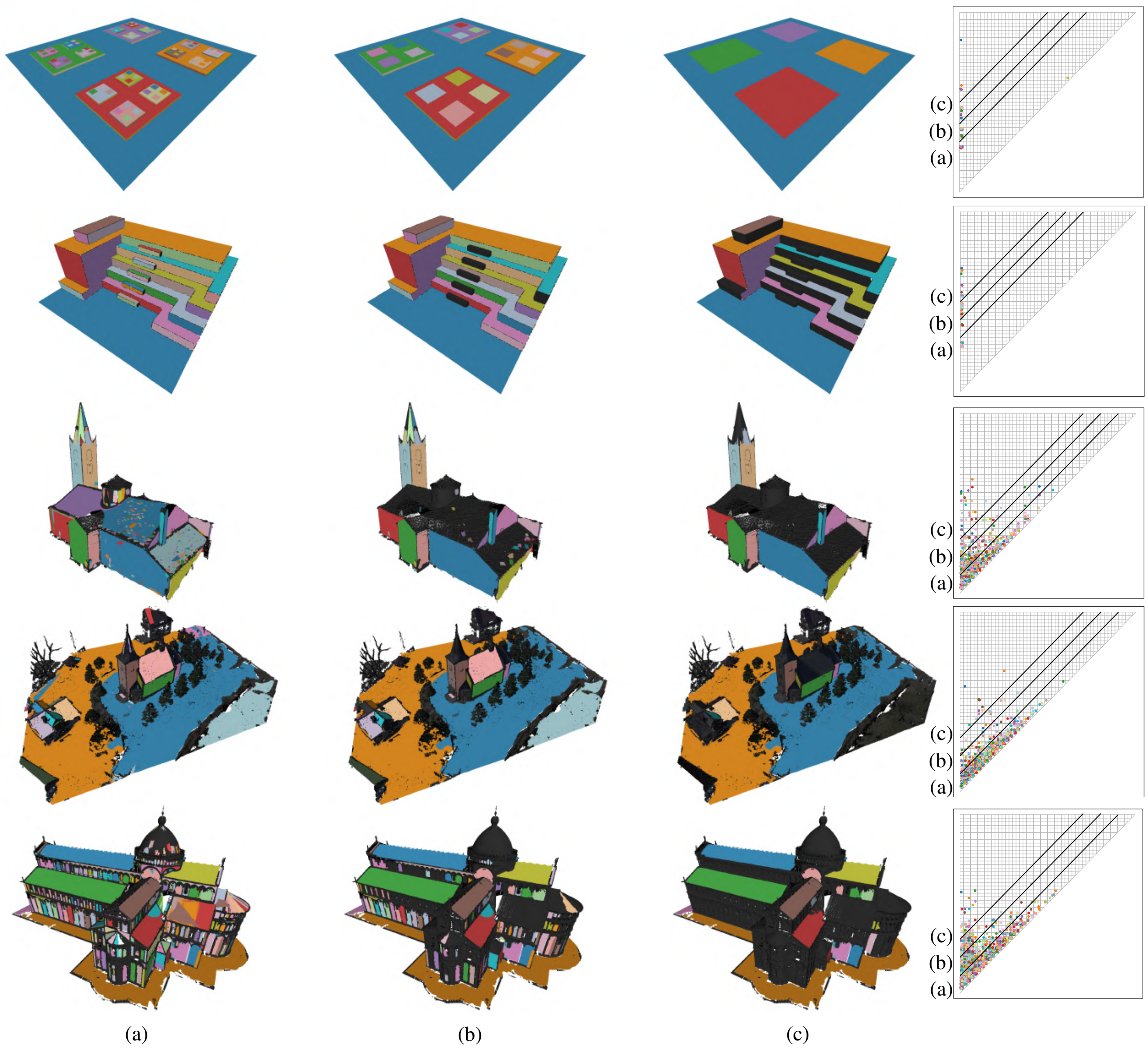


Figure 9: *Persistent components for five scenes (from top to bottom:* `cubes`*,* `stairs`*,* `lans`*,* `church`*,* `pisa`*) with three different persistence thresholds, illustrated on the persistence diagrams (Sec. 6.1).*

imate the points. As such, our tool produces planes with lower coverage and very low geometric error (several orders of magnitude lower than previous work), even at high scales.

Application to polygonal reconstruction. We considered the sets of planes obtained with our method at three different scales and used them as input for a polygonal reconstruction algorithm [NW17] (PolyFit). This algorithm reconstructs a watertight polygonal surface from a set of input planes, optimizing three energy terms related to data fidelity, reconstruction complexity and coverage, under hard constraints that ensure that the resulting mesh is manifold and closed. The resulting meshes for the `lans` and `empire` models are shown in Figures 13 and 14, respectively. We used the Polyfit implementation provided by the authors with default parameters, adding an extra plane at the bottom side of the bounding box to obtain a closed surface. We compare our results against those obtained by extracting the input planes using the CGAL implementation of Ransac [SWK07] with default parameters. Compared to this baseline, using our approach for the input planes selection allows to naturally generate a sequence of meshes at different levels of detail. This is possible thanks to our unique definition of scale, which is not accounted for by Ransac.

Effect of noise on components extraction. We evaluated how noise affects our results by corrupting the synthetic model `cubes` with increasing noise and analyzing the corresponding changes in the persistence diagram. In particular, we consider 4 levels of increasing Gaussian additive noise, corresponding to a standard de-

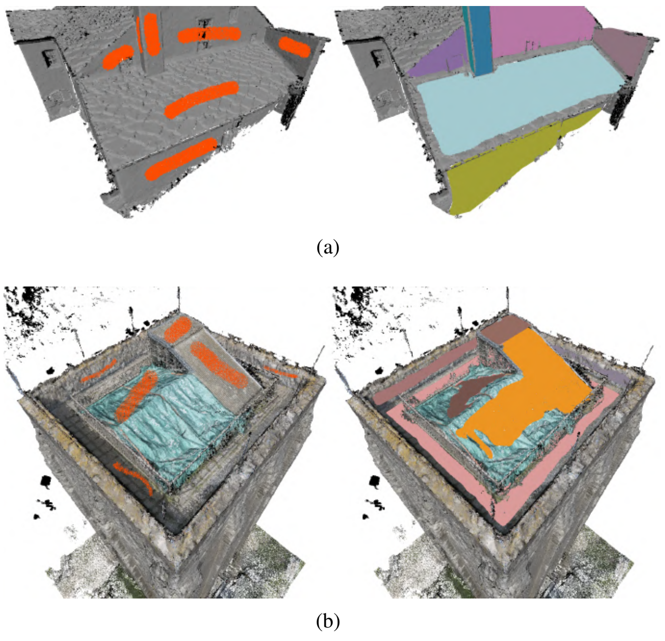


Figure 10: *Interactive reconstruction tool on models* `lans` *(a) and* `loudun` *(b): with just a few rough sketches, the user selects some structures (left) and the tool automatically reconstructs their low-complexity polygonal proxies (right).*

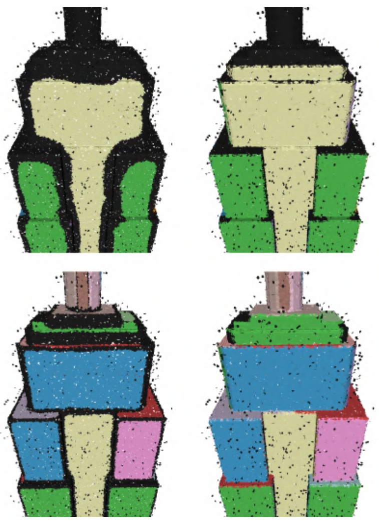


Figure 11: *Effect of extending the initial regions. Bottom: medium scale; top: high scale; left: initial regions; right: extended regions.*

viation σ_{noise} equal to 0.001%, 0.005%, 0.01% and 0.025% of the diagonal of the axis-aligned bounding box. As shown in Figure 8, at low levels of noise the main planar structures emerge already early in the scale-space, since even at the lowest scales the RIMLS reconstruction (Section 5.1) is not affected. For this reason, the corresponding components appear on the left of the diagram. As noise is increased, the point-wise surface reconstruction at low scales becomes unreliable. Hence, there is no region that can be detected as associated to those components at those scales, resulting in higher birth levels. Overall, as noise increases, the diagrams become more cluttered with new points (corresponding to noisy structures), which also appear more spread out and generally shifted towards the right. Nevertheless, the main planes can still be recognized as fairly localized clusters in each diagram.

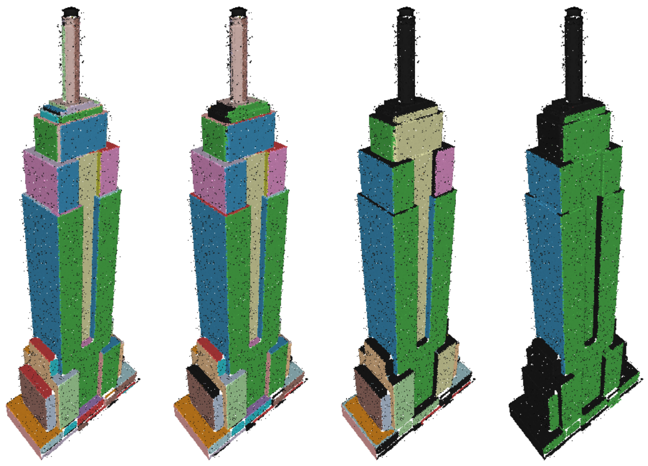


Figure 12: *Planes (stable regions) extracted at the four scales presented in Table 2. From left to right, scale* 1*,* 2*,* 3 *and* 4*.*

Failure cases. Our tests on `cubes` and `tri` reveal that the presence of a single large planar structure spanning the whole input model leads to a component that persists across all the *m* scales considered. In that specific case, the scale-based segmentation tool does not yield meaningful results as all points are associated with the unique component of this structure. In practice however, such situation can easily be detected and fixed by ignoring this largest-scale structure during the search.

8. Conclusions, limitations and future work

This paper introduces a novel method for the extraction of the meaningful structures of a 3D model at multiple scales and for their interactive analysis and exploration. Our approach is based on computing planar regions with similar differential properties at individual scales and on analyzing their stability across the scale-space, which we achieve by studying the topological persistence of a hierarchical graph that stores the regions at different scales. Furthermore, we provide intuitive tools for visualizing the most stable structures discovered, as well as to segment, reconstruct and perform part-based queries on the input model based on these structures. The resulting pipeline is effective and can be applied efficiently to inputs consisting of several millions of points.

Limitations. As many related methods, our approach is restricted to planar structures, since these are ubiquitous in man-made entities and since other smooth objects can be well represented using first-order approximations. Nevertheless, it would be interesting to consider more general types of primitives to define the initial regions,

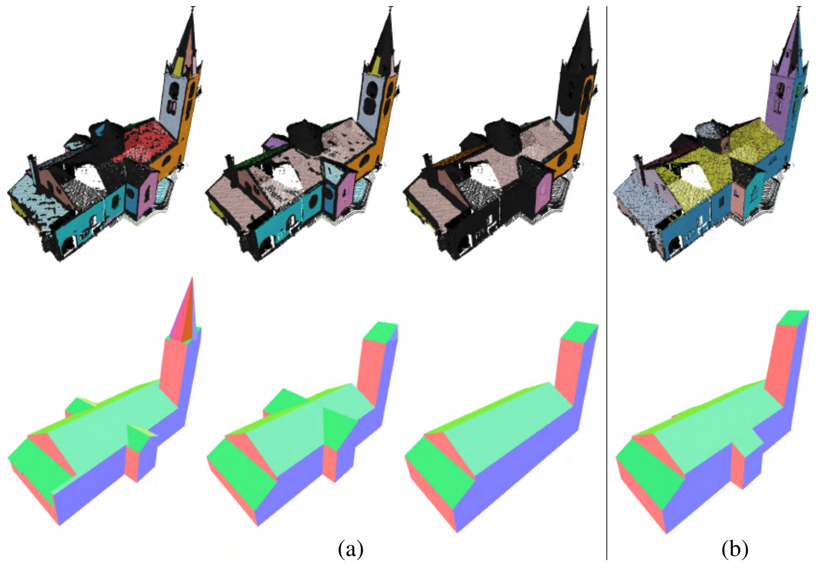


Figure 13: *Polygonal reconstruction of* `lans` *using PolyFit with our planes detected at* 3 *different scales (a) and with those detected by Ransac (b). Top row: planar segmentation; bottom row: reconstructed mesh.*

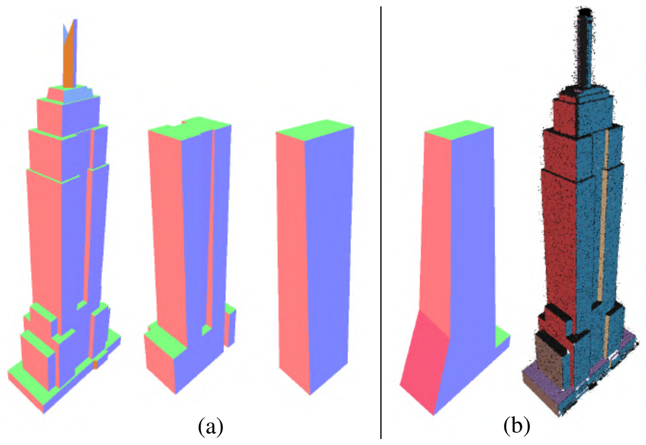


Figure 14: *Polygonal reconstruction of* `empire` *using PolyFit with our planes detected at* 3 *different scales (a) and with those detected by Ransac (b). The planar segmentation obtained with Ransac is shown next to the reconstructed mesh (b); the planar segmentations corresponding to the meshes in (a) are given in Figure 12 (scales* 2, 3 *and* 4*, respectively).*

using curvature (which we compute together with the normal) as an additional feature to drive their extraction. In addition, while we illustrate the behaviour of our pipeline under both synthetic and realistic real-world noise, we did not estimate specifically the maximum levels of noise and outliers that our approach can tolerate. Together with further improving the expressiveness and usability of our interactive tools, these two aspects represent interesting directions for future work.

In addition, while we illustrate the raw information directly provided by our method, eventually filtered with very simple geometric constraints (plane orientation and area), a dedicated UI could include the graph visualization to allow the user to navigate between components. Additional geometric constraints related to points distribution within components, component shape, and component relations (neighbors, orientation deviation, etc.) could also be studied.

Acknowledgments. This research has been partially supported by the CIMI Labex (ANR-11-LABX-0040) and by the grant "Forschungskredit Postdoc" of the University of Zurich (grant number FK-18-022).

References

[AP10] Attene M., Patanè G.: Hierarchical structure recovery of point-sampled surfaces. In *Computer Graphics Forum* (2010), vol. 29, Wiley Online Library, pp. 1905–1920. 2, 3

[ASF*13] Arikan M., Schwärzler M., Flöry S., Wimmer M., Maierhofer S.: O-Snap: Optimization-based snapping for modeling architecture. *ACM Transactions on Graphics 32*, 1 (2013), 6:1–15. 2, 7

[ASZ*16] Armeni I., Sener O., Zamir A. R., Jiang H., Brilakis I., Fischer M., Savarese S.: 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition* (2016). 8

[BELN11] Borrmann D., Elseberg J., Lingemann K., Nüchter A.: The 3D hough transform for plane detection in point clouds: A review and a new accumulator design. *3D Research 2*, 2 (2011), 3. 2

[BK10] Bronstein M., Kokkinos I.: Scale-invariant heat kernel signatures for non-rigid shape recognition. In *Proc. Computer Vision and Pattern Recognition (CVPR)* (2010), pp. 1704–1711. 5

[CLP10] Chauve A.-L., Labatut P., Pons J.-P.: Robust piecewise-planar 3D reconstruction and completion from large-scale unstructured point data. In *Proc. Computer Vision and Pattern Recognition (CVPR)* (2010), pp. 1261–1268. 2, 5

[CM17] Chazal F., Michel B.: An introduction to topological data analysis: fundamental and practical aspects for data scientists. *CoRR abs/1710.04019* (2017). `arXiv:1710.04019`. 6

[DYHS18] Dong Z., Yang B., Hu P., Scherer S.: An efficient global energy optimization approach for robust 3D plane segmentation of point clouds. *ISPRS Journal of Photogrammetry and Remte Sensing* (2018). 2

[EKS83] EDELSBRUNNER H., KIRKPATRICK D., SEIDEL R.: On the shape of a set of points in the plane. *IEEE Transactions on Information Theory 29*, 4 (1983), 551–559. 5

[ELZ02] EDELSBRUNNER H., LETSCHER D., ZOMORODIAN A.: Topological persistence and simplification. *Discrete & Computational Geometry 28*, 4 (2002), 511–533. 6

[FLD18] FANG H., LAFARGE F., DESBRUN M.: Planar shape detection at structural scales. In *Proc. Computer Vision and Pattern Recognition (CVPR)* (2018). 2, 3, 8, 9

[FTK14] FENG C., TAGUCHI Y., KAMAT V. R.: Fast plane extraction in organized point clouds using agglomerative hierarchical clustering. In *Proc. International Conference on Robotics and Automation (ICRA)* (2014), pp. 6218–6225. 3

[GG07] GUENNEBAUD G., GROSS M.: Algebraic point set surfaces. *ACM Transactions on Graphics 26*, 3 (2007), 23. 3, 4

[GGG08] GUENNEBAUD G., GERMANN M., GROSS M.: Dynamic sampling and rendering of algebraic point set surfaces. *Computer Graphics Forum 27*, 2 (2008), 653–662. 4

[GJ*10] GUENNEBAUD G., JACOB B., ET AL.: Eigen v3. http://eigen.tuxfamily.org, 2010. 8

[GLCV19] GUINARD S., LANDRIEU L., CARAFFA L., VALLET B.: Piecewise-planar approximation of large 3D data as graph-structued optimization. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences 4* (2019), 365–372. 2

[HCL18] HU S.-M., CAI J.-X., LAI Y.-K.: Semantic labeling and instance segmentation of 3D point clouds using patch context analysis and multiscale processing. *Transactions on Visualization and Computer Graphics (TVCG)* (2018). 3

[HDD*92] HOPPE H., DEROSE T., DUCHAMPT T., MCDONALD J., STUETZLE W.: Surface reconstruction from unorganized points. In *Proc. ACM SIGGRAPH* (1992), pp. 71–78. 3

[HWS16a] HACKEL T., WEGNER J. D., SCHINDLER K.: Contour detection in unstructured 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 1610–1618. 8

[HWS16b] HACKEL T., WEGNER J. D., SCHINDLER K.: Fast semantic segmentation of 3D point clouds with strongly varying density. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences 3*, 3 (2016), 177–184. 3

[IB12] ISACK H., BOYKOV Y.: Energy-based geometric multi-model fitting. *Int. Journal of Computer Vision 97*, 2 (2012), 123–147. 2

[KYZB19] KAISER A., YBANEZ ZEPEDA J. A., BOUBEKEUR T.: A survey of simple geometric primitives detection methods for captured 3D data. *Computer Graphics Forum 38*, 1 (2019), 167–196. 3

[Lev98] LEVIN D.: The approximation power of moving least-squares. *Mathematics of computation 67*, 224 (1998), 1517–1531. 4

[LWC*11] LI Y., WU X., CHRYSATHOU Y., SHARF A., COHEN-OR D., MITRA N. J.: Globfit: Consistently fitting primitives by discovering global relations. *ACM Transactions on Graphics 30*, 4 (2011), 52. 3

[MDS15] MELLADO N., DELLEPIANE M., SCOPIGNO R.: Relative scale estimation and 3D registration of multi-modal geometry using growing least squares. *IEEE Transactions on Visualization and Computer Graphics (TVCG) 22*, 9 (2015), 2160–2173. 5, 8

[MGB*12] MELLADO N., GUENNEBAUD G., BARLA P., REUTER P., SCHLICK C.: Growing least squares for the analysis of manifolds in scale-space. *Computer Graphics Forum 31*, 5 (2012), 1691–1701. 3, 4

[MMBM15] MONSZPART A., MELLADO N., BROSTOW G. J., MITRA N. J.: Rapter: rebuilding man-made scenes with regular arrangements of planes. *ACM Transactions on Graphics (TOG) 34*, 4 (2015), 103–1. 2, 3, 9

[MPM*14] MATTAUSCH O., PANOZZO D., MURA C., SORKINE-HORNUNG O., PAJAROLA R.: Object detection and classification from large-scale cluttered indoor scans. *Computer Graphics Forum 33*, 2 (2014), 11–21. 2, 5

[MZL*09] MEHRA R., ZHOU Q., LONG J., SHEFFER A., GOOCH A., MITRA N. J.: Abstraction of man-made shapes. In *ACM Transactions on Graphics (TOG)* (2009), vol. 28, p. 137. 2

[NSZ*10] NAN L., SHARF A., ZHANG H., COHEN-OR D., CHEN B.: Smartboxes for interactive urban reconstruction. In *ACM Transactions on Graphics (TOG)* (2010), vol. 29, ACM, p. 93. 7

[NW17] NAN L., WONKA P.: Polyfit: Polygonal surface reconstruction from point clouds. *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), 2372–2380. 10

[ÖGG09] ÖZTIRELI A. C., GUENNEBAUD G., GROSS M.: Feature preserving point set surfaces based on non-linear kernel regression. In *Computer Graphics Forum* (2009), vol. 28, pp. 493–501. 3, 4

[OLA16] OESAU S., LAFARGE F., ALLIEZ P.: Planar shape detection and regularization in tandem. *Computer Graphics Forum 35*, 1 (2016), 203–215. 3

[PERW16] PHAM T. T., EICH M., REID I., WYETH G.: Geometrically consistent plane extraction for dense indoor 3D maps segmentation. In *Proc. Intelligent Robots and Systems (IROS)* (2016), pp. 4199–4204. 2

[PKG03] PAULY M., KEISER R., GROSS M.: Multi-scale feature extraction on point-sampled surfaces. In *Computer Graphics Forum* (2003), vol. 22, pp. 281–289. 3, 4

[PVBP08] POPPINGA J., VASKEVICIUS N., BIRK A., PATHAK K.: Fast plane detection and polygonalization in noisy 3D range images. In *Proc. Intelligent Robots and Systems (IROS)* (2008), pp. 3378–3383. 2

[RvdHV06] RABBANI T., VAN DEN HEUVEL F., VOSSELMAN G.: Segmentation of point clouds using smoothness constraint. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 36*, 5 (2006), 248–253. 2

[SHKF12] SILBERMAN N., HOIEM D., KOHLI P., FERGUS R.: Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision* (2012), Springer, pp. 746–760. 2

[SSS09] SINHA S., STEEDLY D., SZELISKI R.: Piecewise planar stereo for image-based rendering. 2

[SWK07] SCHNABEL R., WAHL R., KLEIN R.: Efficient RANSAC for point-cloud shape detection. In *Computer Graphics Forum* (2007), vol. 26, pp. 214–226. 2, 9, 10

[SWWK08] SCHNABEL R., WESSEL R., WAHL R., KLEIN R.: Shape recognition in 3D point-clouds. In *Int. Conference in Central Europe on Computer Graphics, Visualization and Computer Vision* (2008). 3

[TGDM18] THOMAS H., GOULETTE F., DESCHAUD J.-E., MARCOTEGUI B.: Semantic classification of 3D point clouds with multiscale spherical neighborhoods. In *Int. Conference on 3D Vision* (2018). 3

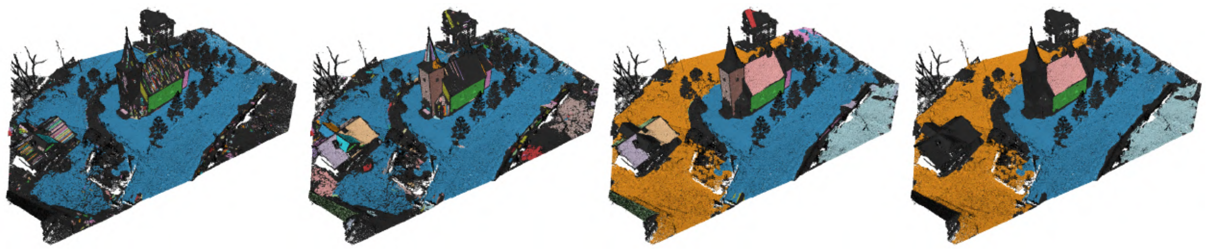
[The19] THE CGAL PROJECT: *CGAL User and Reference Manual*, 4.14.1 ed. CGAL Editorial Board, 2019. URL: `https://doc.cgal.org/4.14.1/Manual/packages.html`. 8

[VLA15] VERDIE Y., LAFARGE F., ALLIEZ P.: LOD generation for urban scenes. *ACM Transactions on Graphics 34*, 3 (2015). 2

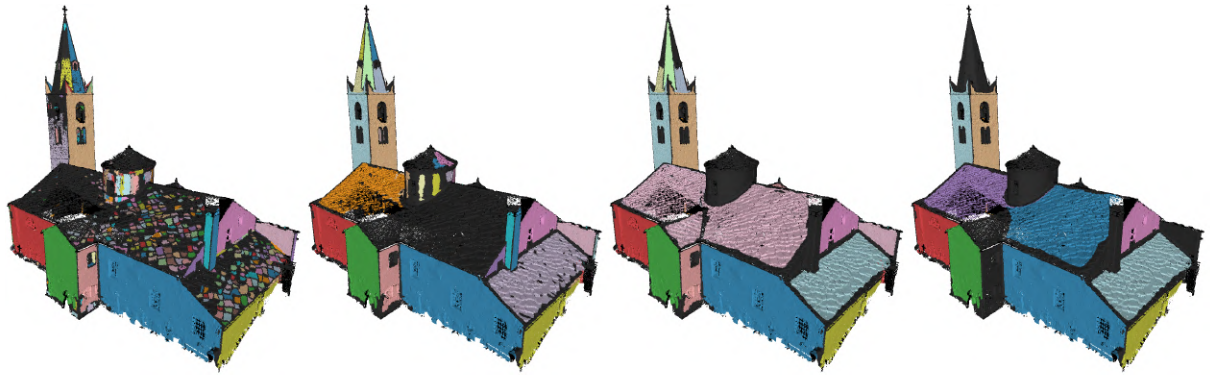
[VTHLB15] VO A.-V., TRUONG-HONG L., LAEFER D. F., BERTOLOTTO M.: Octree-based region growing for point cloud segmentation. *ISPRS Journal of Photogrammetry and Remote Sensing 104* (2015), 88–100. 2

[Wit87] WITKIN A. P.: Scale-space filtering. In *Readings in Computer Vision*. Elsevier, 1987, pp. 329–332. 3

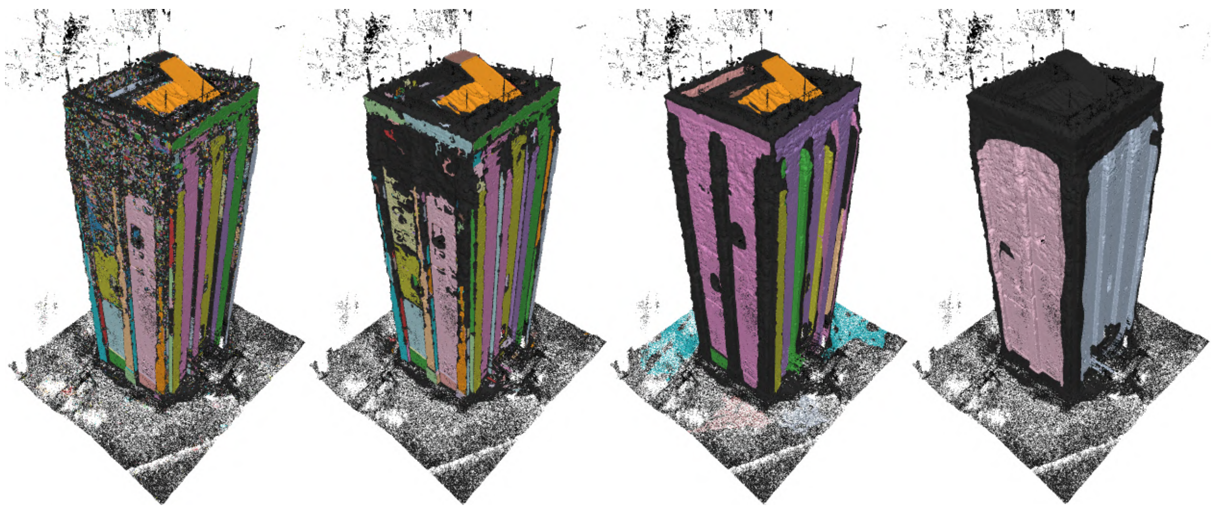
[YCS11] YU J., CHIN T.-J., SUTER D.: A global optimization approach to robust multi-model fitting. In *Proc. Computer Vision and Pattern Recognition (CVPR)* (2011), pp. 2041–2048. 2



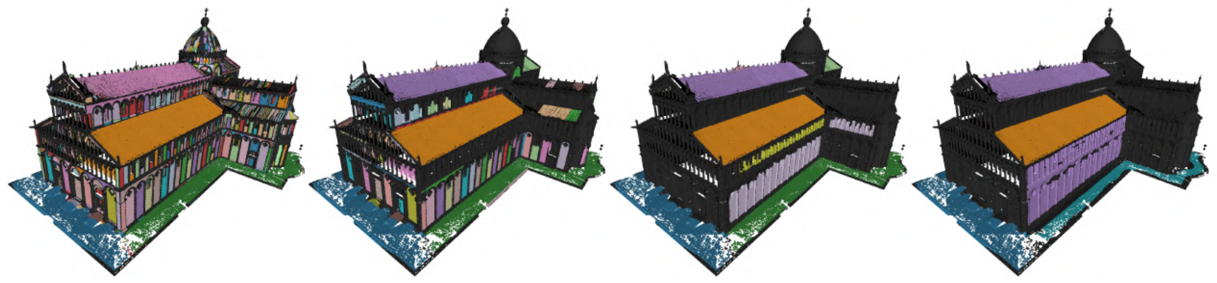
(a)



(b)



(c)



(d)

Figure 15: *Segmentation of models* `church` *(a),* `lans` *(b),* `loudun` *(c) and* `pisa` *(d) based on the most persistent components that include scales 5, 15, 20 and 25 out of the 50. (Sec. 6.2).*

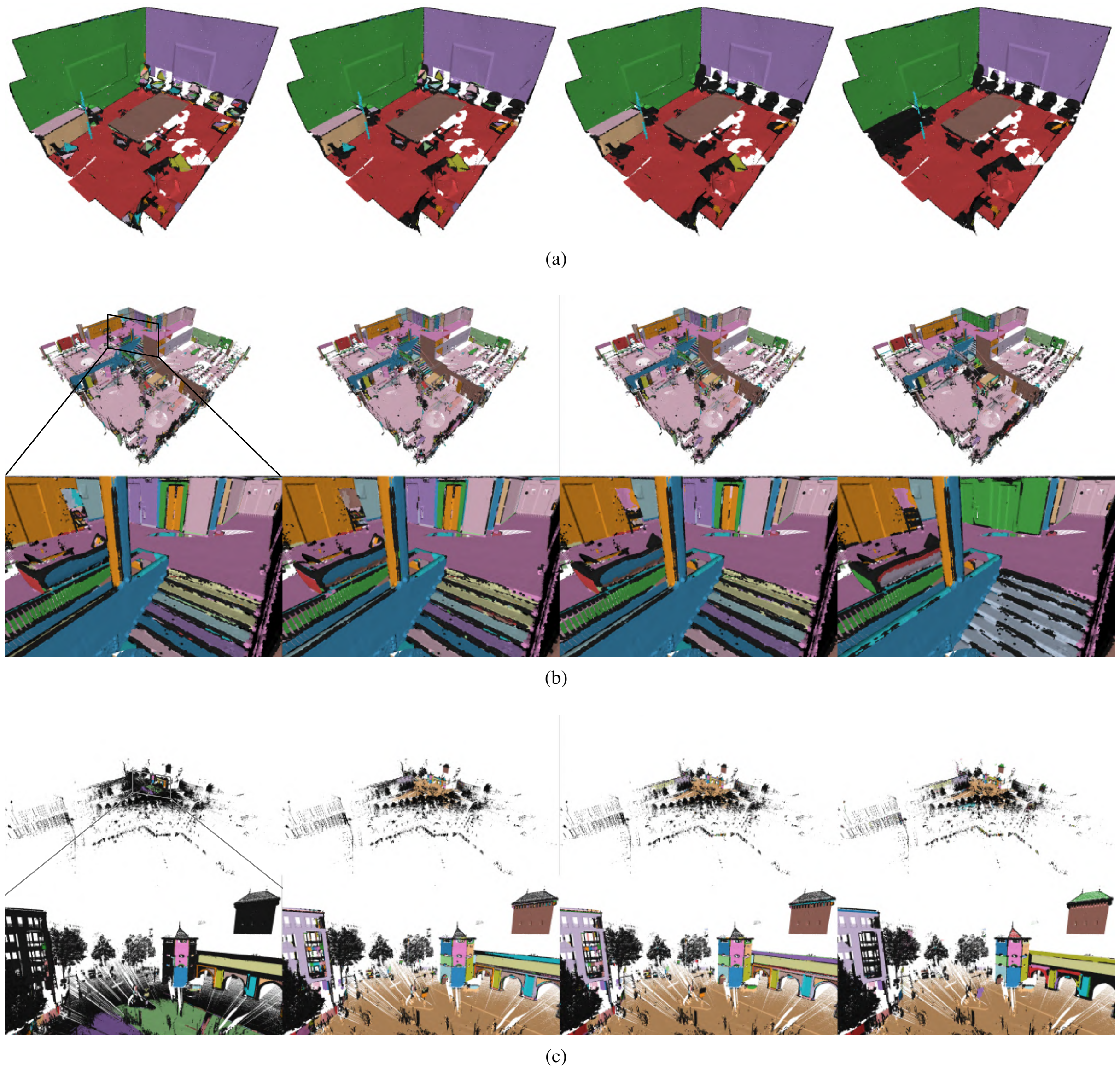
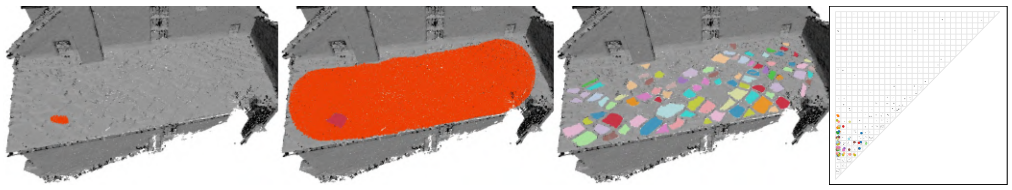
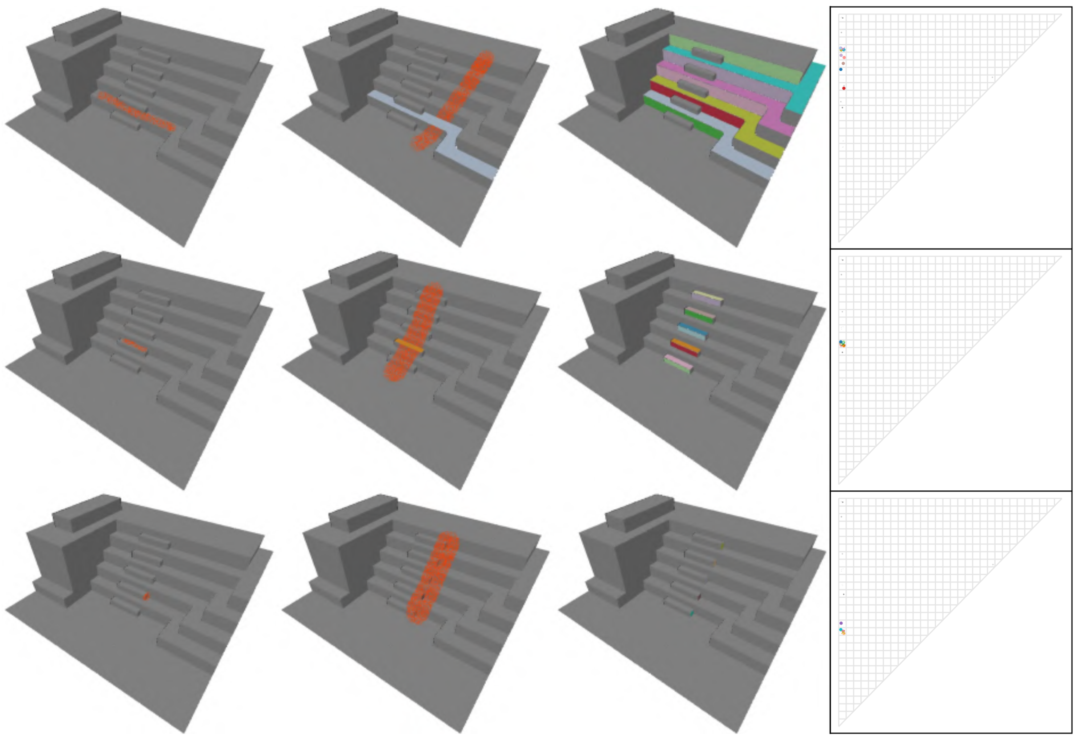


Figure 16: *Segmentation of models* `room` *(a),* `euler` *(b) and* `munich` *(c) based on the most persistent components that include a given scale (with increasing given scale from left to right) (Sec. 6.2).*


(a)


(b)

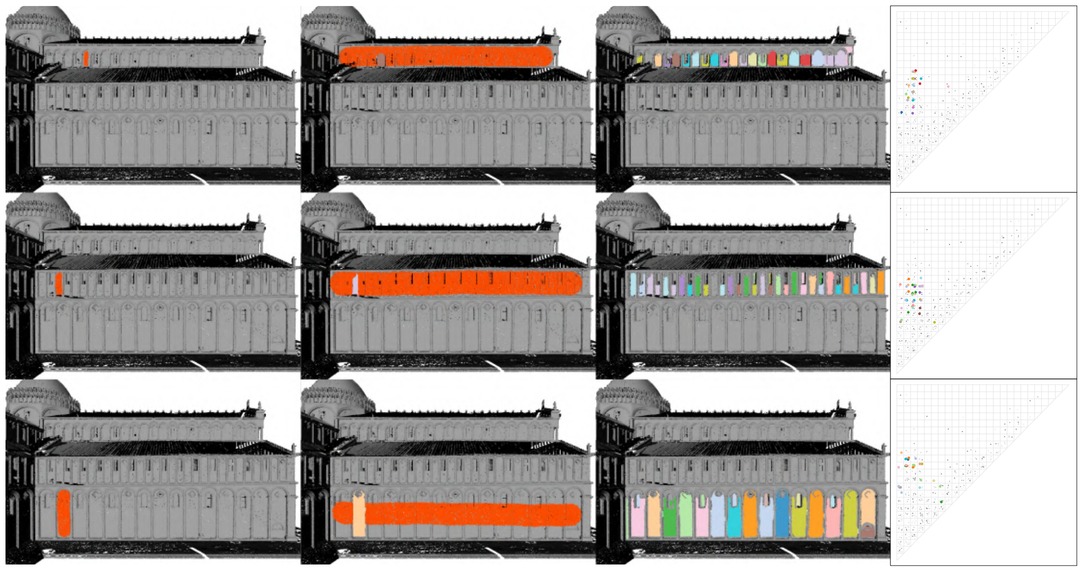

(c)

Figure 17: *Interactive similarity search tool on models* `lans` *(a),* `stairs` *(b) and* `pisa` *(c): with two simple sketches, the user selects a query part (left) and a larger domain part (middle) and the tool automatically extracts all the components of the domain that match the query. Corresponding diagrams highlight matching components in color.*