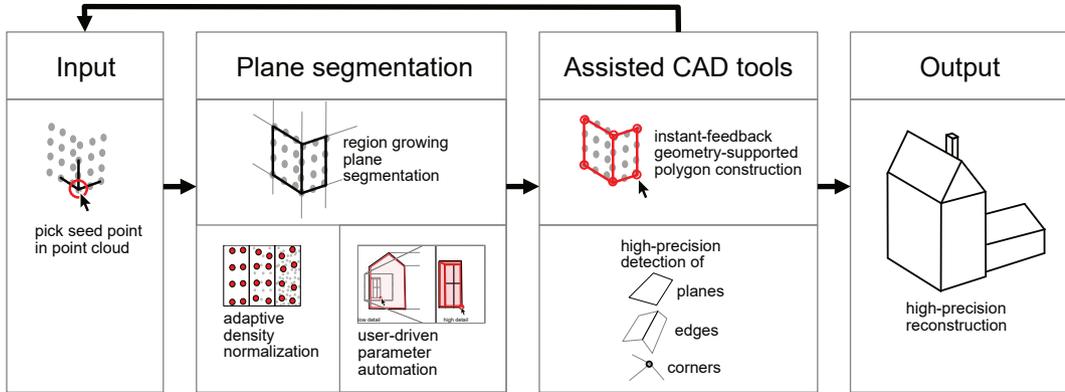


Graphical Abstract

Feature-assisted interactive geometry reconstruction in 3D point clouds using incremental region growing

Attila Szabo, Georg Haaser, Harald Steinlechner, Andreas Walch, Stefan Maierhofer, Thomas Ortner, Eduard Gröller



Highlights

Feature-assisted interactive geometry reconstruction in 3D point clouds using incremental region growing

Attila Szabo, Georg Haaser, Harald Steinlechner, Andreas Walch, Stefan Maierhofer, Thomas Ortner, Eduard Gröller

- Real-world point clouds show strong heterogeneity in size, density, and quality
- Fully automated geometry reconstruction almost always requires human intervention or quality control
- Human-in-the-loop approach avoids cumbersome filter-and-repair post-processing
- Guided technique effectively utilizes human intent to navigate difficult reconstruction scenarios

Feature-assisted interactive geometry reconstruction in 3D point clouds using incremental region growing

Attila Szabo^{a,*}, Georg Haaser^a, Harald Steinlechner^a, Andreas Walch^a, Stefan Maierhofer^a, Thomas Ortner^a and Eduard Gröller^b

^aVRVis Zentrum für Virtual Reality und Visualisierung Forschungs-GmbH, Donau-City-Straße 11, Vienna, 1220, Austria

^bTU Wien, Karlsplatz 13, Vienna, 1040, Austria

ARTICLE INFO

Keywords:

point clouds
interaction
segmentation
reconstruction

ABSTRACT

Reconstructing geometric shapes from point clouds is a common task that is often accomplished by experts manually modeling geometries in CAD-capable software. State-of-the-art workflows based on fully automatic geometry extraction are limited by point cloud density and memory constraints, and require pre- and post-processing by the user. In this work, we present a framework for interactive, user-driven, feature-assisted geometry reconstruction from arbitrarily sized point clouds. Based on seeded region-growing point cloud segmentation, the user interactively extracts planar pieces of geometry and utilizes contextual suggestions to point out plane surfaces, normal and tangential directions, and edges and corners. We implement a set of feature-assisted tools for high-precision modeling tasks in architecture and urban surveying scenarios, enabling instant-feedback interactive point cloud manipulation on large-scale data collected from real-world building interiors and facades. We evaluate our results through systematic measurement of the reconstruction accuracy, and interviews with domain experts who deploy our framework in a commercial setting and give both structured and subjective feedback.

1. Introduction

A major goal in the field of surveying and mapping is to create Computer Aided Design (CAD)-ready geometrical models that accurately describe the as-built conditions of buildings' inside and outside structures. It is important to represent walls, but also more intricate features, such as, roofs, window sills, or a flight of stairs, as shown in Figure 1.

As a means to this end, surveyors capture real-world buildings with terrestrial laser scanners producing 3D point clouds. Depending on the size of the building these typically range from tens of millions to billions of points. This high-detail representation is bulky and impractical for CAD workflows, construction documentation, and as-designed comparisons. To create abstracted, CAD-ready models, for instance an accurately measured 3D plan, surveyors strive to derive edges and corners from point clouds.

Several properties of real-world laser scans make feature derivation challenging: relevant edges and corners are not well captured due to either sampling constraints or scanning occlusions. Further, real-world laser scans contain unwanted points, captured from vegetation, furniture, appliances, or people walking through the scan (can be seen in Figure 1). Finally, high-precision scanning leads to large point clouds generally requiring an out-of-core approach.

Since a majority of human-made structures can be described as piece-wise planar objects, off-the-shelf tools typically allow users to globally fit plane primitives [25] to a point cloud, so the resulting planes may act as shape proxies to support reconstruction in orthographic views. Automatic algorithms often incur over- or undersegmentation due to global scope and parametrization. This requires prior cleaning and subsampling, as well as post-processing to exclude unwanted results and to manually construct missing results. Previous approaches typically do not go far enough in supporting the human operator in these tasks, which we aim to rectify in this work. Our system also focuses on buildings consisting of planar structures. More complex support shapes are considered future work, which is discussed in Section 6.

We propose a novel interaction-based framework that does not require any global pre-segmentation or point cleaning steps (Figures 2,3 left), while offering users a constructive approach to reconstructing corners and edges despite the presence of missing data and unwanted points (Figures 2,3 right). Our human-in-the-loop approach provides algorithmic assistance to the user through on-demand region-growing plane-segmentation with instant feedback enabled by an adaptive point cloud resolution scheme and robust edge and corner detection. Region growing as well as primitive fitting operate *locally* and are evaluated *incrementally*.

Our work is an extension of Steinlechner et al. [26]. There the authors fit plane segments to point clouds and support user interactions, for instance, to prevent pick-through when measuring distances. In this work we use a similar out-of-core-strategy. Our technique supports a full range of tools for reconstruction at any detail level, as seen in Figure 4. Existing interactive reconstruction-based tools require a global

 aszabo@vrvis.at (A. Szabo); haaser@vrvis.at (G. Haaser); steinlechner@vrvis.at (H. Steinlechner); walch@vrvis.at (A. Walch); maierhofer@vrvis.at (S. Maierhofer); inbox@ortner.fyi (T. Ortner); groeller@cg.tuwien.ac.at (E. Gröller)
 <http://ortner.fyi/> (T. Ortner)
ORCID(s): 0000-0002-1322-8704 (A. Szabo); 0000-0002-4567-7942 (A. Walch); 0000-0002-9373-6409 (T. Ortner)
 <https://twitter.com/AszaboKing> (A. Szabo)

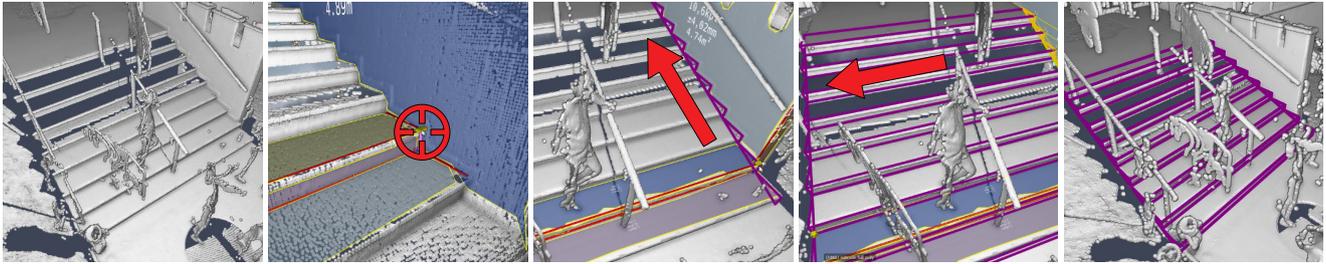


Figure 1: Reconstructing stairs in a real-world point cloud. From left to right, the human operator first zooms in on the stairs and reconstructs the side of a step in detail. Then they zoom out and copy the reconstructed geometry into where points are missing. Finally, they extrude the stair geometry along the tangential direction of the step’s face. The reconstruction of the staircase’s shape is then complete.

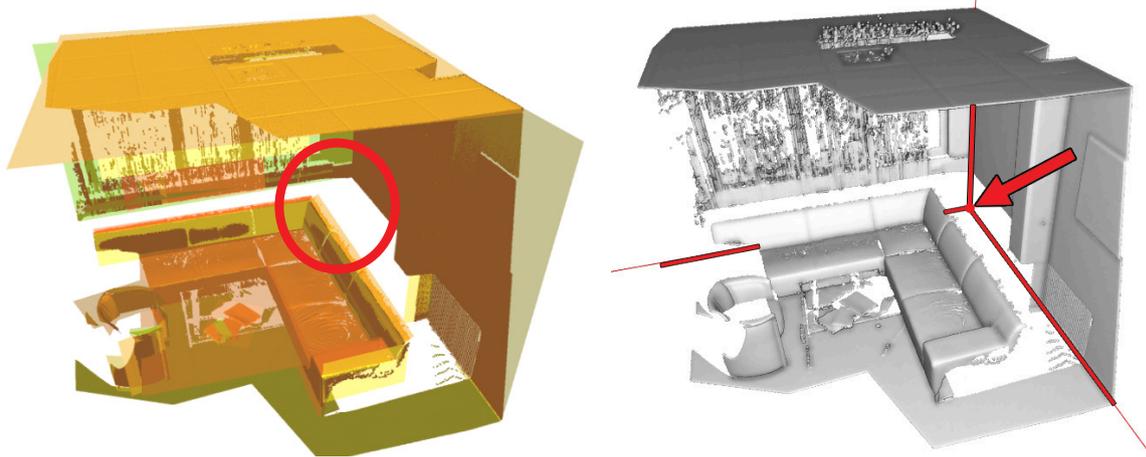


Figure 2: Left: Occlusions cause holes in the automatic segmentation (yellow). The desired corner is missing (red circle). Right: The corner of interest (red arrow) is at the intersection of the walls and the floor.

pre-segmentation of the point cloud [12], [2]. Our approach is orthogonal as all segmentation is local, on-demand, and user-driven with a tightly integrated interaction loop.

An overview of our system is shown in Figure 5.

1.1. Contributions

In summary, we present the following contributions:

- A robust on-the-fly density normalization scheme permitting level-of-detail reconstruction of arbitrary-sized out-of-core point clouds (Section 3.7),
- A novel point cloud interaction framework integrated with on-demand incremental plane segmentation (Sections 3.5), and
- A user-driven workflow for feature-assisted high-precision geometry reconstruction in non-preprocessed point clouds (Sections 4).

1.2. Structure of the Paper

Section 2 reviews recent works in the field of interactive point cloud geometry reconstruction. Sections 3 and 4 detail our incremental region growing and robust interaction loop, respectively. In Section 5 we evaluate our three contributions

with synthetic tests, example showcases, and real-world expert user interviews, respectively. Section 6 discusses strengths and shortcomings of our approach.

2. Related Work

We structure the body of related work into two sections, automatic methods (Section 2.1), and user-driven (Section 2.2) methods that ingest point clouds with real-world artifacts capturing piece-wise planar objects. For an extensive review of surface reconstruction from, and geometric primitives detection in, point clouds, we refer to Berger et al. [4] and Kaiser et al. [11], respectively.

2.1. Automatic Methods

Automatic methods have a longer history of research than user-driven ones. Common approaches attempt to reconstruct a plausible surface model, often planes, either through RANSAC [25] or region growing [22]. Both are available in off-the-shelf tools [23, 20]. RANSAC requires careful parameter tuning to achieve the desired level of detail representation and region growing is sensitive to initial conditions, noise, and outliers [11].

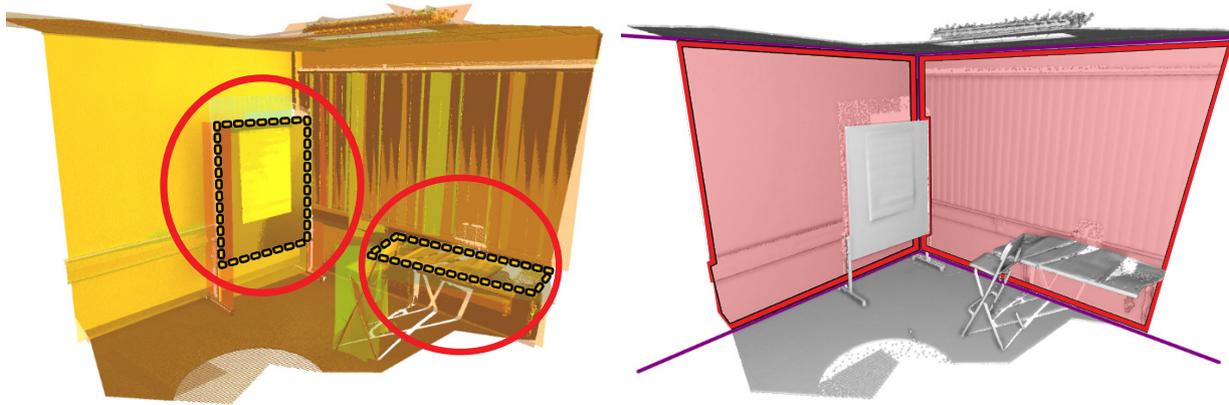


Figure 3: Left: Automatic segmentation (yellow) contains unwanted segments, e.g. whiteboard or desk (red circles). Right: Segments of interest are the two walls (red outlines).

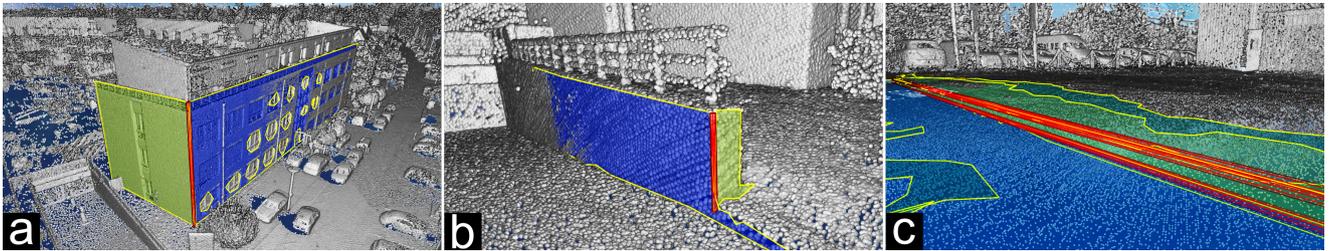


Figure 4: Interactive reconstruction at different degrees of detail. Cloud has 1 billion points. (a) large building ($\sim 20\text{m}$ long). (b) sidewalk structure ($\sim 1\text{m}$ high). (c) roadside curb ($\sim 0.15\text{m}$ high).

Tuning automatic shape detection represents a trade-off between missing important geometric detail versus receiving a large number of primitives, which requires manual cleanup. More advanced methods detect planes and then automatically prune or merge them according to different criteria, such as principal orientations, parallelism, orthogonality, or coplanarity [13, 16, 19, 18]. Since processing a large number of detected shapes is computationally intensive, Bauchet et al. [3] employ a kinetic polyhedral approach after shape detection that performs well for large amounts of measurement data.

2.1.1. Level of Detail

Level-of-detail-aware approaches employ structured metadata to cut down on operations or memory consumption. Real-time point-cloud renderers typically use octree-based data structures to subdivide space into cells of equal point density. They dynamically fill the limited graphics memory with a uniformly distributed subset of points to guarantee optimal image quality. Region-growing segmentation techniques in particular utilize spatial indexing schemes to group points, typically voxel grids or octree-equivalents. Deschaud et al. [6] decompose the point cloud into a voxel grid as a means of batch-processing the region growing algorithm and avoiding costly point-neighborhood searches. Vo et al. [27] use an octree based on point planarity rather than density.

They sidestep the need for point operations almost entirely by comparing representative plane segments instead. For our work we chose a region-growing algorithm equivalent to the implementation in the Point Cloud Library [23]. Extending it with our ad-hoc density normalization scheme does not require a specifically crafted data structure, thus enabling a minimal-effort implementation into existing point renderers.

More recently, Mercier et al. [15] present a level-of-detail-aware reconstruction technique for algebraic surfaces. It uses an octree to store intermediate surface computations and an adaptive traversal scheme to obtain appropriate resolutions for different regions. Their octree traversal is in principle comparable to our adaptive resolution selection (Section 3.7), and could be adapted for planar surfaces.

2.1.2. Feature Detection

2D feature detectors are long-standing image processing steps, including the Canny edge detector and Harris corner detector. They do not generalize straightforwardly, since point clouds exhibit highly irregular sampling patterns and scan artifacts, as compared to regular pixel grids, serving as motivation for much research.

Filter-based approaches exist in 3D, and overcome the aforementioned problems by applying rule-based constraints (e.g. merge close line segments, make lines parallel/orthogonal). Hackel et al. [9] use classifiers to assign points to contour

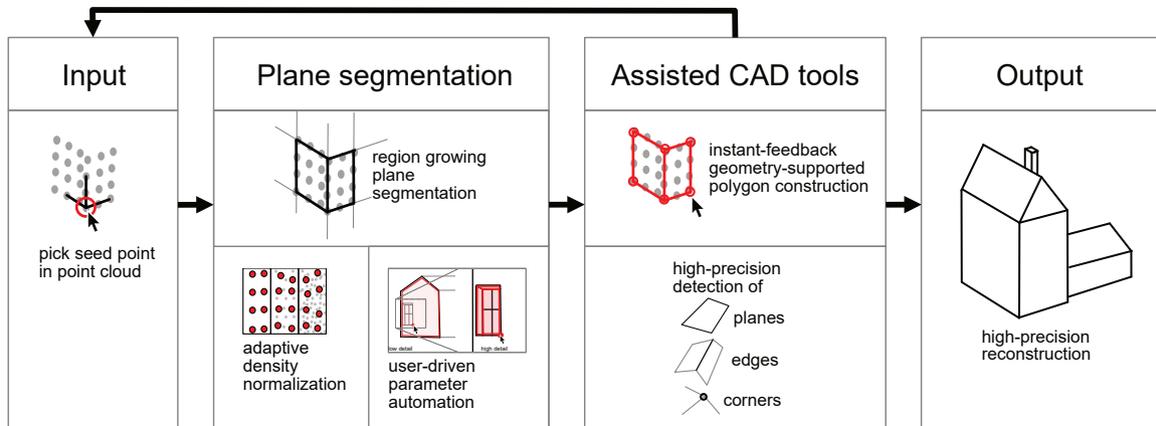


Figure 5: System overview. From left to right, the four labeled boxes represent the major components of our approach. *Input:* In a point cloud a seed point is picked by the human operator from a specific view. *Plane segmentation:* The point cloud is locally segmented into planar regions starting from the seed point. Segmentation parameters are automatically estimated from the user’s view. Our adaptive density normalization scheme ensures consistent convergence across regions of too low or too high density. *Assisted CAD tools:* High-precision planes, edges and corners are synthesized from planar regions and previously-reconstructed geometries, which assist the user in drawing the real shape represented by the point cloud. These support geometries are available to the user instantly after picking the seed point, and progressively get more refined in subsequent iterations. *Output:* Repeated reconstruction of different parts of the point cloud yields the desired final high-precision geometry.

lines and select consistent contour sets as plausible complete outlines of structures. More recently, Wang et al. [28] use deep learning to infer the parameters of contour curves from classified points. Himeur et al. [10] further distinguish edge types (hard/smooth). They provide an interactive element for user-specified edge definitions, being able to handle compromised real-world point clouds. We, in comparison, specifically understand edges as being delineated by large planes, which are commonly used in building planning. Even more sophisticated feature extraction techniques could be used in our system if required by the application.

In contrast to *global* optimization methods, we propose an efficient *local* technique for reconstructing planes, edges, and corners on-demand in a limited spatial region around a user-picked seed point. Our workflow’s constructive nature avoids any prior processing (beyond octree construction), removal of artifacts, time-consuming parameter tuning, or manual clean-up of segmentation results. Furthermore, the user is given a set of tools to deal with noise, missing points, and unwanted portions of a scan based on human decision-making and contextual awareness. Since our interactive approach is tightly coupled with point rendering, the density-based octree re-uses existing infrastructure. Limiting reconstruction space makes the process independent of point cloud size while retaining full-detail accuracy.

2.2. User-driven Methods

Commercial point processing tools, such as ArcGIS or AutoCAD, include user-driven surface reconstruction tailored to the domain of CAD. Workflows consist of a non-interactive and an interactive part, where a global algorithm first finds dominant planes in the subsampled point cloud automatically. Subsequently, the user iterates through orthographic projections on planes of interest. Polygonal

geometry is built in mixed 2D and 3D views using classic construction tools. In comparison, our approach lets the user manipulate the point cloud directly. This avoids the loss of locality caused by multiple views and leverages contextual awareness and semantic information provided by the three dimensional viewpoint.

Chen and Chen [5] recover planar segments and their connectivity through edges and corners automatically. They assist the user in repairing holes through topological reasoning on regularity and symmetry of the polygons. This approach combines multiple global processing steps, with user interaction at the end to complete the result. Our proposed on-demand reconstruction does not require global pre-segmentation and incorporates user interaction directly into the segmentation algorithm.

Many user-driven methods seek a balance between the level of intuition and user feedback that is useful for the reconstruction application. Successful methods tend to tightly integrate the reconstruction algorithm with the form of user interaction [4].

The technique by Nan et al. [17] allows the user to approximately draw shape primitives comprised of axis-aligned boxes. These are compounded and repeated to form complex shapes, which are then iteratively fit to the point cloud. ‘Smartboxes’ is geared towards the reconstruction of large urban environments exploiting regularities and symmetries.

With O-Snap, Arıkan et al. [2] propose a shape-assisted interactive reconstruction system. The user’s construction tools use best-fit plane segments recovered from the point cloud in a preprocessing step as work area. In a background optimization process, the user’s corrections and constructions are constantly refitted to the point cloud. Edges

and corners of polygons are snapped together if they are close enough. O-Snap reconstructs best-fit results based on a global optimization process. In contrast, our technique fundamentally aims at high-precision reconstruction of local point cloud features based on measured data. However, in principle, O-Snap’s continuous polygon optimization loop could be integrated into our approach if required by the application.

Similar to our method, Steinlechner et al. [26] use shape detection on level-of-detail point data to locally extract support geometry. They apply a RANSAC-based method on the rendering data structure. In their approach, segmented planes only assist selection-related point interactions such as picking, brushing, or measurements.

Lejembre et al. [12] find all stable planar features and relationship information through multi-scale analysis of the point cloud. They give the user the ability to query for desired plane segments through brushing gestures. Lejembre et al. provide an integrated reconstruction workflow similar to ours. Conceivably, their global-segmentation based approach could be placed in the same system as our local ad-hoc reconstruction, combining their powerful segment browsing capabilities with our segment-assisted construction tooling.

Following the categorization provided by Berger et al. [4], our approach does not require data-priors, apart from the construction of an octree data structure that most renderers also use for level-of-detail rendering. Our approach applies to *shape classes* of buildings indoor and outdoor, robustly deals with the *artifacts* of *missing data*, *outliers*, and *non-uniform sampling*, and outputs corners and edges. The artifact of *misalignment* does not play a role in survey quality laser scans, where precise manual alignment is commonly performed beforehand.

3. Incremental Region-Growing Point-Cloud Segmentation

In this section we present a segmentation technique based on incremental seeded planar region growing. It works on a density-homogenized representation of a point cloud achieved through an octree-based adaptive resolution scheme. Automatic parameter estimation, allows for intuitive user-driven control of the algorithm. A graphical overview is shown in Figure 5.

The notation (\cdot, \cdot, \cdot) denotes a tuple of values. The boldfaced variable name $\mathbf{p} = (p_x, p_y, p_z)$ stands for the three dimensional point \mathbf{p} and its three components p_x , p_y and p_z .

3.1. Spatial Subdivision Scheme

Octrees, or equivalent spatial subdivision data structures, are commonly used for level of detail rendering and point cloud editing [24]. In our out-of-core implementation, the leaf cells store references to point data, while the inner cells store references to subsampled representations of the contained cells. Apart from creating the octree, no further preprocessing on the point cloud is needed. We use a simple

cell indexing scheme similar to Yoder et al. [29] to find adjacent cells and traverse cell neighborhoods.

3.2. Incremental Plane Regression

A prerequisite for our region growing method is the availability of an *plane regression*. It is later used to define plane segments and incrementally updated given a user-defined seed point (Section 3.4).

Let $R = (\mathbf{S}, \mathbf{Sq}, c, \mathbf{D})$ be an *incremental plane regression* (Equation 1),

$$\begin{aligned} \mathbf{S} &= \sum_{i=0}^c p_i \\ \mathbf{Sq} &= \sum_{i=0}^c (p_{x_i}^2, p_{y_i}^2, p_{z_i}^2) \\ \mathbf{D} &= \sum_{i=0}^c (p_{y_i} * p_{z_i}, p_{x_i} * p_{z_i}, p_{x_i} * p_{y_i}) \end{aligned} \quad (1)$$

where \mathbf{S} is the sum of point coordinates, \mathbf{Sq} is the sum of squared point coordinates, c is the point count and \mathbf{D} is the vector of sums of products between different pairs of point coordinates. Let $\mathbf{p} = (p_x, p_y, p_z)$ be a point, and $R' = (\mathbf{S}', \mathbf{Sq}', c', \mathbf{D}')$ be the *incrementally updated* planar regression produced by adding \mathbf{p} to R . Then R' is calculated as (Equation 2):

$$\begin{aligned} \mathbf{S}' &= \mathbf{S} + \mathbf{p} \\ \mathbf{Sq}' &= \mathbf{Sq} + (p_x^2, p_y^2, p_z^2) \\ c' &= c + 1 \\ \mathbf{D}' &= \mathbf{D} + (p_y * p_z, p_x * p_z, p_x * p_y) \end{aligned} \quad (2)$$

Given a plane regression $R = (\mathbf{S}, \mathbf{Sq}, c, \mathbf{D})$, we compute the associated *covariance matrix* C . It represents the distribution of points in space, and later yields the parameters of the best-fit plane. Using the identity $\bar{\mathbf{p}} = \mathbf{S}/c$, the equation is (Equation 3):

$$C = \begin{bmatrix} Sq_x - \bar{p}_x * S_x & D_z - \bar{p}_x * S_y & D_y - \bar{p}_x * S_z \\ D_z - \bar{p}_x * S_y & Sq_y - \bar{p}_y * S_y & D_x - \bar{p}_y * S_z \\ D_y - \bar{p}_x * S_z & D_x - \bar{p}_y * S_z & Sq_z - \bar{p}_z * S_z \end{bmatrix} * \frac{1}{c-1} \quad (3)$$

The incremental formulation permits calculating the covariance matrix without having to recalculate the point centroid after every point addition. The actual set of included points does not need to be maintained and the memory and calculation overhead for adding a point to the regression is always constant.

Although our formulation is mathematically correct, real-world point clouds can exhibit very large point coordinates. These may lead to numerical problems, mainly due to the involved sum of squares. We shift all points by the coordinates of the first added point, such that the first point has the relative coordinate $(0, 0, 0)$. The covariance matrix

can be *rebased* accordingly after its computation. For this optional improvement, we use an equivalent implementation as published in Ponca [14]. Our formulas can be found in Appendix A.

The evaluation in Section 5.1 shows that this improvement allows us to achieve an accuracy similar to standard approaches on real-world data sets, while maintaining the single-pass character of the approach.

3.3. Plane Synthesis

Given the covariance matrix C , we synthesize the associated regression plane using the *eigendecomposition* of C into its principal components, i.e. finding the roots of the *characteristic polynomial*

$$\det(C - \lambda * I) = 0$$

to obtain the *eigenvalues* $\lambda_i, i \in \{0, 1, 2\}$, sorted by descending magnitude. Solving the specific eigenvalue equation

$$(C - \lambda * I) * \mathbf{v} = 0$$

yield the three associated *eigenvectors* $\mathbf{v}_i, i \in \{0, 1, 2\}, \mathbf{v}_i \neq \mathbf{0}$.

We obtain the unsigned plane normal as the eigenvector \mathbf{v}_2 associated with the smallest eigenvalue λ_2 , tangential and bitangential directions as \mathbf{v}_0 and \mathbf{v}_1 , and the plane origin as the last added point.

3.4. Initial Seed

An incremental plane regression R_0 is initialized from a seed point \mathbf{p}_0 plus neighboring points. The seed point is chosen by the human operator via point picking. In case of an edge or corner seed, two or three plane regressions are initialized for the participating planes.

In order to find the participating planes, we place a spherical *seed region* of radius r_s around \mathbf{p}_0 . Within the seed region, seed points are chosen from the point cloud at a specific resolution, corresponding to octree cells at a certain level. We suggest an adaptive resolution technique, which is further described in Section 3.7. It ensures an appropriate subsampling rate for constant point density, even in the presence of severe anisotropy (e.g., floor around scanner). The absolute subsampling rate, i.e., the *desired point density* d , is chosen with our automatic parameter estimation technique, further described in Section 3.8. We apply RANSAC plane fitting [25] to find the initial planes, ranking them by inlier counts as stability measure. A plane regression is initialized for each plane. Optionally we discard the least stable planes in case of noisy scans using a variance-minimizing threshold [21]. The combination of these systems gives control over the degree of detail, i.e., the size and complexity of the desired reconstruction results. This is independent of heterogeneity in point density or actual complexity of the scanned region.

3.5. An Interactive Region Growing Loop

A point's distance to a regression is its minimal Euclidean distance to the regression's associated plane. The decision to add a newly encountered point \mathbf{p} to a regression R is made if the point's distance $distance(R, \mathbf{p})$ to the

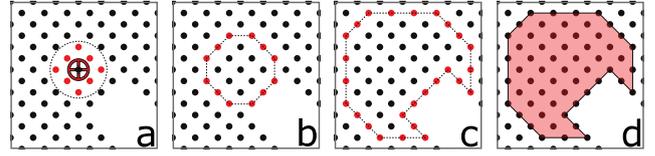


Figure 6: Interactive region growing loop. Steps (a) through (d) happen in sequence.

regression is less than the *plane threshold* t_p . If true, we call the point an *inlier point*. t_p represents the maximum acceptable point distance and controls the precision of the reconstruction. We refer to a regression together with its inlier points as a *segment*.

Given initial seed regressions, the core reconstruction loop essentially consists of repeatedly adding neighboring points to the regressions if they lie within the plane threshold t_p . We thereby gradually expand outwards the set of *border points* (inlier points that have unvisited points in their neighborhood). After the current octree cell is exhausted, and if any points were added to a regression, we communicate the current result using a *progress callback* (Section 3.6). Then repeat the process for all bordering cells. In summary, the region growing algorithm is as follows:

1. With increasing distance to the seed point, find unvisited point \mathbf{p} closer to a regression R 's border than the *search radius* r , and, if $distance(R, \mathbf{p}) < t_p$, $update(R, \mathbf{p})$. Repeat until all candidate points in the current cell have been visited. Update the average neighbor distance d_{avg} with the actual distance between \mathbf{p} and R 's border.
2. Run the progress callback.
3. If any point was added to any regression, repeat steps 1 through 3 for this cell's neighboring cells, with increasing distance to the seed point. Optionally, if d_{avg} differs considerably from the desired density d , adjust the levels of neighboring cells according to our adaptive resolution scheme (Section 3.7).
4. The algorithm terminates after all candidate cells have been visited. The first few iterations of the algorithm are illustrated in Figure 6.

3.6. Progress Callback

Leveraging incrementality, i.e. every intermediate result is also a valid final result, we return the current regression plane and *regression polygon*, i.e. a flat polygon showing the outline of the border points projected onto the regression plane. We use a space carving technique equivalent to Alpha Shapes [7] to triangulate the regression polygon. We also preserve some useful per-segment statistics from the algorithm as output: point count, polygon area, and the *regression variance* λ_2 . λ_2 was calculated in Section 3.3, and represents the mean squared error of the regression inliers along the plane's normal direction.

If multiple regression planes exist in the current plane segmentation, we facilitate additional user interactions by finding stable edges and corners (Section 4.1).

3.7. Adaptive Resolution

Point clouds are typically captured at different densities, depending on environmental factors such as size, distance to the scanner, and so on. Variations in point density have significant implications on performance and termination behavior of segmentation algorithms. Parameters like the point search radius r lose their meaning across regions with different densities. Globally downsampling the point cloud to a constant density is undesirable for reconstructing detailed structures, such as staircases, whose shapes would be lost. We propose an adaptive resolution approach. It dynamically adjusts cell levels during the region growing process, based on the observed point density d_{avg} and the desired point density d .

Octree cells correspond to regions in space, and *parent* cells contain downsampled versions of its eight contained *children* cells. The region growing algorithm places cells in a processing queue, in order of distance to the seed point. If d_{avg} significantly differs from d , these cells are replaced in the queue by cells of higher or lower level, representing lower or higher sampling rates and point densities, respectively.

A cell can simply be replaced by its children since they describe the same three dimensional space. Placing a cell's parent in the queue, however, is more involved. The parent comprises regions that have potentially already been processed, and should not be visited a second time. To avoid discarding information, we lock the current level. We completely finish processing the cells inside partially visited cells, and delay the transition to the parent level until this is complete. Our per-cell scheme is not optimal in speed since the reconstruction happens at a too fine detail level during the delay period. An alternative would incrementally remove already-processed areas from the regression and reconstruct them anew on a coarser level. The benefits of this additional optimization were negligible for our use case.

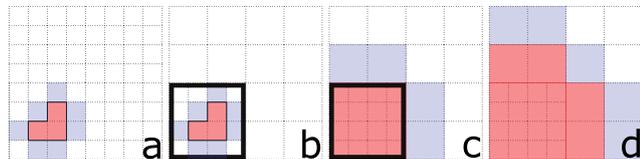


Figure 7: Adaptive resolution switch to a lower cell level. Steps (a) through (d) happen in sequence.

In summary, the adaptive resolution algorithm consists of two cases:

Increase point density, in case $d_{avg} \ll d$. Remove the current cell from the queue and replace it with its eight children. The cell level is now increased by 1. Continue processing the queue as usual.

Decrease point density, in case $d_{avg} \gg d$. Identify all cells inside the parents of completed cells or parents of cells in the queue. *Mark* those cells. Continue processing the queue only for cells that are marked, until all marked cells have been visited. For all remaining cells in the queue, if $d_{avg} \gg d$ is still true, replace them with their parents. The

cell level is now decreased by 1. Continue processing the queue as usual. This second case is illustrated in Figure 7.

The thresholds for the level switch comparisons are free parameters. We used $d_{avg} < 0.5 * d$ for density increase, and $d_{avg} > 4.0 * d$ for density decrease, which worked reasonably well in our test setups. Our choice is based on the observation that planes are roughly split into four equal parts in the octree level transition. In the case of density increase, we err on the side of caution. A too-fine point sampling is preferable to a too-coarse one in terms of accuracy. The introduced sampling errors are further analyzed in Section 5.1.

3.8. Automatic Parameter Estimation

Our incremental region growing point cloud segmentation depends on the following parameters. We derive *intuitive controls* for the parameters, primarily based on the concept of moving the cursor and the camera closer to the site of reconstruction. This automatically increases the degree of reconstruction detail:

- \mathbf{p}_0 : seed point. \mathbf{p}_0 is chosen by picking a point from the point cloud using the cursor.
- r_s : seed radius in $[m]$. r_s is approximately 10% of the screen width at the depth of \mathbf{p}_0 .
- d : desired point density (degree of detail) for the plane segmentation. Measured in $[\text{points per } m^3]$. d is proportional to the projected size of a pixel around \mathbf{p}_0 . Experimental values are between 10000 (closest to camera) and 100 (farthest away from camera), with a logarithmic falloff.
- r : point neighbor search radius in $[m]$. r is twice the estimated average point distance, depending on the point cloud's overall scale.
- t_p : plane distance threshold measured in $[m]$. t_p remains as a free user parameter representing the desired precision of the plane regression.

Our objective is to provide human operators with direct and intuitive ways of controlling the algorithm. In particular, the initial seed selection (Section 3.4) heavily determines the reconstruction and can be chosen to fit a user's particular needs. By moving the camera and zooming in, the users decide what constitutes a "good" or "bad" initialization. They are guaranteed to obtain the most salient planar features at the degree of detail they are currently viewing the point cloud at. Figure 8 gives an example.

4. Assisted Geometry Construction Tools

In this section we present our methodology of robust edge and corner reconstruction (Section 4.1) and assemble feature-assisted modeling tools for interactive geometry reconstruction (Section 4.2).



Figure 8: Zoom out (left) to obtain a planar regression of the entire house wall. Zoom in (right) to capture details in a local part of the wall.

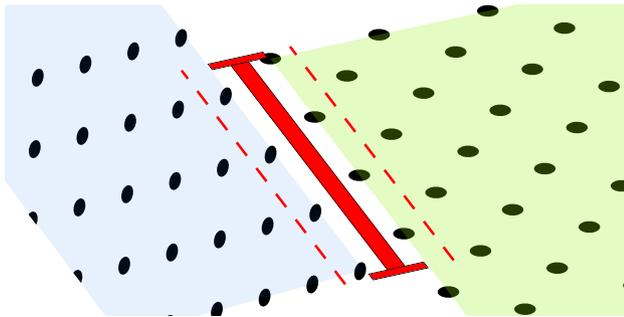


Figure 9: Stable edge support (red) between two plane segments (green and blue) where both segments have inlier border points (inside dotted lines).

4.1. Edge and Corner Reconstruction

We automatically reconstruct the *maximum likelihood* edges and corners as intersections between regressions. Given a region growing result, all two- and threefold selections of regressions are intersected if:

- the regression planes are not parallel,
- the regression polygons have non-vanishing areas, and
- after intersecting the regression planes, all regression polygons have vertices in proximity to the intersection line/point.

The resulting set of intersections has elements of two categories, which receive stability measures called *supports*:

Corners. The intersection between three planes results in a corner. The viable corner’s support is comprised of the three regression variances.

Edges. The intersection between two planes results in an edge. For a viable edge, the participating segments have border points closer than $2 * r$ to the intersection line. Projecting these border points onto the line gives a 1D distribution. The 95th percentile of it is the *support range* and produces the stable start- and endpoints of the line segment describing the edge (illustrated in Figure 9).

4.2. Feature-Assisted Modeling Tools

Based on our experience with experts and novice users, we identify two principal modes of shape assistance for reconstruction workflows: *Cursor snapping* and *direction finding*.

Cursor Snapping enables the human to point out dominant structures: maximum likelihood edges and corners (Figure 10(a)).

Direction finding allows the user to select directions in three dimensional space. The human operator chooses a direction by simply pointing at a plane to select the plane normal. Alternatively, pointing at an edge or a corner selects one of the planes’ tangents through subtle pointer movement. This interaction is shown in Figure 10(b) and 10(c). We aim at offering easy-to-understand tools for the user by only adhering to simple pointing-based selection techniques.

Leveraging the described support mechanisms, we implement an essential set of *feature-assisted CAD tools* (shown in Figure 11):

- *Free polygon construction* assisted by pointer snapping (Figure 11(a)).
- *Moving and copying polygons* assisted by pointer snapping and optional direction constraining (Figure 11(b)).
- *Polygon extrusion* constrained along a fixed direction, further assisted by polygon snapping (Figure 11(c)).

5. Evaluation

We assess our technique in three different ways by determining the:

- *reconstruction accuracy* of our out-of-core incremental region growing algorithm with synthetic test cases,
- *effectiveness* of our workflow and interactions with example use cases and time measurements, and
- *applicability in real-world scenarios* through interviews with surveying experts based on a version of our framework implemented in their application.

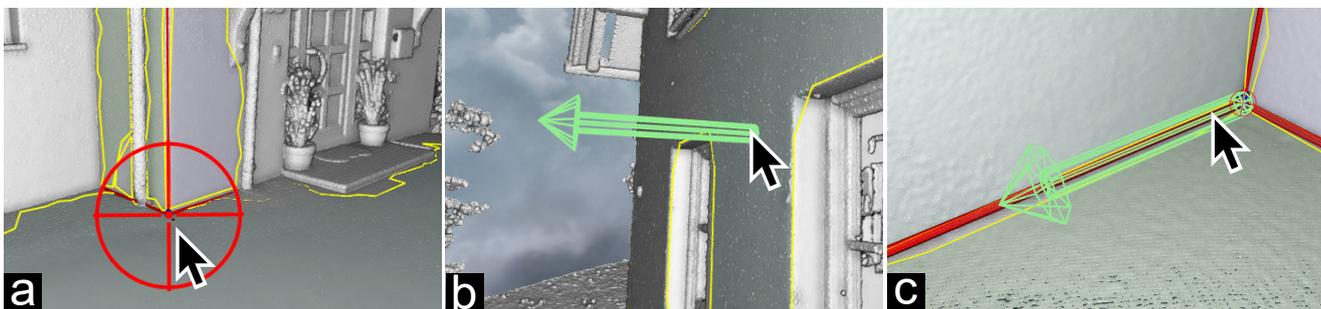


Figure 10: Shape-assisted interactions. (a) corner snapping. (b) selecting plane normal. (c) selecting plane tangent.

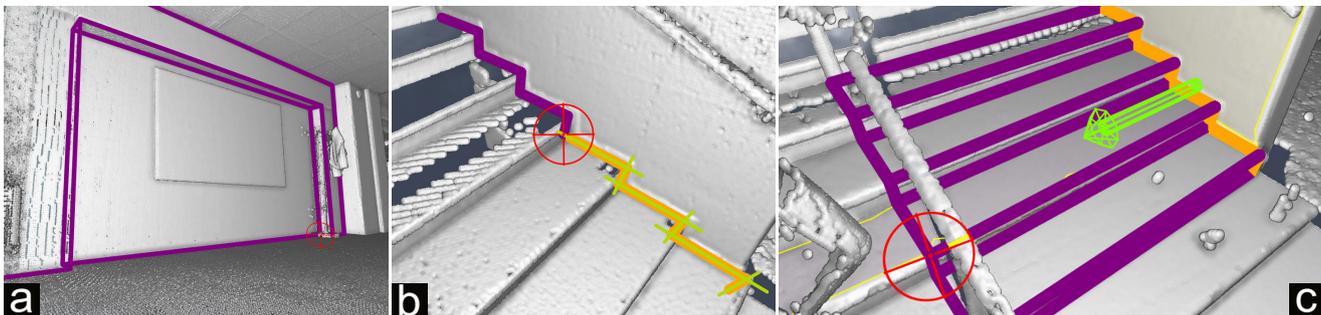


Figure 11: Plane-assisted CAD tools. (a) polygon construction. (b) polygon movement. (c) polygon extrusion.

The performance of the region growing method is crucial for interactivity. While our incremental region growing formulation (Section 3.2) is independent of the point cloud size, interactions depend on the chosen level of subsampling (Section 5.1). On consumer-level hardware (Intel i5-4690K, Nvidia GTX 1080) the extraction of geometrical features and region growing typically took between 0.1 and 1 seconds depending on plane size and never slowed down interaction, especially since the first results appear immediately and refinement happens until full convergence.

5.1. Reconstruction Accuracy

We evaluate two kinds of reconstruction accuracies for our proposed technique: The *absolute reconstruction error* measured as a function of input noise to the precision of reconstructed features, and the *effect of subsampling* as the loss of precision if lower point densities are chosen.

The absolute reconstruction error (as Root Mean Square Error of inlier points to their regression planes) is evaluated in comparison to the well-established robust RANSAC plane detection algorithm by Schnabel et al. [25]. We chose this particular algorithm as baseline because our collaboration partners work with it regularly and deem it sufficiently accurate for real-world scenarios. Implementations are freely included in open-source point cloud processing tools such as CloudCompare [1]. We create a synthetic scenario with known input noise, and process the input by both algorithms in a controlled fashion. Our synthetic scenario is generated using a simulated laser scanner that is placed inside a room-sized box. It scans its surroundings in regular patterns, introducing depth measurement errors ranging between one

millimeter and a few centimeters. The measurement error is simulated in a realistic way, i.e., the virtual laser scanner produces angular and depth measurement errors similar to a real one, leading to realistic anisotropy and error-to-distance behaviors. All tests produce between 10^5 and 10^6 points, which is small enough to fit into the core on typical hardware, and are repeated at least 200 times. Pre-defined user inputs are simulated to reconstruct the corners of the synthetic room. The accuracy of them is then compared to that of the RANSAC plane detection and the subsequent recovery of the same corners.

The evaluation results are illustrated in Figure 12. The graph clearly shows that our incremental region growing algorithm is roughly in line with the established technique. Both approaches produce reconstructions approximately an order of magnitude more precise to the ground truth than the scale of the point cloud noise. This evaluation shows that our incremental reconstruction approach suffers no compromise in accuracy if applied to the base case (in-core point cloud and no user input).

The use of subsampling in our level-of-detail region growing leads to a certain loss of precision. To evaluate this in a real-world context, we chose a high-resolution (1 mm - 2 mm point distance) laser scan of a building interior and selected a well-captured corner. As opposed to the previous analysis, we now recover a feature of a scan with a known, fixed noise level at different subsampling rates. We repeatedly reconstructed the corner with our plane segmentation at different point densities (segmentation parameter d). This measures the impact of our adaptive resolution scheme on the reconstruction accuracy. The point densities

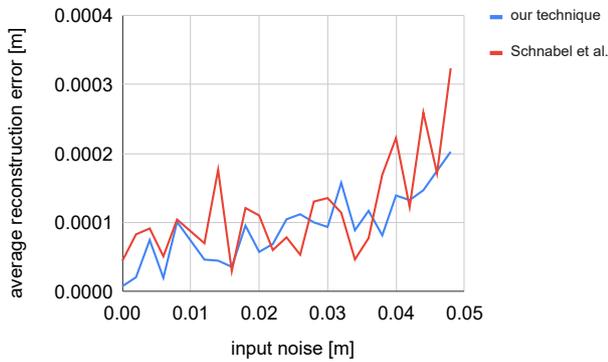


Figure 12: Input noise vs. reconstruction error at maximum detail. Lower is better. Our proposed system in blue, RANSAC plane detection by Schnabel et al. [25] in red. Synthetic data set.

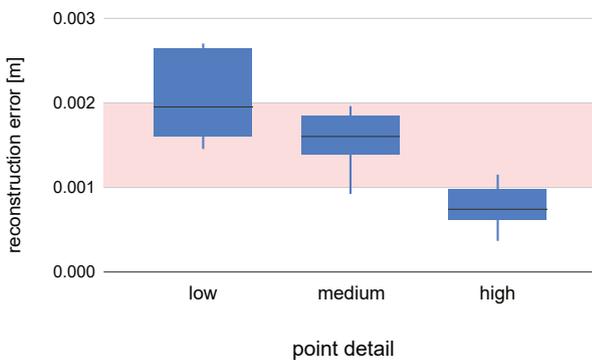


Figure 13: Degree of point detail vs. reconstruction error. Lower is better. Red area shows the approximate scanner noise. Our detail levels are: *low* (>2cm resolution), *medium* (1mm-2cm), and *high* (<1mm). Real-world data set, dimensions: approx. 7m × 5m × 2.5m.

are categorized into *low detail* cases with more than 2 cm between points, *medium detail* cases ranging from 1 mm to 2 cm between points, and *high detail* with less than 1 mm between scan points. The values correspond to detail levels commonly referenced in the domain of urban surveying.

Figure 13 shows the results. The input scanner noise is highlighted as red area and lies between 1 mm and 2 mm. In the low detail reconstruction, the output error lies between 1.5 mm and 3 mm. In the medium detail reconstruction, the output error is roughly equal to the input noise. In the high detail reconstruction, the output error lies below the input noise, at approximately 0.5 mm to 1 mm. These results show that the loss of precision introduced by subsampling in our technique stays within tolerable bounds. The domain experts found these accuracy ranges appropriate for typical reconstruction workflows that involve multiple levels of detail.

5.2. Example Use Cases

To demonstrate the capabilities of our technique, we reconstructed example scenarios from real-world use cases. The examples' scopes are similar to that of common tasks in the domain of as-built surveying. The point clouds were supplied by our collaboration partners and were not cleaned or downsampled beforehand. These data sets range from approx. 10 million to over 1 billion (and, in real applications, up to 10+ billion) points.

To put this number into perspective, we found it only possible to load up to 50 million points into comparable interactive systems like Arikan et al. [2] and Lejemle et al. [12]. Beyond 50 million points, memory and runtime constraints of global pre-processing proved prohibitive. Our local technique, with adaptive subsampling, keeps memory and runtime proportional to the size and detail of the reconstruction, rather than the size of the point cloud.

All tests were done on average consumer hardware, i.e., Intel i5-4690K, Nvidia GTX 1080.

The first scenario, 'office interior', was reconstructed by an expert user. They are an industrial provider of surveying services, with more than five years of experience and familiar with the traditional reconstruction workflow. The expert is not affiliated with this paper. The expert reconstructed the inside layout of an office building, using our demonstrator GUI. It uses our interactive technique. They compared it with an expert tool that is being employed by our collaboration partners and has limited interaction assistance. The expert tool uses RANSAC-based global plane segmentation and allows the user to select and intersect planes to find edges. The user ended up reconstructing approximately 92 polygons. The result is visible in Figure 14(c). It took them approximately 20 minutes with our technique. Using the comparison tool with limited shape assistance, they required approximately 105 minutes. The main reason for this longer time is that the user had to run the global RANSAC segmentation repeatedly with different parameters. They were unable to find one set of parameters that was appropriate for the dense parts as well as the sparse parts of the point cloud. The user did not have this problem with our technique, since our instant feedback interaction allowed them to find locally appropriate segmentation parameters on the fly.

The remaining example reconstruction scenarios are shown in Table 1, with screenshots in Figure 14 and descriptions in the caption. Scenarios were selected to be representative of various real-world architectural disciplines: building layouts, building facade reconstruction, outdoor and indoor details, and varying sampling rates and occlusions/noise. The selection of three plane segments to identify the corner point at their intersection has been an interaction that was used particularly often, especially in cases of scanning shadows due to occlusion. Whenever the users encountered a hole in the point cloud, they usually decided to extrapolate surrounding plane segments, or guess the missed shape from a different, symmetrical part of the building. This showcases the value of human understanding

scenario	#points	polygons	time taken
spiral staircase (14a)	~15 million	45	~15 min.
house facade (14b)	~50 million	49	~12 min.
office interior (14c)	~150 million	92	~20 min.
factory layout (14d)	~1 billion	22	~8 min.

Table 1

Example scenarios reconstructed with our technique. Screenshots are shown in Figure 14.

and contextual awareness to infer shapes from missing point data.

The reconstruction example scenarios resulted in approximately 20-40 polygons and were completed in a matter of minutes. Users subjectively estimated they would need more than an hour for similar reconstructions without shape assistance. They could navigate towards their reconstruction targets, and then invoke our on-demand plane segmentation. They quickly found reconstruction parameters appropriate for the current local region and reconstruction goal, regardless of the point cloud size and complexity.

5.3. Expert Interviews

We conducted interviews with users who have varying degrees of experience: two domain experts who work with point clouds professionally, three intermediate users who had some point cloud experience, and two novice users who had no prior experience in this domain. The two domain experts work in terrestrial and building surveying, with more than three years of experience. The intermediate users apply point cloud tools regularly (at least two years of experience), but are not familiar with high-precision reconstruction. The novice users have no professional experience with point clouds or reconstruction. None of the respondents are co-authors of this paper. We asked them to complete minimal reconstruction tasks, answer usability questions, and give their subjective opinions on our technique.

The domain experts are accustomed to a workflow that involves manually drawing a polygon on an orthographic projection of a part of the point cloud. This is supported by coarse global pre-segmentation and direct point picking. Common usability issues of this workflow stem from the abrupt viewpoint switches between different orthographic depictions and the high cognitive load of selecting and repairing the output of global segmentation.

All respondents reported a strong feeling of presence in the data set, always being aware which part of the point cloud they were currently viewing. They zoomed the camera in and out of the point sets to switch between coarse and fine-grained reconstruction parameters. Consequently, they alternated between overview and detail views of the data on their screens. Our automatic parameter estimation naturally supports such focus-and-context interaction types.

One task was to reconstruct a rectangular wall on a building facade, of which one corner was missing due to a scan shadow. The intermediate and expert users intuitively selected surrounding planes and found the missing corner as

the intersection point. All participants stated that they were confident in the interaction and that they felt the resulting geometry was identical to what they had intended. In the previous workflow, this task is more difficult. The automatic plane pre-segmentation needs to be adjusted to specific parameters to correctly select the participating planes for intersection.

Another task was to reconstruct the volume of a window reveal, of which the bottom side was missing. The participants applied the extrusion tool to construct the volume starting from the top side and snapping to the bottom. The participants positively commented on the simplicity of the interaction, being able to complete it from a single viewpoint. In the previous workflow, this task requires multiple viewpoint switches to capture the entire three-dimensional volume.

Ultimately, the novice and intermediate users found our approach to be a novel and interesting way of interacting with point clouds. This was evident from most of them staying to explore and reconstruct more point sets after the interview was over. The expert users welcomed our interaction techniques for being able to handle common tasks with ease that are otherwise cumbersome without feature support. An example is the ability to select plane normals or tangential directions with a single click. They found our ability to interact with point clouds of arbitrary size without compromising reconstruction accuracy an essential asset for modern high-resolution laser scanners.

5.4. Real-world Application

To indicate the usability of our approach, we briefly touch on usage of our interaction system in real-world applications. We implemented our technique in an out-of-core octree and point cloud rendering system, which can handle point clouds of arbitrary size, in our experiments up to tens of billions of points. This exemplifies how our approach can be integrated in existing point cloud tooling.

The project partners and domain experts in urban surveying at rmDATA use our system in their commercial product rmDATA 3DWorx [8]. They deploy it to expert users in the domains of as-built surveying and urban reconstruction. A domain expert independently measured lengths on buildings from which we have laser scans. Our reconstructed measurements agree with their precise surveying, showing 0.35 mm of difference on average on a length of 1 m.

5.5. Comparison with Interactive Tools

For completeness, we briefly compare the reconstruction experience using our tool compared to similar published work, O-Snap (Arikan et al. [2]) and Lejemble et al. [12], applied to their publicly available demonstration data sets.

The clear advantage of O-Snap lies in a fast reconstruction as long as the data is available. As noted in their evaluation, reconstruction takes longer in regions where building parts are not included in the original data. Furthermore, their global automatic optimization focuses on creating a watertight mesh, which makes the precision of the reconstructed geometry difficult to comprehend.

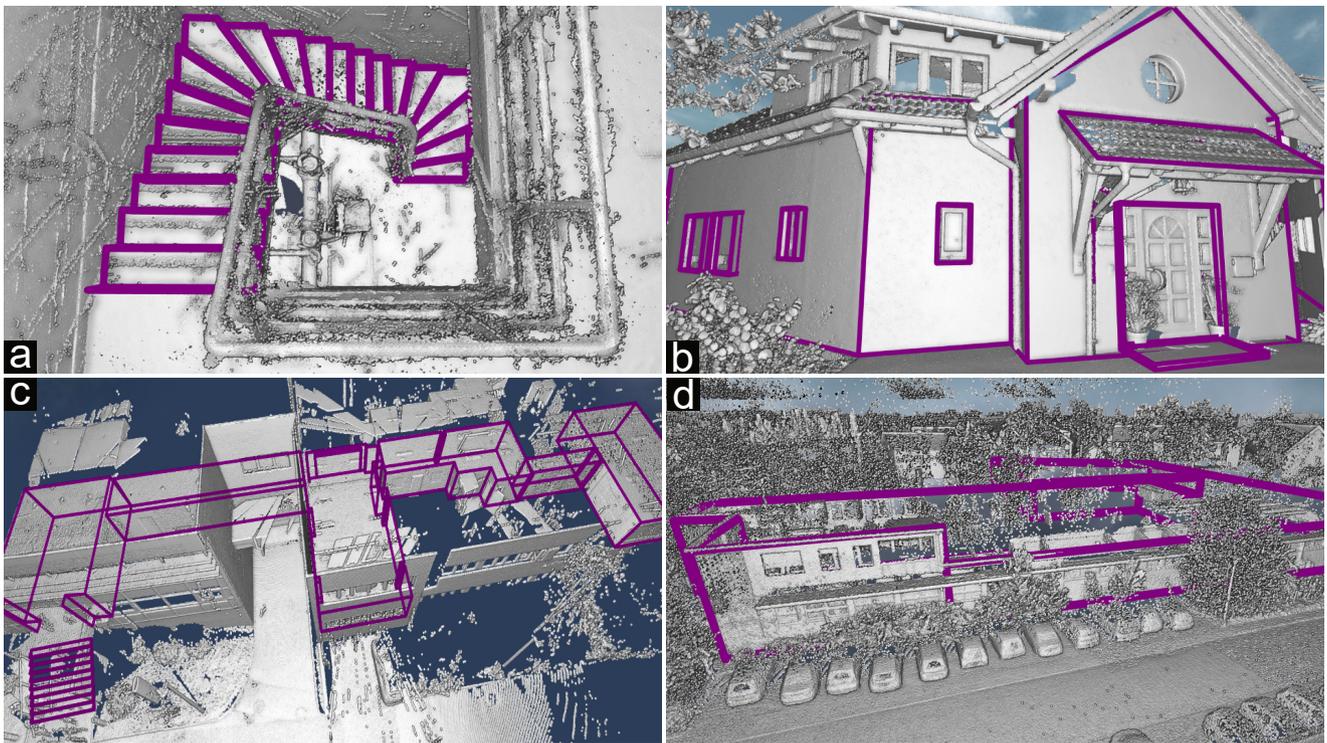


Figure 14: Reconstructed scenarios from Table 1. (a) Indoor spiral staircase. (b) House facade. (c) Indoor hull of an office building that was scanned both from inside and outside. Traditionally prone to severe oversegmentation, this frequent scenario was straightforwardly reconstructed using our method by moving the camera through the rooms and clicking on the corners. (d) Factory layout. The facade is severely obscured by trees, cars, and moving people, which are common artifacts in mobile mapping. This scenario was reconstructed using our method by clicking on the dominant planes from a zoomed-out view, and ad-hoc filtering the noise.

Lejemble et al.’s technique is more similar to ours in terms of user interaction. It is also unable to use missing regions for modeling, but their effective segment-browsing toolset notably speeds up the process. Their tooling could be adapted to work within our local reconstruction loop.

In summary, all three approaches provided adequate ways to obtain the desired geometry, but each one is fundamentally motivated by different reconstruction scenarios and requirements. In comparison, our technique took approx. 5-10 minutes longer than in the compared works. It, however, had a clear focus on completing reconstructions where others failed, and on modeling the geometry as accurately as possible.

6. Discussion and Future Work

Our interactive and localized approach to point cloud reconstruction makes for lightweight interactions. These are intuitively clear to humans and produce little cognitive load. Further, they interface directly with a point cloud. All of these are valuable properties. Novice and expert users quickly adapted to our technique due to its intuitive nature and reported gaining new insight into the structure of point clouds they interacted with.

Currently, in our technique, shapes other than plane segments are not considered. An interesting research direction

is the integration of other support geometries, such as parametric surfaces, or the automatic detection of different kinds of edges [9], [10] or parametric curves [28]. It remains to be investigated whether they could be assembled into sensible user interactions outside of highly specialized contexts.

Our feature-assisted CAD tooling requires no specialized point cloud infrastructure and can easily be integrated with existing interaction systems, such as O-Snap [2]. As future work, a mesh optimization loop similar to O-Snap’s could be used to evolve our loose collection of reconstructed polygons into watertight meshes on-the-fly.

7. Conclusion

In this paper we present an interaction framework for feature-assisted point cloud geometry reconstruction using on-demand planar region growing. The core contributions are a density-normalized out-of-core point cloud reconstruction technique, an on-demand plane-segmentation interaction framework, and a feature-assisted high-precision geometry reconstruction workflow. Our evaluation shows that we achieve the stated goal of providing reconstruction capabilities at arbitrary precision and level of detail on large out-of-core point clouds without preprocessing.

A. Rebasement Incremental Plane Regression

Given an incremental plane regression $R = (\mathbf{S}, \mathbf{S}\mathbf{q}, c, \mathbf{D})$ as defined in Section 3.2, and a new *reference point* \mathbf{r} , with the previous reference point denoted as \mathbf{r}_0 . Let $\mathbf{d} = \mathbf{r} - \mathbf{r}_0$, we calculate the *rebased incremental plane regression* $R^R = (\mathbf{S}^R, \mathbf{S}\mathbf{q}^R, c, \mathbf{D}^R)$ which re-centers the synthesized values around \mathbf{r} using the following Equations 4:

$$\begin{aligned} \mathbf{S}^R &= \mathbf{S} + c * \mathbf{d} \\ \mathbf{S}\mathbf{q}^R &= \mathbf{S}\mathbf{q} + 2 * (d_x * S_x, d_y * S_y, d_z * S_z) + c * (d_x^2, d_y^2, d_z^2) \\ v_x &= d_x * S_y + d_y * S_x + c * d_x * d_y \\ v_y &= d_x * S_z + d_z * S_x + c * d_x * d_z \\ v_z &= d_y * S_z + d_z * S_y + c * d_y * d_z \\ \mathbf{D}^R &= \mathbf{D} + (v_x, v_y, v_z) \end{aligned} \quad (4)$$

B. Conflicts of interests/competing interests

The authors have no competing financial or non-financial interests to disclose.

C. Acknowledgment of Funding

This work at VRVis is funded by BMK, BMDW, Styria, SFG, Tyrol and Vienna Business Agency in the scope of COMET-Competence Centers for Excellent Technologies (879730) which is managed by FFG. Part of the work has been funded by the Vienna Science and Technology Fund (WWTF) and the City of Vienna (Grant ID: 10.47379 / ICT22055).

References

- [1] , 2009. CloudCompare 2.X. <http://www.cloudcompare.org/>. Accessed: Feb. 2023, GPL License.
- [2] Arikian, M., Schwärzler, M., Flöry, S., Wimmer, M., Maierhofer, S., 2013. O-snap: Optimization-based snapping for modeling architecture. *ACM Transactions on Graphics (TOG)* 32, 1–15.
- [3] Bauchet, J.P., Lafarge, F., 2020. Kinetic shape reconstruction. *ACM Transactions on Graphics (TOG)* 39, 1–14.
- [4] Berger, M., Tagliasacchi, A., Seversky, L.M., Alliez, P., Guennebaud, G., Levine, J.A., Sharf, A., Silva, C.T., 2017. A survey of surface reconstruction from point clouds, in: *Computer Graphics Forum*, Wiley Online Library. pp. 301–329.
- [5] Chen, J., Chen, B., 2008. Architectural modeling from sparsely scanned range data. *International Journal of Computer Vision* 78, 223–236.
- [6] Deschaud, J.E., Goulette, F., 2010. A fast and accurate plane detection algorithm for large noisy point clouds using filtered normals and voxel growing, in: *3DPVT*, Hal Archives-Ouvertes Paris, France.
- [7] Edelsbrunner, H., Mücke, E.P., 1994. Three-dimensional alpha shapes. *ACM Transactions on Graphics (TOG)* 13, 43–72.
- [8] rmDATA GmbH, 2021. rmDATA 3DWorx. <https://www.rmdatagroup.com/produkte/rmdat3dworx/>. [Online; Accessed: 2023-01-27].
- [9] Hackel, T., Wegner, J.D., Schindler, K., 2016. Contour detection in unstructured 3d point clouds, in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1610–1618.
- [10] Himeur, C.E., Lejembre, T., Pellegrini, T., Paulin, M., Barthe, L., Mellado, N., 2021. Pcednet: A lightweight neural network for fast and interactive edge detection in 3d point clouds. *ACM Transactions on Graphics (TOG)* 41, 1–21.
- [11] Kaiser, A., Ybanez Zepeda, J.A., Boubekour, T., 2019. A survey of simple geometric primitives detection methods for captured 3d data, in: *Computer Graphics Forum*, Wiley Online Library. pp. 167–196.
- [12] Lejembre, T., Mura, C., Barthe, L., Mellado, N., 2020. Persistence analysis of multi-scale planar structure graph in point clouds, in: *Computer Graphics Forum*, Wiley Online Library. pp. 35–50.
- [13] Li, Y., Wu, X., Chrysathou, Y., Sharf, A., Cohen-Or, D., Mitra, N.J., 2011. Globfit: Consistently fitting primitives by discovering global relations, in: *ACM SIGGRAPH 2011 papers*, pp. 1–12.
- [14] Mellado, N., Lejembre, T., Guennebaud, G., Barla, P., et al., 2020. Ponca: a point cloud analysis library. <https://github.com/poncateam/ponca/>. [Online; Accessed: 2023-01-27].
- [15] Mercier, C., Lescoat, T., Roussillon, P., Boubekour, T., Thiery, J.M., 2022. Moving level-of-detail surfaces. *ACM Transactions on Graphics (TOG)* 41, 1–10.
- [16] Monszpart, A., Mellado, N., Brostow, G.J., Mitra, N.J., 2015. Rapter: rebuilding man-made scenes with regular arrangements of planes. *ACM Trans. Graph.* 34, 103–1.
- [17] Nan, L., Sharf, A., Zhang, H., Cohen-Or, D., Chen, B., 2010. Smart-boxes for interactive urban reconstruction, in: *ACM SIGGRAPH 2010 papers*, pp. 1–10.
- [18] Nan, L., Wonka, P., 2017. Polyfit: Polygonal surface reconstruction from point clouds, in: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2353–2361.
- [19] Oesau, S., Lafarge, F., Alliez, P., 2016. Planar shape detection and regularization in tandem, in: *Computer Graphics Forum*, Wiley Online Library. pp. 203–215.
- [20] Oesau, S., Verdie, Y., Jamin, C., Alliez, P., Lafarge, F., Giraudot, S., Hoang, T., Anisimov, D., 2021. Shape detection, in: *CGAL User and Reference Manual*. 5.3 ed. CGAL Editorial Board. URL: <https://doc.cgal.org/5.3/Manual/packages.html#PkgShapeDetection>. [Online; Accessed: 2023-01-27].
- [21] Otsu, N., 1979. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics* 9, 62–66.
- [22] Rabbani, T., Van Den Heuvel, F., Vosselmann, G., 2006. Segmentation of point clouds using smoothness constraint. *International archives of photogrammetry, remote sensing and spatial information sciences* 36, 248–253.
- [23] Rusu, R.B., Cousins, S., 2011. 3D is here: Point Cloud Library (PCL), in: *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China.
- [24] Scheiblauer, C., Wimmer, M., 2011. Out-of-core selection and editing of huge point clouds. *Computers & Graphics* 35, 342–351.
- [25] Schnabel, R., Wahl, R., Klein, R., 2007. Efficient ransac for point-cloud shape detection, in: *Computer Graphics Forum*, Wiley Online Library. pp. 214–226.
- [26] Steinlechner, H., Rainer, B., Schwärzler, M., Haaser, G., Szabo, A., Maierhofer, S., Wimmer, M., 2019. Adaptive pointcloud segmentation for assisted interactions, in: *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pp. 1–9.
- [27] Vo, A.V., Truong-Hong, L., Laefer, D.F., Bertolotto, M., 2015. Octree-based region growing for point cloud segmentation. *ISPRS Journal of Photogrammetry and Remote Sensing* 104, 88–100.
- [28] Wang, X., Xu, Y., Xu, K., Tagliasacchi, A., Zhou, B., Mahdavi-Amiri, A., Zhang, H., 2020. Pie-net: Parametric inference of point cloud edges. *Advances in neural information processing systems* 33, 20167–20178.
- [29] Yoder, R., Bloniarz, P.A., 2006. A practical algorithm for computing neighbors in quadtrees, octrees, and hyperoctrees., in: *MSV*, pp. 249–255.