Noname manuscript No.
(will be inserted by the editor)

ASPIRE: Automatic Scanner PosItion REconstruction

Georgios-Tsampikos Michailidis · Renato Pajarola

Received: date / Accepted: date

Abstract The recent advances in 3D laser range scanning have led to significant improvements in capturing and modeling 3D environments, allowing the creation of highly expressive and semantically rich 3D models from indoor environments, generally known as *building information models* (BIMs). Despite the capabilities of state-of-the-art methods to generate faithful architectural 3D building models, the majority of them rely explicitly on the prior knowledge of scanner positions in order to reconstruct them successfully. However, in real-world applications this metadata information gets typically lost after the point cloud registration, which means that none of these methods could work in practice and the creation of their building models would be impossible. Therefore, we present a novel pipeline that allows to automatically and accurately reconstruct the original scanner positions under very challenging conditions, without requiring any prior knowledge about the environment or the dataset. Being independent from laser range scanner manufacturers, it can be applied to almost every real-world LiDAR application. Our method exploits only information derived from the raw point data and is applicable to all scientific and industrial applications, where the original scan positions typically get lost after registration by the proprietary software provided by the scanner manufacturers. We demonstrate the validity of our approach by evaluating it on several real-world and synthetic indoor environments.

Keywords LiDAR reconstruction · interiors reconstruction · point cloud processing · point pattern analysis

G.-T. Michailidis
University of Zurich, 8050 Zurich, Switzerland
Tel.: +41 44 635 71 27
E-mail: gtmichail@ifi.uzh.ch

R. Pajarola
University of Zurich, 8050 Zurich, Switzerland
Tel.: +41 44 635 43 70
E-mail: pajarola@ifi.uzh.ch

1 Introduction

In recent years, many methods have taken full advantage of new advances in acquisition technology, in order to generate faithful 3D architectural *building information models* (BIMs), which are used in many diverse application areas such as building construction, facility maintenance, engineering, visualization, navigation and other. Such models can reduce the time and costs of various tasks around building maintenance and construction, significantly enhancing the project performance and productivity.

A typical starting point for generating such 3D models is to scan the environment from multiple positions using a LiDAR device, yielding small point clouds that represent parts of the scene and which are usually associated with some metadata information, such as the scanner position coordinates. The raw point clouds are then registered to a common reference frame in order to compose a large joint 3D point cloud that represents the whole scene, which is then further processed for generating the architectural 3D building model. Thanks to the many recent research efforts, current state-of-the-art modeling methods can cover almost all aspects of real-world scenarios and applications, such as reconstructing the architectural building structures (e.g. walls, ceiling and floor) [1–9], handling occlusions and clutter [1–3,8,7], detecting wall openings (i.e. windows and doors) [1, 2,5,10,7,11], while they can also semantically partition the building space into individual rooms [4,3,9,8,7].

Despite the variety of these methods, the majority of them exhibit a common characteristic, i.e. they rely on the prior knowledge of the scanner positions in order for their modeling pipelines to be successfully executed (see also Section 2). However, after registering the raw point clouds, which is usually performed by proprietary software developed by the scanner manufacturers, the metadata information (e.g. the scanner positions) from each individual scan typically

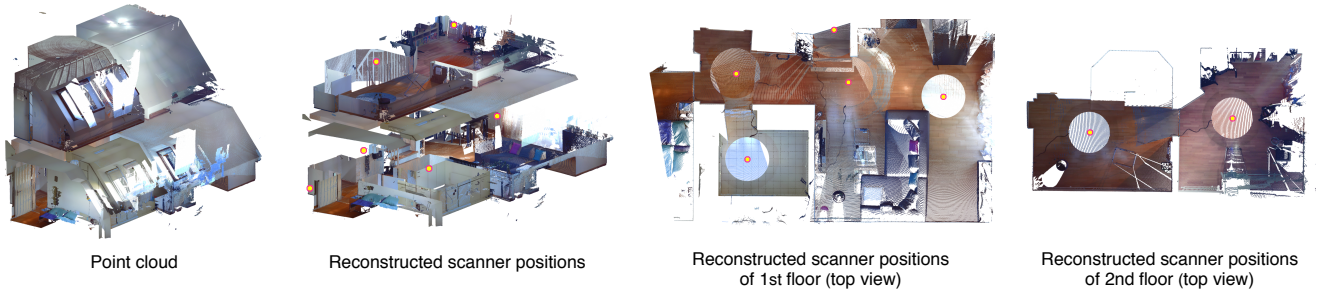


Fig. 1 We employ a descriptive, spatial point pattern analysis to the point distribution of the point clouds, in order to reveal the features which will allow us to retrieve automatically the original scanner viewpoint positions.

gets lost since it is not considered or exposed by the software, and only point-related data (e.g. point coordinates and color) remain in the registered point cloud. Thus in many real-world scenarios, where only the registered point clouds are considered, the scanner positions are often not available and, consequently, many of the above-mentioned methods would not work in practice, since without this information their pipelines will fail. Hence these methods commonly assume that the scanner positions are known a priori (e.g. from explicitly recording the scanner positions), imposing a hard restriction to the reconstruction process, limiting their automatic applicability and increasing their dependency to external factors. Moreover, due to their direct dependency on this metadata information, when it is not available, they apply various computation methods to generate new synthetic scanner poses in arbitrary positions in order to simulate the real ones and execute successfully their tasks (e.g. in [9]). However, based on their typical usage in the modeling pipelines, arbitrary or approximate scanner positions can create wrong localizations, failures or imperfections during occlusion detection, imprecision to object detections and wrong semantic interpretations.

In this paper we introduce a novel method for automatically reconstructing the real scanner viewpoint positions, without requiring any prior knowledge about the environment or the dataset. Our method is very efficient and robust, and it can compute the original scanner position coordinates exploiting only information derived from the raw point data (see also Fig. 1). In particular, our method is applicable mainly to point clouds where the original scan positions are either not available (e.g. in individual raw point clouds, since many point cloud formats do not store the scanner positions) or lost due to registration, and it is considered especially beneficial for industrial and scientific applications that process registered and large point clouds, such as in scan-to-BIM workflows, architectural 3D modeling, localization of building components, building re-trofits, etc. Therefore, it can be applied to the majority of indoor point clouds, and being independent from the laser scanner manufacturers makes it suitable for almost every real-world LiDAR application. We demonstrate the effectiveness of our proposed method by applying it to a variety of real-world and synthetic indoor datasets under various challenging conditions.

2 Related Work

To better comprehend the extent to which the state-of-the-art methods rely on the widely-used assumption that the scanner positions should be known a priori in order to be successfully executed, we provide a brief overview of the related literature below, emphasizing mainly on the usage and exploitation of scanner positions. Since our method is explicitly applicable to point clouds acquired by LiDAR devices, methods relying on RGB-D datasets, such as in [12], are not included in our literature review. Additionally, since we focus on BIM applications, recent approaches such as in [13, 14], which only semantically segment the point clouds without reconstructing their geometric, volumetric or parametric building models are outside the scope of automatic creation of BIMs and therefore are also excluded from our analysis.

In [1,2], major tasks such as occlusion labeling, wall openings detection and wall surface reconstruction are performed by relying explicitly on the prior knowledge of scanner positions, while in [5] the prior knowledge of viewpoints is used in a similar manner for occlusion handling, but also for hierarchically clustering the building elements and reconstructing the wall openings. In [15], the planar model fitting relies on the consistent assignment of point-wise normal vector orientations according to the scanner positions, from which the walls, ceiling, and floor are later reconstructed. The viewpoints are also used in [6] for detecting occlusions and semantically segmenting the source point cloud, as well as for classifying the points and estimating the orientations of the wall surfaces. In a more recent method [9], authors do not assume the prior knowledge of scanner positions, but they compute synthetic ones in arbitrary positions in order to segment the 3D space and reconstruct the room layout. In [7], the point cloud segmentation and room layout partitioning rely explicitly on the prior that the initial set of room

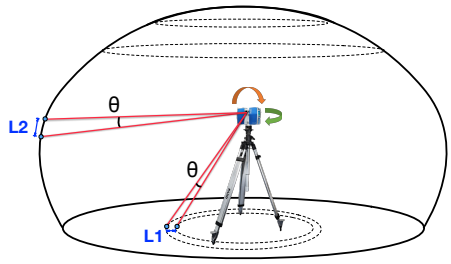


Fig. 2 Typical laser scanning setup, showing the rotational movement around the scan position, as well as the angle θ of the angular scanning increments. Notice that the scan density changes as we proceed farther away from scanner position (i.e. $L1 < L2$).

labels is directly provided by the scanner positions, while the reconstruction of the wall elements and the detection of wall openings rely also on the same prior information. The methods in [3,8] assume that the number of scans and the scanner positions are known a priori in order to extract candidate walls, to employ viewpoint-based visibility computations for occlusion detection and walls' reconstruction, but also to extract the correct number and location of rooms (space partitioning). Occlusion handling and wall openings detection are also performed in [16] with the help of the same prior knowledge, while a different method [11], which focuses only on the reconstruction of complex wall openings from building interiors, relies again on viewpoints in order to be successfully executed.

In the next section, before providing a detailed description of our pipeline, we briefly review the fundamental operation of a typical laser range scanner (LiDAR) and mention the key data acquisition principles on which our method relies for reconstructing the original scanner positions.

3 Laser Scanner Fundamentals

To reason about our proposed method, we first have to consider some facts that characterize the typical point cloud acquisition process. A typical laser scanner (phase-shift or time-of-flight) emits a laser beam and uses a rotational mechanism to deflect it to different directions. The laser scanners from main manufactures (e.g. Leica, Riegl, Faro, etc.) present a field of view of almost 360°, forming a closed spherical scan region around the scan position and excluding only a small part under the laser scanner due to the presence of the supporting tripod, as illustrated in Fig. 2.

During the acquisition phase, the scanning quality depends mainly on the distance between the laser sensor and the object surface, as well as on the selected angular resolution (known also as scan density), assuming that for short-range applications such as the acquisition of indoor environments, other factors such as the environmental conditions and beam divergence do not affect the laser measurements notably. [17]. In particular, the scan density specifies the smallest possible increment of the angle between two successive points and causes the scanned surface areas near the scanner position to have a higher density of samples (i.e. points) than the areas farther away (see also Fig. 2).

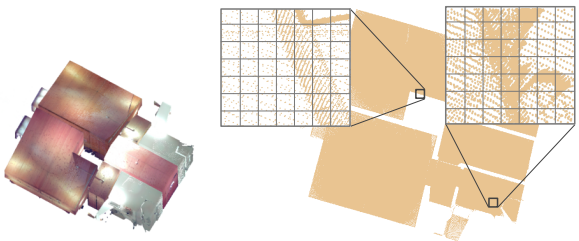
Based on the above scanning characteristics, most scenes acquired by a laser scanner exhibit a fairly regular circular scanning pattern on surface areas near and around the scanner position, while the density of points in these areas is prominently higher than in other areas farther away.

4 Method Overview

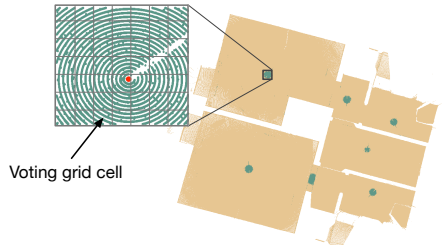
Relying on the intrinsic scanning characteristics of the acquisition process, the main idea of our method is to employ descriptive, spatial point pattern analysis to the point distribution of the raw point cloud, in order to reveal the features which will allow us to retrieve the original scanner viewpoint positions. Our analysis focuses on the detection of the circular scanning patterns and the high density of points near the scanner positions, which can clearly indicate the accurate spatial position of the scanning device.

Specifically, the input to our pipeline can be a set of merged cluttered 3D point clouds representing any kind of interior environment and acquired by any laser scanning device as outlined above. A great advantage of the proposed method is that it is independent of the architectural shape of building interiors, and therefore, it can be applied to environments with complex room layouts, arbitrarily oriented planar or curved walls and sloped ceilings. Furthermore, it is quite robust against outliers, noise and heavily cluttered scenes, and it is not affected by occlusions.

Given the (merged) point cloud (Fig. 3(a)), our method computes its bounding box and generates a uniform 2D grid of cells, into which the raw 3D points are projected and classified (Fig. 3(b)). Performing spatial and point pattern analysis on the point distribution of each cell, we identify strongly reliable *voting cells*, i.e. cells which can indicate the scanner positions with high confidence, using a linear *support vector machine* (SVM) (Fig. 3(c)), and we cluster their points based on their spatial point characteristics. Typically, the most reliable voting cells lie in flat surface areas near the LiDAR devices, where the circular scanning patterns are strong and clear. Then a circle model fitting algorithm is applied to each curve-shaped point cluster in voting cells, in order for its circle center to cast a vote in a Hough transform space for the potential scanner position. Using non-maximum suppression, we detect the cells where the scanner devices are located, and by further analyzing the scanner position estimates inside those cells, we find their exact location within an accuracy of tenths of millimeters.



(a) Input point cloud. (b) 2D grid plane, into which the raw 3D points are projected and classified.



(c) Detected voting grid cells after the SVM classification.

Fig. 3 The main phases of our method for the reconstruction of scanner positions. The input point cloud (a), the 2D grid of cells (b) and the detected voting cells (c), which will cast votes for the scanner positions. The image insets indicate the clear difference of point distributions between randomly selected cell areas and the cells near and around the scanner positions. The finally retrieved scanner positions are indicated with a red dot inside the insets in (c).

5 Detection of Voting Cells

Our method starts with detecting, in the input point cloud and its projection into the 2D cell grid, the areas in which the point distributions clearly exhibit circular scanning patterns of LiDAR devices, which we will use as reliable voting areas to cast votes for the potential scanner positions. In multi-story buildings, each floor could be first identified by performing a histogram analysis along the vertical axis similar to [18, 19].

5.1 Space Partitioning

The target underlying the projection of points into the 2D plane and its discretization into a grid of uniform cells is to reduce the dimensionality of the scanner position reconstruction problem and to optimally increase the number of the detected voting cells, particularly in cases where large parts of the floor surface are occluded by the interior objects. Moreover, by discretizing the continuous 2D space into a grid of uniform cells, we are efficiently balancing between computational performance and position accuracy for localizing the scanner devices.

Before projecting the points of point cloud to the 2D plane, we compute for the input raw point cloud its bounding box (BB), which provides a simplified shape approximation of the scene and allows us to demarcate the equivalent 2D space in which the points will be projected. This bounded space is then subdivided into a uniform grid of cells, where the 3D points will be projected into and assigned to the respective cells.

To partition the 2D plane into a grid map, we define first the size of the grid cells. Since these cells will be used for evaluating the information derived from their embodied points and intuitively will determine also the level of expressiveness of the features derived from them, we define the grid cells' size considering its impact on the computational performance, the intrinsic scanning characteristics (see Sec. 3) and the scanning patterns generated under different indoor environments. For instance, although a large cell size contains more points and could natively increase the localization accuracy of scanner positions by improving feature expressiveness, it could also be easier distorted by adjacent fuzzy point distributions, such as from overlapping scans or from projected points belonging to nearby interior objects. On the other hand, a small cell size may allow a finer and more reliable classification, but it would result also to an increased grid map resolution, leading to higher memory consumption and computational costs. Considering, moreover, that indoor environments typically contain many non-structural objects like furniture, the reliable detection of voting cells in such cluttered environments is very challenging and makes the requirement of high resolution grid maps necessary. Thus in our pipeline it can be set according to the point sampling distance and the localization accuracy requirements of the application, while in our tests we set the default cell size conservatively to a small $1cm$, allowing finer and more reliable cell classifications, as demonstrated in our experiments.

After creating the 2D grid map, we project the raw 3D points into the corresponding cells, which given the orientation of the floor can easily be done by an orthogonal projection. Quite often the floor plane is aligned with a particular coordinate system plane, such as e.g. the x, y-plane, which allows just dropping a coordinate (e.g. z). In addition, considering that the computation of cell features, as described in Section 5.2, requires frequent spatial range searches between the points contained in each cell, and taking into account the demanding processing and data storage requirements for handling large-scale datasets, we employed an efficient Kd-tree data structure [20], which is fully optimized for accelerating range and nearest-neighbor searches, and can optimally organize and classify the data points in the grid map.

5.2 Feature Extraction

To eventually classify a grid cell as a reliable voting cell or not, we need a set of features which will be capable to

discriminatively represent the spatial point distributions of the cells. Considering that the point distribution is strongly affected by various factors, such as the distance from laser scanner, the scanning resolution, the projected points of adjacent indoor objects and the overlapping scans, we incorporated descriptive features based on spatial statistics and point pattern analysis, whose combination permits efficient cell classification under the aforementioned vague scanning conditions. Features relying mostly on point intensities, color, texture, shape or convexity are very likely to underperform under the challenges of this process, while on the other hand, descriptive features measuring the variance or the statistical dispersion of points inside a cell are simple yet more suitable to characterize adequately a point distribution in the present context.

The features we used, which are general enough to account for cell diversity and point distribution variability, are:

Median nn-distance (ψ_1) This is the median length of all pairwise Euclidean distances between a point and its nearest neighbors in the cell. More precisely, denoting the spatial point distribution of each cell of size N by $\boldsymbol{\mathcal{P}}^s = (p_i^s : i \in \{1, 2, ..., N\})$ [21], for any point $p_i^s \in \boldsymbol{\mathcal{P}}^s$ in cell s the nearest neighbor distance (nn-distance) from p_i^s to all other points in $\boldsymbol{\mathcal{P}}^s$ is given by

$$d_i^s = d_i(\boldsymbol{\mathcal{P}}^s) = \min\big\{d(p_i^s, p_j^s) : p_j^s \in \boldsymbol{\mathcal{P}}^s, j \neq i\big\}, \quad (1)$$

where $d(.,.)$ is the Euclidean distance. Consequently, the median nn-distance is given by the following equation

$$d_{\text{med}}^s = \text{median}(\boldsymbol{\mathcal{D}}^s) \quad (2)$$

where $\boldsymbol{\mathcal{D}}^s = \{d_i^s\}$ is the set of all nn-distances between any point p_i^s and its nearest neighbor in cell s.

Mean nn-distance (ψ_2) This feature indicates the mean distance of all pairwise Euclidean distances between a point and its nearest neighbors in the cell. In a similar manner as the median nn-distance, it is given by

$$\widetilde{d^s} = \frac{1}{N}\sum_{i=1}^{N} d_i^s, \quad (3)$$

where the notation and constants are the same as above.

Height range (ψ_3) Represents the height difference between the two highest and lowest points in a cell.

$$d_{z-\text{range}}^s = \max_z(\boldsymbol{\mathcal{P}}^s) - \min_z(\boldsymbol{\mathcal{P}}^s), \quad (4)$$

Here, $\max_z$ and $\min_z$ indicate the maximum and minimum values along the z-axis of all points in $\boldsymbol{\mathcal{P}}^s$.

Standardized height dispersion (ψ_4) Quantifies the statistical dispersion of points along the z-direction inside the cell and penalizes the cells that exhibit large height variations, combining the performance of measures from descriptive and robust statistics. Specifically, it is expressed proportionally to the amount of points in each cell as

$$d_{\text{disp}}^s = \frac{d_{z-\text{range}}^s - \text{MAD}(\boldsymbol{\mathcal{D}}^s)}{N \cdot \text{STD}(\boldsymbol{\mathcal{D}}^s)}, \quad (5)$$

where STD is the standard deviation of all distances in $\boldsymbol{\mathcal{D}}^s$, and MAD is a robust estimator of the variability indicating the median absolute deviation from the median [22] and is computed by

$$\text{MAD}(\boldsymbol{\mathcal{D}}^s) = \text{median}\left\{\left|d_i^s - d_{\text{med}}^s\right|\right\} \quad (6)$$

Skewness of nn-distances (ψ_5) Measures the asymmetry in the distribution of nn-distances inside a cell and quantifies it as a value of spread. Typically the voting cells are skewed left, so using the Pearson's moment coefficient of skewness [23] we compute it as

$$\text{skew}(\boldsymbol{\mathcal{D}}^s) = \frac{\frac{1}{N}\sum_{i=1}^{N}\left(d_i^s - \widetilde{d^s}\right)^3}{\text{STD}(\boldsymbol{\mathcal{D}}^s)^3} \quad (7)$$

Kurtosis of nn-distances (ψ_6) Measures the normality of the distribution of nn-distances inside a cell and indicates how prone to outliers the cell is by measuring the tail heaviness relative to that of the normal distribution. It is given by

$$\text{kurt}(\boldsymbol{\mathcal{D}}^s) = \frac{\frac{1}{N}\sum_{i=1}^{N}\left(d_i^s - \widetilde{d^s}\right)^3}{\left(\frac{1}{N}\sum_{i=1}^{N}\left(d_i^s - \widetilde{d^s}\right)^2\right)^2} - 3 \quad (8)$$

where we subtract 3 from the raw kurtosis index to provide a figure relative to the normal distribution [23].

Histogram bin counts in XY (ψ_7) To compute these features, we first uniformly quantize the cell space into a particular number of bins based on d_{med}^s and we then generate a histogram for its x, y main coordinate axis. The *number of bins along each axis direction that contain at least one point* is used as a feature in our cell classification process, since it reveals the underlying shape of the point distribution.

Histogram bin counts in Z (ψ_8) Similar to the computation of the previous feature, we uniformly quantize the space between the minimum and the maximum height values of points contained in a cell based on d_{med}^s, and we then generate the corresponding histogram. The *number of bins containing at least one point* is used as the last feature in our classification method.

The above listed features can adequately capture the point distributions in the 2D grid cells and they have proven to be robust in practice, producing the least number of false negatives since the point distributions between the good voting and other cells typically varies a lot.

5.3 Outlier Removal

Before proceeding to the actual description of our cell classification approach, it is worth mentioning that, since we are performing a confirmatory data analysis for quantifying the extent to which the point distributions of voting cells deviate from those of the other cells (a process similar to statistical hypothesis testing), the detection and omission of anomalous data points in point patterns is quite beneficial [24]. Although their amount in voting cells is not high, considering also the inherent ambiguities introduced to cell classification by the scanning process as described earlier, we tried to increase the discriminance of cell classification by removing points that spatially deviate a lot from the other points in a cell and may adversely lead to cell misclassification.

Therefore, we added a data cleansing stage to our pipeline, which detects and rejects *spatial outliers* that present local spatial instabilities with respect to the neighboring points, which could bias and mislead the training process of our machine learning method, resulting in a less accurate model and ultimately in poorer results.

For that reason, we exploit the pairwise nn-distances (Eq. 1) to reject all points in a cell which are greater than the γ-scaled MAD away from the median nn-distance (Eq. 2). The inlier distances are given by

$$\begin{aligned} &\boldsymbol{\mathcal{D}}^s_{\text{filt}} = \{d^s_i \quad | \, d^s_i \leq \gamma \cdot d^s_{\text{outlier}}\} \\ &\text{with } d^s_{\text{outlier}} = \beta \cdot \text{MAD}(\boldsymbol{\mathcal{D}}^s), \end{aligned} \tag{9}$$

where the correction factor β is required to make MAD an unbiased estimate of the standard deviation for Gaussian data and is equal to the reciprocal of the quantile function Φ^{-1} [25,22,26], while γ is the outlier rejection threshold and was set to 3 according to [27,26]. An example demonstrating the effectiveness of this cleansing measure in point distributions of voting cells is presented in Fig. 4, while in a similar manner the noisy points from the other cells are eliminated too.

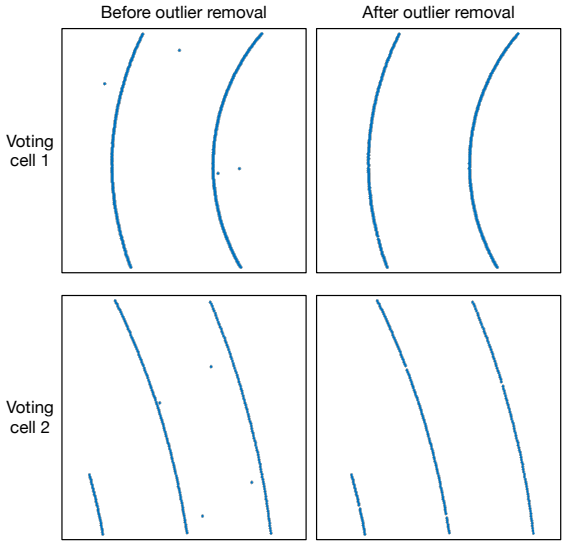


Fig. 4 Top view of two voting cells, showing the rejection of spatial outliers, i.e. noisy points causing local spatial instabilities.

5.4 Cell Classification

Based on an 8-dimensional feature vector $\Psi = (\psi_1, \psi_2, ..., \psi_8)$ created by the features introduced above, an SVM classification process learns to classify a point distribution signature for every given input cell, expecting that spatially similar point patterns have similar signatures. This task can be formulated as a binary supervised classification problem, where a given set of training samples (i.e. cells) $S = \{(\Psi_1, h_1), (\Psi_2, h_2), ..., (\Psi_s, h_s)\} \subseteq \ominus \times \mathcal{L}$, with $h_s \in \mathcal{L}= \{-1, +1\}$ being the target label set, can be used by a classifier $\mathbf{w}$ for learning the distribution patterns of voting cells from the training samples S and then predict reliably the label h_s for any unknown Ψ_s based on the equation $\mathbf{w}^T \Psi_s >$ τ. In our approach, we used linear classifiers trained via LIBLINEAR [28], while the usage of more advanced classification approaches (e.g. based on neural networks) cannot be fully justified, since our classification method has proven to be quite robust in practice. The resulting labeled grid map after the SVM classification has a form similar to Fig. 3(c).

6 Scanner Position Reconstruction

Focusing in this stage only on the detected voting areas and considering that their point distributions mainly form one or more curved-like shapes which constitute a small part of the characteristic circular point pattern around the scanner position, we reliably reconstruct the scanner positions following a conceptually straightforward strategy, as illustrated in Fig. 5.

6.1 Clustering the Points of Voting Cells

First, we cluster the points of each voting cell based on a recent method called *Gaussian density distance* (GDD) [29], which relies mainly on features such as the point density and pairwise point distance in order to naturally cluster the cell points based on a *Gaussian mixture models* (GMM). It can be applied in cases where point distributions present variable densities or pairwise point distances, differentiating efficiently the point clusters based on point densities and considering the spatial gradual variations in pairwise point distances. Fig. 5(a) presents an example of clustering a voting cell by applying the GDD method.

Our target here is to cluster the points $\boldsymbol{\mathcal{P}}^s$ of a voting cell s into l clusters $\mathcal{Q}^s = \{\mathcal{Q}^s_1, \mathcal{Q}^s_2, \ldots, \mathcal{Q}^s_l\}$, each having

a curved-like shape, to which we will fit a circle model c as a circular scanning pattern hypothesis in order to approximate the original circular scanning pattern from which they were generated. The problem of fitting a circle to a set of points can be approached using an *orthogonal least squares* (OLS) [30,31] fit by minimizing the continuous function

$$\boldsymbol{E_g}(a, b, R_l) = \sum d(p_i^s, c_l)^2, \tag{10}$$

where $d(p_i^s, c_l)$ stands for the Euclidean (geometric) distance between the point p_i^s in cluster $\mathcal{Q}_l^s$ and the circle curve c_l (usually called residual), (a, b) is the circle center and R_l the radius of this circle. Using the difference-of-squares geometric error criterion [32], this problem has a closed form and the above distances can explicitly be defined by the formula

$$\begin{aligned} d(p_i^s, c_l)^2 &= r_i - R_l \quad \forall R_l \geq 0 \\ \text{with} \quad r_i &= \sqrt{(x_i - a)^2 + (y_i - b)^2}. \end{aligned} \tag{11}$$

We should mention here that, although LS methods are sensitive to outliers and their results might be quite erroneous in cases with high noise, it was not necessary to use more robust algorithms like in [33,34] to accomplish the model fitting, since the point distributions of our voting cells contain only a very minor amount of noisy points due to our previous noise elimination processing step. The point clusters presented in Fig. 5(a) constitute a representative example which exemplifies our argument.

6.2 Recovery of the 2D Scanner Position

The conceptual basis of fitting a circle model to the clustered points of a voting cell is to allow each cluster to vote for a scanner location corresponding to its circle center, which represents actually a scanning position proposal. Using an enhanced Hough voting scheme, our method casts for each fitted circle model a weighted vote for the corresponding scanner position proposal into a discrete voting space, establishing correspondences between the potential scanner locations and a discrete confidence map supporting them. Then, the scores of all votes in the voting space are accumulated and the regions with the highest voting scores in the map are detected using a suppression or mode-seeking method. Put it simply, this process would be equivalent to finding the local maxima of intensities in an image, interpreting here as intensities the total number of proposals per grid cell. Fig. 6 shows one such an example from a voting area of the building shown in Fig. 3, where the cell intensities of scan position proposals are presented as a heatmap image.

The conceptual basis of the proposed voting model relies, among other factors, on the fact that the curved shape

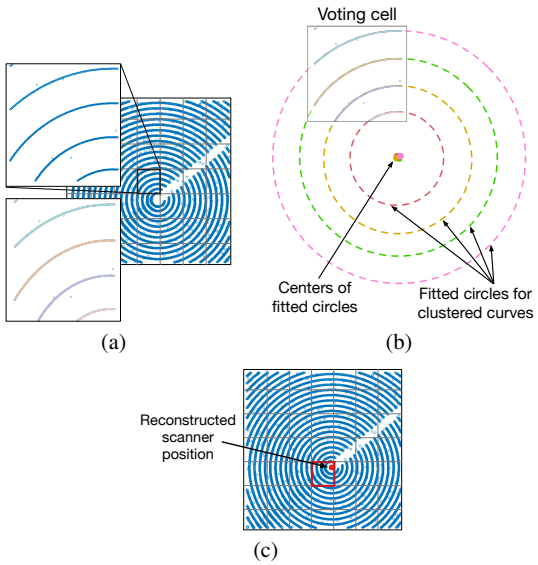


Fig. 5 (a) After the detection and classification of voting cell areas (here presented in blue), the points of voting cells are clustered using the GDD algorithm. (b) Then a robust circle fitting method is applied to the points of each cluster and the corresponding circle centers are computed. (c) These centers represent scanning position proposals which cast votes for the scanner locations in a voting space, where using non-maximum suppression we detect the cells with the maximum proposals in the grid map (cell in red), while applying mean-shift to the Gaussian KDE of the scanner position proposals of the detected cells, we accurately reconstruct the original 2D scanner coordinates (red dot).

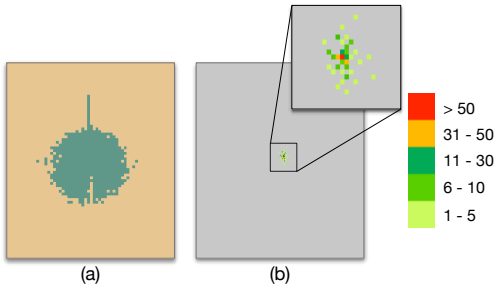


Fig. 6 (a) Example classification of voting cells (in green) around a scanner location in an example region from the building model shown in Fig. 3. (b) Heatmap of the estimated circle center locations from the voting cells, with more than 800 votes, of which the majority lies in the red cell.

of point distributions gets softer farther away from the scanner locations, restricting the accurate circle fitting and, consequently, decreasing the consistent voting for scanner positions. Thus we assign larger weights to the votes originated from point clusters closer to the scanner proposals, as well as from those presenting smaller magnitude of residuals from the circle fitting model. Hence the weights are computed by

$$w_l = \frac{k_l}{RMSE_l} \cdot e^{-R_l}$$

where k_l is a scaling factor and $RMSE$ is the root mean squared error, which quantifies the goodness-of-fit of the fit-

ted circle model to the data points. It is computed by

$$RMSE_l = \sqrt{\frac{\sum_{i=1}^{N_l} d(p_i^s, c_l)^2}{N_l}},$$

where N_l is the total number of points in cluster $\mathcal{Q}_l^s$. Note that due to this weighted voting approach, our method is capable to efficiently eliminate inconsistencies or distortions in the voting process, while more complex probabilistic voting schemes (such as in [35,36]) will not add any additional value to the extracted results, since our robust voting scheme seems to cover effectively all possible cases.

Eventually, we detect the regions (i.e. cells) in the grid map where the scanner devices are located by applying non-maximum suppression in the confidence map of the Hough space [37]. To accurately reconstruct the 2D scanner coordinates, we further analyze the distribution of scanner position proposals inside those cells, using a Gaussian kernel density estimate (KDE) for the probability density function of the proposals and applying mean-shift to find the mode of the density function. Finally, finding the nearest neighboring proposal to the computed mode value, we reconstruct the final 2D scanner positions with an accuracy in the tenths of millimeter (see also Tbl. 1). Note that the use of Hough-based methods to identify the circular scanning patterns and consequently finding their centers will practically fail, since the circular shapes get typically distorted, deformed or interrupted under the above-mentioned real-world challenging conditions (see Sec. 5.2). Our approach is very robust and can accurately recover the 2D scanner positions even if the scanning pattern is distorted, since only a few (theoretically one) reliable voting cells are adequate for detecting the exact scanner positions.

6.3 Recovery of the Scanner's Height

Assuming that the 2D coordinates of the scanner position have been already recovered (see Section 6.2), we reconstruct the height of the LiDAR device at that position by analyzing trigonometrically the geometry of the laser setup, as illustrated in Figs. 2 and 7. Specifically, considering that the angular scan resolution remains constant during the scanning process, and following the intuition that every concentric circle of the circular scanning pattern constitutes the base of a cone whose apex coincides with the 3D scanner position, we create a vertical line ($l3$) that passes through the recovered 2D scanner position P, emulating the common axis of these cones. By creating line segments connecting the points of the voting cells to a common point SP on the cones' axis, we approximate the apex of the concentric cones by comparing the angles (e.g. θ_1 and θ_2) between the line segments of three consecutive point clusters inside the same voting cell (e.g. from points $P1$, $P2$ and $P3$) until

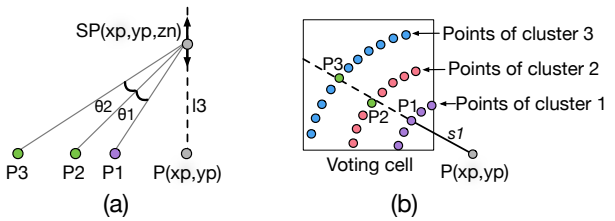


Fig. 7 (a) Approximating the 3D scanner position SP by comparing the angles between the line segments created by aligned points from different clusters inside the voting cell. (b) Selection of aligned points inside a voting cell by finding the nearest points (e.g. $P2$ and $P3$) to the lines (e.g. $s1$) that start from the scanner position P and pass from each point of the nearest cluster to P (e.g. $P1$).

they converge, i.e. until $\theta_1 \cong \theta_2$. The position where the θ angles converge constitutes a 3D scanner position proposal, and after repeating this process for all voting cells, we apply a similar mode-seeking approach as in Section 6.2 along the cones' axis in order to find the most likely height position of the scanning device. Note that in order to reduce inconsistencies and keep only indicative θ angles, our approach only considers points with the same height inside the voting cells, while we compare only the angles between the line segments created by aligned points of different clusters with respect to the scanner position P, as depicted in Fig. 7(b).

7 Results and Discussion

In this section, we present our experimental results and discuss the insights derived from them, focusing mainly on real-world and synthetic datasets used by state-of-the-art methods (e.g. from [3,38,8,9]) in the field of 3D BIM reconstruction. The buildings of the datasets we used present diverse properties in architectural design and unique characteristics that challenge our reconstruction pipeline and prove the general validity and performance of our method. An overview of all datasets and their statistics are provided in Figs. 8 and 9, while Tbl. 1 shows indicatively the reconstructed scanner position coordinates from some of these datasets along with their absolute distance errors, according to the equation $\Delta E = \sqrt{(x_{orig} - x_{rec})^2 + (y_{orig} - y_{rec})^2 + (z_{orig} - z_{rec})^2}$.

In Fig. 8, the 'Cottage' [8] building model constitutes a very challenging dataset that presents steep sloped ceilings which distort the circular point patterns and create ellipsis when projected onto the 2D grid map, making the circle model fitting difficult and ultimately leading to the distribution of votes over a line. However, this voting pattern is not affecting the final reconstruction accuracy, since these false positive votes do not distract the mode-seeking and non-maximum suppression steps, as also demonstrated by the numerical results in Tbl. 1. The results from this dataset also demonstrate the effectiveness of our method against other similar scenarios, e.g. when the floor is not horizontal or the scanner is not mounted vertically to the floor, where

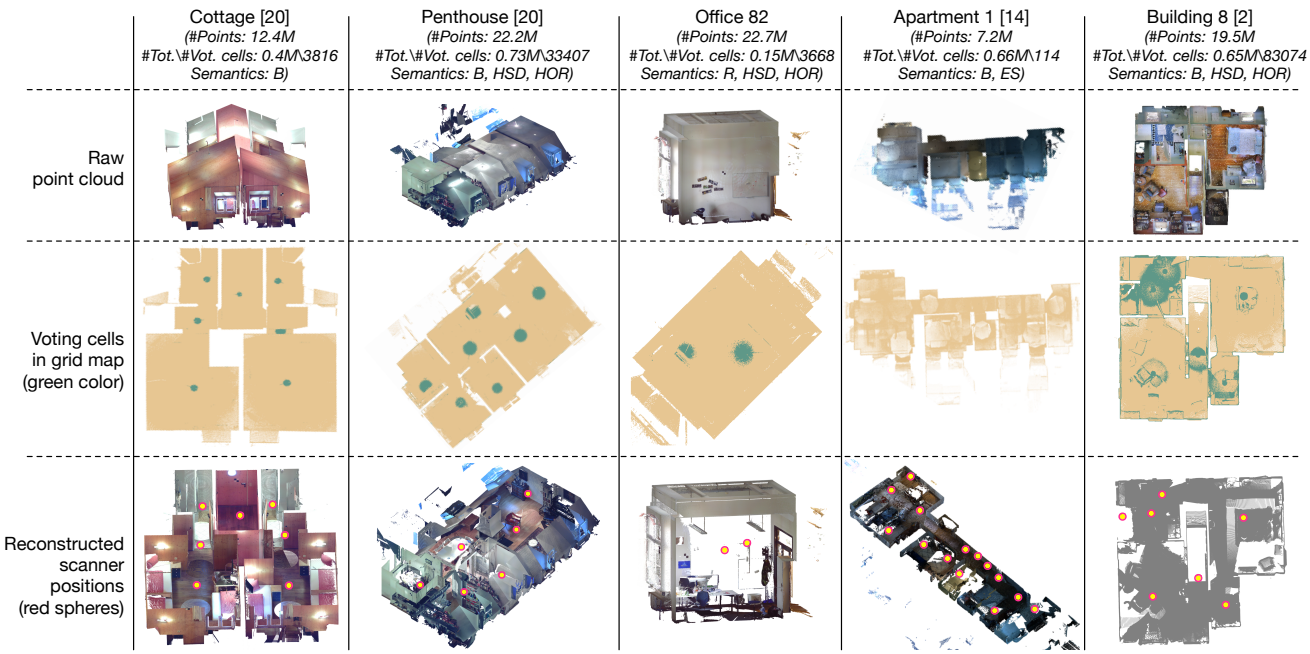


Fig. 8 Scanner position reconstruction results and statistics (no. of points, no. of total and voting cells) from real-world datasets from [38,8,9]. Semantics describe if the model is a room (R) or building (B), if it presents high scan density (HSD), high overlapping regions (HOR) or if it was extremely subsampled (ES).

the scanning pattern will be similarly distorted presenting an elliptical shape.

Another, even more challenging environment with many slanted walls, more structural details such as the window alcoves, and less regular room shapes is the 'Penthouse' [8] dataset in Fig. 8. Despite the high levels of clutter and the distortion of point pattern from the slanted walls, our method manages again to correctly reconstruct all scanner positions, as it is depicted in the bottom row of Fig. 8. The third dataset in which we evaluated our method on is the 'Office G82' (Fig. 8), which represents an exceptionally difficult setting for laser scanning since it is quite small (approx. 4.80×4.10m) and includes many interior objects and furniture that not only provide very limited space for scanning the room, but also cause a highly cluttered scene with many occluded regions. Typically, under these demanding conditions and considering the size of the room, one scan from a central position would be adequate to capture its shape and structure. However, in order to challenge and test our method under scanning conditions where scanners are placed very close to each other creating dense and highly overlapped regions, and where the environment has limited flat floor surfaces, we scanned the room twice. Nevertheless, the reconstructed scanner positions are accurately computed, as it is depicted in Tbl. 1, despite the challenges and the difficult scanning conditions. We should notice here that, in general, clutter and outliers outside of the building are effectively ignored by our method and they do not contribute to the voting process for the scanner positions. However, clutter inside the

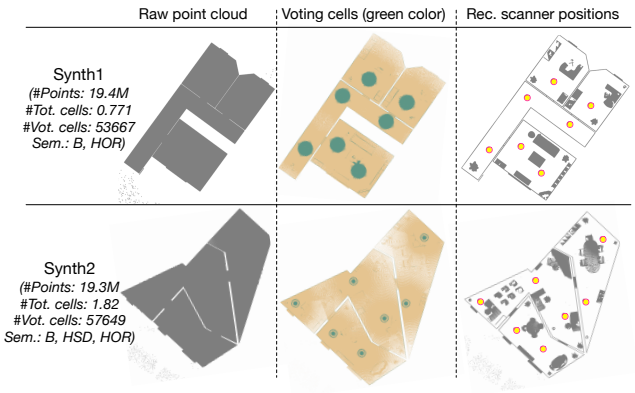


Fig. 9 Scanner position reconstruction results from the synthetic datasets [3].

building can affect the local distribution of points when projected onto the 2D plane and might distort the point pattern. Therefore, it is quite possible that this typical and uniform circular shape formed by the voting cells around the potential scanner positions, which is clearly evident in 'Cottage' and 'Penthouse' datasets in Fig. 8, might be distorted and corrupted, and only a few randomly scattered voting cells may be detected which, however, are still sufficient for our method. The datasets 'Apartment 1' and 'Building 8' in Fig. 8 constitute typical examples of such cases. Especially the 'Apartment 1' [38] building model constitutes the most challenging dataset in our test bench because the point cloud was originally subsampled and discretized, restricting our cell classification process and allowing only very few voting cells to be identified. Nonetheless, there were created

sufficient point patterns necessary for the detection of some voting cells, which consequently allowed for the localization of the scanner positions, although two of the original scanner positions were not possible to be reconstructed due to the limited number of voting cells. On the other side, the 'Building 8' [9] represents the interior of a house which was scanned with very high resolution, and therefore, many voting cells were detected (more than 10^4) in different regions of the grid map. However, due to the existence of occlusions and clutter near the LiDAR devices, the detected voting areas do not all form the typical circular shape around the scanner positions. Nevertheless, our method can estimate the actual scanner positions very accurately based on the detected voting cells, as it is verified in Tbl. 1.

In our testing setup we additionally used two synthetic datasets, which are shown in Fig. 9. These datasets were virtually scanned from several positions to simulate the results of 3D laser range scanning, as it is described in [3]. Although they were artificially corrupted with additive Gaussian noise, our method managed to correctly recognize the adequate number of voting cells in order to accurately reconstruct all scanner positions.

8 Conclusions and Future Work

In this work, to the best of our knowledge we have presented the first method for re-engineering and reconstructing the original scanner viewpoint positions from raw and merged point clouds, lifting the most widely-used assumption by recent 3D modeling methods until now that the true scanner positions should be known a priori, in order the generation of 3D BIM models to be performed successfully. Our approach allows the accurate localization of LiDAR devices under very challenging indoor environments and without requiring any prior knowledge about the environment or the dataset, while it is independent from the laser scanner manufacturers. Thus, it can be efficiently applied to merged real-world and large-scale indoor point clouds where the original scan positions are typically lost or not available. Although the performance degrades under certain conditions, such as for highly sub-sampled point clouds where the spatial point pattern analysis becomes very challenging due to the limited number of available points, our method is generally suitable for almost every real-world LiDAR application, which is verified by our evaluation.

Despite its novelty and generalized applicability, our pipeline presents also some space for further improvements. As was mentioned in Sec. 5.2, the fixed cell size introduces some restriction, and although it can be optimally set by the user according to the application requirements, it limits our method's flexibility and adaptability. Therefore, in future work we foresee to replace the fixed grid map by a quadtree or some other spatial data structure, adapting dynamically the cell size based on the available local point information. Furthermore, we would like also to explore the applicability of our method to the more general scenario of reconstructing not only the static position of terrestrial laser scanners (TLS), but also the trajectory of mobile laser scanners (MLS) and other RGB-D sensors.

Acknowledgements We acknowledge Dr. Claudio Mura, Prof. Yasutaka Furukawa, Prof. Satoshi Ikehata, Prof. Reinhard Klein and Dr. Sebastian Ochmann for the acquisition of the 3D point clouds. The 3D scanning of the datasets Cottage, Penthouse, G82, Synth1 and Synth2 has partially been supported by the EU FP7 People Programme (Marie Curie Actions) under REA Grant Agreement no. 290227.

Compliance with Ethical Standards

Conflict of interest G.-T.Michailidis declares that he/she has no conflict of interest. R. Pajarola declares that he/she has no conflict of interest.

References

1. Antonio Adan and Daniel Huber. 3D reconstruction of interior wall surfaces under occlusion and clutter. In *Proceedings Symposium on 3D Data Processing, Visualization, and Transmission*, pages 275–281, 2011.
2. Xuehan Xiong, Antonio Adan, Burcu Akinci, and Daniel Huber. Automatic creation of semantically rich 3D building models from laser scanner data. *Automation in Construction*, 31:325–337, May 2013.
3. Claudio Mura, Oliver Mattausch, Alberto Jaspe Villanueva, Enrico Gobbetti, and Renato Pajarola. Automatic room detection and reconstruction in cluttered indoor environments with complex room layouts. *Computers & Graphics*, 44:20–32, 2014.
4. Sven Oesau, Florent Lafarge, and Pierre Alliez. Indoor scene reconstruction using feature sensitive primitive extraction and graph-cut. *ISPRS Journal of Photogrammetry and Remote Sensing*, 90:68–82, 2014.
5. Mattia Previtali, Marco Scaioni, Luigi Barazzetti, and Raffaella Brumana. A flexible methodology for outdoor/indoor building reconstruction from occluded point clouds. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-3:119–126, 08 2014.
6. Adam Stambler and Daniel Huber. Building modeling through enclosure reasoning. In *Proceedings International Conference on 3D Computer Vision, Workshop on 3D Computer Vision in the Built Environment*, pages 118–125, 2014.
7. Sebastian Ochmann, Richard Vock, Raoul Wessel, and Reinhard Klein. Automatic reconstruction of parametric building models from indoor point clouds. *Computers & Graphics*, 54:94–103, February 2016.
8. Claudio Mura, Oliver Mattausch, and Renato Pajarola. Piecewise-planar reconstruction of multi-room interiors with arbitrary wall arrangements. *Computer Graphics Forum*, 35(7):179–188, October 2016.

9. Rares Ambrus, Sebastian Claici, and Axel Wendt. Automatic room segmentation from unstructured 3D data of indoor environments. *IEEE Robotics and Automation Letters*, 2(2):749 – 756, April 2017.
10. Georgios-Tsampikos Michailidis and Renato Pajarola. Automatic reconstruction of wall features under clutter and occlusion. In *In Proceedings Computer Graphics International*, 2015.
11. Georgios-Tsampikos Michailidis and Renato Pajarola. Bayesian graph-cut optimization for wall surfaces reconstruction in indoor environments. *The Visual Computer*, 33(10):1347–1355, October 2017.
12. Tianjia Shao, Weiwei Xu, Kun Zhou, Jingdong Wang, Dongping Li, and Baining Guo. An interactive approach to semantic modeling of indoor scenes with an RGBD camera. In *ACM Transactions on Graphics*, volume 31, pages 136:1–136:11, 2012.
13. Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3D semantic parsing of large-scale indoor spaces. In *Proceedings IEEE International Conference on Computer Vision and Pattern Recognition*, 2016.
14. Lyne P. Tchapmi, Chris Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. SEGCloud: Semantic segmentation of 3D point clouds. In *International Conference of 3D Vision*, 2017.
15. Victor Sanchez and Avideh Zakhor. Planar 3D modeling of building interiors from point cloud data. In *Proceedings IEEE International Conference on Image Processing*, pages 1777 – 1780, 2012.
16. Shayan Nikoohemat, Michael Peter, Sander Oude Elberink, and George Vosselman. Exploiting indoor mobile laser scanner trajectories for semantic interpretation of point clouds. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W4:355–362, 2017.
17. Wolfgang Boehler and Andreas Marbs. Investigating laser scanner accuracy. Technical report, German University FH Mainz, 2003.
18. Antonio Adan, Blanca Quintana Galera, Andres Vazquez, Alberto Olivares, Eduardo Parra, and Samuel Prieto. Towards the automatic scanning of indoors with robots. *Sensors*, 15(5):11551–11574, 2015.
19. Jingdao Chen, Yihai Fang, and Yong Cho. Unsupervised recognition of volumetric structural components from building point clouds. In *Proceedings International Workshop on Computing in Civil Engineering*, pages 34–42, 2017.
20. Jan Elseberg, Stéphane Magnenat, Roland Siegwart, and Andreas Nüchter. Comparison of nearest-neighbor-search strategies and implementations for efficient shape registration. *Journal of Software Engineering for Robotics*, 3:2–12, 2012.
21. Peter J. Diggle. *Statistical Analysis of Spatial Point Patterns*. Edward Arnold, 2nd edition, 2003.
22. Peter J. Huber and Elvezio M. Ronchetti. *Robust Statistics*. Wiley, 2nd edition, February 2009.
23. Lihua An and S. Ejaz Ahmed. Improving the performance of kurtosis estimator. *Computational Statistics & Data Analysis*, 52(5):2669 – 2681, 2008.
24. Chang-Tien Lu, Dechang Chen, and Yufeng Kou. Algorithms for spatial outlier detection. In *Proceedings IEEE International Conference on Data Mining*, pages 597–601, 2003.
25. Peter J. Rousseeuw and Christophe Croux. Alternatives to the median absolute deviation. *Journal of the American Statistical Association*, 88(424):1273–1283, 1993.
26. Ronald K. Pearson. Exploring process data. *Journal of Process Control*, 11(2):179–194, 2001.
27. Ronald K. Pearson. Outliers in process modeling and identification. *IEEE Transactions on Control Systems Technology*, 10(1):55–63, January 2002.
28. Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.
29. Emre Güngör and Ahmet Özmen. Distance and density based clustering algorithm using gaussian kernel. *Expert Systems with Applications*, 69:10–20, 2017.
30. Walter Gander, Gene H. Golub, and Rolf Strebel. Least-squares fitting of circles and ellipses. *BIT Numerical Mathematics*, 34(4):558–578, Dec 1994.
31. Sung Joon Ahn, Wolfgang Rauh, and Hans-Jürgen Warnecke. Least-squares orthogonal distances fitting of circle, sphere, ellipse, hyperbola, and parabola. *Pattern Recognition*, 34(12):2283 – 2303, 2001.
32. Giuseppe Calafiore. Approximation of n-dimensional data using spherical and ellipsoidal primitives. *IEEE Transactions on Systems, Man, and Cybernetics*, 32(2):269 – 278, March 2002.
33. Jieqi Yu, Zheng Haipeng, Sanjeev R. Kulkarni, and H. Vincent Poor. Two-stage outlier elimination for robust curve and surface fitting. *EURASIP Journal on Advances in Signal Processing*, 2010.
34. Abdul Nurunnabi, Yukio Sadahiro, and Roderik Lindenbergh. Robust cylinder fitting in three-dimensional point cloud data. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-1/W1:63–70, 2017.
35. Bastian Leibe, Ales Leonardis, and Bernt Schiele. Combined object categorization and segmentation with an implicit shape model. In *European Conference on Computer Vision Workshop on statistical learning in computer vision*, pages 17–32, 2004.
36. Juergen Gall, Angela Yao, Nima Razavi, Luc Van Gool, and Victor Lempitsky. Hough forests for object detection, tracking, and action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(11):2188 – 2202, 2011.
37. Oliver J. Woodford, Minh-Tri Pham, Atsuto Maki, Frank Perbet, and Björn Stenger. Demisting the hough transform for 3D shape recognition and registration. *International Journal of Computer Vision*, 106(3):332–341, 2014.
38. Satoshi Ikehata, Hang Yan, and Yasutaka Furukawa. Structured indoor modeling. In *Proceedings IEEE International Conference on Computer Vision*, 2015.

DS	$X_{rec}(\Delta X$‰$)$	$Y_{rec}(\Delta Y$‰$)$	$Z_{rec}(\Delta Z$‰$)$	ΔE
G82	2.99925 (.0243)	2.9981 (.0067)	2.4171 (.0248)	0.000966
	1.48286 (.0074)	2.98112 (.0017)	1.65184 (.0979)	0.001622
Cottage [8]	7.72914 (.0015)	4.660471 (.0163)	1.04721 (.4227)	0.004475
	3.702994 (.0084)	5.300328 (.0025)	1.0173 (.3674)	0.003766
	5.863674 (.0092)	5.661893 (.0097)	1.0235 (.1659)	0.001862
	7.925154 (.0048)	6.10037 (.0265)	1.01933 (.5128)	0.005459
	4.34194 (.0109)	2.092657 (.1234)	1.03109 (.5736)	0.006440
	7.04589 (.0062)	2.937977 (.0242)	1.0278 (.2636)	0.002828
	8.620154 (.0160)	3.706677 (.0426)	1.01553 (.4180)	0.004719
Apartment 1 [38]	16.4417 (.3533)	9.668548 (2.3012)	1.8127 (.2711)	0.225221
	15.593 (.6255)	11.1643 (.6990)	1.81342 (.0124)	0.124093
	12.3648 (4.2750)	12.5321 (4.3340)	1.81441 (.0037)	0.726620
	14.8922 (.5565)	8.3903 (1.1823)	1.81483 (.0133)	0.128079
	13.367 (.5598)	9.3191 (.5111)	1.81733 (.0175)	0.088486
	11.4141 (1.4649)	8.3232 (2.5214)	1.82503 (.2569)	0.265929
	9.09904 (.3434)	6.8956 (.6992)	1.82785 (.3720)	0.057519
	8.4232 (.3890)	8.2315 (.0937)	1.81269 (.3582)	0.034170
	10.016 (.0454)	9.8505 (.7663)	1.815732 (.0206)	0.075051
	6.9295 (.1644)	7.7261 (.0831)	1.82178 (.0215)	0.013070
	7.3192 (.1307)	5.6604 (.7732)	1.82534 (.0005)	0.044470
	3.9552 (1.3687)	4.0757 (1.4437)	1.81625 (.0562)	0.078849
	2.6506 (.7374)	3.0341 (.9258)	1.81588 (.0721)	0.033953
	3.0647 (.8524)	5.9195 (.0770)	1.8285 (.3107)	0.026903
Building 8 [9]	6.06468 (.0115)	5.0339 (.0119)	1.1762 (.0595)	0.001158
	1.81368 (.0165)	8.6339 (.0069)	1.1751 (.0340)	0.000781
	3.91417 (.0051)	7.7825 (.0103)	1.1757 (.0170)	0.000849
	2.36437 (.0169)	3.8862 (.0747)	1.1758 (.0255)	0.002943
	5.16342 (.0108)	9.0837 (.0044)	1.1761 (.0510)	0.000913
	1.81391 (.0039)	4.7335 (.0042)	1.175 (.0425)	0.000543
	0.51311 (.1693)	4.83321 (.0019)	1.1749 (.0510)	0.001061
Penthouse [8]	4.498257 (.0371)	1.799142 (.0209)	2.2775 (.0923)	0.002709
	3.224586 (.0164)	4.354123 (.0298)	2.2726 (.0629)	0.002001
	4.643164 (.0128)	7.28859 (.0031)	2.319 (.0933)	0.002253
	1.662534 (.0714)	7.551153 (.0645)	1.9251 (.1795)	0.006089
	4.588724 (.0025)	10.90727 (.0019)	2.2672 (.0147)	0.000407
	2.432537 (.0448)	9.896112 (.0107)	1.9015 (.0018)	0.001518
	6.141467 (.0093)	5.685274 (.0015)	1.9768 (.0152)	0.000650
Synth1 [3]	2.9052 (.0358)	4.4681 (.0134)	1.006 (.6000)	0.006119
	0.90426 (.0111)	2.6691 (.0150)	1.0043 (.4300)	0.004320
	0.90443 (.0299)	8.0683 (.0050)	1.0024 (.2400)	0.002448
	5.10454 (.0074)	8.36823 (.0056)	1.0011 (.1100)	0.001255
	5.60412 (.0007)	10.6681 (.0056)	1.0103 (1.0300)	0.010318
	1.90427 (.0058)	10.4676 (.0105)	1.0022 (.2200)	0.002462
	5.60397 (.0034)	3.6693 (.0164)	1.0003 (.0300)	0.000697
Synth2 [3]	10.4764 (.0057)	5.9641 (.0067)	1.5063 (.4200)	0.006341
	7.2281 (.0152)	8.9639 (.0067)	1.5032 (.2133)	0.003437
	2.9788 (.0605)	13.4651 (.0045)	1.4978 (.1467)	0.002905
	10.7281 (.0103)	15.2148 (.0020)	1.4921 (.5267)	0.007982
	13.9779 (.0064)	9.9641 (.0040)	1.5091 (.6067)	0.009153
	16.9767 (.0018)	11.4651 (.0052)	1.4993 (.0467)	0.000970
	20.4764 (.0029)	19.4634 (.0057)	1.50012 (.0080)	0.001259

Table 1 The reconstructed scanner position coordinates (in m), their deviation from the original ones (in ‰), and the absolute distance error ΔE (in m).