DATA-DRIVEN MODELING OF BUILDING INTERIORS FROM LIDAR POINT CLOUDS

J. Sanchez[1], F. Denis[1], F. Dupont[1], L. Trassoudaine[2], P. Checchin[2,*]

[1] Univ Lyon, LIRIS, UMR 5205 CNRS, Université Claude Bernard Lyon 1,
43, bd du 11 novembre 1918, 69622 Villeurbanne CEDEX, France
(julia.sanchez, florence.denis, florent.dupont)@univ-lyon1.fr
[2] Institut Pascal, UMR 6602, Université Clermont Auvergne, CNRS, SIGMA Clermont, F-63000 Clermont-Ferrand, France
(laurent.trassoudaine, paul.checchin)@uca.fr

KEY WORDS: Point cloud, reconstruction, modeling, building interiors

ABSTRACT:

This paper deals with 3D modeling of building interiors from point clouds captured by a 3D LiDAR scanner. Indeed, currently, the building reconstruction processes remain mostly manual. While LiDAR data have some specific properties which make the reconstruction challenging (anisotropy, noise, clutters, etc.), the automatic methods of the state-of-the-art rely on numerous construction hypotheses which yield 3D models relatively far from initial data. The choice has been done to propose a new modeling method closer to point cloud data, reconstructing only scanned areas of each scene and excluding occluded regions. According to this objective, our approach reconstructs LiDAR scans individually using connected polygons. This modeling relies on a joint processing of an image created from the 2D LiDAR angular sampling and the 3D point cloud associated to one scan. Results are evaluated on synthetic and real data to demonstrate the efficiency as well as the technical strength of the proposed method.

1. INTRODUCTION

Since the 1990's, architects and civil engineering professionals use digital models to perform simulations and accurate computations before construct a building. The new challenge is to obtain as-built models from existing buildings, which would notably be useful in case of renovations. For this purpose, the LiDAR technology is mostly used to scan building interiors because of its accuracy and its efficiency. In this work, we are particularly interested in reconstructing the envelopes of the scanned scenes. The building structure is commonly composed of a main set of polygons which are easy to represent in 3D. The major scientific lock to the modeling of this structure lies in the accurate location of the planes, the polygonal contours and the polygons' connections. The difficulty of this task comes mostly from the acquisition faults of a LiDAR device *i.e.,* sampling anisotropy, measurement noise, occlusion problems, etc.

In view of the planar feature of the building interiors, the polygonal and polyhedral reconstructions seem to be the most suitable. They are notably easy to transfer to a BIM (*Building Information Model*) format used in the field of civil engineering. Both are based on a precise segmentation of planar primitives in the scene (Macher et al., 2017) (Boulch, Marlet, 2016) (Ochmann et al., 2016). The most known method to extract planar primitives in point clouds, namely Hough transform, RANSAC and region growing, suffer from the anisotropy of sampling in the 3D environment, and require complex configuration. Then, the primitives are generally extrapolated to recover a watertight envelope. However, this extrapolation can lead to false reconstructed areas.

In this work, we aim at reconstructing indoor building envelopes as a piecewise planar structure. A new method of polygonal modeling from point clouds is then proposed. Like various authors (Chauve et al., 2010) (Boulch, Marlet, 2016), we choose to work on individual scans after capture process to maintain the link with the 2D angular sampling of the LiDAR. The method relies on the estimation of normals in order to achieve 2D region growing allowing to extract the plane primitives and their contours directly in an image. An intersection study is then carried out in order to classify the interactions between planes (occlusions or connections) and the contours are modeled by a set of line segments taking account of openings and clutters. Finally, the set of connected polygons with holes is then represented in the form of a mesh.

The existing methods in the field of interior scene modeling is first detailed in Section 2. Then, our approach to model a scan is presented in Section 3. Finally, the method is evaluated on real data (Section 4) before concluding on the advantages and drawbacks of such modeling.

2. RELATED WORK

Building modeling from point clouds generally breaks down into two distinct tasks: segmentation, which corresponds to labeling of points according to the sampled object, and adjustment of geometric models to points. These two tasks are carried out by a multitude of different methods which vary according to:

- the reconstructed object: a complete building, a building floor, a room or a scan;
- initial data: 3D coordinates, color, normal, photographs, individual or multiple registered scans, etc.
- the desired accuracy;
- the initial geometrical hypotheses;
- special requirements: wateriness, modeling openings, etc.

Consequently, the following paragraphs draw up a non-exhaustive inventory of the tasks achievable in a processing chain aiming at reconstructing building interiors in different contexts.

* Corresponding author

2.1 Breakdown into subsets of points

Building modeling can be carried out from a point cloud integrating all the scans from the different poses of the scanner, after registration. In this context, a few methods propose to decompose the global cloud into sub-clouds which can be segmented and/or modeled independently. (Huber, 2011), (Oesau et al., 2014), (Khoshelham, Díaz-Vilariño, 2014) and (Macher et al., 2017) break down the point cloud into floors. (Ochmann et al., 2016) and (Macher et al., 2017) further decompose the point cloud into rooms. Some authors also filter the cloud to remove occluding objects (Oesau et al., 2014, Sanchez, Zakhor, 2012). These methods are based on geometric priors of the scan (verticality, walls width) and are sensitive to anisotropy of the 3D point clouds. Here, we will prefer to work on LiDAR scans directly to address these issues.

2.2 Envelope element detection

In order to be modeled, the elements of the envelope of a room must be detected, *i.e.,* the points of the different walls, floor and ceiling must be labeled. To do this, most methods seek to detect planes in scenes.

2.2.1 Detection of 3D planes In a majority of building reconstruction methods, the search for planes is carried out in the point cloud. The classic 3D Hough transform (Duda, Hart, 1971) can be used. It consists in creating a surface of potential planes' parameters for each point and to build a 3D histogram (accumulator) from these parameters to detect the surface intersections while taking account of noise. However, this approach is slow and encounters discretization problems which make it dependent on the chosen bin width and on the orientation of the histogram. Some extensions have been introduced to solve such issues (Yl-Jski, 1994), (Borrmann et al., 2011).

The RANSAC paradigm (Fischler et al., 1981) consists in fitting planes on triplets of points randomly extracted from the cloud. Then, the plane corresponding to the most points (inliers) relatively to a given distance threshold, is selected. This method has been extended taking account of normal information (Bretar, Roux, 2005) or constraining the models to find horizontal or vertical planes (Thomson, Boehm, 2015). (Torr, Zisserman, 2000) introduce MSAC (*M-estimator SAmple Consensus*) and MLESAC (*Maximum Likelihood Estimation SAmple Consensus*) in which the selection criterion of the plane is replaced to depend on the residues of the inliers relatively to the studied model. This kind of methods is efficient, fast and widely used in an indoor scene reconstruction context. (Tarsha-Kurdi et al., 2007) assert that RANSAC has better noise management and greater efficiency than the 3D Hough algorithm. A combination of the RANSAC and Hough methods is proposed by (Xu et al., 1990) in their method called Randomized Hough Transform (RHT).

3D region growing on an image is another solution. It consists in starting from a seed pixel and in growing a region pixel by pixel according to a criterion related to the selected seed. Region growing has been performed on range images (Besl, Jain, 1988) (Fitzgibbon et al., 1997). It has been extended to 3D point clouds by defining a 3D local neighborhood (Chauve et al., 2010) (Xiong et al., 2013). However, a neighborhood in a point cloud is difficult to define, mainly because of the non-uniformity of the sampling. In both cases, the criterion generally used to enlarge the regions is the residue of neighboring points relatively to the local plane defined on the seed (Pu, Vosselman, 2006, Poppinga et al., 2008), but the similarity of the normals can also be taken into account (Deschaud, Goulette, 2010, Poullis, You, 2009, Boulch et al., 2014). The main limitation of region growing and RANSAC is the complexity of their configuration in order to limit the overlap of one region on another, while taking into account the effect of measurement noise and normals estimation.

In the indoor reconstruction context, some authors base their approach on the assumptions of the Manhattan world (Coughlan, Yuille, 1999). (Budroni, Boehm, 2010) add the hypothesis that the floor and the ceiling are parallel to the $\{x, y\}$ plane, using a sweeping procedure (translating and rotating planes). (Vanegas et al., 2012) use these assumptions to gather the points according to their spatial proximity and their neighborhood similarity to local primitives (planes, 90° edges and 90° corners). Although the effectiveness of these methods has been proven, the construction assumptions on which they are based reduce their applicability to real data.

2.2.2 2D plane detection A possible assumption to make is that the walls are perpendicular to the ground plane which is horizontal. The 3D plane search problem can then turn into a search for 2D lines after projection of the points onto the horizontal plane. (Macher et al., 2017) perform 2D straight line detection on projections using MLESAC 2D to guide subsequent wall detection by MLESAC 3D. (Oesau et al., 2014) fit locally the best of two possible models: one line, or an intersection of two lines. Then, they correct the resulting local lines by a bilateral filter and group them by a region growing. This method has the advantage of being able to detect fine planes and to reconstruct curved wall under the shape of a set of fine vertical planes.

2.3 Modeling

2.3.1 Polygonal modelisation The envelope elements can be modeled independently with polygons. To do this, the characteristics of each underlying plane are computed and the outline of the polygon is detected and modeled. Some authors choose to directly use the RANSAC algorithm (Thomson, Boehm, 2015) (Sanchez, Zakhor, 2012). Then, the first ones use the convex hull, while the second ones prefer the α*-shape* to detect and model the outline of the polygon. The α*-shape* of a point cloud is a hull closer to the data than the convex hull. However, it is sensitive to the sampling anisotropy and its configuration can be complex (Boulch et al., 2014). With the same objective, (Boulch et al., 2014) prefer to directly work on the 2D angular frame used by the LiDAR sensor. The detected primitives can then be represented on an image and local lines are detected in the form of pixels. (Macher et al., 2017) detect the vertical edges as the ends of segments in the 2D projection of the wall onto the floor. The other edges are modeled as intersections. These methods do not study the interactions between planes and lead to independent polygonal modeling for each plane.

2.3.2 Polyhedral modelisation To recover these interactions, a majority of methods seek to directly reconstruct a polyhedron by locating its faces on an arrangement of lines or planes. The classical method described by (Budroni, Boehm, 2010) consists in projecting the walls in the horizontal plane in the shape of line segments. Then, the segments are extended until they intersect the bounding box. The points of the cloud are also projected in the arrangement and the cells are

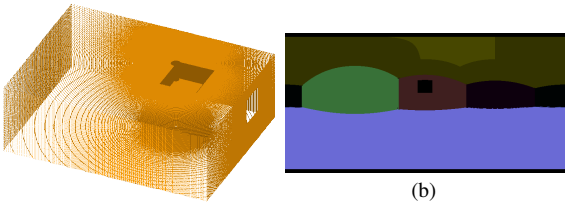
labeled according to their occupation by points or not. The segments separating two cells of different labels are then extruded to model the walls. Similarly, (Ochmann et al., 2016) work on registered point clouds and seek the walls separating rooms in 2D. To do so, they optimize a cost function to label the cells per room. Other authors are working on 3D plane arrangements. These arrangements can be constructed by extruding 2D arrangements (Oesau et al., 2014) or by extending 3D planar primitives (Chauve et al., 2010) (Boulch et al., 2014). As (Ochmann et al., 2016), the authors cited above propose to carry out an optimization which allows to label the cells accordingly to the measured points (with ray tracing (Oesau et al., 2014) or visibility constraints (Boulch et al., 2014)) while limiting the complexity of the surface. These methods make it possible to obtain a watertight, credible reconstruction. However, they are sensitive to the anisotropy of the sampling and the authors use weighting mechanisms to overcome this problem (Oesau et al., 2014) (Ochmann et al., 2016) (Boulch et al., 2014). Moreover, many artifacts can appear in the reconstruction (forgetting a wall, copies, false intersections) and openings are not modeled.

Aware of the shortcomings of the methods listed above, we wish to propose a new approach to model indoor scenes with polygons, solving such issues and yielding information on interactions between surface pieces (occlusions, or connections). The method should allow the modelisation of openings and lead to minimal extrapolation of LiDAR data.

3. INDOOR SCENE MODELING

3.1 Overview

The LiDAR device captures the world around its position with distance measurements for a collection of rays w.r.t. two orientation angles φ and θ. The objective of our method is to take advantage of the regularity of the sampling in this 2D coordinate system to improve the point cloud segmentation in the 3D space. Indeed, the point cloud can also be represented under the form of an image; φ being associated with rows and θ being associated with columns. We call these data $\{\varphi, \theta\}$ images. Each 3D point of the dataset is associated with a pixel. Figure 1 presents such an image, in which the colored elements $\{R, G, B\}$ of a pixel q_i express the values $\{d_{\mathbf{p}_i}\mathbf{n}^x_{\mathbf{p}_i}, d_q\mathbf{n}^y_{\mathbf{p}_i}, d_{\mathbf{p}_i}\mathbf{n}^z_{\mathbf{p}_i}\}$ where $\mathbf{n}_{\mathbf{p}_i}$ is the normal of the local plane adjusted in $\mathbf{p}_i$ and $d_{\mathbf{p}_i}$ is its distance to the origin. This image identifies polygons as groups of pixels of similar colors. Note that the points which never returned to the source correspond to empty pixels (black pixels in figure). The other are qualified as "full" pixels. Hence, a segmentation of this image



(a) (b)

Figure 1. Simulated capture of a room by a LiDAR laser scanner. (a) Point cloud. (b) $\{\varphi, \theta\}$ image.

is carried out to detect these groups and their contour lines as a set of pixel/point pairs. The modeling of the polygons is then carried out in two stages. First, straight line segments are fitted to the outline and secondly, the vertices of the polygons are deduced by analyzing the intersections of these segments. All the method parameters defined in the following are summarized in Table 1, with their values used in all of our tests.

3.2 Image segmentation into planar surfaces

Our goal is to segment the planar primitives from the $\{\varphi, \theta\}$ image. We call Π the set of planar primitives. A primitive π_k belonging to Π and modeling a subset of points in the cloud $\mathcal{S}$, is defined by:

- P^{π_k}: a set of 3D points;
- Q^{π_k}: the corresponding pixels;
- $\mathbf{n}^{\pi_k}$: the normal to π_k;
- d^{π_k}: its distance from the origin.

We want to detect these primitives and to compute their features using a regions growing process. Like (Boulch et al., 2014), we choose to achieve the region growing on the image considering the neighbors residues relatively to the local plane defined on the seed with the threshold τ_d. The regions are sets of pixels. Each seed g^{r_k} is defined by its pixel/point pair $q^{g^{r_k}}/\mathbf{p}^{g^{r_k}}$ and its local plane, $\pi^{g^{r_k}}$. To select a seed, we test all the pixels of the scan successively per line and per column. p_i is selected to create a seed if q_i does not belong to any region, the maximum angular difference between the normal in p_i and the neighboring normals is less than a threshold τ^g_α, and if the maximum residue of a p_i's neighbors relatively to its local plane is less than a threshold τ^g_d. The neighborhood is defined by the 8 pixels surrounding q_i, and $\pi^{g^{r_k}}$ is determined by averaging the 3D points and normals in q_i's neighborhood.

A problem relating to the use of a region growing is that, depending on the chosen residual error, a region may partially overlap another one. Moreover, a region can propagate on a surface of different orientation if points are found aligned with the studied region. (Boulch et al., 2014) propose to add a new threshold on the resemblance of the local normals of the points with the seed to mitigate this problem but that can complicate the parameters setting of the algorithm that we want to keep as simple as possible. We use a variant to the region growing in order to resolve these ambiguities: after a region has been segmented, its pixels are removed from the list of potential seeds, but they are brought into play again in the growth of the following regions. Hence, a pixel can be selected in several regions. The points are then reallocated to the regions with the most similar local planes *i.e.,* the region corresponding to the smallest angular deviation between the studied point normal and the seed's local plane normal. This method allows the region growing to cover a maximum of pixels of the image, it prevents overlapping of regions corresponding to intersecting planes while alleviating the algorithm configuration.

After segmenting the planar regions R, the set of primitives Π can be determined. For each region r_k, a primitive π_k is created: Q^{π_k} contains the pixels of r_k, P^{π_k} is the set of corresponding 3D points, $\mathbf{n}^{\pi_k}$ is determined by PCA on the whole set P^{π_k} and finally, d^{π_k} is the scalar product of P^{π_k}'s centroid with $\mathbf{n}^{\pi_k}$. When 3D points of a region are distant from the seed, the angular error of the local plane normals implies strong points' residues and several regions can then be detected on the same planar primitive. Figure 2 shows a schematic example of such a situation. Therefore, the characteristics of the different segmented planes are compared and, if they are close together and the regions have pixels in common, the planes are merged: their

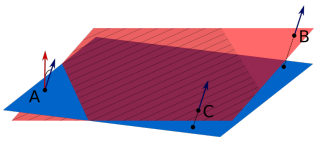


Figure 2. Representation of a (hatched) region grown on a red flat surface from the seed A associated with the inaccurate blue local plane. B is wrongly discarded from the region. Similarly, another region will be grown from B in which A will be discarded.

points and pixels are pooled and a new pair of characteristics (normal and distance) is computed.

At the end of this process, an image is obtained where all the planar regions are segmented. The points which do not belong to any region are assumed to belong to non-planar objects. The image is then filtered to eliminate false detections of non-planar objects due to the error of normals estimation and to the measurement noise. If the objects contain less than S_{min} pixels, they are reassigned to the nearest plane in the image. An example of these processes result is given in Figure 3 for the cloud presented in Figure 1(a). The colors correspond to the regions.

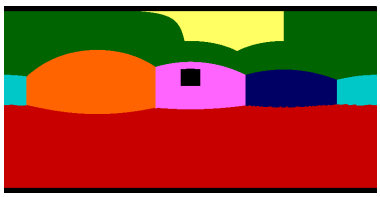


Figure 3. Segmentation of the $\{\varphi, \theta\}$ image represented Figure 1(b)

3.3 Contour line detection

In order to model polygons, we want to detect the edges of the planar primitives previously extracted, and to label them. A contour line can either separate two primitives, indicate an opening or the presence of a non-planar object. Examples of contour lines are shown in Figure 4. The set of contour lines is called $\mathcal{L}$. Each contour line ℓ_l belonging to $\mathcal{L}$ is defined by:

- P^{ℓ_l}: a set of 3D points;
- Q^{ℓ_l}: the set of corresponding pixels;
- Π^{ℓ_l}: a set of interacting primitives. $\Pi^{\ell_l} \subset \Pi$;
- e^{ℓ_l}: a label which can be: "interaction with primitive", "opening", or "interaction with object".

We consider here the study of the contour of a particular primitive named π, and we note π^C the complement of π.

All the pixels Q^π are isolated in a binary image (see Figure 5(a)). The edge pixels of π are then extracted from the image along with their closest neighbors in π^C. This new set of pixels is named Q^π_{bound}. To create each contour line, one must group the pixels Q^π_{bound} corresponding to the same element (a neighboring planar primitive, a non-planar object or an opening). An example of such a grouping is given in Figure 5(b), the labels, corresponding to colors, identify the object interacting with the studied primitive. For π, a group of contour pixels extracted from Q^π_{bound} allows to create a line ℓ as follows:

- Q^ℓ is the group of pixels and P^ℓ their corresponding 3D points;
- Π^ℓ contains the plane and the interacting plane where applicable;
- The label e^ℓ is selected accordingly to the neighborhood. If the pixels are close to another plane, the label is "interaction with primitive"; if the pixels have no full neighbor, the label is "opening"; otherwise, the label is "interaction with object".

It is to note that if ℓ is an interaction between primitive, then $\#\Pi^\ell = 2$ otherwise $\#\Pi^\ell = 1$.

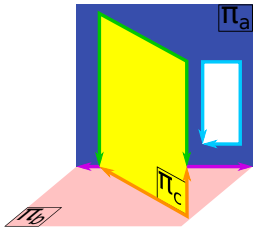
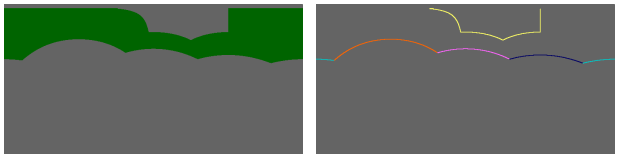


Figure 4. Example of contour lines for three interacting planar primitives. The orange line represents the interaction of the primitives π_b and π_c, the green line represents the interaction of π_a and π_c, the purple line represents the interaction of π_a and π_b, and the blue line shows the opening in the primitive π_a.



(a) (b)

Figure 5. Outline detection in a planar primitive. (a) Planar primitive extracted from $\{\varphi, \theta\}$ image, (b) labeled contours

3.4 Contour line modeling

We assume that the contour lines can be modeled by a set of straight line segments. Therefore, we will try to adjust these models and label them. An example of contour line modeling is given in Figure 6. The contour lines of the plane π_c (on the left), are modeled by segments (on the right). The set of straight line segments modeling the contour lines is named Δ. A straight line segment δ_d belonging to Δ is defined by:

- $\mathbf{t}^{\delta_d}$: its guiding vector;
- $\{\mathbf{p}^{\delta_d}_{init}, q^{\delta_d}_{init}\}$ and $\{\mathbf{p}^{\delta_d}_{fin}, q^{\delta_d}_{fin}\}$: the point/pixel pairs of its ends;
- ℓ^{δ_d}: its referent contour line, $(\ell^{\delta_d} \in \mathcal{L})$;
- P^{δ_d}: the set of 3D points modeled, $(P^{\delta_d} \subset P^{\ell^{\delta_d}})$;
- Q^{δ_d}: the set of corresponding pixels, $(Q^{\delta_d} \subset Q^{\ell^{\delta_d}})$;
- e^{δ_d}: a label which can be: "occlusion", "intersection", "interaction with object" or "opening".

The segments are built successively, after a segment δ has been computed, the points P^δ and the pixels Q^δ are removed from P^ℓ and Q^ℓ respectively, and new segments are sought in the remaining points. Depending on their nature, the contour lines are modeled by different methods. An additional labeling step is necessary in order to separate the lines containing only occlusion segments from lines containing an intersection segment. We carry out a connectivity test: for a line ℓ of this type, we determine the pixels which could be attributed to an intersection

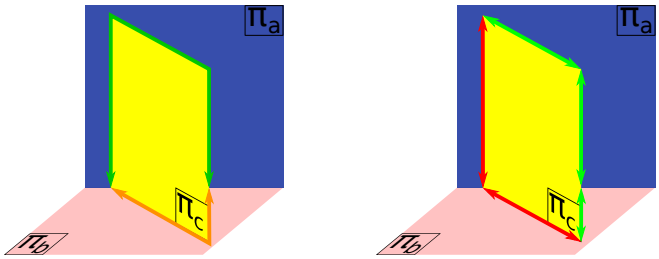


Figure 6. Contour lines and modeled straight line segments. (a) Contour lines of π_c interacting with π_a and π_b. (b) Straight line segments to model contour. The green segments correspond to occlusions. The red segments correspond to intersections

between the planes of the set Π^ℓ. We call this new set of potential pixels $Q^{\ell *}$. Figure 7 gives an example of the result obtained for the contour of the planar primitive studied (see Figure 5). If the number of pixels included in $\{Q^\ell \cap Q^{\ell *}\}$ is sufficient, the line must contain an intersection, otherwise, we deduce that it is a set of occlusion segments. In the example of Figure 7, most of the contour lines have pixels in common with their theoretical intersection lines and are therefore labeled as intersections, except the yellow line which results only from an occlusion.

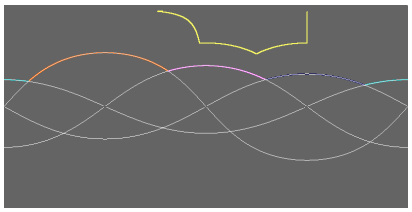


Figure 7. Theoretical lines (white) for connectivity test. The colored lines compose the primitive outline detected.

3.4.1 Intersection segment After detecting a line segment of intersection δ by this connectivity test, its characteristics can be computed as follows:

- the guiding vector $\mathbf{t}^\delta$ is deduced by cross product from the normals of the planes;
- Q^δ is defined by $Q^\delta = Q^{\ell^\delta} \cap Q^{\ell^\delta *}$. P^δ contains the corresponding 3D points;
- To determine $\mathbf{p}^\delta_{init}$ and $\mathbf{p}^\delta_{fin}$, one computes the values of projections of all the segments corresponding points along the directing vector $\mathbf{t}^\delta$. The minimum and maximum of these values correspond to the projections of $\mathbf{p}^\delta_{init}$ and $\mathbf{p}^\delta_{fin}$, respectively.

3.4.2 Opening segment In order to model an opening segment δ, we seek to adjust a line on the points of a line ℓ labeled as "opening". For this, we use a 2D RANSAC procedure. In fact, the points of the opening belong to a single plane ($\Pi^\ell = \{\pi^\ell\}$). Therefore, the search for lines takes place in the coordinate system of this plane. RANSAC is carried out with a maximum number of tests M and a membership threshold to the straight line named τ_{delta}. A maximum distance between two consecutive points of the same segment is configured simultaneously in pixels at the value of τ^q_{diff} and in spatial distance at the value of τ^p_{diff}. The previously defined threshold S_{min} is also used to define a minimum number of pixels in a segment. A least squares refinement is finally performed on the inlier points. Each set P^δ is then defined by the points at a distance less than τ_δ the line adjusted by least squares. The ends ($\mathbf{p}^\delta_{init}$ and $\mathbf{p}^\delta_{fin}$) of the segment are computed by projection of the points P^δ on the line obtained and by selection of the minimum and maximum projection values respectively. After this step, the segments are finally replaced in 3D.

3.4.3 Occlusion segment Unlike an opening line, a contour line containing an occlusion has 3D points on two separate planes ($\Pi^\ell = \{\pi^\ell_1, \pi^\ell_2\}$). We then seek to model from the same contour line pairs of segments, the first of which is in π^ℓ_1 and the second of which is in π^ℓ_2. To do this, a first segment δ_1 is adjusted on the points P^ℓ belonging to π^ℓ_1 using, as before, a 2D RANSAC procedure, coupled with a least squares adjustment. Then, a second segment δ_2 is adjusted on the points P^ℓ belonging to π^ℓ_2 and whose pixels are close to the pixels modeled by δ_1. Then, in each pair of segments $\{\delta_1, \delta_2\}$, δ_1 is projected onto the plane π^ℓ_2 and δ_2 is projected onto the plane in the direction of the laser beam from the sensor. An example of this kind of projection is given in Figure 8. The adjusted segments are represented by a solid line and the projected segments are represented by dotted lines. Then, for each plane, an average is computed between the features of the detected segment and the projected one, in order to obtain two final segments, one in the plane π^ℓ_1 and the other in the plane π^ℓ_2.

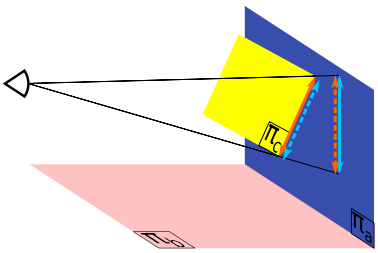


Figure 8. Detection of occlusion straight line segments from contour lines of interaction between planes. The solid lines represent the segments detected by RANSAC 2D on the planes and the dashed lines represent their projections on the other plane

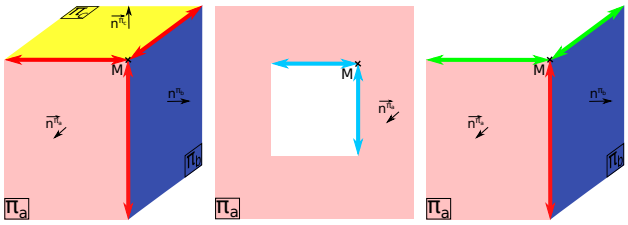
For the interactions with non-planar objects, the difference with the preceding modeling resides in the fact that the outline of the object is not modeled.

3.5 Corner modeling

The last step is to connect the line segments to obtain closed polygons. For this, we are looking for the vertices of the polygons. We call $\mathcal{C}$ the set of all the corners in the scene. A corner c_n belonging to $\mathcal{C}$ is defined by:

- $\{\mathbf{p}^{c_n}, q^{c_n}\}$: a pair 3D point/pixel;
- Δ^{c_n}: a set of line segments ($\Delta^{c_n} \subset \Delta$);
- e^{c_n}: a label which can be: intersection of three planes, intersection of two lines, continuity of two lines.

The different types of corners defined by their labels are illustrated in Figure 9. A corner is then computed depending on its label. The algorithm consists, first, in selecting a set of potential corners $\mathcal{C}^*$ relative to the set of line segments Δ. Then, these potential corners are compared with the measured ends of the segments in order to be selected or not in the set $\mathcal{C}$. This process is described in detail below.



(a) (b) (c)

Figure 9. Kinds of possible corners. (a) intersection of three planes ; (b) intersection of two coplanar straight lines ; (c) intersection of three straight lines.

3.5.1 Intersection of three planes We select the pairs of line segments labeled as intersections $\{\delta_\alpha, \delta_\beta\}_{\delta_\alpha, \delta_\beta \in \Delta}$ that have a plane in common. This common plane is named $\pi^{\delta_{\alpha,\beta}}$. The union of the sets Π^{δ_α} and Π^{δ_β} then contains three planes, named $\{\pi^{\delta_\alpha}, \pi^{\delta_\beta}, \pi^{\delta_{\alpha,\beta}}\}$. A new potential corner c^* is created whose point $\mathbf{p}^{c^*}$ is the intersection of these three planes. The segments δ_α and δ_β are then contained in Δ^{c^*}. Finally, we seek if a third segment represents the intersection between the planes π^{δ_α} and π^{δ_β}. If this segment exists, it is added to Δ^{c^*}.

3.5.2 Intersection of two lines The intersection of two lines corresponds to pairs of segments $\{\delta_\alpha, \delta_\beta\}_{\delta_\alpha, \delta_\beta \in \Delta}$ which have a common plane and of which at least one of the segments is not an intersection. This plane is then named $\pi^{\delta_{\alpha,\beta}}$. A new potential corner c^* is created whose point $\mathbf{p}^{c^*}$ is the intersection of the straight lines of these segments in the plane $\pi^{\delta_{\alpha,\beta}}$. The set of line segments associated with is then $\Delta^{c^*_n} = \{\delta_\alpha, \delta_\beta\}$.

3.5.3 Continuity of two lines When a primitive occludes different other primitives, the same edge can be divided into several consecutive segments (the green segments in Figure 6(b), are an example of such a case). These different segments are reconnected. For this, for each pair of segments $\{\delta_a, \delta_b\}$, if they have a plane in common, if they meet a parallelism constraint ($\arccos(|\mathbf{t}^{\delta_a} \cdot \mathbf{t}^{\delta_b}|) < \tau_{//}$) and if their ends are close (according to the threshold distance τ_c^p), then, a potential corner is created and added to $\mathcal{C}^*$ to link the two segments.

3.5.4 Selection of corners and fusion For a line segment δ, we extract the potential corners whose corresponding set of segments contains δ (*i.e.,* for a corner c^*, $\delta \in \Delta^{c^*}$). We then compute the distances between the point $\mathbf{p}^{c^*}$ and the ends of δ ($\mathbf{p}^\delta_{init}$ and $\mathbf{p}^\delta_{fin}$). After evaluating all the 3D distances between ends and potential corners, for each end, the corner c^* for which the distance is the smallest is selected, *i.e.,* the one $\mathbf{p}^{c^*}$ with the closest point. If this distance is less than a named threshold τ_c^p and if the distance between the pixel q^{c^*} and the pixel of the end is less than a threshold τ_c^q, the corner is added to $\mathcal{C}$ and the ends of the segment δ are modified accordingly.

Finally, as can be seen in Figure 9(c), it is possible that the corner (called M in the image) is the intersection of three lines without being an intersection of three planes. In this case, we will get two distinct corners, computed from two intersections of lines. Therefore, the last step of the algorithm consists of detecting these duplicates and merging them. The detection is carried out by looking for the corners computed from a common segment which are separated by a distance less than the threshold τ_c^p. The average point is computed between the points of the two corners.

3.6 Final modeling

An example of the result obtained after the construction of segments and corners is given in Figure 10 for the synthetic example studied in this section (see Figure 1). From this result, the last step is to close the polygons and create the holes by gathering corners to form cycles. Starting from a corner chosen randomly in a polygon, the segments having this corner as end are studied, and the end of one other is selected. This process is repeated iteratively until retrieving the initial corner. If a segment is connected only by one end, we look for the corner of the closest unconnected polygon to the other end in order to close the cycle. Moreover, if a cycle closure involves crossing an edge, the corner closest to the intersection is removed and the route is continued. Various cycles are created until all segments have been traversed. They represent the outline, openings or occluded areas of the polygon. A mesh can then be created

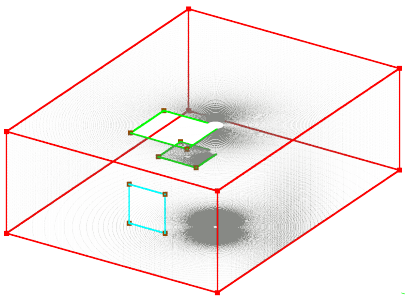


Figure 10. Contour segments of the polygons and corners detected by our method, superimposed on the point cloud from a LiDAR simulation in a synthetic scene. The edge labels are: intersection (red), occlusion (green) and opening (cyan). The corner labels are: intersection of three planes (red), intersection of two segments (brown)

by triangulation on the corners by imposing the edges of the polygons. For the same planar primitive, the faces located in the inner cycles are removed. The constrained Delaunay triangulation implemented by the CGAL library was used in the reference plane of the polygons to perform this task. The mesh result of the example used in this section (see Figure 1) is represented in Figure 11.

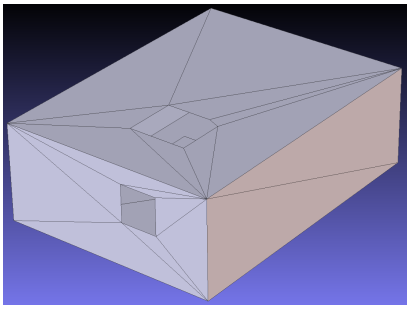


Figure 11. Mesh obtained from a synthetic point cloud.

3.7 Settings

For all the reconstructions performed, we use the same set of parameters as detailed in Table 1. This set was chosen in relation to the physical characteristics of the indoor environments without any calibration and the algorithm is robust to its modification.

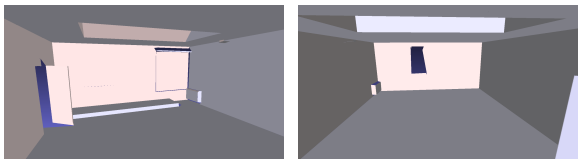
	Symbol	Description	Value
General	S_{min}	Minimal number of points/pixels	4
	$\tau_{//}$	Maximal angle value to consider parallelism	10°
Seed selection	τ_d^g	Maximum residue of a neighbor to the seed's local plane	2 cm
	τ_α^g	Maximum angular difference between neighbor normals	8°
Region growing	τ_d	Maximum distance to the seed's local plane	8 cm
RANSAC	M	Maximum number of tests	50 000
	τ_δ	Maximum distance to a straight line	2 cm
	τ_{diff}^p	Maximum distance between two consecutive points of a straight line	10 cm
	τ_{diff}^q	Maximum distance between two consecutive pixels of a straight line	5px
Corner selection	τ_c^p	Maximum distance to a potential corner	15 cm
	τ_c^q	Maximum distance to a potential corner	10px

Table 1. Parameters of the proposed method

4. EVALUATION

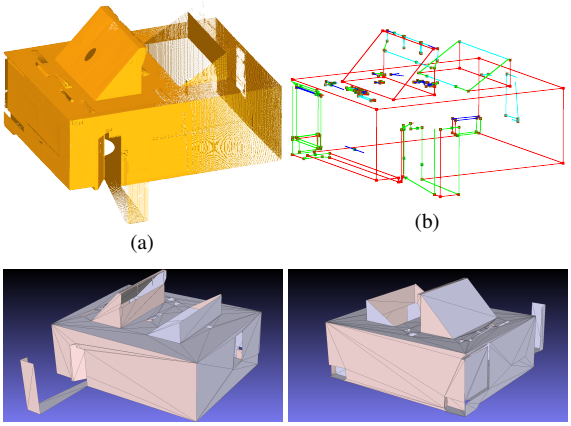
Our method is successfully evaluated on simulated data (see Figure 11) with an RMS error of 1.3×10^{-5} m between acquired points and the mesh structure. We validated visually our algorithm on various LiDAR scans, and we present the results for two challenging ones in this section. These data were collected by a Leica P20 scanner. First, a room is scanned and modeled by our method. The scan contains 2 million points. The modeling result is obtained in 2 minutes (one thread on processor Intel i57440HQ, 2.80 GHz) and is shown in Figure 13. The point/segment reconstruction in Figure 13(b) highlights the relationships between the planes and the objects in the scene which are correctly labeled. The short intersection segments are all detected but a few occlusion segments could not be highlighted by the RANSAC algorithm. The reconstruction contain 84 planar primitives. They are represented as a mesh in Figures 13(c) and 13(d), we note that the sampling anisotropy of the point cloud is well handled by our method. Thin polygons such as the door border or the ceiling grids are detected if they are sampled by at least 3 points across their width. Small occluding objects are also detected: the emergency lighting above the door, the radiators, the video projector, the blackboard, etc.However, we note that the points reflected on the windows can corrupt the reconstruction (see Figure 12(b)). In addition, the ceiling lights are not fully detected because they lay in the area of noise. We measured an RMSE of 8 mm between acquired points and the resulted mesh which mainly correspond to the noise level of the Leica sensor. Moreover, 98.6% of the total points are represented by one of the planar primitive (*i.e.,* are located at less than 20 cm of a primitive).

Second, another example of scan is tested in a scene including a staircase sampled by 2.5 million points (see Figure 14). This scene is particularly challenging because some walls are at a strong oblique angle to the LiDAR rays and it contains a lot of details and fine planes. The scene is visually correctly reconstructed in five minutes: all primitives are detected (87 polygons), fine planes as well (see the doors' width, the banisters or the box on top left part zoomed in Figure 14(c)) and the connections are correctly assigned. Residual problems concern curved shapes such as the discoid targets placed in the foreground (on the floor), and in the background (upstairs), to help the registration, whose contours are modeled by polygons which causes a loss of information. The RMSE between initial points and the reconstructed structure is 8 mm as previously and 99.6% of the points are represented by one of the planar primitive.



(a) (b)

Figure 12. 3D views of generated building model, (a) a first inside view of the room, (b) another view of the scanned scene



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

Figure 13. 3D reconstruction of a classroom (a): LiDAR 3D point cloud. (b) Segments and corners. The edge labels are: intersection (red), occlusion (green), interaction with object (blue) and opening (cyan). The corner labels are: intersection of 3 planes (red), intersection of 2 segments (brown). (c), (d): mesh model obtained.

5. CONCLUSION

Building modeling is commonly divided into two stages, the segmentation of the point cloud and the fitting of models on these points. The mainly used segmentation methods require complex configuration and are difficult to adapt to any LiDAR acquisition context. In the model fitting process, existing methods often lead to a strong geometrical deviation of the modelisation relatively to the studied scene. We then proposed a new method allowing to solve the problems of modeling from 3D LiDAR scans. It is based on the simultaneous processing of the points in the 3D space and in the angular 2D frame used by the LiDAR device. This process allows a better handling of the sampling anisotropy than existing methods, and, as we do not use lines nor planes arrangement, it allows a lower extrapolation of the data. Finally, the proposed approach allows extracting occlusions, connections and openings information with a good accuracy. The direct step derived from this procedure would be to gather all scan models into one unique object by registration. A method of global registration for point clouds (Sanchez et al., 2017) could be extended as it uses geometrical features that can be easily extracted from our new polygonal model.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-V-2-2020-395-2020 | © Authors 2020. CC BY 4.0 License.

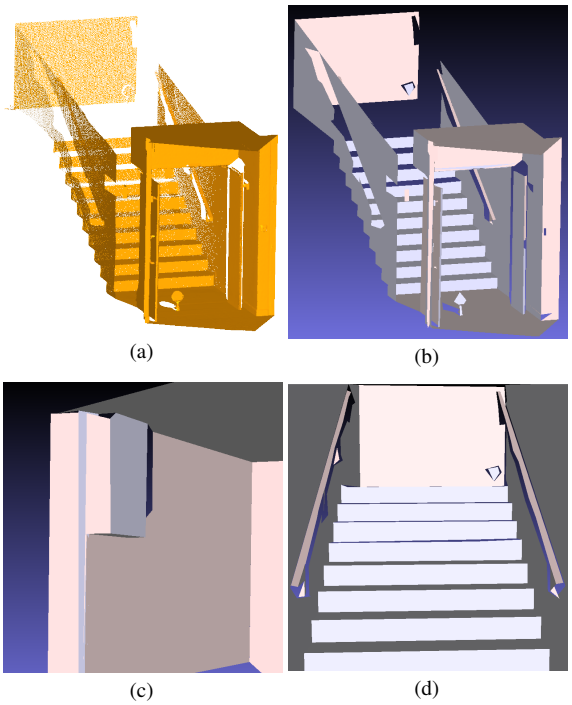


Figure 14. 3D reconstruction of a staircase. (a) LiDAR 3D point cloud, (b) mesh model obtained, (c) zoom on a box, (d) zoom on banisters and their stands.

ACKNOWLEDGEMENTS

This work has been sponsored by the RobotEx Equipment of Excellence (ANR-10-EQPX-44) and the LabEx IMobS3 (ANR-10-LABX-16-01).

REFERENCES

Besl, P., Jain, R., 1988. Segmentation through variable-order surface fitting. *IEEE Trans. on PAMI*, 10(2), 167–192.

Borrmann, D., Elseberg, J., Lingemann, K., Nüchter, A., 2011. The 3D Hough Transform for plane detection in point clouds: A review and a new accumulator design. *3D Research*, 2(2), 1–13.

Boulch, A., Gorce, M. d. L., Marlet, R., 2014. Piecewise-planar 3D reconstruction with edge and corner regularization. *Eurographics Symposium on Geometry Processing*, 33(5), 55–64.

Boulch, A., Marlet, R., 2016. Deep learning for robust normal estimation in unstructured point clouds. *Computer Graphics Forum*, 35(5), 281–290.

Bretar, F., Roux, M., 2005. Hybrid image segmentation using LiDAR 3D planar primitives. *ISPRS Proceedings. Workshop Laser scanning, the Netherlands*, 78, 12–14.

Budroni, A., Boehm, J., 2010. Automated 3D Reconstruction of Interiors from Point Clouds Automated 3D Reconstruction of. *International Journal of Architectural Computing*, 08(01), 55–74.

Chauve, A. L., Labatut, P., Pons, J. P., 2010. Robust piecewise-planar 3D reconstruction and completion from large-scale unstructured point data. *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 1261–1268.

Coughlan, J. M., Yuille, A. L., 1999. Manhattan world: Compass direction from a single image by bayesian inference. *Int. Conf. on Computer Vision*, 2, IEEE, 941–947.

Deschaud, J.-E., Goulette, F., 2010. A fast and accurate plane detection algorithm for large noisy point clouds using filtered normals and voxel growing. *3D Data Processing, Visualization, and Transmission*.

Duda, R. O., Hart, P. E., 1971. Use of the Hough Transformation to Detect Lines and Curves in Pictures. *Technical Note, Artificial Intelligence Center, SRI International*, 36.

Fischler, M., Bolles, R., Bruckstein, A., 1981. Random Sample Consensus: A paradigm for model fitting with application to image analysis and automated cartography. *Communications of the ACM*, 24(6), 381–395.

Fitzgibbon, A., Eggert, D., Fisher, R., 1997. High-level model acquisition from range images. *Computer-Aided Design*, 29(4), 321-330.

Huber, P. J., 2011. Robust statistics. *International Encyclopedia of Statistical Science*, 1248–1251.

Khoshelham, K., Díaz-Vilariño, L., 2014. 3D modelling of interior spaces: Learning the language of indoor architecture. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, 40(5), 321–326.

Macher, H., Landes, T., Grussenmeyer, P., 2017. From Point Clouds to Building Information Models: 3D Semi-Automatic Reconstruction of Indoors of Existing Buildings. *Applied Sciences*, 7(10), 1030.

Ochmann, S., Vock, R., Wessel, R., Klein, R., 2016. Automatic reconstruction of parametric building models from indoor point clouds. *Computers and Graphics (Pergamon)*, 54(5), 94–103.

Oesau, S., Lafarge, F., Alliez, P., 2014. Indoor scene reconstruction using feature sensitive primitive extraction and graph-cut. *ISPRS Journal of Photogrammetry and Remote Sensing*, 90, 68–82.

Poppinga, J., Vaskevicius, N., Birk, A., Pathak, K., 2008. Fast plane detection and polygonalization in noisy 3D range images. *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 3378–3383.

Poullis, C., You, S., 2009. Automatic reconstruction of cities from remote sensor data. *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2775–2782.

Pu, S., Vosselman, G., 2006. Automatic extraction of building features from terrestrial laser scanning. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(5), 25-27.

Sanchez, J., Denis, F., Checchin, P., Dupont, F., Trassoudaine, L., 2017. Global registration of 3D LiDAR point clouds based on scene features: application to structured environments. *Remote Sensing, MDPI*, 9(10).

Sanchez, V., Zakhor, A., 2012. Planar 3d modeling of building interiors from point cloud data. *IEEE International Conference on Image Processing*, 1777–1780.

Tarsha-Kurdi, F., Landes, T., Grussenmeyer, P., 2007. Hough-Transform and Extended Ransac Algorithms for Automatic Detection of 3D Building Roof Planes From Lidar Data. *ISPRS Workshop on Laser Scanning 2007 and SilviLaser 2007*, XXXVI(1), 407–412.

Thomson, C., Boehm, J., 2015. Automatic geometry generation from point clouds for BIM. *Remote Sensing*, 7(9), 11753–11775.

Torr, P. H., Zisserman, A., 2000. Mlesac : A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1), 138–156.

Vanegas, C. A., Aliaga, D. G., Benes, B., 2012. Automatic extraction of Manhattan-world building masses from 3D laser range scans. *IEEE transactions on visualization and computer graphics*, 18(10), 1627–1637.

Xiong, X., Adan, A., Akinci, B., Huber, D., 2013. Automatic creation of semantically rich 3D building models from laser scanner data. *Automation in Construction*, 31, 325–337.

Xu, L., Oja, E., P., K., 1990. A new Curve DetectionMethod: Randomized Hough Transform (RHT). *Pattern Recognition Letters*, 11, 331–338.

Yl-Jski, Kiryati, N., 1994. Adaptive Termination of Voting in the Probabilistic Circular Hough Transform. *IEEE Trans. on PAMI*, 16(9).