



**HAL**  
open science

# Towards Real-Time 3D Editable Model Generation for Existing Indoor Building Environments on a Tablet

Adrien Arnaud, Michèle Gouiffès, Mehdi Ammi

► **To cite this version:**

Adrien Arnaud, Michèle Gouiffès, Mehdi Ammi. Towards Real-Time 3D Editable Model Generation for Existing Indoor Building Environments on a Tablet. *Frontiers in Virtual Reality*, 2022, 3, pp.782564. 10.3389/frvir.2022.782564 . hal-03880873

**HAL Id: hal-03880873**

**<https://hal.science/hal-03880873v1>**

Submitted on 1 Dec 2022

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Towards Real-Time 3D Editable Model Generation for Existing Indoor Building Environments on a Tablet

Adrien Arnaud<sup>1</sup>, Michèle Gouiffès<sup>1,\*</sup> and Mehdi Ammi<sup>2</sup>

<sup>1</sup>LISN, CNRS, Université Paris Saclay, France

<sup>2</sup>LIASD, University Paris 8; ammi@ai.univ-paris8.fr

Correspondence\*:

Michèle Gouiffès

michele.gouiffes@lisn.upsaclay.fr

## 2 ABSTRACT

3 This paper describes a mobile application that builds and updates a 3D model of an indoor  
4 environment, including walls, floor and openings, by a simple scan performed using a tablet  
5 equipped with a depth sensor. This algorithm is fully implemented on the device, does not require  
6 internet connection and runs in real-time, *i.e.* at 5 frames per second. This is made possible by  
7 taking advantage of recent AR frameworks, by assuming that the structure of the room is aligned  
8 on an Euclidean grid and by simply starting the scan in front of a wall. The wall detection is  
9 achieved in two steps. First, each incoming point cloud is segmented into planar wall candidates.  
10 Then, these planes are matched to the previously detected planes and labeled as ground, ceiling,  
11 wall, openings or noise depending on their geometric characteristics. Our evaluations show that  
12 the algorithm is able to measure a plane-to-plane distance with a mean error under 2 cm, leading  
13 to an accurate estimation of a room dimensions. By avoiding the generation of an intermediate  
14 3D model, as a mesh, our algorithm allows a significant performance gain. The 3D model can  
15 be exported to a CAD software, in order to plan renovation works or to estimate energetic  
16 performances of the rooms. In the user experiments, a good usability score of 75 is obtained.

17 **Keywords:** Computer vision, mobile devices, planes detection, time consistency

## 1 INTRODUCTION

18 Creating a 3D model of an existing building has found many applications, such as the generation of a  
19 BIM<sup>1</sup> of the building or obtaining geometrical information about the building (dimensions, surfaces, etc...)  
20 Whereas a 3D model of a new building is created during the conception phase, older buildings have  
21 frequently to be modeled. This process is often done manually for smaller buildings and is very tedious.  
22 For large scale buildings, laser scanners are used to generate high resolution 3D point clouds. These laser  
23 data serve as a basis to identify the structure of the reconstructed building. In addition, the process for  
24 exporting a BIM model can be partially automated (Macher et al., 2015).

---

<sup>1</sup> BIM stands for: Building Information Model

25 However performing an automatic generation of a 3D editable model of buildings is a complex task.  
26 Most of the previous works have focused on recognizing the global structure (grounds, ceiling, walls and  
27 openings) of the reconstructed building. Moreover, the use of laser scanners and LIDAR is costly and the  
28 scan is not performed in real time.

29 When only a rough global structure of the building is needed, or when the housing is small, it is possible  
30 to simplify the process.

31 With the recent release of depth sensors integrated onto tablets and with the enhancement of their com-  
32 puting capabilities, it is now possible to use these devices to perform a real-time 3D reconstruction. This  
33 can be bought at an affordable price by companies or by private individuals who want to renovate their  
34 housings. The measure of each dimension of the room and the edition in a CAD software, are very time  
35 expensive tasks, that can be made easier by the simple scan, as proposed in this paper.

36 This paper presents an application that allows any user to generate a 3D editable model of an existing  
37 indoor environment using a tablet or a smartphone, by simply walking inside the building and scanning  
38 the walls. The resulting model can then be exported in a CAD<sup>2</sup> software for modification, for renovation  
39 or decorating purposes and for estimating the energetic performances. The device has to include a visual  
40 odometry system, which is the case with modern AR APIs such as Google ARCore<sup>3</sup>, so that each RGB-D  
41 data issued from the sensors can be expressed in a same absolute coordinates frame. The proposed algorithm  
42 uses the computing capabilities of the device to generate a 3D editable model on-the-fly, including the walls  
43 and openings of the rooms. Thus, it avoids the generation of an intermediate 3D mesh, which is costly in  
44 terms of memory and computing resources Arnaud et al. (2016). A RGB-D sensor is used, which provides  
45 a temporal sequence of color and depth data captured at 5 frame per second. First, a planar segmentation  
46 is performed in each depth image of the sequence and the extracted planes are matched to the previously  
47 detected ones. Through this temporal analysis, walls are extracted and described in terms of geometry and,  
48 the global 3D model is updated.

49 The proposed algorithm assumes that the walls, the ceiling, and the floor are aligned on a Euclidean  
50 grid, which is a common assumption when working with building data Coughlan and Yuille (1999). It  
51 also assumes that the scan starts in front of a wall. These reasonable assumptions lead to considerable  
52 simplifications of the reconstruction process.

53 The previous work Arnaud et al. (2018) was a first attempt of real-time planes detection and matching  
54 using a mobile device, and was dedicated to the segmentation method. Color, luminance edges and point  
55 cloud were analyzed together through a bottom-up segmentation process, which combined a region growing  
56 method with a merging. Although the process was real-time (the data was processed in approximately  
57 200ms), some experiments have shown that, in some situations, only 40 % of the planar surfaces were  
58 correctly detected. The present paper details each component of the application, from the extraction of  
59 3D planes to their export towards a CAD software. Concerning the 3D scanning and modeling, which  
60 is the most time-consuming task of the application, a top-down hierarchical segmentation is achieved  
61 directly on the point cloud, without any pre-processing. A first stage is dedicated to the detection of rough  
62 planar categories, which are then separated into parallel planes. By capturing the most dominant planes, the  
63 method proves less sensitive to noise than the top-down strategy. In addition, the multi-threading is made  
64 possible by an adapted tree sorting of the points. The planar surfaces that are extracted in two successive  
65 frames are matched temporally, then walls and openings are identified. To finish, this paper explains the

---

<sup>2</sup> CAD for Computer Aided Design

<sup>3</sup> <https://developers.google.com/ar/>

66 different components of a more comprehensive mobile application, which is able to create and export a 3D  
67 model that can be used and edited in a CAD software, to plan renovation works and to estimate energetic  
68 performance.

69 In the rest of this paper, some previous works on 3D segmentation and classification are described (sec 2).  
70 Then, the planar segmentation algorithm is developed in section 3 while the temporal matching and the  
71 model export are detailed in Section 4. The results of the evaluations conducted for these algorithms are  
72 presented in section 5 and discussed.

## 2 RELATED WORKS

73 3D modeling of indoor environments has been the subject of numerous research studies, as noticed by the  
74 recent review Kang et al. (2020). One of the strategies consists in capturing and mapping a dense colored  
75 point cloud Henry et al. (2012) into a real coordinate system, in which it is possible to navigate virtually.  
76 From depth data, a 3D mesh of the whole scene can first be estimated, and eventually simplified Liang et al.  
77 (2020). Since our main objective is to propose a stand-alone 3D modeling application for mobile devices,  
78 such as tablets or smartphones, such dense models are not optimal since they require a large amount of  
79 memory resources. In addition, we intend to produce a 3D model in real-time, without connecting to a  
80 distance webservice. Indeed, internet connection is not available on all worksites. Since the sensor captures  
81 one point cloud each 200ms, it is necessary to compute a 3D model in this period of time, otherwise  
82 the next point cloud can be lost. For that purpose, 3D segmentation techniques are promising (see 2.1).  
83 From the point cloud, these techniques directly detect planar structures from the point cloud and convert  
84 them into parameterized shapes (for example by simply storing the coordinates of their 4 corners). All the  
85 surfaces can be locally viewed as planar surfaces Tatavarti et al. (2017) that can be further analyzed using  
86 classification techniques (see 2.2) and recognizing methods.

### 87 2.1 3D segmentation

88 Concerning 3D segmentation, model fitting algorithms are popular due to their simplicity. The most  
89 commonly used algorithms are inspired by RANSAC Schnabel et al. (2007) or 3D Hough transform  
90 Borrmann et al. (2011). In an iterative way, RANSAC randomly picks a group of points and refine the  
91 coefficients of a given parameterized model. Although the quality of the estimation depends on the number  
92 of iterations, and on the quality of the selected points, it provides a faster and more accurate planes detection  
93 than 3D Hough transform Tarsha-Kurdi et al. (2007).

94 The use of region growing algorithms can speed-up the segmentation task by restricting the analysis in  
95 the neighborhood of a few seed points, before merging the resulting clusters. However, their performance  
96 is tightly linked to the choice of the seed points Grilli et al. (2017).

97 Erdogan et al. Erdogan et al. (2012) perform a planar segmentation of a depth map using an adaptation of  
98 the Superpixels algorithm Fulkerson et al. (2009) originally designed for 2D images. The generated clusters  
99 are then merged using the Swendsen-Wang sampler Swendsen and Wang (1987); Barbu and Zhu (2005).

100 Papon et al. Papon et al. (2013) propose the Voxel Cloud Connectivity segmentation (VCCS) algorithm.  
101 This is a generic 3D segmentation algorithm which selects seed points in a regular grid from the input  
102 point cloud and generates clusters around these seed points. This algorithm outputs a set of clusters and  
103 an adjacency graph that describes the connectivity between them. Points are gathered together based on a  
104 similarity function that depends on the spatial coordinates, the normal vector and the color of each point,  
105 with a weight for each parameter. For planar segmentation, VCCS is parameterized so that the orientation

106 of the normal vector has a predominant weight in the computation of the similarity criterion between two  
107 points.

108 Planar segmentation can also be viewed as a clustering problem. Clustering techniques can be supervised,  
109 with a known number of classes. One of the most popular supervised clustering techniques is the K-means  
110 algorithm Jain (2010); Celebi et al. (2013), which consists in selecting  $k$  elements in the input dataset as the  
111 centers of the clusters. The remaining elements are associated to the nearest center. Then, the centroids are  
112 updated. The process is repeated until the algorithm converges. K-means algorithms are efficient when data  
113 are well separated. Non-supervised algorithms, such as DBSCAN Ester et al. (1996), or ISODATA, used  
114 for example in Holz et al. (2011), do not require the number of classes but group the elements depending  
115 on a density criterion. Then, it detects clusters when their density is higher than a fixed threshold, which  
116 highly depends on the scene to be analyzed.

117 The reader can refer to Grilli et al. (2017); Nguyen and Le (2013) for more detailed information about  
118 3D segmentation.

## 119 **2.2 Classification and structure recognition**

120 The results of a planar segmentation algorithm are used as a basis to identify the components of an input  
121 3D point cloud. Indeed, it allows a fast recognition of structural elements Verma et al. (2006); Ochmann  
122 et al. (2015), or the classification of a room furniture, once the main planar surfaces have been subtracted  
123 Deng et al. (2017). Many of these algorithms train classifiers to label the points Ren et al. (2012); Gupta  
124 et al. (2013); Lai et al. (2014). For example, Silberman et al. Silberman et al. (2012) propose a segmentation  
125 algorithm that uses RGB-D data from indoor scenes. After a RANSAC-based planar segmentation, the  
126 resulting regions are grouped into structure categories using logistic regression.

127 Lee et al. Lee et al. (2009) describe a method for detecting walls, ground and ceiling in an indoor scene  
128 using only RGB images. They use predefined patterns about the lines of the RGB images to infer the  
129 building structure, assuming that the building structure is aligned along a Manhattan grid Coughlan and  
130 Yuille (1999).

131 Verma et al. Verma et al. (2006) propose a 3D segmentation algorithm that can identify the external  
132 structure of buildings in an outdoor point cloud captures by a LIDAR scanner. Assuming that the point  
133 cloud has been acquired from an aerial scanner, the authors focus on detecting the ceilings of the buildings.  
134 For this purpose, they perform a planar segmentation of the points cloud and remove the vertical planes.  
135 Then, the shape of the buildings is inferred using predefined patterns.

136 Many works deal with automatically generating a BIM model from laser data Adan and Huber (2011);  
137 Jung and Joo (2011); Jung et al. (2014); Macher et al. (2015). In practice, this is a complex task that cannot  
138 be fully automated but algorithms can detect the global shape of the building and export a pre-generated  
139 model that can be refined manually in a CAD application. Let us also mention the existence of semantic  
140 segmentation using deep learning techniques Zhang et al. (2020). These approaches would require resources  
141 that are not available on simple mobile devices.

## 142 **2.3 Proposed solution**

143 RANSAC-like algorithms can deal with many outliers but can require a large and variable number of  
144 iterations to converge, depending on the quality of the points that are randomly selected on the surface to  
145 be parameterized. Region growing algorithms are efficient and are fast when the size of the clusters to be

146 generated is constrained. However, their performance is dependent on the initialization and on the quality  
147 of the input data.

148 In the proposed method, a K-means algorithm is used to perform a planar segmentation of each input  
149 point cloud. The convergence speed depends in particular on the choice of the initial centers. The closer  
150 they are to the real centers, the faster the convergence. In most standard buildings, it can be assumed that  
151 the wall candidates have six main orientations, and possibly these walls can be aligned to a regular grid, as  
152 in a Manhattan World. The proposed method first achieves a planar segmentation, detailed hereafter in  
153 section 3, of the 3D point cloud captured by the sensor. Then, the planes are temporally matched between  
154 successive frames and labeled as walls, floors or ceilings, as described in section 4.

### 3 REAL-TIME PLANAR SEGMENTATION

155 In order to reach real-time execution, *i.e.* 5 fps as imposed by the device, the user is recommended to start  
156 the scan of the indoor environment by facing one of the walls. Thus, the reference coordinate frame  $\mathbf{R}_0$   
157 is aligned with the structure of the building and the orientations of the normal vectors are well-defined.  
158 After a brief overview on the proposed algorithm, we go further into detail by explaining successively the  
159 different stages of the method.

#### 160 3.1 Overview of the algorithm

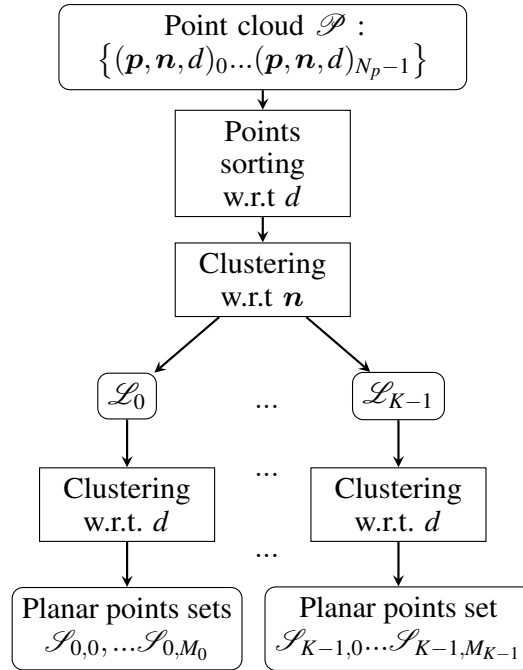
161 The detection of the planar surfaces is illustrated on Fig. 1. Let  $\mathbf{R}_0$  be the coordinates frame with axes  
162  $\mathbf{u}_x$ ,  $\mathbf{u}_y$  and  $\mathbf{u}_z$ . The input is the 3D point cloud (of  $N_P$  points  $P$ ) to be segmented, noted  $\mathcal{P}$ . Each point  $P$   
163 is characterized by :

- 164 • a coordinate vector  $\mathbf{p} = [x, y, z]^\top$
- 165 • a normal vector  $\mathbf{n} = [n_x, n_y, n_z]^\top$ . The computation of  $\mathbf{n}$  is not detailed here, but the reader can refer to  
166 Arnaud et al. (2018) for further details.
- 167 • a distance  $d$  from the origin of  $\mathbf{R}_0$ , defined as:  $d = -(n_x \cdot x + n_y \cdot y + n_z \cdot z)$ .

168  $\mathcal{L}_k$  stands for the  $k^{\text{th}}$  set of points  $P$ ,  $k \in [0, K - 1]$ , with similar norm vectors. To finish,  $\mathcal{S}_{k,m}$  is the  
169 notation used for the  $m^{\text{th}}$  (with  $m \in [0, M_k - 1]$ ) set of points in  $\mathcal{L}_k$  which share the same properties  $\mathbf{n}$  and  
170  $d$ . Notice in  $\mathcal{S}_{k,m}$ , points do not necessary form a connected component but belong to the same *planar*  
171 *structure* in the real scene. The planar segmentation consists in detecting these sets  $\mathcal{S}_{k,m}$  (referred to *planar*  
172 *structures* for sake of simplicity).

173 The segmentation is performed in two steps. First, the normal vectors are clustered with respect to their  
174 orientation using an adaptation of the K-means algorithm Jain (2010) that is described in 3.3. For each  
175 orientation category, a second clustering (detailed in 3.4) is made according to  $d$ , in order to separate  
176 parallel surfaces, that is planes of similar orientation located at different distances. This is detailed in 3.4.

177 In terms of implementation, two independent threads are used, one for the data pre-processing (storing,  
178 normals computation), the other one for the segmentation itself. The first thread computes the normal  
179 vectors at each point of the surfaces. Concerning the sorting detailed in 3.2, the data are divided into two  
180 groups (as shown by Fig. 1), depending on whether  $d$  is lower or higher than the mean distance  $\bar{d}$  computed  
181 on the whole point cloud. Here also, this allows to use 2 threads for the sorting.



**Figure 1.** Overview of the planar segmentation algorithm. It consists of two clustering phases : the first one is for the normals' space, and the second one separates parallel planes.

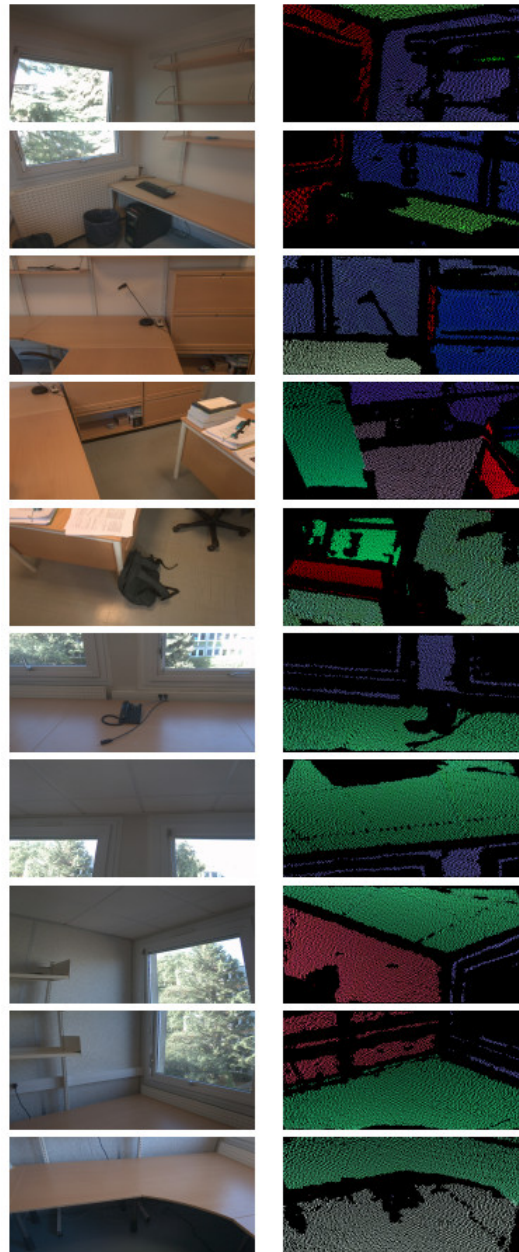
## 182 3.2 Points sorting

183 The input points are sorted using a tree sort algorithm. First, each descriptor  $P$  is stored in a binary tree  
 184  $\mathcal{T}$ . The insertion of a descriptor  $P$  in a node is made using the distance  $d$  as comparison criterion. If  $d$   
 185 is inferior to the current node  $d$  value, then  $P$  is inserted into the left child of the node, otherwise, it is  
 186 inserted into the right one. The list is then sorted by recursively browsing the binary tree  $\mathcal{T}$  starting by the  
 187 left. The mean complexity of this sorting method is  $O(n \log(n))$ . The best benefits of the algorithm in terms  
 188 of performance, are obtained for large amount of data. Consequently, the whole point cloud is sorted first,  
 189 before creating any cluster, instead of performing one sorting per cluster. In this way, the probability to  
 190 build an unbalanced tree is minimized, which would lead to a complexity of  $O(n^2)$  for the spatial sorting.  
 191 Moreover, the sorting algorithm is easier to parallelize on different threads of equal workload.

## 192 3.3 Normals clustering

193 The normals clustering is a K-means algorithm Celebi et al. (2013) which is constrained by predefined  
 194 centroids related to the main planar surfaces that can be found in the building. Assuming the building  
 195 structure is aligned on a Euclidean grid, the normal vectors of the walls candidates have 6 possible  
 196 orientations, one for each axis direction. By starting the capture in front of a wall, the reference frame  $\mathbf{R}_0$   
 197 is aligned to this Euclidean grid, and the normal vectors for the walls, the ground and the ceiling are along  
 198 the different axes of  $\mathbf{R}_0$ .

199 The original K-means algorithm has been adapted so that it can classify the input normal vectors into at  
 200 most  $K$  classes ( $K = 6$ ), but can use less classes. Moreover, only points that have a normal vector close  
 201 to one of the  $\mathbf{R}_0$  axes will be considered. It starts by the initialization with the centers  $\hat{\mathbf{o}}_k, k \in [0, K - 1]$ ,  
 202 corresponding to the 6 possible orientations aligned with axes  $\mathbf{u}_x, \mathbf{u}_y, \mathbf{u}_z$ . These centers are refined to  
 203 match the actual orientation of the normals when  $\mathbf{R}_0$  is not perfectly aligned with the building structure.



**Figure 2.** Planar segmentation using our method. Left column: RGB images of the scene to be segmented. Right column: representation of the corresponding 3D points cloud with one color per plane orientation. Only the planes that are aligned with the building structure are detected.

204 Let  $\{\mathbf{o}_k/k \in [0, K - 1]\}$  be the final centers (*i.e.* the mean characteristics of the clusters), and  $\mathcal{L}_k$  the labeled  
 205 set where each element of  $\mathcal{P}$  is labeled with the corresponding class label  $k \in [0, K - 1]$ .

206 First of all, each  $\mathbf{o}_k$  is initialized with the corresponding  $\hat{\mathbf{o}}_k$ , and  $\mathcal{L}_k$  is initialized with the elements  
 207 of  $\mathcal{P}$  labeled with  $+\infty$ . Then, for each iteration of the algorithm, a first loop iterates over each element  
 208  $(\mathbf{p}, \mathbf{n}, d) \in \mathcal{L}_k$  and compares them to each center  $\hat{\mathbf{o}}_k$ . If the Euclidean distance  $\Delta_k = \delta(\mathbf{n}, \mathbf{o}_k)$  is under a  
 209 threshold  $\varepsilon_n$ , then  $k$  is kept as a candidate label. The argmin value  $k_0$  for the candidate labels is kept if it  
 210 exists. Otherwise, the label is set at  $+\infty$  and the corresponding point data is pruned from the process.



211 Once each element of  $\mathcal{L}_k$  has been labeled, each  $\mathbf{o}_k$  is updated with the mean normal value of the elements  
 212 of  $\mathcal{L}_k$  labeled with  $k$ . If no element of  $\mathcal{L}_k$  is labeled with  $k$ , then  $\mathbf{o}_k$  is set to  $[0, 0, 0]$ .

213 This process is repeated  $N$  times. Similarly, the process could be repeated until convergence. This is a  
 214 form of constrained E-M approach.

215 In terms of implementation, since the clustering is made independently on each point, the work can be  
 216 made by several threads, for instance 4 threads in this work. We have also used the SIMD<sup>4</sup> instructions  
 217 of the processor to accelerate the execution. Note that Euclidean distance has been preferred to angular  
 218 distance, because products and sums are faster to compute (and even faster in SIMD) than trigonometric  
 219 functions.

### 220 3.4 Distance clustering

221 After normals clustering, a maximum of  $K$  clusters  $\mathcal{L}_k, k \in [0, K - 1]$  is formed. For each of these clusters,  
 222 points have to be separated into parallel planes. Each cluster  $\mathcal{L}_k$  contains  $L_k$  points to be sorted. Let  
 223  $l \in [0, L_k - 1]$  be the index of the points inside a cluster  $\mathcal{L}_k$ . In each cluster  $\mathcal{L}_k$ ,  $M_k$  planar structures have  
 224 to be determined. Let  $\mathcal{S}_m$  with  $m \in [0, M_k - 1]$  denote the  $m^{\text{th}}$  set of points forming a planar structure in  
 225  $\mathcal{L}_k$ .

226 First of all, in each set  $\mathcal{L}_k$ , the  $L_k$  points  $P_l(\mathbf{p}_l, \mathbf{n}_l, d_l)$  are sorted by ascending distances  $d_l$ . Once sorted,  
 227 the clustering is straightforward using a DBSCAN method. Subsequent points  $P_{l-1}$  and  $P_l$  are considered  
 228 as belonging to the same structure  $\mathcal{S}_m$  when they have close distances, that is when the deviation between  
 229 their distances  $|d_l - d_{l-1}|$  is less than a threshold  $\varepsilon_d$ . Otherwise, a new plane object is created and initialized  
 230 with  $P_l$ .

### 231 3.5 Parameters

232 Table 1 shows the values of the parameters used to perform the planar segmentation.

<i>Param</i>	<i>Value</i>	<i>Description</i>
$N$	5	Max of iterations
$\varepsilon_n$	0.10	Maximal distance to add a point to a cluster
$\varepsilon_d$	0.03	Minimal distance to separate parallel planes

**Table 1.** Parameters for the planar segmentation.

233 Since the algorithm is fast to converge, a maximum number of iterations  $N = 5$  is enough for the K-means  
 234 algorithm. The distance  $\varepsilon_n$  has been fixed to 0.10, so normal vectors that are not aligned with the initial  
 235 frame  $\mathbf{R}_0$  are not added to any cluster. The parameter  $\varepsilon_d$  has been set to 0.03. This value is limited by the  
 236 precision of the depth sensor and the algorithm.

237 Figure 2 shows a few examples of the final segmentation process. The left column shows images from  
 238 the scene to be reconstructed, while the right column shows the segmentation results, where a different  
 239 color is given to each plane category.

<sup>4</sup> SIMD for Single Instruction Multiple Data

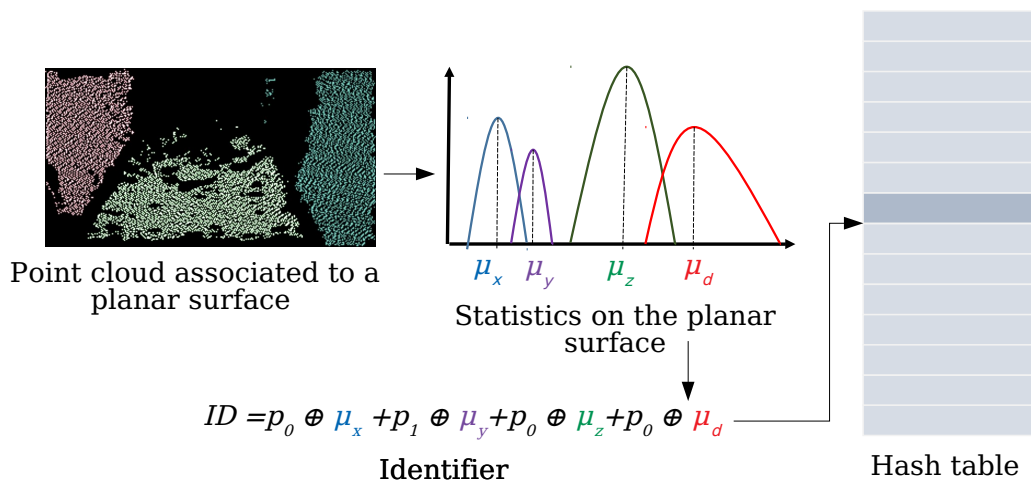
## 4 FROM TEMPORAL MATCHING TO THE EXPORT OF THE MODEL

240 The current planes are matched with the planes detected in the previous frame. Thus, similar planes are  
 241 merged and their geometric characteristics are updated, leading progressively to a 3D model of the whole  
 242 scene. This section first describes the creation and the update of the 3D model, and then explains how the  
 243 recognition of walls, ground or ceiling is made.

### 244 4.1 Planes matching

245 All this process is based on the ability to match the currently detected planes (at time  $t$ ) to the previous  
 246 ones (at time  $t - 1$ ). Using a motion tracking algorithm, all the 3D coordinates are expressed in a same  
 247 reference frame  $R_0$ .

248 The method described in Arnaud et al. (2018) is used to match similar planes. For each cluster  $\mathcal{S}$   
 249 extracted in the previous planar segmentation, the histograms of each parameter, *i.e.*  $n_x, n_y, n_z, d$ , are  
 250 computed (see Fig. 3). In theory, all points of  $\mathcal{S}$  have the same parameters, while in practice they are  
 251 distributed following a normal law around the actual parameters. These distributions are used to create a  
 252 unique identifier ID for each plane, which is used to store the plane in a hash table. Thus the memory is  
 253 dynamically allocated when new data is available. This identifier ID is built using the four mean values  
 254  $ID = (\mu_x, \mu_y, \mu_z, \mu_d)$  of the distributions of  $n_x, n_y, n_z, d$  on the corresponding planar surface. Thus, when  
 255 a new plane is detected in the current frame, its statistical characteristics are used either to retrieve the  
 256 previous corresponding plane when it exists with a complexity  $O(1)$ , or, to create a new plane and the  
 corresponding entry in the hash table.



**Figure 3.** Illustration of the hash table. On the left: a point cloud segmented into 3 planar surfaces. For each planar surface, four histograms are computed. The statistics are used to build an identifier that is used as a key to address the hash table.

257

### 258 4.2 Drifts correction

259 The estimation of the global position of the tablet is performed natively using a Visual Inertial Odometry  
 260 algorithm Li and Mourikis (2012). However, there are still positioning errors that accumulate over time,  
 261 causing imprecise temporal matching. To correct these possible drifts, an accelerated version of the Iterative  
 262 Closest Point (ICP) algorithm Besl and McKay (1992) has been implemented.

263 Let  $\mathcal{P}_t$  and  $\mathcal{P}_{t+1}$  be two point clouds captured respectively at times  $t$  and  $t + 1$ . Each point  $P \in \mathcal{P}$  is  
 264 described by its spatial position  $\mathbf{p}$ , its color  $\mathbf{c}$  and its normal vector  $\mathbf{n}$ . ICP consists in iteratively searching  
 265 for correspondences between two points clouds and in estimating the affine transform that minimizes a  
 266 global distance. Our variant of the ICP, described hereafter, accelerates the process by a fast sorting, a  
 267 pruning of the points that are too far to be considered as homologous, and by exploiting a distance based  
 268 on location, color and geometry .

#### 269 4.2.1 Finding correspondences

270 For each point in  $\mathcal{P}_{t+1}$ , a match is searched in  $\mathcal{P}_t$ . Considering that  $\mathcal{P}_{t+1}$  and  $\mathcal{P}_t$  are nearly aligned,  
 271 two points  $P' \in \mathcal{P}_{t+1}$  and  $P \in \mathcal{P}_t$  can be considered for matching if their distance  $\delta(P', P)$  is under a  
 272 threshold  $\varepsilon$ .

273 Assuming a small movement of the device in the time interval  $[t, t + 1]$ , the correspondences are searched  
 274 in a local neighborhood  $\mathcal{W}_{P,u,v}$  around each point  $P$  indexed by  $(u, v)$ <sup>5</sup>.

275 The algorithm 1 details the procedure for creating a set of correspondences  $\mathcal{C}$ . Two points  $P$ , and  $P'$   
 276 are matched if they are similar in terms of location  $\mathbf{p}$ , color  $\mathbf{c}$  and normals  $\mathbf{n}$ , according to the following  
 277 similarity function  $\delta(P, P')$ :

$$\delta(P, P') = \frac{\omega_p \delta_p(\mathbf{p}, \mathbf{p}') + \omega_c \delta_c(\mathbf{c}, \mathbf{c}') + \omega_n \delta_n(\mathbf{n}, \mathbf{n}')}{\omega_p + \omega_c + \omega_n}$$

278 where  $\delta_p$ ,  $\delta_c$  and  $\delta_n$  stand for Euclidean distances, computed respectively on spatial location, color and  
 279 normals. The weights  $\omega_p$ ,  $\omega_c$  and  $\omega_n$  are used to give more or less importance to each component.

---

**Algorithm 1:** Correspondences searching [between](#) two ordered point clouds.

**Input :** point clouds  $\mathcal{P}(t + 1)$  and  $\mathcal{P}(t)$

**Output:** set of correspondences  $\mathcal{C}$

---

```

1:  $\mathcal{C} \leftarrow \emptyset$ 
2:  $k \leftarrow 0$ 
3: for all  $P \in \mathcal{P}(t + 1)$  do
4:    $\Delta_{min} \leftarrow \varepsilon$ 
5:   for all  $P' \in \mathcal{W}_{P,u_0,v_0}$  do
6:      $\Delta \leftarrow \delta(P, P')$ 
7:      $P_0 \leftarrow \emptyset$ 
8:     if  $\Delta < \Delta_{min}$  then
9:        $\Delta_{min} = \Delta$ 
10:       $P_0 \leftarrow P'$ 
11:    end if
12:  end for
13:  if  $\Delta_{min} < \varepsilon$  then
14:     $\mathcal{C}[k] \leftarrow (P, P')$ 
15:     $k \leftarrow k + 1$ 
16:  end if
17: end for

```

---

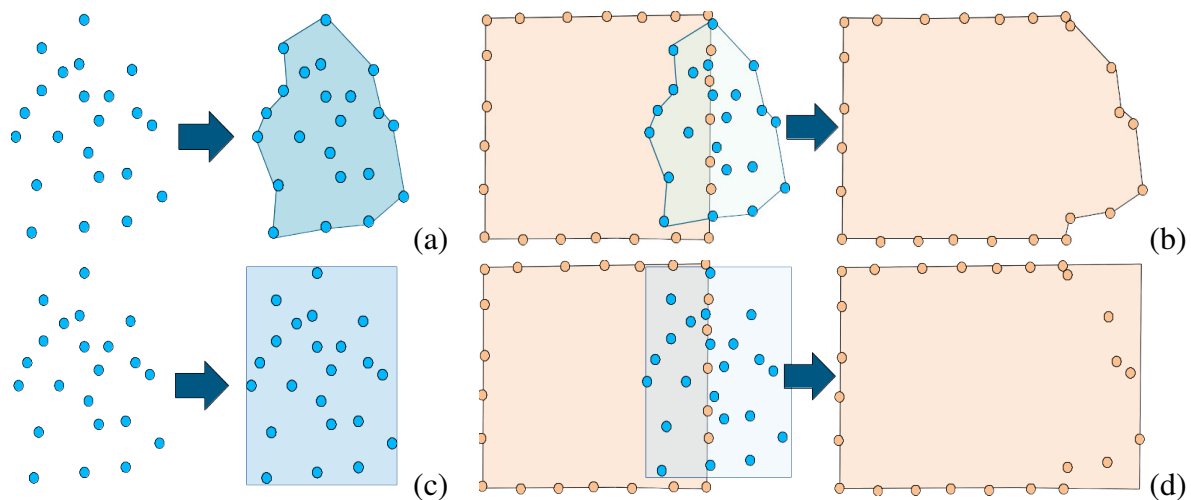
<sup>5</sup> Note that 2 coordinates are enough since the points lie on the same planar surface.

## 280 4.2.2 Alignment of the point clouds

281 Then, the two points clouds are aligned geometrically. To this end, the homography transform is estimated.  
 282 First, the translation  $T$  is estimated as the distance between the centroids  $\mathbf{o}'$  and  $\mathbf{o}$  of the spatial positions  
 283 of  $\mathcal{P}_t$  and  $\mathcal{P}_{t+1}$ . Then, the rotation  $R$  is estimated by computing the mean rotation required to align the  
 284 normal vectors on their mean normal vector.

## 285 4.3 Estimating the boundaries of each plane

286 Once the point cloud has been separated into different planar structures, each of them can be represented  
 287 by its area and its boundaries. Among the various possible techniques, two solutions have been considered,  
 288 as illustrated by Fig. 4.

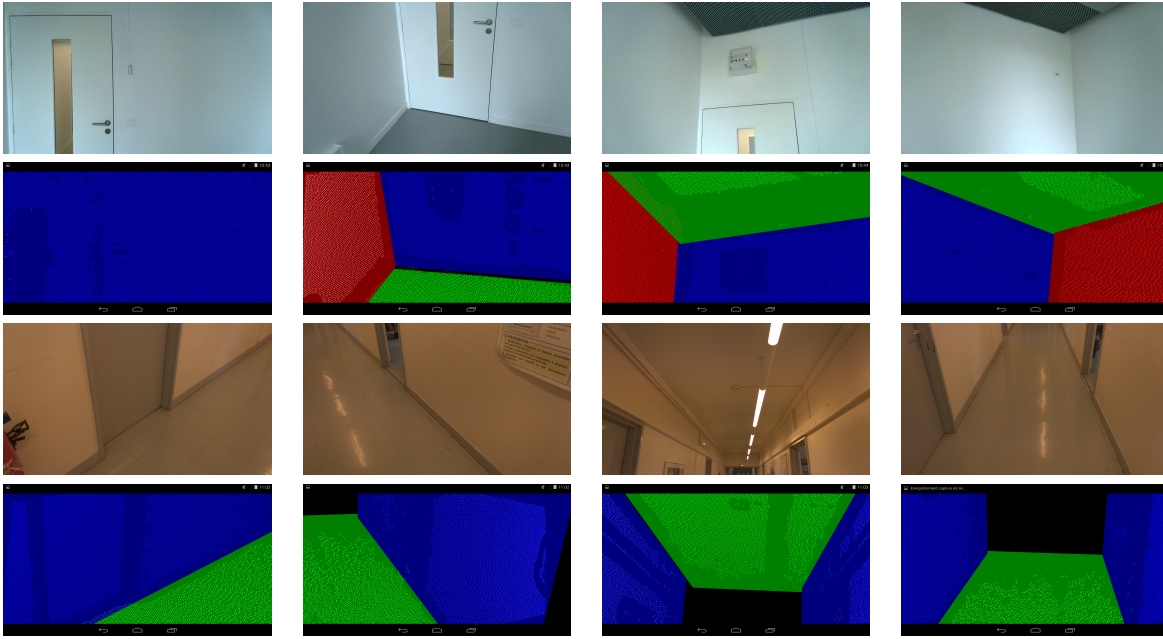


**Figure 4.** Two techniques considered for plan boundaries. The first one consists in computing the full concave hull of each cluster after the segmentation stage (a), and then in repeating this algorithm in order to merge this cluster (detected at time  $t + 1$ ) with one of the previous clusters (detected at time  $t$ ) (b). The second technique consists in computing the minimal bounding box of a cluster (c), and to update it after merging with a previous cluster (d).

289 The first one (see Fig. 4(a) and (b)) consists in computing and updating the full concave hull of a plane  
 290 using the KNN-based method developed by Moreira et al. Moreira and Santos (2007). In the latter, the  
 291 concave hull is defined as a polygon that best describes the region occupied by a set of points in a plane,  
 292 *i.e.* the minimal envelope or the *footprint* of these points. This technique allows the modeling of walls of  
 293 non-rectangular shape, but has two disadvantages. First, the concave hull is a growing list of points, and  
 294 the spatial resolution and growth are limited. In addition, this algorithm is time consuming. The second  
 295 option, illustrated by Fig. 4(c) and (d), consists in using the minimal bounding box of the planes, which is  
 296 satisfactory when walls are rectangular, as it is the case in our work.

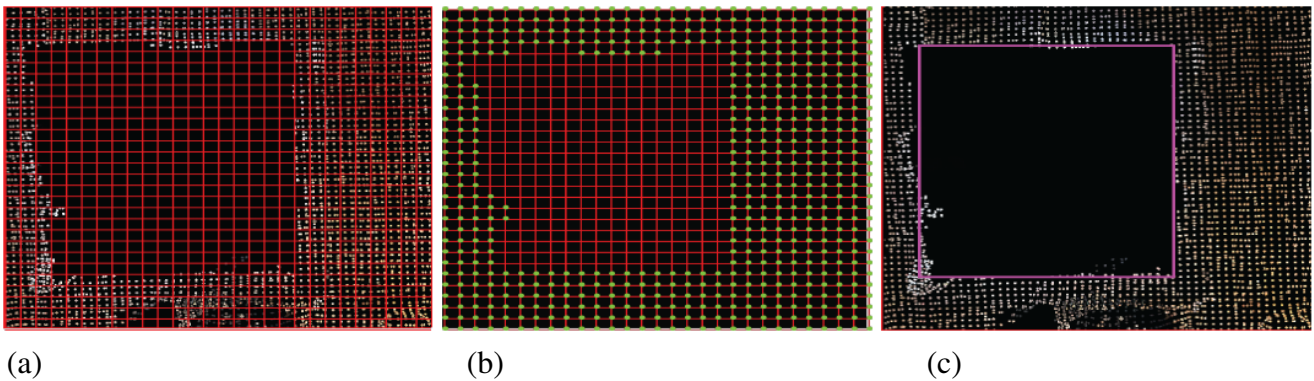
## 297 4.4 Walls and openings identification

298 First, the height  $H$  of the room is computed. This is made by finding the planes corresponding to the  
 299 ground and the ceiling, *i.e.* the planes that are orthogonal to the  $z$  axis. The floor and ceiling are respectively  
 300 the lowest and highest planes. Then, all of the planes that are orthogonal to the ground are considered as  
 301 potential walls. This is confirmed when its height is at least 80% of  $H$ . Examples of walls detection are  
 302 presented in Fig. 5.



**Figure 5.** Examples of walls identification. The ground and ceiling are displayed in green color, walls are drawn in blue and red.

303 Once the structural planes identified, the openings (*i.e.* doors and windows) are detected by assuming  
 304 that windows are transparent and doors are open. Therefore, the infrared light are not reflected 6(a). Thus,  
 305 the openings appear as void rectangles, *i.e.* areas for which no input data is available. Each wall is analyzed  
 306 individually in 2D, and each dimension is sub-sampled as described in Figure 6(b). The rectangular shapes  
 307 are detected using the ratio of the size (number of pixels) of the black area over the size of the minimal  
 308 bounding box (Figure 6(c)). A region is considered as an opening when the ratio is close to 1.



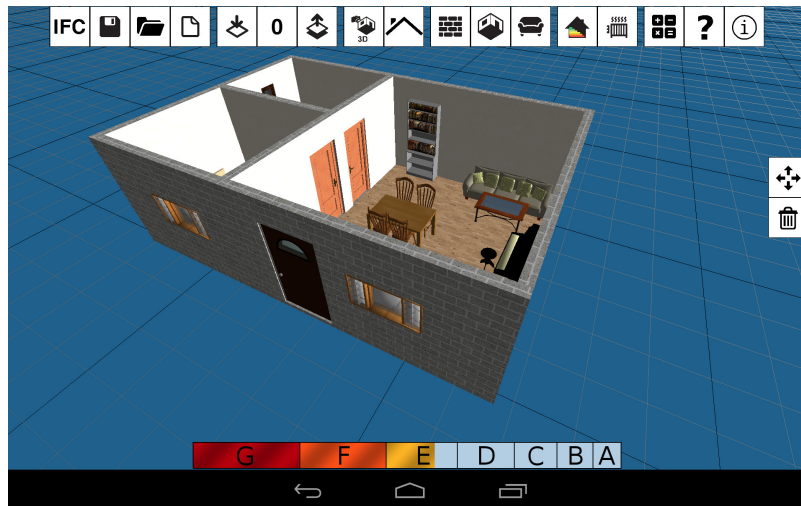
**Figure 6.** Illustration of the openings detection. (a) Infrared light is not reflected by glass. (b) Sub-sampling of the data. (c) Detection of the minimal bounding box.

#### 309 4.5 Exporting the model to a CAD software

310 After processing, the information needed to create a 3D model are exported in a .xml file. For each wall or  
 311 opening, the following parameters are saved in this file: an identifier, the coordinates of its four corners and  
 312 the orientation. The resulting 3D model can be edited in the CAD software, for example to plan renovation  
 313 and decorating works (see Fig. 8). In the application *Plan 3D Energy* that we developed (see Figure 7) with



**Figure 7.** A 3D model viewed in the software *Plan 3D Energy*.



**Figure 8.** Example of applications: decorating, evaluating energetic performances.

314 the society RPE, it is possible to specify the characteristics of the building and the materials for each wall  
 315 and each opening. These estimations, together with the data that our algorithm can provide, it is possible  
 316 to estimate the energy losses of the room<sup>6</sup>, and consequently its energetic performances. It represents  
 317 a useful tool to sensitize users to make energetic responsible choices regarding their interior renovation  
 318 works. To evaluate the quality of the energetic estimation when using the proposed system, five scans of  
 319 a room have been performed, and the energy losses are compared to the estimations made with the real  
 320 dimensions ( $310kWhm^{-2}year^{-1}$ , letter E). Using our 3D models, the estimated energetic performances are  
 321  $306kWhm^{-2}year^{-1}$  in average, and systematically lead to the same letter E.

## 5 EVALUATIONS

322 This section presents the evaluations of the planar segmentation algorithm used as a basis for wall detection.

### 323 5.1 Material

324 Developments were made on a Google Tango Yellowstone tablet, equipped with a Nvidia Tegra K1  
 325 processor and 4 GB RAM, and running on Android 4.4. It embeds a depth sensor and a motion tracking  
 326 algorithm based on Virtual Inertial Odometry Li and Mourikis (2012).

<sup>6</sup> The energy losses of one surface is the surface multiplied by its heat transfer coefficient defined by the 3CL10 norm according to the main building features : year of construction, type of insulation, materials of the building.



**Figure 9.** Setup used for evaluation precision.

## 327 5.2 Evaluation of the precision and reliability

328 Both the reliability and the precision of the planes detection are evaluated. For this purpose, the device is  
 329 first put in front of an empty wall at a fixed distance  $x$ . Then, another planar surface is put between the  
 330 device and the wall, at a fixed distance  $D$  from the wall. Fig. 9 illustrates the setup. For each configuration,  
 331 the procedure is tested on 100 successive point clouds. Two measures are used : the number  $n$  of times the  
 332 algorithm detects exactly 2 planes; if so, the distance  $d$  between the two detected surfaces. The results are  
 333 presented in table 2.

334 The mean error  $\bar{\delta}$  for the estimation of  $D$  is 1.46 cm, and the mean percentage of frames where exactly  
 335 two planes are detected  $\bar{n}$  is 77.15%. In comparison, in Arnaud et al. (2018), the error was approximately  
 336 2.1 cm, and two planes were correctly detected in 40.5 % of the cases. More detailed results are given in  
 337 Fig 10.

338 Thus, it can be seen that the clustering method provides a better precision in most cases, and is able  
 339 to distinguish parallel walls with a high reliability if their inter-plane distance is up to 10 cm. For each  
 340 algorithm, some errors occur for low inter-planes distances  $D$ . This is due to the norms estimation, which  
 341 is not accurate enough on the edges of the depth map. Most failures occur when two planes are close from  
 342 each other. In terms of execution time, the clustering approach is 5 to 10 times faster than the growing  
 343 regions strategy.

## 344 5.3 Evaluation of the 3D model

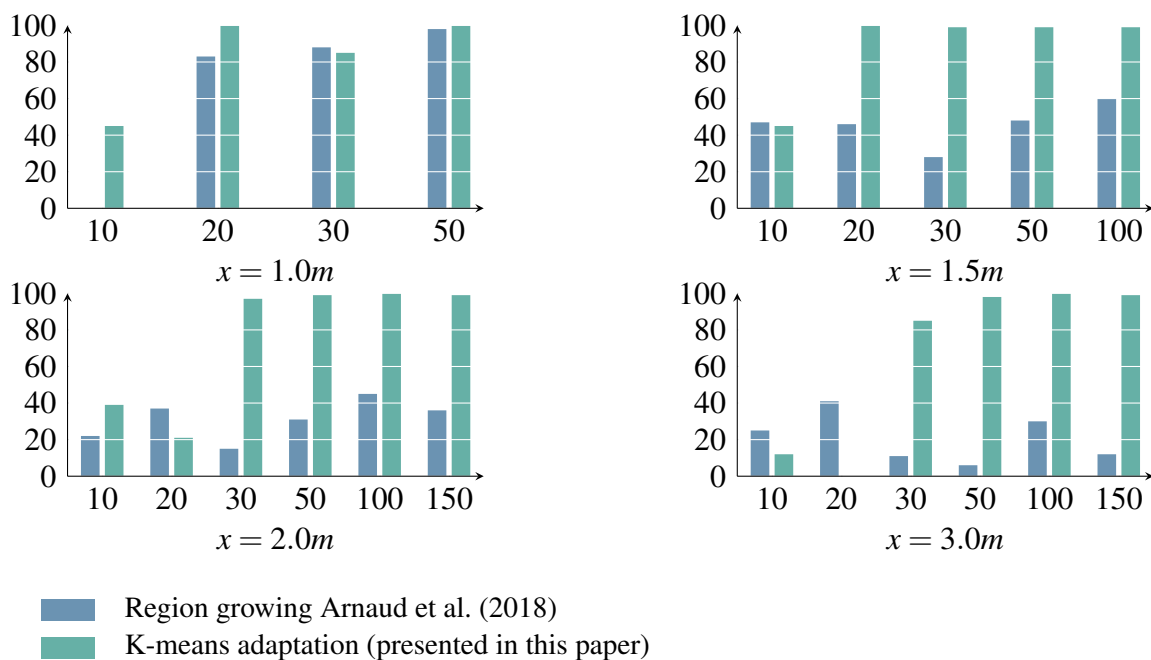
345 Once the scan is made, we have a 3D model representing a set of walls and their geometric dimensions.  
 346 Then the estimated dimensions can be compared with the real ones, measured by using a laser meter.

347 The three rooms used for evaluation are shown in Fig. 11. First, the rooms are scanned, the walls are  
 348 identified, as explained in section 4.4, and their dimensions are estimated. The experience is repeated 10  
 349 times for each room.

350 The table 3 synthesizes the results using the precision and execution times for three methods. The first  
 351 one is the accelerated implementation of CHISEL algorithm Klingensmith et al. (2015), with the use of  
 352 RANSAC for planar detection Arnaud et al. (2016). This version does not run in real-time and the execution

$x$	$D$	$d$	$n$	$x$	$D$	$d$	$n$
100	10	10	45	150	10	13	45
	20	22	100		20	23	100
	30	32	85		30	31	99
	50	50	100		50	55	99
					100	101	99
200	10	14	39	300	10	20	12
	20	22	21		20	-	0
	30	32	97		30	34	85
	50	51	99		50	51	98
	100	100	100		100	100	100
	150	150	99		150	150	99
$\bar{\delta} = 1.46 \text{ cm}$				$\bar{n} = 77.15 \%$			
Average results obtained with method Arnaud et al. (2018)							
$\bar{\delta} = 2.1 \text{ cm}$				$\bar{n} = 40.5 \%$			

**Table 2.** Results of the evaluation of the segmentation algorithm for 100 measures.  $x$  represents the distance between the tablet and the furthest plane,  $D$  the real distance between the two planes and  $d$  the measured one. All these distances are expressed in centimeters.  $n$  represents the number of valid measured frames for each condition.



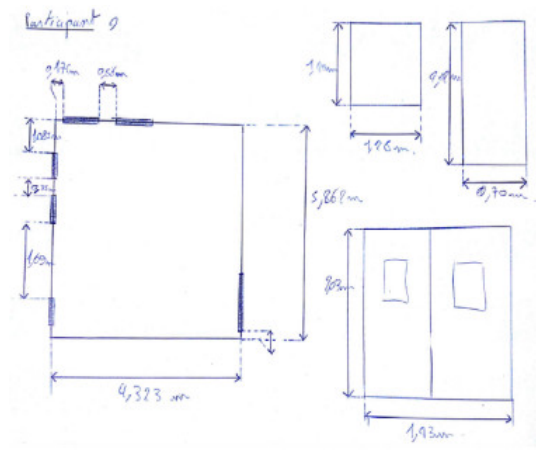
**Figure 10.** Comparison of the percentage of frames where two planes were detected  $n$  (ordinate axis) for both planar segmentations algorithms depending on the inter-planes distance  $D$  (abscissa) for each different configuration of  $x$ .

353 time varies from 200 ms to 500ms. Concerning the algorithm detailed in Arnaud et al. (2018), which  
 354 estimates the 3D mesh of the room before achieving the bottom-up segmentation, it just reaches real-time  
 355 execution. The mean error is approximately 5% of the real dimensions, with a maximum error of 25 %.  
 356 The maximum error is obtained for the meeting room, where the ceiling lights distort the measurements of  
 357 the depth sensor. The proposed approach reduces the errors, as also shown in table 2 and it is faster. Note  
 358 also that the furniture in the scenes (the clutter) do not harm the detection of the walls because each wall is





**Figure 11.** Images of the rooms used for evaluation.



**Figure 12.** Example of sketch drawn by an user during the usability experiment.

359 partly visible. Of course, errors could occur in the following situations: when a wall is totally occluded  
 360 from the floor to the ceiling by a piece of furniture, when the room is not square or rectangular.

Method	Precision	Execution Time
Segmentation of the 3D mesh + RANSAC Arnaud et al. (2016)	5-17%	200 to 500ms
Fusion of RGB, contours + bottom-up segmentation Arnaud et al. (2018)	3 – 25%	≈ 200ms
<b>Proposed approach</b>	3-15%	≈ 100 ms

**Table 3.** Synthetic comparison of three different methods. The precision is given as a percentage w.r.t the real dimensions.

361 **5.4 Evaluation of the usability**

362 Some experiments have been conducted to evaluate the usability of the system. First, we compare the time  
 363 needed for manually measuring the dimensions of a room using a laser meter, with the time needed to scan  
 364 and generate the model with the proposed system. Ten people of different ages, genders and professions  
 365 participated to the experiment. 419 ( $\pm 73$ ) seconds were needed to scan the room with the laser meter,  
 366 whereas it took 247 ( $\pm 44$ ) seconds to get the 3D model when using the proposed system. Let us underline  
 367 that the 419 seconds do not include the modeling stage, *i.e.* entering the dimensions manually in a CAD  
 368 software. Fig. 12 shows an example of sketch made by one the user after measuring the room. We applied  
 369 the SUS questionnaire Brooke (1996) to evaluate the usability of the system. Users answer 10 questions  
 370 formulated in such a way that they are generic and apply to any type of service or system. A score close to  
 371 100 indicates excellent satisfaction.

372 Using a SUS questionnaire Brooke (1996), a mean usability score of 75 was obtained which correspond  
373 to a *good* satisfaction, which shows that the application can be easily understood by inexperienced users.  
374 Among the 10 participants, 2 users found that the use of the tablet did not bring any advantage compared to  
375 a manual measure. The usability score was higher than 75 for seven participants, with a maximum score of  
376 93 for three persons.

## 6 CONCLUSION AND FUTURE WORKS

377 This paper has described an application that runs on a tablet equipped with a depth sensor that generates  
378 and updates a 3D model of an indoor environment. The 3D model is built in real-time and uses exclusively  
379 the computing capabilities of the tablet. This has been made possible by making assumptions (the building  
380 structure is aligned on an Euclidean grid), by using adapted data structures (hash tables and binary trees),  
381 by combining a fast planar segmentation using hierarchical clustering, by achieving code acceleration  
382 (SIMD) and multi-threading.

383 Our evaluations show that the planar segmentation algorithm is able to distinguish parallel planes if they  
384 are separated by more than 10 cm, with a very high accuracy, and that the planes placement accuracy is  
385 less than 2 cm. Compared to previous work, the precision and speed have been significantly improved.  
386 This accuracy can be further improved by using a more accurate depth sensor. Some user experiments have  
387 also shown that it takes approximately twice less time to scan a room compared to the measurement using  
388 a laser meter. A usability score of 75 was obtained. In addition, the 3D model can be edited in a CAD  
389 software, for example to estimate the energetic performances, to plan renovation and decorating works.

390 One of the perspectives of this work is the improvement of the proposed application, while maintaining  
391 the real-time performance, which is key for a good user experience. Concerning the 3D planar segmentation,  
392 it would be interesting to extend the work to more complex rooms where walls are not perpendicular, or  
393 when some of the walls are totally made of glass. Regarding the analysis of the walls, the key elements of  
394 the rooms, such as openings, could be made by using a semantic segmentation Zhang et al. (2013) or by  
395 recent deep learning techniques. To finish, the parameters of the algorithms have been selected manually.  
396 Even if they are satisfactory for all the scenes we studied, a calibration is a component that could be useful  
397 in other contexts (in order to update the parameters automatically).

398 In a more technical concern, the proposed application has been developed on Android for the Tango  
399 Platform, which has a depth sensor and a visual odometry system. This is unfortunately not the case for all  
400 devices, but more and more Android models are proposed at an affordable price Taneja (2nd April 2020).  
401 When not available on the device, a visual odometry algorithm can be installed Li and Mourikis (2012).  
402 More generally, the future application could be available in different versions, depending on the targeted  
403 device and its components. This requires consequent engineering work, which is out of the scope of this  
404 work.

## 7 ACKNOWLEDGMENT

405 This work was funded by Rénovation Plaisir Energie (RPE).

## REFERENCES

406 Adan, A. and Huber, D. (2011). 3d reconstruction of interior wall surfaces under occlusion and clutter. In  
407 *2011 International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*  
408 (IEEE), 275–281

- 409 Arnaud, A., Christophe, J., Gouiffès, M., and Ammi, M. (2016). 3D reconstruction of indoor building  
410 environments with new generation of tablets. In *VRST '16 Proceedings of the 22nd ACM Conference on*  
411 *Virtual Reality Software and Technology*. 187–190
- 412 Arnaud, A., Gouiffès, M., and Ammi, M. (2018). On the fly plane detection and time consistency for indoor  
413 building wall recognition using a tablet equipped with a depth sensor. *IEEE Access* 6, 17643–17652
- 414 Barbu, A. and Zhu, S.-C. (2005). Generalizing Swendsen-Wang to sampling arbitrary posterior probabilities.  
415 *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27, 1239–1253
- 416 Besl, P. J. and McKay, N. D. (1992). A method for registration of 3-D shapes. *IEEE Trans. Pattern Anal.*  
417 *Mach. Intell.* 14, 239–256. doi:10.1109/34.121791
- 418 Borrmann, D., Elseberg, J., Lingemann, K., and Nüchter, A. (2011). The 3D Hough Transform for plane  
419 detection in point clouds: A review and a new accumulator design. *3D Research* 2, 1–13
- 420 Brooke, J. (1996). SUS-A quick and dirty usability scale. *Usability evaluation in industry* 189, 4–7
- 421 Celebi, M. E., Kingravi, H. A., and Vela, P. A. (2013). A comparative study of efficient initialization  
422 methods for the k-means clustering algorithm. *Expert systems with applications* 40, 200–210
- 423 Coughlan, J. M. and Yuille, A. L. (1999). Manhattan world: Compass direction from a single image by  
424 bayesian inference. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International*  
425 *Conference on (IEEE)*, vol. 2, 941–947
- 426 Deng, Z., Todorovic, S., and Latecki, L. J. (2017). Unsupervised object region proposals for RGB-D indoor  
427 scenes. *Computer Vision and Image Understanding* 154, 127–136
- 428 Erdogan, C., Paluri, M., and Dellaert, F. (2012). Planar segmentation of RGB-D images using fast linear  
429 fitting and markov chain monte carlo. In *Computer and Robot Vision (CRV), 2012 Ninth Conference on*  
430 *(IEEE)*, 32–39
- 431 Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering  
432 clusters in large spatial databases with noise. In *Kdd*. vol. 96, 226–231
- 433 Fulkerson, B., Vedaldi, A., and Soatto, S. (2009). Class segmentation and object localization with  
434 superpixel neighborhoods. In *Computer Vision, 2009 IEEE 12th International Conference on (IEEE)*,  
435 670–677
- 436 Grilli, E., Menna, F., and Remondino, F. (2017). A review of point clouds segmentation and classification  
437 algorithms. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information*  
438 *Sciences* 42, 339
- 439 Gupta, S., Arbelaez, P., and Malik, J. (2013). Perceptual organization and recognition of indoor scenes from  
440 RGB-D images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.  
441 564–571
- 442 Henry, P., Krainin, M., Herbst, E., Ren, X., and Fox, D. (2012). RGB-D mapping: Using Kinect-style  
443 depth cameras for dense 3D modeling of indoor environments. *The International Journal of Robotics*  
444 *Research* 31, 647–663
- 445 Holz, D., Holzer, S., Rusu, R. B., and Behnke, S. (2011). Real-time plane segmentation using RGB-D  
446 cameras. . In *Robot Soccer World Cup.*. Springer, Berlin, Heidelberg. 306–317
- 447 Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern recognition letters* 31, 651–666
- 448 Jung, J., Hong, S., Jeong, S., Kim, S., Cho, H., Hong, S., et al. (2014). Productive modeling for  
449 development of as-built BIM of existing indoor structures. *Automation in Construction* 42, 68–77
- 450 Jung, Y. and Joo, M. (2011). Building information modelling (BIM) framework for practical  
451 implementation. *Automation in construction* 20, 126–133
- 452 Kang, Z., Yang, J., Yang, Z., and Cheng, S. (2020). A Review of Techniques for 3D Reconstruction of  
453 Indoor Environments. *ISPRS Int. J. Geo-Inf* 9

- 454 Klingensmith, M., Dryanovski, I., Srinivasa, S., and Xiao, J. (2015). Chisel: Real time large scale 3d  
455 reconstruction onboard a mobile device using spatially hashed signed distance fields. *Robotics: Science  
456 and Systems XI*
- 457 Lai, K., Bo, L., and Fox, D. (2014). Unsupervised feature learning for 3D scene labeling. In *Robotics and  
458 Automation (ICRA), 2014 IEEE International Conference on (IEEE)*, 3050–3057
- 459 Lee, D. C., Hebert, M., and Kanade, T. (2009). Geometric reasoning for single image structure recovery. In  
460 *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on (IEEE)*, 2136–2143
- 461 Li, M. and Mourikis, A. I. (2012). Improving the accuracy of EKF-based visual-inertial odometry. In *IEEE  
462 Int. Conf. on Robotics and Automation (ICRA) (IEEE)*, 828–835
- 463 Liang, Y., He, F., , and Zeng, X. (2020). 3d mesh simplification with feature preservation based on whale  
464 optimization algorithm and differential evolution. *Integrated Computer-Aided Engineering 27*, 417–435
- 465 Macher, H., Landes, T., and Grussenmeyer, P. (2015). Point clouds segmentation as base for as-built BIM  
466 creation. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences 2*,  
467 191
- 468 Moreira, A. and Santos, M. Y. (2007). Concave hull: A k-nearest neighbours approach for the computation  
469 of the region occupied by a set of points
- 470 Nguyen, A. and Le, B. (2013). 3D point cloud segmentation: A survey. In *RAM*. 225–230
- 471 Ochmann, S., Vock, R., Wessel, R., and Klein, R. (2015). Automatic reconstruction of parametric building  
472 models from indoor point clouds. *Computers & Graphics 54*. doi:10.1016/j.cag.2015.07.008
- 473 Papon, J., Abramov, A., Schoeler, M., and Worgotter, F. (2013). Voxel cloud connectivity segmentation-  
474 supervoxels for point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern  
475 Recognition*. 2027–2034
- 476 Ren, X., Bo, L., and Fox, D. (2012). Rgb-(d) scene labeling: Features and algorithms. In *Computer Vision  
477 and Pattern Recognition (CVPR), 2012 IEEE Conference on (IEEE)*, 2759–2766
- 478 Schnabel, R., Wahl, R., and Klein, R. (2007). Efficient ransac for point-cloud shape detection. In *Computer  
479 graphics forum (Wiley Online Library)*, vol. 26, 214–226
- 480 Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. (2012). Indoor segmentation and support inference  
481 from RGB-D images. In *European Conference on Computer Vision (Springer)*, 746–760
- 482 Swendsen, R. H. and Wang, J.-S. (1987). Nonuniversal critical dynamics in Monte Carlo simulations.  
483 *Physical review letters 58*, 86
- 484 [Dataset] Taneja, A. (2nd April 2020). Top 10 Smartphones With A Dedicated Depth  
485 Sensor Camera To Capture Perfect Bokeh Shots. [https://www.cashify.in/  
486 top-10-smartphones-with-a-dedicated-depth-sensor-camera-to-capture-perfect](https://www.cashify.in/top-10-smartphones-with-a-dedicated-depth-sensor-camera-to-capture-perfect)  
487 Accessed 2020/10/01
- 488 Tarsha-Kurdi, F., Landes, T., Grussenmeyer, P., et al. (2007). Hough-transform and extended RANSAC  
489 algorithms for automatic detection of 3D building roof planes from LIDAR data. In *Proceedings of the  
490 ISPRS Workshop on Laser Scanning*. vol. 36, 407–412
- 491 Tatavarti, A., Papadakis, J., and Willis, A. R. (2017). Towards real-time segmentation of 3D point cloud  
492 data into local planar regions. In *SoutheastCon, 2017 (IEEE)*, 1–6
- 493 Verma, V., Kumar, R., and Hsu, S. (2006). 3D building detection and modeling from aerial LIDAR data. In  
494 *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on (IEEE)*, vol. 2,  
495 2213–2220
- 496 Wang, D., Hassan, O., Morgan, K., and Weatherill, N. (2006). Efficient surface reconstruction from  
497 contours based on two-dimensional Delaunay triangulation. *International journal for numerical methods  
498 in engineering 65*, 734–751

- 499 Zhang, D., He, F., Tu, Z., Zou, L., and Chen, Y. (2020). Pointwise geometric and semantic learning  
500 network on 3D point clouds. *Integrated Computer-Aided Engineering* 27, 57–75
- 501 Zhang, J., Kan, C., Schwing, A. G., and Urtasun, R. (2013). Estimating the 3D Layout of Indoor Scenes  
502 and Its Clutter from Depth Sensors. In *IEEE International Conference on Computer Vision (ICCV)*.  
503 1273–1280