Room Floor Plan Generation on a Project Tango Device

✉Vincent Angladon[1,2], Simone Gasparini[1], and Vincent Charvillat[1]

[1] Université de Toulouse; INPT – IRIT; 118 Route de Narbonne, F-31062 Toulouse, France,
{vincent.angladon, simone.gasparini, vincent.charvillat}@irit.fr,
[2] Telequid; Toulouse, France

Abstract. This article presents a method to ease the generation of room floor plans with a Project Tango device. Our method takes as input range images as well as camera poses. It is based on the extraction of vertical planar surfaces we label as wall or clutter. While the user is scanning the scene, we estimate the room layout from the labeled planar surfaces by solving a shortest path problem. The user can intervene in this process, and affect the estimated layout by changing the label of the extracted planar surfaces. We compare our approach with other mobile applications and demonstrate its validity. ***Index Terms*—** Floor plan, Mobile device, RGB-D cameras, 2D/3D mapping

1 Introduction

Floor plan generation is the problem of generating a drawing or a digital model to scale of an existing room or building. A common workflow consists in taking individual measurements reported on a freehand sketch of the floor, then in using a CAD software to draw the floor plan. This approach is mostly manual and requires the user to collect all the measurements, typically with a laser range finder. The task is not trivial, as it requires the correct use of the measurement tool and the collection of all the necessary measurements to fully define the floor plan: missing measurements may lead to incomplete plans, thus requiring costly do-overs on site. Moreover, the task can be challenging when dealing with furnished or cluttered environments, preventing, *e.g.*, the direct measurement of certain distances. In order to provide more automatic and efficient solutions, the use of desktop or mobile applications performing the drawing on-site has gained a lot of interest over the last years.

With the latest advancement in 3D scanning technologies, 3D scanners placed on tripods have become an interesting solution for the generation of the floor plan. The generated 3D point cloud enables to perform Building Information Modeling (BIM) and floor plan generation in a semi-automatic way, through the use of a dedicated software. The operating cost is generally higher than the previous method, due to the investment of the scanner and the scanning time which can be quite long when there are several small rooms. The automatic creation of floor plans or BIM from such scanners has been extensively researched for single rooms [1,2] and multiple rooms [3,4,5,6,7,8]. These methods, are called *offline* because they start the processing after the scan is complete and all the data is available.

At the opposite, for handheld scanners, *online* methods provide results incrementally during the scanning process. Several online approaches have been proposed [9,10,11]

to extract some walls of indoor scenes, but none of them focus on the creation of single or multiple room floor plans.

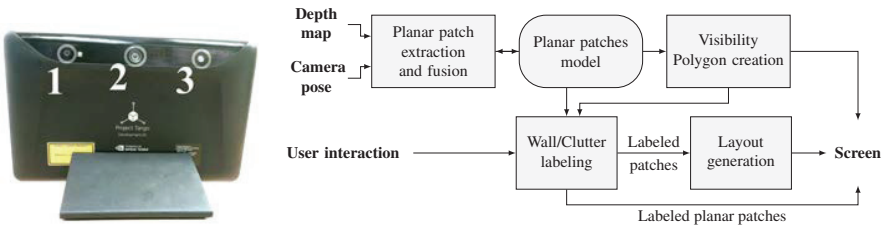
On common smartphones, user-driven approaches have been proposed [12,13,14]. Knowing the phone orientation and the vertical distance of the device to the floor (assumed constant and calibrated), they estimate the distances between the user and the room corners. More recently, as mobile devices started to sport depth sensors, user-driven approaches have been proposed for the Project Tango mobile devices: the user is required to manually select the walls with FloorPlanEx [15] or the edge corners of the indoor scene [16].

In this paper, we propose an online approach for the Tango Tablet Development Kit(TDK) Fig. 1a, which is a tablet equipped with a depth sensor and a fish eye camera running on Android. It offers dedicated libraries to perform localization and 3D reconstructions in the form of 3D textured meshes. Our method can generate a room floor plan with optional user interaction. We rely on the ability of this device to localize itself and capture depth maps, in order to have a better understanding of the scene and hence reduce the user efforts. Given the complexity of the task and the diversity of the scenes, the user interaction is yet helpful to improve the robustness of the approach: large occlusions, lack of physical separation between rooms, missing data, incorrect depth perception (*e.g.* glasses and mirrors), and other issues that could be challenging for a fully automatic system, can be easily solved by the operator. [17] enumerates and classifies all the issues related to the problem of indoor mapping and modeling. While the Tango TDK addresses most of the *acquisitions and sensors* problems (variable lighting conditions, sensor fusion, mobility support), our works also tackles *acquisition* problems (variable occupancy support), *data structure and modeling* issues (real-time modeling, dynamic abstraction), *visualization* problems (on mobile visualization, real-time change visualization) and *legal issues* (user privacy).

2 Overview of the Method

Hypotheses. We assume the considered rooms are made of a horizontal ceiling and floor, and vertical planar walls, the latters not necessarily orthogonal w.r.t. each other. They can contain clutter (furniture, movable objects, ...) occluding the walls. With the Tango TDK, we observed objects located $4.2\,\mathrm{m}$ away suffer from a depth uncertainty over $25 \pm 9\,\mathrm{mm}$, and lack of texture (*e.g.* walls and ceiling of uniform color) can lead to incorrect camera poses from the Tango motion tracking. Therefore, our approach is limited to rooms with reasonable texture and where all the walls and the ceiling can be observed by the depth sensor without too many efforts (medium size room with a $5\,\mathrm{m}$ ceiling height maximum).

Pipeline. Figure 1b summarizes the proposed pipeline. The Tango Tablet Development Kit middleware provides at each iteration the depth map of the scene, as well as the camera pose. The depth map is processed in order to extract sets of 3D points corresponding to candidate walls, what we call *planar patches* (detailed in Sect. 3). Thanks to the known camera pose, the planar patches are brought into a global world coordinate system, so that they can be associated and then fused with the existing patches of



(a) (b)

Fig. 1: (a) Tango Tablet Development Kit image sensors: RGB-IR camera (1), Fisheye camera (2), IR pattern emitter (3). (b) Our pipeline.

the model (Sect. 4). Whenever the model is updated, the visibility polygon of the discovered area(s) is also updated (Sect. 5), which can be used as a visual feedback for the user. Furthermore, the visibility polygon is used by the labeling module to automatically classify the planar patches as *wall* or *clutter* (Sect. 6), thus allowing the disambiguation between actual walls and other objects that may be lying inside the room. This classifier was trained beforehand against 900 manually labeled vertical planar regions from our training dataset. This last task can take advantage of the interaction of the user, who can correct and change the automatic labeling of the vertical planar patches into wall or clutter (Sect. 7). Finally, the layout generation module computes the room layout from the labeled vertical planes and the boundary given by the visibility polygon (Sect. 8).

3 Planar Patches Extraction

A planar patch is defined as a list of 3D points associated with the equation of the fitted infinite plane and the boundaries of this patch. We define the planar patches extraction problem as a function which takes as input a range image (also called depth map) and returns a list of planar patches.

The common strategies to extract planar patches are the Hough Transform approaches [18], the region growing algorithms [19], the normal map segmentation [20] and the split and merge approaches [21].

In our preliminary tests, we noticed that normal map segmentation approaches were too sensitive to the data noise and required higher computational time. We chose instead a region growing approach, similar to the algorithm described by Poppinga *et al.* [19]. In this approach, a random seed point and its neighbors are picked and extended by taking into consideration neighboring points. A plane is estimated on this set of points and a new point is considered valid when its distance to the plane is small enough. The planar patch keeps growing iteratively until no valid point can be found in the neighborhood of the patch. A new seed point is picked until all points have been considered. We made some modifications to the original algorithm in order to get better results with the point cloud provided by the Tango TDK. We ignore the 3D points with a distance to the camera farther to $3.7\,\mathrm{m}$ as they revealed to be very noisy. To cope with the low density

of the depth map, the neighboring selection is not pixel-wise but performed on a 3×3 pixel window. In a post-processing step, we remove isolated points of the planar patches with a neighborhood pixel analysis performed in the segmented depth map. This ensure our planar patches are more compact. Finally, we perform a validation of the planar patch with the RANSAC algorithm. Without implementing the optimizations suggested by [19], we can process 10k points in $47\,\mathrm{ms}$ with our C++ implementation.

The ceiling and the floor are incrementally estimated with the minimal and maximal height of the considered vertical planar patches and validated with the fitting of horizontal planes at the considered heights. In the following sections, we consider only the vertical planar patches for which we compute a rectangle boundary.

4 Planar Patches Model Update

We create a model of planar patches in the world coordinates which is initialized with the patches of the first frame. For each successive frame, we compute planar patches which are associated with the model with the help of an association function. We update the associated planar patches of the model with a fusion function.

Planar Patch Association. Given a list of planar patches from the model and a list of planar patches from the current frame the association function computes a list of planar patches pairs where each patch appears only once. We consider a distance between planar patches which takes into account the normal angle difference of the planes and the mean distance of the points-to-plane distances, using the points of the rectangle boundary of the planar patches. This distance defines the candidate pairs of associated patches, which are validated with an overlapping test between the rectangle boundaries.

Planar Patch Fusion After two planar patches have been associated, we need to create a new planar patch combining the two. We compute and update the covariances matrices of the planes using [22]. The plane equations of the fused patch is updated with a Principal Component Analysis, in order to avoid the costly storage of the 3D points associated with the planar patches. We then update the rectangular boundary to include the two planar patches.

5 Visibility Polygon Computation

During a scan, the user needs to know which parts of the scene he visited or not. We provide this information with a top-down 2D view representation of the area observed by the camera in the form of a visibility polygon, similarly to [9], but computed differently from the vertical planar patches model and the history of the camera positions. We denote s^{v} the line segment obtained from the projection of a vertical planar patch on a horizontal plane. We consider the visibility view polygon P^s associated to each line segment s^{v}, which is formed by the union of all the triangles formed by the camera position and the two extremities of s^{v}, see Fig. 2. Each triangle is made of one *wall segment* and two *frustum segments*. The geometric union of all the polygons P^s forms the visibility polygon.

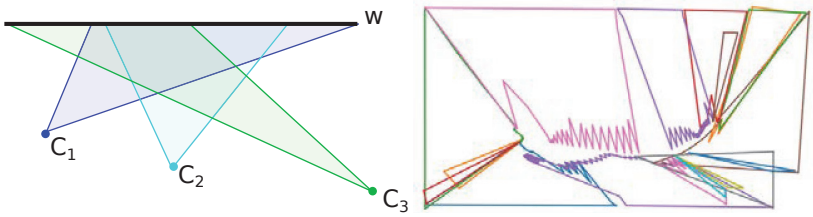


Fig. 2: Left: the three blue, cyan and green triangles represent the part of the camera frustum viewing $\mathrm{s^v}$ and associated with the camera poses c_1, c_2 and c_3 respectively. Their union form P^s: the visibility view polygon associated with the line segment $\mathrm{s^v}$. Right: the visibility polygon is the union of the visibility view polygons P^s, here represented with different colors.

We implemented the computation of the visibility polygon with the help of the geometry engine GEOS [23].

6 Wall-Clutter Separation

While clutter often consists of irregular shapes, such as plants, sofas, *etc*., which are eliminated in planar primitive approaches, it can also consist of piecewise planar shapes such as cupboards, radiators, *etc*. In this section, we propose a method to classify the planar primitive as wall or clutter in order to reduce the number of potential irrelevant planes considered by the room layout estimations component. The segmentation of the primitives can be performed individually [6,24,4], *i.e*. using features on each primitive considered independently, or globally [7,2], *i.e*. considering the adjacent primitives, which provide contextual information.

Our objective is to build the room layout incrementally, during the scan progress, which means all the adjacencies are only known at the end of the scan. Therefore we favored a classification with an individual approach. Our features take the form of a vector (d^c, d^f, l, d^v). They include the distances d^c between the highest point of the vertical planar patch and the estimated ceiling and d^f between the lowest point of the vertical planar patch and the estimated floor. Intuitively, a planar patch both close to the ceiling and floor is likely to be a wall. Similarly, a segment $\mathrm{s^v}$ with a longer length l has a higher probability to correspond to a wall. We also consider the distance $d^v = \max_{p \in \mathrm{s^v}} d(p, \partial\,\mathrm{P^v})$ between a segment $\mathrm{s^v}$ and the exterior boundary of the visibility polygon $\partial\,\mathrm{P^v}$. A high distance d^v corresponds to a segment with at least one extremity far from the visibility polygon, which is likely to correspond to clutter.

We compute a wall probability $P(\mathrm{s^v})$ for each segment $\mathrm{s^v}$ of the model with a Multi-layer Perceptron classifier using one hidden layer and a logistic sigmoid activation, trained on our dataset. When a new frame is ingested, we compute $P(\mathrm{s^v})$ if $\mathrm{s^v}$ was modified since the previous frame or the ceiling/floor estimation changed. Our implementation based on the Python Scikit-learn module can label 50 planar patches in $29\,\mathrm{ms}$ on average.

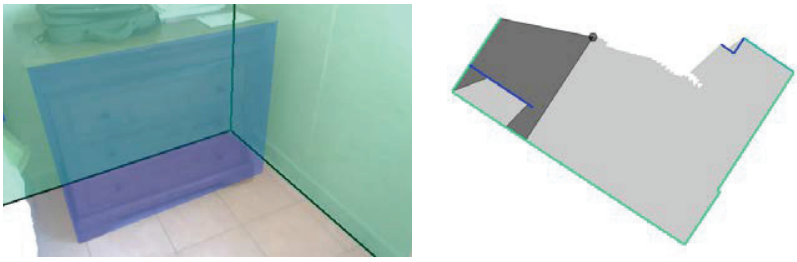


Fig. 3: Visualization of the scan progress of the scene House2. Planar patches classified as wall and clutter are represented in green and blue respectively. Intermediate colors represent intermediate probabilities. Left: augmented view of the device camera with the detected and classified planar patches. The estimated room layout is displayed with black lines. Right: visibility polygon in light gray, camera view polygon in black.

7 User Interaction

Automatic approaches for floor plan generation or scan to BIM can be prone to errors. These errors can come from missing data, clutter, sensor noise or some specificities in the scene (*e.g.* small walls, large openings, cavities in the walls, obstacles fully covering a wall, *etc.*). Depending on the progress of the scan, it may not be possible for an algorithm to determine whether a planar patch corresponds to a wall or clutter.

The user interaction schemes proposed in the literature are usually corrective [25,7,26], where the user intervenes at the end to correct a proposed solution (commonly considered for offline approaches), or user-driven [14,12,13,16,15] where no solution can be computed without user interaction (mostly found in online approaches).

We wanted to propose a collaborative interaction scheme where the interaction would be optional and could be performed at any time. The first way to interact is to move the Tango TDK to make it observe new parts of the scene. A first view displays the camera image augmented with the detected planar patches colored relatively to their wall probability and the estimated layout, while a second view provides a top view of the scene with the visibility polygon and the estimated layout too. The two views shown in Fig. 3 are updated at the frame rate of the depth sensor, giving an immediate feedback to the user who can decide to visit the area(s) with missing data. The user can interchange the wall/clutter labels of the detected patches by touching them in the augmented view, which directly affects the room layout estimation component. At the end of the scan, we let the user perform further editions: suppress forgotten clutter patches and merge planar patches. At the time of writing, we only designed a desktop prototype which can replay a recorded scan or process live data transmitted by the application. The presented results were obtained by performing the interactions on replays of the scans.

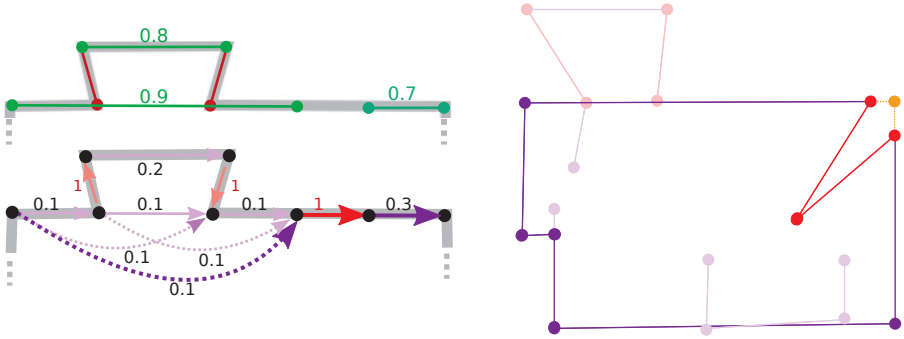
8 Room Layout Estimation

The Room Layout Estimation takes as input the model of line segments s^{v} coarsely classified as wall or clutter and generates a simplified floor plan (see Fig. 1b). In order to compute the room layout, we need to retrieve the topological relationships between the walls, *i.e*. their adjacencies.

Related Work. A simple approach is to consider the topology of the room can be recovered by connecting adjacent segments [27,2,24], which works well for rooms with low clutter, and when the walls are well separated from the clutter. The use of cell-complex is very popular for both single rooms [1,28] and multiple rooms [5,6,7,8] layout estimation. The line segments are replaced by infinite lines which partition the 2D space into polygonal cells. A graph connecting the adjacent cells is defined: instead of considering topological information on the segments, adjacency relationship between the cells is taken into account. The inside/outside label of the cells are computed with a graph-cut algorithm. This approach is robust to missing data because the extension of the segments is automatically considered, but it does not simplifies the layout.

Weighted graph construction. In our approach, the visibility polygon provides the topology, and we try to compute a simple chain of segments s^{v} which explains the visibility polygon. We create a graph from the adjacency relationships between the segments. For simplification purpose, the segments with a low wall probability (inferior to 0.5 in practice) are discarded. The remaining segments S are very likely to be close to the boundary of the visibility polygon, but they might not cover this boundary completely due to missing data. We complete the boundary with segments s^c from the relative complement of S with respect to the visibility polygon. A node is created at each segment extremity and each segment intersection as illustrated in Fig. 4a. The nodes belonging to the same segments are linked by an edge. We assign to each edge e a weight proportional to $1 - P(\mathrm{s}^{\mathrm{v}})$, where s^{v} is the segment associated to e. The edges corresponding to *frustum segments* and completion segments s^c are assigned a penalty weight. With this design, the shortest path, which minimizes the sum of the weights of its edges, gives more importance to segments with a high wall probability and favors solutions with a low number of planar patches.

Solving and discussion. We use the Dijkstra's algorithm to find a cycle in our graph which minimizes the sum of our weights. To avoid trivial solutions, we use a start point on the segment with a high $P(\mathrm{s}^{\mathrm{v}})$, an endpoint which is adjacent, and we remove the edge between them. As shown in Fig. 4a, the graph may contain several cycles which can lead to incorrect layouts. In order to avoid finding an incorrect cycle, we impose the segments along the visibility polygon to correspond to directed edges following a clockwise orientation in the visibility polygon. This method cannot handle the cycles non adjacent to the visibility polygon, which revealed to be non-existent in our dataset. In a post-processing step, we replace, whenever possible, the chains of frustum/completion segments by lines segments extending their adjacent segments, otherwise we simply join them. Figure 4b illustrates the solution computed from a simplified scene. We also apply the Ramer-Douglas-Peucker algorithm [29] with a low threshold



(a) Creation of a weighted graph (bottom) from the set of segments with their probabilities (top). The visibility polygon is represented by a gray thick line. Some edges are overlapping, for visualization purpose, we represented them with a curved dashed arrow line. Red edges correspond to segments from s^c (including *frustum edges*).

(b) Room layout toy example illustrating various cases: open door on the top, missing data in the upper right corner, clutter at the bottom. Orange dashed lines correspond to segments created during the post-processing step.

Fig. 4: Toy examples illustrating the graph creation and the solving. Faded colors represent the segments which are not part of the solution. Red edges correspond to frustum edges, purple for the other edges.

(1 cm) to simplify near parallel adjacent segments. Contrary to the offline approaches mentioned earlier, ours can deliver an immediate room layout. Our Python implementation of this component takes 200 ms to generate the graph, and 11 ms to compute the shortest path and apply the post-processing steps.

For a satisfying user experience, the layout of previously seen areas should not change when the user visits a new part of the scene. This behavior cannot be guaranteed with an offline approach. When the ceiling and the floor are detected, our method is suitable for incremental changes of the model: the wall probability $P(\mathrm{s}^{\mathrm{v}})$ of the previously observed segments s^{v} does not change, which means the computed path restricted to the previously seen segments is the same and has the same cost.

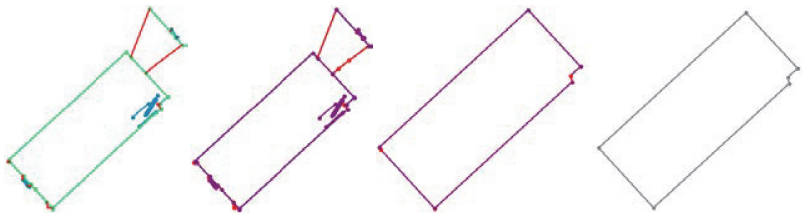


Fig. 5: Four steps of our room layout estimation : the extracted segments, the generated graph, the shortest path solution, and the layout obtained after post-processing. See Fig. 3 and Fig. 4 for the signification of the colors.

9 Experimental Results

In this section, we compare –the geometry accuracy of– the room floor plans generated with our approach, Magic Plan [14] (run on iPad Air 1, prior to ARKit release) and FloorPlanEx from Google [15].

Evaluation protocol. We considered five indoor scenes Lab1^{MW}, Lab2, House1^{MW}, House2^{MW} and House3^{MW}, where MW denotes the scenes respecting the Manhattan World assumption. The ground truth room layouts of these scenes were created with a Bosh DLE 50 laser ranger finder. We evaluated the geometry accuracy of the obtained layouts with the ground truth and the reproducibility of the measurements by repeating the measurements five times. Each estimated layout was aligned with the ground truth by computing the transformation which minimizes the distances between their corresponding vertices. The mean of these distances defines our residual error. We also evaluated the user effort during the use of the considered mobile applications. Magic Plan and FloorPlanEx are user-driven applications where the user selects the walls and the corners, respectively. The number of interactions is equal to the number of corners (plus one for Magic Plan). We did not count the interactions required for Magic Plan calibration processes. For our approach, we evaluated the number of labels corrections on the planar patches and the number of post-scan modifications.

Results and analysis. Table 1 and Fig. 6 show the obtained results. Magic Plan estimates the camera-to-corner distances from the device orientation instead of taking advantage of a localization module or a depth sensor. Consequently, even the best results of the application are less accurate than the results obtained with the other approaches. Due to the amount of clutter, most of the corners were captured on the ceiling, which reduces the accuracy of the measurements. Magic Plan assumes the angle between two consecutive walls is $90°$ or $45°$, for this reason, the results are unsatisfactory on the scene Lab2 which does not follow the MW assumption. We can also observe the residual increases with the area of the room, which is coherent when there is a small error with the device height estimation.

For selling or renting a property in France, the Alur law defines the maximal error of the measured area to 5%. The area errors from the FloorPlanEx application and our approach are inferior or equal to this threshold, which may not be enough for some official uses. The results show our method is generally more accurate and provides more repeatable results than FloorPlanEx. One explanation is that we consider the 3D points from multiple frames to estimate the planes of the walls when the FloorPlanEx only considers the points from one frame.

The last column of Table 1 describes the degree of interaction. It confirms our approach generally requires fewer screen interactions than user-driven approaches, except for the scene House3^{MW} which contained a fireplace, high furnitures and many curtains. The Lab1^{MW} was also quite challenging because of the presence of a high cupboard and pillars which were incorrectly labeled as wall.

Scene	Method	Mean area err.	Max area err.	Mean residual	$\sigma_{resid.}$	Min residual	Max residual	Number Interact.
Lab1^{MW} ($25\,m^2$)	Ours	2.3 %	**4.2 %**	**29 mm**	**14 mm**	**14 mm**	**48 mm**	**3.25**
	FloorPlanEx	**2.2 %**	4.5%	47 mm	26 mm	18 mm	84 mm	4
	Magic Plan	12 %	17 %	164 mm	52 mm	106 mm	231 mm	5
Lab2 ($47\,m^2$)	Ours	**1.1 %**	**2.4 %**	**38 mm**	**3 mm**	**35 mm**	**43 mm**	**0.75**
	FloorPlanEx	3.3 %	4.3 %	73 mm	19 mm	45 mm	100 mm	6
	Magic Plan	15 %	24%	264 mm	65 mm	199 mm	329 mm	7
House1^{MW} ($11\,m^2$)	Ours	**1.8 %**	**2.4 %**	**30 mm**	12 mm	**14 mm**	**44 mm**	**0.5**
	FloorPlanEx	2.6 %	4.5 %	53 mm	**9 mm**	38 mm	60 mm	6
	Magic Plan	4.9 %	8.8 %	66 mm	18 mm	46 mm	87 mm	7
House2^{MW} ($13\,m^2$)	Ours	**3.0 %**	5.0 %	**29 mm**	12 mm	**17 mm**	**49 mm**	**0**
	FloorPlanEx	3.3 %	**4.2 %**	41 mm	**11 mm**	28 mm	58 mm	4
	Magic Plan	5.4 %	9.2 %	50 mm	23 mm	22 mm	89 mm	5
House3^{MW} ($48\,m^2$)	Ours	**1.9 %**	**2.3 %**	**32 mm**	**5 mm**	**27 mm**	**37 mm**	7.5
	FloorPlanEx	2.8 %	4.0 %	105 mm	23 mm	78 mm	144 mm	**6**
	Magic Plan	15.4 %	24.9 %	233 mm	79 mm	163 mm	344 mm	7

Table 1: Results of the geometry accuracy and reproducibility comparison experiment.

10 Conclusion

We presented an online approach to estimate and measure the room layout of an indoor scene using a Project Tango mobile device. The various components of our pipeline were evaluated on a desktop computer, using scans recorded from a Project Tango mobile device. Compared to existing online works, the proposed method relies on scene understanding to compute the room layout. In order to cope with the complexity of some rooms, the user can interact in a cooperative way to add or remove walls among the observed planes. The comparison with other mobile applications demonstrates that in general, we achieve higher accuracy. In term of efficiency, the number of user interactions depends on the complexity of the scenes, which may contain clutter data incorrectly classified as walls. Some limitations of the proposed method are implicitly associated to the Tango TDK: drift of the device pose and surfaces not handled by the depth sensors, such as mirrors and glasses. Applications of our works include floor plan generation, area estimation and room reconstruction for virtual reality games.

References

1. Budroni, A., Boehm, J.: Automated 3d reconstruction of interiors from point clouds. International Journal of Architectural Computing **8**(1) (2010) 55–73
2. Xiong, X., Adan, A., Akinci, B., Huber, D.: Automatic creation of semantically rich 3d building models from laser scanner data. Automation in Construction **31** (2013) 325–337
3. Xiao, J., Furukawa, Y.: Reconstructing the worlds museums. International journal of computer vision **110**(3) (2014) 243–258

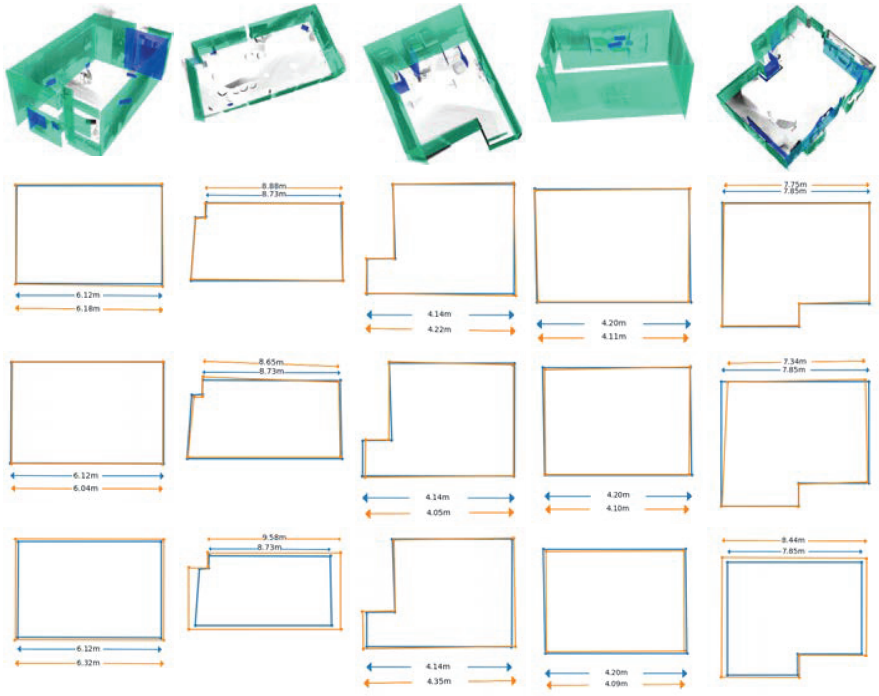


Fig. 6: First row: the point clouds and the labeled planar patches (blue: clutter, green: walls) of the scanned rooms: Lab1^{MW}, Lab2, House1^{MW}, House2^{MW} and House3^{MW} (from left to right). Second row: comparison of our approach in orange with the ground truth in blue. Third and fourth row : comparison of the room layout obtained with FloorPlanEx and Magic Plan in orange and the ground truth in blue.

4. Stambler, A., Huber, D.: Building modeling through enclosure reasoning. In: 3D Vision (3DV), 2014 2nd International Conference on. Volume 2., IEEE (2014) 118–125
5. Oesau, S., Lafarge, F., Alliez, P.: Indoor scene reconstruction using feature sensitive primitive extraction and graph-cut. ISPRS Journal of Photogrammetry and Remote Sensing **90** (2014) 68–82
6. Mura, C., Mattausch, O., Villanueva, A.J., Gobbetti, E., Pajarola, R.: Automatic room detection and reconstruction in cluttered indoor environments with complex room layouts. Computers & Graphics **44** (2014) 20–32
7. Mura, C., Mattausch, O., Pajarola, R.: Piecewise-planar reconstruction of multi-room interiors with arbitrary wall arrangements. In: Proceedings of the 24th Pacific Conference on Computer Graphics and Applications. PG '16, Goslar Germany, Germany, Eurographics Association (2016) 179–188
8. Ochmann, S., Vock, R., Wessel, R., Klein, R.: Automatic reconstruction of parametric building models from indoor point clouds. Computers & Graphics **54** (2016) 94–103
9. Zhang, Y., Luo, C., Liu, J.: Walk&sketch: create floor plans with an rgb-d camera. In: UbiComp. (2012)

10. Zhang, Y., Xu, W., Tong, Y., Zhou, K.: Online structure analysis for real-time indoor scene reconstruction. ACM Transactions on Graphics (TOG) **34**(5) (2015) 159
11. Dzitsiuk, M., Sturm, J., Maier, R., Ma, L., Cremers, D.: De-noising, stabilizing and completing 3d reconstructions on-the-go using plane priors. In: Robotics and Automation (ICRA), 2017 IEEE International Conference on, IEEE (2017) 3976–3983
12. Rosser, J., Morley, J., Smith, G.: Modelling of building interiors with mobile phone sensor data. ISPRS International Journal of Geo-Information **4**(2) (2015) 989–1012
13. Pintore, G., Agus, M., Gobbetti, E.: Interactive mapping of indoor building structures through mobile devices. In: 3D Vision (3DV), 2014 2nd International Conference on. Volume 2., IEEE (2014) 103–110
14. Sensopia: MagicPlan, Create a floor plan in just a few minutes. http://www.magic-plan.com (2012)
15. Google: java_floor_plan_example create a floor plan by using the depth sensor of a Google Tango device to detect and measure walls in a room. https://goo.gl/3Ns3XY (2016)
16. Google: Measure: augmented reality measurement application for Google Tango devices. https://goo.gl/2RyPnz (2016)
17. Zlatanova, S., Sithole, G., Nakagawa, M., Zhu, Q.: Problems in indoor mapping and modelling. Acquisition and Modelling of Indoor and Enclosed Environments 2013, Cape Town, South Africa, 11-13 December 2013, ISPRS Archives Volume XL-4/W4, 2013 (2013)
18. Borrmann, D., Elseberg, J., Lingemann, K., Nüchter, A.: The 3d hough transform for plane detection in point clouds: A review and a new accumulator design. 3D Res. **2**(2) (March 2011) 32:1–32:13
19. Poppinga, J., Vaskevicius, N., Birk, A., Pathak, K.: Fast plane detection and polygonalization in noisy 3d range images. In: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems. (Sept 2008) 3378–3383
20. Holz, D., Holzer, S., Rusu, R.B., Behnke, S.: Robot soccer world cup xv. Springer-Verlag, Berlin, Heidelberg (2012) 306–317
21. Feng, C., Taguchi, Y., Kamat, V.R.: Fast plane extraction in organized point clouds using agglomerative hierarchical clustering. In: 2014 IEEE International Conference on Robotics and Automation (ICRA). (May 2014) 6218–6225
22. Lee, T.K., Lim, S., Lee, S., An, S., Oh, S.Y.: Indoor mapping using planes extracted from noisy rgb-d sensors. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. (2012)
23. OSGeo: GEOS, Geometry Engine, Open Source. http://goo.gl/RqHPM6 (2000)
24. Murali, S., Speciale, P., Oswald, M.R., Pollefeys, M.: Indoor scan2bim: Building information models of house interiors. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). (2017)
25. Arikan, M., Schwärzler, M., Flöry, S., Wimmer, M., Maierhofer, S.: O-snap: Optimization-based snapping for modeling architecture. ACM Transactions on Graphics (TOG) **32**(1) (2013) 6
26. Pintore, G., Ganovelli, F., Gobbetti, E., Scopigno, R.: Mobile mapping and visualization of indoor structures to simplify scene understanding and location awareness. In: European Conference on Computer Vision, Springer (2016) 130–145
27. Valero, E., Adán, A., Cerrada, C.: Automatic method for building indoor boundary models from dense point clouds collected by laser scanners. Sensors **12**(12) (2012) 16099–16115
28. Previtali, M., Barazzetti, L., Brumana, R., Scaioni, M.: Towards automatic indoor reconstruction of cluttered building rooms from point clouds. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences **2**(5) (2014) 281
29. Ramer, U.: An iterative procedure for the polygonal approximation of plane curves. Computer graphics and image processing **1**(3) (1972) 244–256