Indoor Scan2BIM: Building Information Models of House Interiors

Srivathsan Murali, Pablo Speciale, Martin R. Oswald and Marc Pollefeys[1]

***Abstract*— We present a system to generate building information models (BIMs) of house interiors from 3D scans. The strength of our approach is its simplicity and low runtime which allows for mobile processing applications. We consider scans of single floor, Manhattan-like indoor scenes for which our method creates metric room layouts by detecting walls and performing a subsequent reasoning about their neighborhood relations. The output of our method is a 3D BIM with hierarchical semantic annotations for individual rooms being refined by walls, ceilings, floors and doors. A variety of experiments demonstrate the effectiveness of our approach. Our reconstruction results compare well to other state-of-art methods in both reconstruction quality as well as runtime.**

I. Introduction

Original floor plans and 3D building information models (BIMs) of a property are often not available or outdated due to past modifications. Hence, interior designers and builders often recreate building floor plans using distance measures acquired using point-to-point measuring devices. This is a tedious and time consuming task which we aim to automatize.

In this paper, we propose a system that automatically generates 3D room layouts from indoor building data, either obtained from image-based matching algorithms or mobile 3D depth sensors like Google Project Tango [1], [2], Microsoft Kinect [3] or Microsoft HoloLens [4]. The sensor data captured by these devices is used to generate a 3D mesh of the indoor environments.

The use of mobile scanning devices aids in reducing the amount of missing geometry caused by occlusions due to objects in the environment. The user can instantly check the model completeness and re-scan certain scene parts if necessary. Mobile devices also reduce the time required to acquire the input 3D mesh. Unlike laser scanners, devices such as the Project Tango tablet can be made available at lower costs, making scans of indoor environments accessible and easier to use by everyone.

Compared to outdoor scene reconstruction, indoor scenes are challenging since they usually contain high amounts of clutter. In this work we are aiming for an indoor reconstruction which inherently segments major structural elements like walls, floors and doors from furniture, movable objects, clutter and scanning noise. Our goal is to obtain a compact and simplified representation of the scene, jointly with semantic relations between walls, floors and doors as well as their grouping into rooms.

[1]The authors are with the Computer Vision and Geometry Group, Institute for Visual Computing, Computer Science Department, ETH Zürich. *stindiva@student.ethz.ch,* {*pablo, moswald, marc.pollefeys*}*@inf.ethz.ch*.

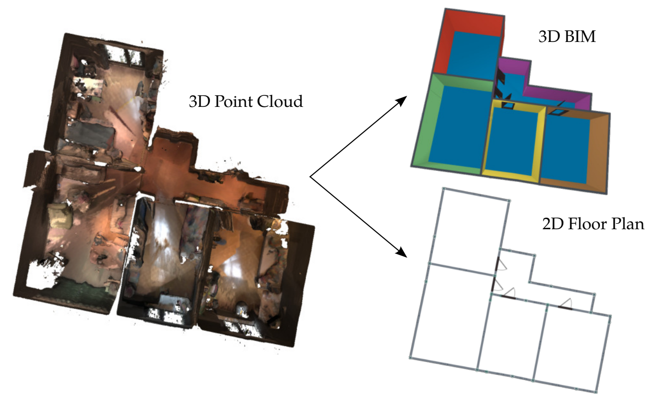


Fig. 1: Given a mesh of a 3D scanned apartment, our method fully automatically computes a floor plan and a 3D building information model at interactive processing time. Our method also provides semantic output in form of plane labels (wall, ceiling, floor, door) and room labels (depicted with different colors) which can be useful for further processing such as semantic reasoning or higher-level navigation.

As illustrated in Fig. 1, our system uses the 3D mesh produced to output a 3D indoor building information model which can then be further processed by interior design software to place furniture and appliances, either offline or on a tablet.

Since our approach is fast and fully automatic, building information models can be acquired with robots, e.g. UAVs. The extracted BIM can also be used for robot navigation and semantic reasoning, since the extracted model contains major structure elements like walls and doors. Movable objects and furniture can then be identified as the difference between BIM and scanned model data.

Moreover, the proposed method attempts to solve the problem in a simpler and efficient way; of course, with the risk of losing some details due mainly to the strong assumptions imposed to the environment, the Manhattan assumption [5]. Nevertheless, providing a solution that can be employed by a regular user in a mobile device, to build a model with similar accuracy as state-of-the-art methods in less time. Then, the focus of the method is to provide a quick solution obtained in seconds, on devices like Tango, instead of minutes (or probably hours, in some cases) using more sophisticated methods where precision is the ultimate goal instead of practicality.

The paper is organized in five sections. Section II describes the previous research on indoor building reconstruction. Section III describes our proposed system. It discusses the methods used to detect walls and how to cluster them into rooms and generate 3D models. Section IV presents the qualitative and quantitative results as well as evaluates the performance and the last Section VI concludes the paper.

II. Related Work

The field of indoor scene reconstruction has attracted lots of research in the past and it is difficult to classify all these works into meaningful categories. We provide an overview of relevant works and focus more on method properties which are important to our problem setting and therefore many approaches appear in multiple categories.

State of the Art in Industry. Multiple commercial software systems exist to solve the indoor reconstruction task, e.g. Google Tango [2] or Matterport's [6] scanning software. Both systems yield compelling and highly detailed 3D reconstruction results, but they are focused on the accurate reconstruction of all details in a scene rather than the creation of minimal model representations that semantically separate the building structure from furniture, objects and clutter. The Matterport system [6] requires a special commercial depth sensor and the processing is offline and not real-time capable, while the Google Tango system [2] runs in real-time on mobile devices.

Real-time Approaches. An early work that tries to simplify scanned 3D geometry is presented in [7] in which an EM-algorithm is applied to fit planar polygons to the input data in real-time. Beyond the Google Tango framework [1], [2] itself the recent work in [**?**] builds on the Tango framework and targets near real-time performance for detecting planes which are then leveraged to order to denoise and complete scanned 3D data. Similar to their and our goal, there are many research works aiming for a denoised, simplified and more compact model representation by approximating surface parts with simple primitives like planar planar polygons. Other real-time or close to real-time approaches include [9], [10].

Primitive-Fitting Approaches. Many works try to simplify the model geometry by replacing larger surface parts with a previously fitted primitive and the simplest case of primitive fitting is the detection of planes in a point cloud [7], [11], [12], [13], [10], [**?**]. Schnabel et al. [14] present a general method for efficient RANSAC-based detection of basic primitives like planes, cylinders, spheres, etc. within point clouds.

Contour-based Approaches. A number of works simplify the detection of walls as a 2D problem by projecting all points onto a plane and a subsequent Hough transform on the point location histogram is used to identify walls [15], [16], [17], [9], [18].

Beyond simple primitive fitting many works focusing on urban scenes restrict the problem search space by assuming special relations between primitives, in particular, many works follow the so-called **Manhattan-World assumption** [19], [11], [10] in which all planes are assumed to be oriented either parallel or orthogonal among each other. Straub et al. [20] slightly relax the Manhattan world assumption by looking for a mixture of Manhattan frames within a probabilistic approach. A recent and similar work to ours is by Li et al. [10] which find a tesselation of the input point cloud into data-aligned cuboids which are subsequently labeled as occupied or empty.

Higher Level Approaches. Several works go beyond the reconstruction of surfaces and their compact representation but also reason about higher level primitive dependencies and semantics, for instance the neighborhood relationship between walls, ceilings and floors, as well as the segmentation or detection of rooms. Monszpart et al. [13] compute a coupled set of planar primitives and a corresponding primitive relation graph from the previously detected planes.

Room segmentation in [17], [12], [9] is carried out by using the geometry and some kind of visibility information of the environment. Beyond the detection of rooms, the works in [21], [12], [22] detect other features such as doors and windows. Kim et al. [23] focus on indoor scenes and in particular on the shape learning and reconstruction of repetitive and frequently occurring objects and furniture. Similarly, Shao et al. [24] aim to reconstruct furniture but further require user-interaction for a semantic labeling.

In sum, a large amount of the works mentioned above, e.g. [16], [11], [12], [17], [13], [18], [25], have rather long processing times from several minutes to many hours and are therefore not suited for mobile interactive processing. Our method aims at bridging the gap between obtaining a compact, accurate and simplified building representation with higher level semantic annotation and fast processing times to allow for mobile applications. Compared to the rather slow and sophisticated reconstruction methods we are trading method generality and a bit of accuracy for speed.

A. Contributions

We present an efficient and simple method for Manhattan-like indoor modeling. We propose a simple but effective scheme for room and door detection via the construction of a wall graph representing a cuboid scene tessellation and clustering to group them into rooms. Our approach is simple to implement, runs at interactive runtimes and compares well to state-of-the-art approaches with similar runtime.

III. Automatic Indoor Modelling

A. System Overview

Our system architecture is simple, straightforward and thus easy to implement. As depicted in Fig. 2 we consider four major processing steps:

1) **Input Filtering** (III-B) to obtain a homogeneous sampling and reduce the amount of data for faster processing.
2) **Wall Detection** (III-C) to identify major structuring elements in the point cloud data.
3) **Room Layout Detection** (III-D) to reason occupied and free space, as well as semantically group walls into different rooms and identify connections via doors.
4) **Model Generation** (III-E) to create a final semantically labeled building information model.

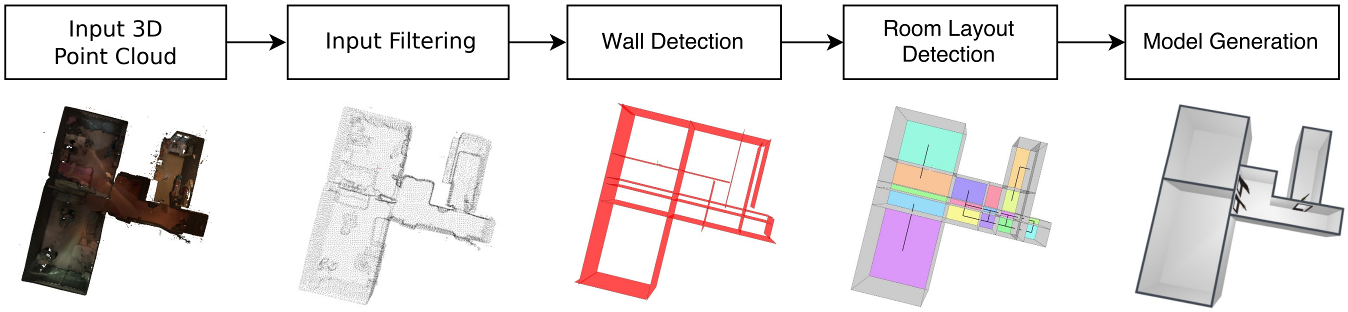


Fig. 2: System overview with the major processing steps: **1)** voxel grid filtering reduces the number of 3D points and homogenizes their density, **2)** wall detection via plane candidate selection **3)** room layout detection via reasoning about detected walls, and **4)** model generation.

The following sections will detail each of these processing steps separately.

B. Input Filtering

The system takes a 3D mesh of the indoor building environment as the input. The 3D mesh is assumed to be from an indoor scene that adheres to the Manhattan world assumption [5]. The 3D mesh is generated by running the 3D reconstruction pipeline on the sensor data from the Google project tango. Fig. 2 (first image) shows an example of the original input data. The units of the vertices in the 3D mesh are assumed to be in meters.

We apply a voxel grid-based filtering in order to obtain a point cloud with homogeneous density as well as to reduce the amount of input data, thereby improving total runtime. To this end, we fit a voxel grid to the input point cloud. Then, for each voxel we replace all input points within the voxel by their centroid. This approach is slower than approximating them with the voxel center, but it represents the underlying surface more accurately and avoids any bias towards the voxel grid.

The voxel grid merely helps to obtain a regular resampling of the input point cloud in which point densities can vary heavily. We choose a uniform voxel size of 0.05 meters (5 cm) edge length for all three dimensions. Fig. 2 (second image) shows an example output of the voxel grid filtering.

C. Wall Detection

Indoor environments are composed of planar surfaces such as the ceiling, floor and walls. The purpose of the this processing step is to detect these three surface types. Later on, these planar surfaces will be grouped together to form rooms in section III-D. The detection and labeling of walls can be divided into the following three sub-tasks and which are detailed subsequently: 1) Plane Detection, 2) Manhattan World Fitting, 3) Plane Labeling.

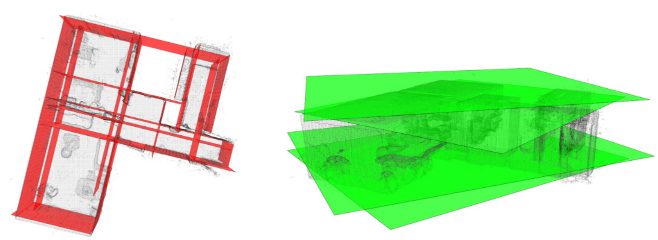
1) Plane Detection: Planes are detected using One-Point RANSAC Model fitting [26]. The hypothetical model of planes is computed using the surface normals and the position of random vertices in the mesh.

The inlier set is selected by adding points that adhere to two conditions. The point must have a surface normal that is parallel to the hypothetical plane normal and the distance to the plane is less than 20cm. The threshold value was selected based on the wall thickness encountered in various houses. The system is modeled to produce a single plane for a walls with thickness under 20cm.

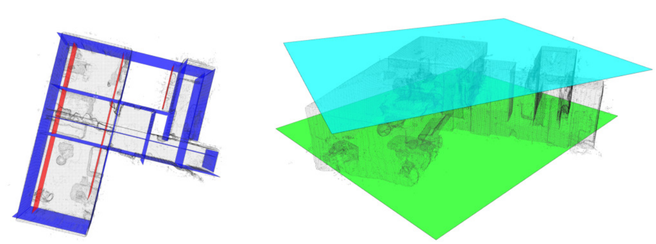
In order to increase the accuracy of the plane model, it is further refined by fitting a plane to the entire inlier set. The final model also includes a bounding box around the plane points. It is faster to work with plane boundaries and the plane equation compared to point based operations.

The principal components of the inlier set are found using PCA, with the first principal component used as the plane's normal. The centroid of the inlier set is used to find the offset of the plane from the origin.

Fig. 3a displays a sample of vertical and horizontal planes that were detected during the plane detection step.



a) Detected Planes



b) Labeled Planes

Fig. 3: Wall detection steps: **a)** RANSAC-based plane detection yields vertical and horizontal planes; **b)** The plane labeling step divides the planes into Wall (blue), Clutter (red), Floor (green) and Ceiling (cyan).

2) Manhattan World Fitting: The Manhattan world assumption [5] enforces that planar surfaces are aligned with one of the three dimensional directions. Planes detected by RANSAC are rotated along their normals to orient the two axes of the plane with the up direction (i.e. the z-axis).

This is done because the environments in the datasets adhere to the Manhattan world assumption where all the planes are oriented in mutually orthogonally directions to each other. Hence, to make the processing easier, the planes detected are adjusted to have similar properties.

The orientation of the planes is updated to align with the Manhattan axes, i.e., the plane's axes are either parallel or perpendicular to the z-axis. The normals and the x-axis of the planes are set to either the given axes or perpendicular to the given axes.

3) Plane Labeling: The planes are labeled as either ceiling, floor, walls or clutter. In this step we further assume that the scene only consists of a single ground floor and ceiling. The point cloud is also assumed to be oriented correctly with respect to gravity (The Z axis is aligned with the up direction).

Horizontal planes with the highest point density are chosen as the candidates for ceiling and floor. Since, the point cloud is aligned with the gravity, the plane with the higher offset is deduced as the ceiling and the one with the lower offset to be the floor.

Scene height is calculated as the perpendicular distance between the floor and the ceiling planes.

Vertical planes can be divided into walls and clutter. In this step, the vertical planes are iterated through to sort them into the following types:

- **Walls:** Vertical planes with a height greater than **90%** of the scene height.
- **Clutter:** Every other plane in the scene.

The height threshold on the wall is implemented to differentiate vertical planes caused by walls and clutter (such as cupboards, tables, television, etc.). Fig. 3b depicts the planes with their classified label for the example dataset by different colors.

D. Room Layout Detection

From the previous step, the walls, ceiling and floor have been segmented from the clutter. In the subsequent room detection step the walls are clustered together in order to form rooms.

Most modern indoor environments can be approximated by a union of axis-aligned cuboids. Following this idea, we check the detected planes for intersections and build a graph of connected walls. We then find cycles in this wall graph to construct cuboids. Afterwards, these cuboids are clustered together to form rooms.

To aid in finding cuboids in rooms, the walls are intersected with each other and split at intersections. Only walls that are perpendicular to each other are intersected. Fig. 4a shows an example of an output after intersections.

1) Wall Graph: The graph is represented an array of adjacency lists, i.e., the list of connected nodes for each node (each node is a wall in this case). Edges between two walls is added when two walls are connected according to the following conditions:

- The walls must be physically connected by an edge.
- The walls must be perpendicular to each other.

The graph is built by iterating through the list of walls to find the walls connected to each of them. Fig. 4b shows an example of such a graph.

2) Cuboid Detection: Cuboids are detected by finding cycles of four continuous walls in the wall graph. A part of the graph is considered a cycle if the following conditions are met:

- Each of the four walls are connected to two other walls in the cycle.
- All the walls do not intersect at the same point.

Cycle detection The algorithm to detect cycles in an undirected graph is based on the depth first search algorithm. This algorithm is exponential in the number of edges in the graph if cycles of all lengths are required to be found. However, the detection of cycles with a specific size can be solved in polynomial time.

The algorithm works by searching for cycles that start from each wall node. If a cycle is found it is added to the list of cycles taking into consideration to not add duplicate cycles. Fig. 4c shows the cuboids found using the plane graph on the sample dataset.

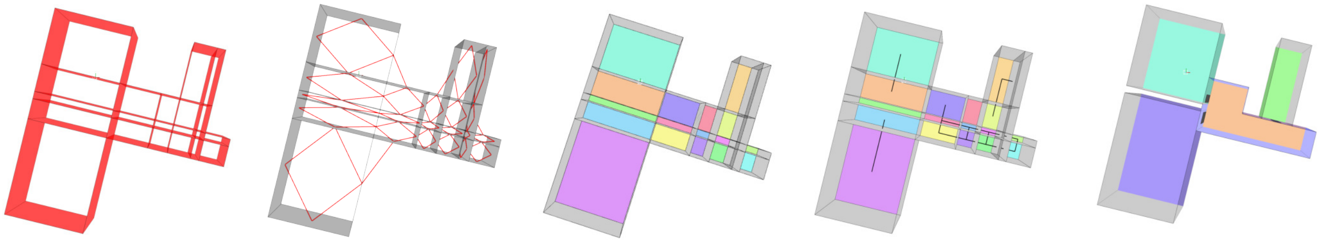
3) Cuboid Connection Classification: The rooms are modeled to be made up of cuboids, hence, the connection between adjacent cuboids needs to classified. The connection can be classified by analyzing the points that are on the plane of the common side of adjacent cuboids. The cuboid connection classification was inspired by the room connection classification in [12].

The cuboids from adjacent rooms would have a solid wall as the common side unless there is a door in the wall. Whereas, the cuboids from the same room would have empty space in the common side.

The points that are on the common side of two cuboids are projected onto its plane and a 2D occupancy map across the area of the plane is generated. All patches of empty space (i.e. patches that do not have points) are found using connected components labelling algorithm [27]. A rectangle which fits within area covered by the largest patch is found.

We consider three possible types of cuboid connections: disconnected, door and merge. Representative cases of these connection types are sketched in Fig. 5. By analyzing the proportional size of the bounding rectangle enclosing any empty space in the wall, we classify the connection type in the following way.

- **Disconnected:** The connection between two cuboids is classified as disconnected if the area covered by the rectangle is less than 10% of the common side. Fig. 5a shows such an example. Such small amounts of empty space could have been generated due to noise or incomplete scans.
- **Door:** Doors can be detected if the rectangle enclosing the empty space can be fit to one of the door models as in Fig 5b. The model uses the average dimensions of doors as seen in modern apartments, which are 2m high and 0.8m wide. The position and size of the rectangle are matched to the models. If the two neighboring



a) Detected walls. **b)** Wall graph. **c)** Detected cuboids. **d)** Cuboids clusters. **e)** Rooms.

Fig. 4: Room layout detection steps: **a)** walls detected in previous phase are intersected; **b)** the wall graph built by iterating through the list of walls in order to analyze wall connectivities; **c)** connected walls are clustered together to form cuboids; **d)** the cuboids are clustered based on their connection type; **e)** cuboids are merged to form rooms.

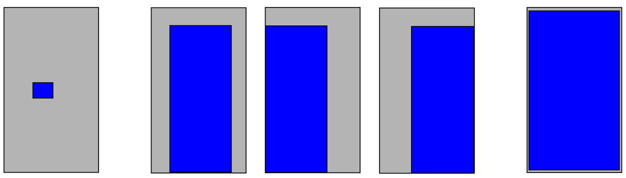


Fig. 5: Visualization of the wall connection type classification. Detected cuboids are either merged, disconnected or connected via doors and thereby grouped into rooms by classifying the wall connection type into: disconnected, door and merge. The figure shows typical examples of wall types which are classified based on the size of the rectangle covering empty space.

cuboids have matching door positions, the connection between the two cuboids is classified as door.

- **Merge:** If a rectangle does not fit to neither the door model nor the disconnected type, the connection between the two cuboids gets classified as merge.

4) Clustering: The final step of the Room Layout detection is clustering the cuboids together into rooms. The information of the connection type found in the previous step is used to cluster the cuboids.

Clustering of cuboids into rooms is done by finding pairs of neighboring cuboids having a connection type of merge. Fig. 4d gives an example of the cuboid graph. The edges between cuboids is represented as black lines.

Each cluster of cuboids is merged together to form rooms. When the cuboids are merged, the common side between cuboids are removed and other sides are added to the list of walls for each room. The list of walls are ordered in a way that each wall is adjacent to the wall next to it. If the adjacent walls in the list are coplanar, they are merged to form a single wall. Fig. 4e shows an example output when the cuboid clusters are merged to form rooms.

E. Model Generation

We use the information from all previous steps to build a semantically labeled building information model which can be used for further processing. In our experiments we exported the generated indoor models to the format read by Planner5D [28], a home interior design application that enables the user to design and view 3D models of buildings. Exporting to an home designing application enables the user to correct small errors and add for instance new furniture to the 3D model. Fig. 6 depicts the output of our approach next to a user-edited building model as viewed in the designer.

IV. Results

A. Qualitative Results

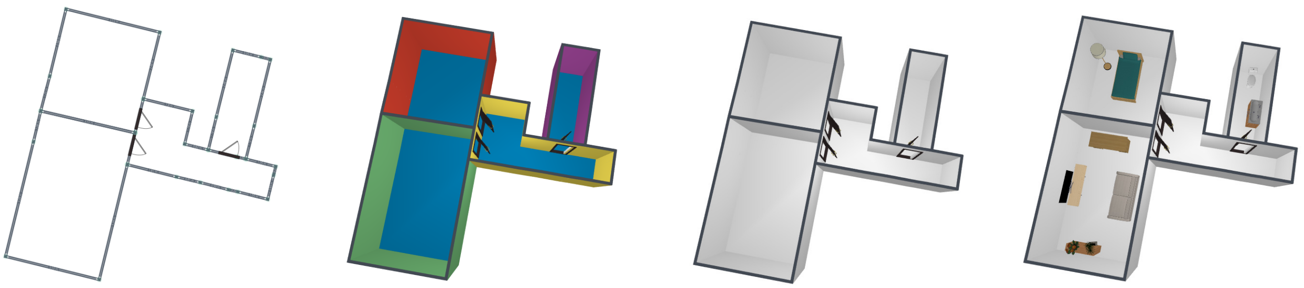
Our system has been tested on five different real world datasets. These datasets were acquired with a Google Tango tablet using a mesh resolution of 0.03 meters.

The five datasets **`Apt1`**, **`Apt2`**, **`Apt3`**, **`Apt4`** and **`Office1`** are 3D meshes of indoor environments with varying levels of clutter and noise. The datasets **`Apt1`**, **`Apt2`**, **`Apt3`** and **`Apt4`** are used to test the system on everyday clutter that is typically found in apartments, such as the furniture and electronic appliances. **`Apt1`** consists of four rooms and one corridor. **`Apt2`** consists of two rooms and a short corridor. **`Apt3`** is made up of three rooms and a corridor. **`Apt4`** consists of four rooms and a small corridor. The final dataset **`Office1`** is acquired in an office environment. The office environment is made up of rooms that have large glass windows and glass walls with doors which cause holes in the mesh. This dataset is made up of three rooms and a small corridor.

Fig. 7 shows the results for all 5 datasets. The building models of datasets **`Apt1`**, **`Apt2`** and **`Apt3`** are successfully generated. Dataset **`Apt4`** poses a problem as one of the rooms in the dataset contains a large hole created due to the presence of a mirror which causes a wall to be not detected. Nevertheless, the systems performs well to generate a model for the other rooms.

The office environment in dataset **`Office1`** contains a high ceiling and large windows and glass walls that can confuse the cuboid connection type classification. Thought the system is able retrieve the model of the dataset correctly. The door in one of the rooms is not detected properly.

It should be noted that the shape of the rooms and position of doors in the datasets have been recognized correctly and



2D Floor Plan | Semantic Labels | 3D BIM | Edited BIM with new furniture

Fig. 6: System output and BIM data export. The output of our method be used to generate 2D floor plans, label semantics and 3D models. The generated building information model can be exported to Planner5D [28] which enables the user to view or edit the model, and to design the interior appearance of the apartment by placing furniture and fixtures.

the floor plans generated by the system match well to the ground truth obtained by manual measurements.

Dataset	`Apt1`	`Apt2`	`Apt3`	`Apt4`	`Office1`
Mean Error [meters]	0.06	0.02	0.05	0.65	0.08
Wall Detection [sec]	0.61	0.29	0.49	0.64	0.61
Layout Detection [sec]	0.07	0.01	0.03	0.07	0.07

TABLE I: **Quantitative results and runtimes**. First row gives the average error in the dimensions of the walls of the indoor environments with respect to the ground truth. The following two rows give the execution times of the Wall Detection and the Room Layout Detection phases.

B. *Quantitative Results*

The performance of the system is measured quantitatively in terms of the total execution time and the mean error in the dimensions of the model generated by the system. Table I shows the summary of the quantitative results for each of the datasets.

Runtimes. The runtimes in Tables I, II correspond to a single-threaded and unoptimized C++ implementation of our method. All experiments were conducted on an Intel Core i7-4770 CPU @ 3.40 GHz machine with 16 GB RAM running a recent Linux distribution. The experiments demonstrated that our method is able to generate building information models for all tested datasets in under one second. It is worth noting that many steps in our method like for example the input filtering, or the wall detection steps can be easily parallelized for which we expect significant speed-ups of our method. Hence our approach is suitable for interactive processing on mobile devices.

Mean Error. The mean error in the dimensions of the models generated are computed by comparing the values with the ground truth data. The error was calculated as the absolute difference between the model and the ground truth. The ground truth data was acquired by measuring the width, length and height of the rooms at multiple locations with multiple measurements which were then averaged to minimize measurements errors. Overall experiments our system was able to get a mean error of less than 0.1 meters on average.

C. *Comparison with other methods*

In the Fig. 8 we visually compare the results from our method against [10], which is the closest related work in terms of accuracy and execution time (see Table II). From the *Top View* the result looks very similar, but from *Side View* we can observe that there are multiple internal planes that have not been removed. This might be because the method was mainly designed to deal with building exterior structures rather than the indoor environments.

Another big difference with respect to [10] is that our method can distinguish between rooms. In this regard, our method is similar to [12] in that capability. On the other hand, in contrast to state-of-the-art methods like [17], [12], [25], [13], the proposed method attempts to solve the problem in a simpler and efficient way; of course, with the risk of loosing details mainly due to the strong assumptions imposed to the environment (Manhattan-World assumption).

Dataset / Time [sec]	`Apt1`	`Apt2`	`Apt3`	`Apt4`	`Office1`
Ours	**0.68**	**0.30**	**0.52**	**0.73**	**0.68**
Li *et al.* [10]	0.93	0.49	0.96	1.39	0.70
Monszpart *et al.* [13]	> 60.00	> 60.00	> 60.00	> 60.00	> 60.00

TABLE II: **Runtime comparison.** Our method is consistently faster than the approach by Li *et al.* [10]. While their method often provides an over-segmentation of the scene, our approach recognizes the room layout and doors much more robustly and reliably (see Fig. 8). The approach by Monszpart *et al.* [13] requires significantly more computation time and does not yield superior results.

V. Limitations and Future Work

The current implementation of our system has a few limitations due to our assumptions regarding the indoor environments. We assume that the walls adhere to the Manhattan assumption which means they are aligned to the three dominant axes in the scene. While dropping the Manhattan assumption would make the plane detection step slower, it would allow to model a larger variety of scenes. In particular, the system could accommodate for non-cuboidal shapes of rooms. Furthermore, our method currently does not support varying floor levels or multi-floor buildings.

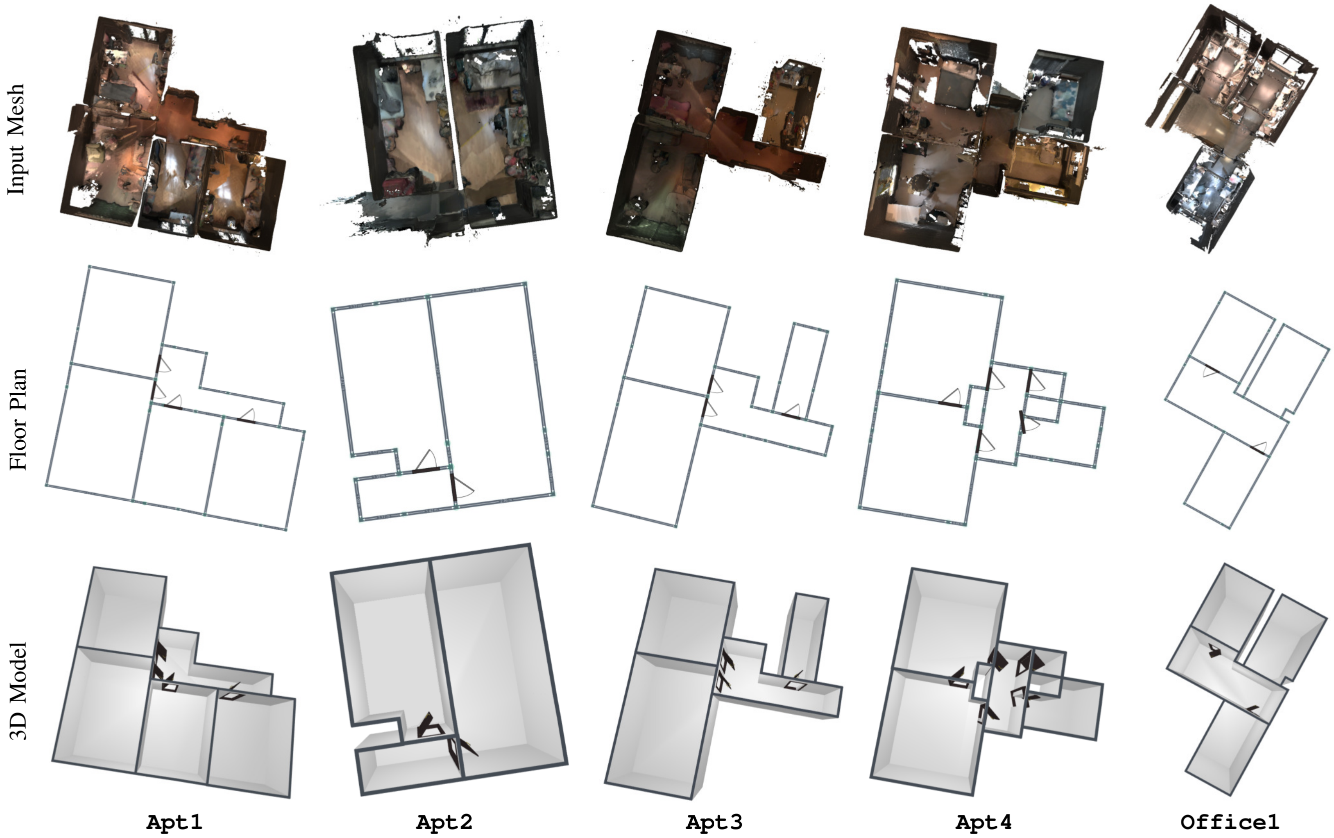


Fig. 7: Qualitative results of our approach on five indoor datasets. Although the data is misses scanned parts and the rooms contains a lots of furniture, our method reliably computes the room layouts a variety of scenes. Note that the top-right room in **`Apt4`** has not been recognized properly, which considerably effects the mean error in Table I. Nevertheless, this can be easily corrected by the user since the model is exported to an interior design software format.

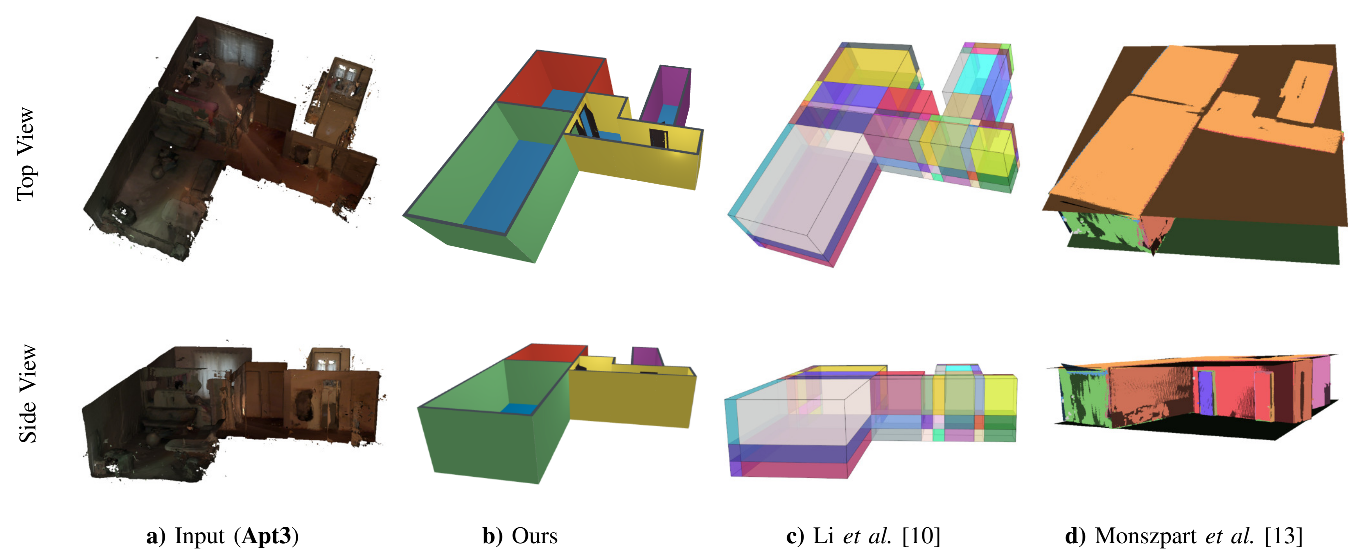


Fig. 8: **Comparison to [10] and [13]:** Results from related work. From *Top View* we can observe their results are similar to Ours, but from the *Side View* undesired internal planes are also visible. For [13], the results were obtained using the following parameters: $scale = 0.2$, $angle\_limit = 20$, $pairwise = 0.01$, and $pop\_limit = 100$. For [10], we used the default settings.

VI. Conclusion

We presented a simple and efficient approach for the creation of a 3D building information model from a given 3D scan. This enables the user to quickly generate compact models of indoor environments using existing depth sensor based scanning technologies such as the Google Tango tablet or the Microsoft Kinect.

Multiple experiments demonstrated the usefulness, accuracy and efficiency of our approach. In future work we will investigate how much a drop of the Manhattan assumption in the plane detection can be compensated by leveraging tree-like data structures and parallelization.

We consider our work as a progress towards a solution that can be employed by non-expert users on a mobile device or by a robot in the context of navigation, in order to build a model with sufficient accuracy – for the application at hand – in less time. Therefore, the focus of this method is to provide a quick solution obtained in less than a second (Table II), instead of minutes or hours required by more sophisticated methods like [17], [12], [25], [10], [13], where higher precision and generality is prioritized over practicality.

Acknowledgements

This research received funding from the European Union Horizon 2020 research and innovation programme (grant No. 637221). We further thank Yasutaka Furukawa [12], Minglei Li and Liangliang Nan [10], and Aron Monszpart [13] for providing their software and their assistance for generating comparative results on our datasets.

References

[1] S. Lynen, T. Sattler, M. Bosse, J. Hesch, M. Pollefeys, and R. Siegwart, "Get out of my lab: Large-scale, real-time visual-inertial localization," in *Proceedings of Robotics: Science and Systems*, Rome, Italy, July 2015.

[2] Google Project Tango, https://get.google.com/tango/, [Online, accessed 2017-02-22].

[3] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, Oct 2011, pp. 127–136.

[4] Microsoft HoloLens, https://www.microsoft.com/microsoft-hololens/, [Online, accessed 2017-02-22].

[5] J. M. Coughlan and A. L. Yuille, "Manhattan world: Orientation and outlier detection by bayesian inference," *Neural Computation*, vol. 15, no. 5, pp. 1063–1088, 2003.

[6] Matterport, "Capturing device and software," https://matterport.com/, [Online, accessed 2017-02-22].

[7] S. Thrun, C. Martin, Y. Liu, D. Hahnel, R. Emery-Montemerlo, D. Chakrabarti, and W. Burgard, "A real-time expectation-maximization algorithm for acquiring multiplanar maps of indoor environments with mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 3, pp. 433–443, 2004.

[8] M. Dzitsiuk, J. Sturm, R. Maier, L. Ma, and D. Cremers, "De-noising, stabilizing and completing 3D reconstructions on-the-go using plane priors," in *arXiv:1609.08267*, September 2016.

[9] E. Turner and A. Zakhor, "Multistory floor plan generation and room labeling of building interiors from laser range data," in *Computer Vision, Imaging and Computer Graphics-Theory and Applications*. Springer, 2014, pp. 29–44.

[10] M. Li, P. Wonka, and L. Nan, "Manhattan-world urban reconstruction from point clouds," in *ECCV*, 2016.

[11] V. Sanchez and A. Zakhor, "Planar 3d modeling of building interiors from point cloud data," in *ICIP*. IEEE, 2012, pp. 1777–1780.

[12] S. Ikehata, H. Yang, and Y. Furukawa, "Structured indoor modeling," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1323–1331.

[13] A. Monszpart, N. Mellado, G. Brostow, and N. Mitra, "RAPter: Rebuilding man-made scenes with regular arrangements of planes," *ACM SIGGRAPH 2015*, 2015.

[14] R. Schnabel, R. Wahl, and R. Klein, "Efficient ransac for point-cloud shape detection," *Computer Graphics Forum*, vol. 26, no. 2, pp. 214–226, June 2007.

[15] B. Okorn, X. Xiong, B. Akinci, and D. Huber, "Toward automated modeling of floor plans," in *Proceedings of the Symposium on 3D Data Processing, Visualization and Transmission*, vol. 2, 2010.

[16] A. Adan and D. Huber, "3d reconstruction of interior wall surfaces under occlusion and clutter," in *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2011 International Conference on*. IEEE, 2011, pp. 275–281.

[17] C. Mura, O. Mattausch, A. J. Villanueva, E. Gobbetti, and R. Pajarola, "Automatic room detection and reconstruction in cluttered indoor environments with complex room layouts," *Computers & Graphics*, vol. 44, pp. 20–32, 2014.

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

[19] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, "Manhattan-world stereo," in *CVPR*. IEEE Computer Society, 2009, pp. 1422–1429.

[20] J. Straub, G. Rosman, O. Freifeld, J. J. Leonard, and J. W. Fisher III, "A mixture of manhattan frames: Beyond the manhattan world," in *CVPR*, 2014.

[21] S. Ochmann, R. Vock, R. Wessel, and R. Klein, "Automatic reconstruction of parametric building models from indoor point clouds," *Computers & Graphics*, vol. 54, pp. 94–103, 2016.

[22] H. Macher, T. Landes, and P. Grussenmeyer, "Point clouds segmentation as base for as-built bim creation," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 2, no. 5, p. 191, 2015.

[23] Y. M. Kim, N. J. Mitra, D. Yan, and L. J. Guibas, "Acquiring 3d indoor environments with variability and repetition," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 138:1–138:11, 2012.

[24] T. Shao, W. Xu, K. Zhou, J. Wang, D. Li, and B. Guo, "An interactive approach to semantic modeling of indoor scenes with an RGBD camera," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 136:1–136:11, 2012.

[25] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese, "3d semantic parsing of large-scale indoor spaces," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*.

[26] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[27] K. Suzuki, I. Horiba, and N. Sugie, "Linear-time connected-component labeling based on sequential local operations," *Computer Vision and Image Understanding*, vol. 89, no. 1, pp. 1–23, 2003.

[28] Planner 5D, http://www.planner5d.com, [Online, accessed 2017-01-16].