Automatic Digitization and Orientation of Scanned Mesh Data for Floor Plan and 3D Model Generation

Ritesh Sharma[1*] [0000−0003−1160−3918], Eric Bier[2] [0009−0004−2265−386X], Lester Nelson[2] [0000−0001−8556−7644], Mahabir Bhandari[3] [0000−0003−1951−9876], and Niraj Kunwar[3] [0000−0002−6345−7652]

[1] University Of California, Merced, USA rsharma39@ucmerced.edu
[2] Palo Alto Research Center, USA {bier,lnelson}@parc.com
[3] Oak Ridge National Laboratory, USA {bhandarims,kunwarn1}@ornl.gov

Abstract. This paper proposes a novel approach for automatically generating accurate floor plans and 3D models of building interiors using scanned mesh data. Unlike previous methods, which begin with a high resolution point cloud from a laser range-finder, our approach begins with triangle mesh data, such as from a Microsoft HoloLens headset. The approach includes generating two types of floor plans, a "pen-and-ink" style that preserves details and a drafting-style that reduces clutter, and processing the 3D model for use in applications by aligning it with coordinate axes, annotating important objects, dividing it into stories, and removing the ceiling. The performance of each step is evaluated on commercial and residential buildings, and experiments are conducted to assess quality and dimensional accuracy. Our approach demonstrates promising potential for automatic digitization and orientation of scanned mesh data, enabling floor plan and 3D model generation in various applications such as navigation, interior design, furniture placement, facilities management, building construction, and HVAC design.

Keywords: Clustering based methods · Floor plans · Augmented Reality· 3D Models.

1 Introduction

Floor plans are useful for many applications including navigating in building interiors; remodeling; efficient placement of furniture; placement of pipes, heating, ventilation, and air conditioning (HVAC) design, and preparing an evacuation plan for emergencies. Depending on the application, different kinds of floor plan are appropriate. For remodeling building interiors or designing HVAC systems, users may prefer a drafting-style floor plan that focuses on planar walls and removes furniture and other clutter. For furniture placement, navigation, or evacuation planning, users may prefer a more detailed floor plan that shows the positions of furniture, cabinets, counter tops, and similar features. In either case, producing a floor plan can be a time-consuming process, requiring expert skills, such as measuring distances and angles or entering data into a CAD program. Furthermore, this process may need to be done more than once because a building changes when walls and furniture are moved, added, or removed. So it is valuable to be able to generate floor plans automatically with little or no training.

In order to generate floor plans automatically, it helps to begin with accurate data that can be collected automatically. Laser range finders, smartphones, tablets, and augmented reality headsets are some of the devices that have made it easier to collect high-resolution building data in the form of RGBD images, point clouds, and triangle meshes. In this paper, we propose a method for generating drafting-style and pen-and-ink-style floor plans by leveraging incomplete or not so well captured triangle mesh data. This approach offers an efficient way to generate both types of floor plans accurately, making it a valuable tool for a wide range of applications.

Main Contribution We describe a new method for generating accurate floor plans using poorly captured triangle mesh data from Microsoft HoloLens 2. The main contributions are:

- A modified Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm, using blocks to capture wall height and thickness.
- An orientation-based clustering method for finding walls facing any arbitrary orientation.
- The use of k-means clustering to rotate the mesh to the principal axes and to identify the floor and ceiling.

This approach generates two kinds of precise floor plans from incomplete mesh data.

* Worked on this research work during his internship at Palo Alto Research Center.

2 Related Work

Floor plans are crucial for various applications, such as navigation, furniture placement, remodeling, sensor placement, and HVAC design. Software approaches to floor plan creation depend on data availability and data format. Our work builds on previous research on data collection and floor plan computation.

Data collection. Indoor environments can be captured in several formats, including RGBD images, point clouds, and triangle meshes. Zhang et al. [26] uses panoramic RGBD images as input and reconstructs geometry using structure grammars, while [19] uses a 3D scan to extract plane primitives and generates models using heuristics. A single image is used in some deep learning methods [7, 10, 12, 13, 27] to generate cuboid-based layouts for a single room. Detailed semi-constrained floor plan computations for a complete house require processing a 3D scan of the house [14]. While a complete scan increases accuracy, it also increases computing requirements and time. In our approach, we begin with a triangle mesh generated by a HoloLens 2, using its Spatial Mapping software [16], which has been extensively surveyed by Weinmann et al. [24].

Floor plan computations Early methods [1, 3, 20, 25] relied heavily on image processing techniques, such as histograms and plane fitting, to create floor plans from 3D data. While [20] creates a floor plan by detecting vertical planes in a 3D point cloud, [3] uses planar structure extraction to create floor plans. These techniques rely heavily on heuristics and were prone to failure due to noise in the data.

There has been significant progress in floor plan computation using graphical models [4, 8, 11]. Such models [9] have also been used in recovering layouts and floor plans from crowd-sourced images and location data.

Pintore et al. [21] characterizes several available input sources (including the triangle meshes that we use) and output models and discusses the main elements of the reconstruction pipeline. In addition, it identifies several systems for producing floor plans, including FloorNet [14], and Floor-SP [6].

Monszpart et al. [17] introduced an algorithm that exploits the observation that distant walls are generally parallel to identify dominant wall directions using k-means. Similarly, our approach also utilizes k-means to identify walls in all directions, not just the dominant ones.

Cai et al. [5] leverage geometric priors, including point density, indoor area recognition, and normal information, to reconstruct floorplans from raw point clouds of indoor scene.

In contrast to Arikan et al. [2], which employed a greedy algorithm to find plane normal directions and fit planes to points with help from user interaction, our approach works in an automatic manner. Our approach also differs from the work presented in [18], which focuses on removing clutter and partitioning the interior into a 3D cell complex. In contrast, our method specifically divides the building into separate walls.

Our work is closely related to the proposals in [20] and [23].In [20], the floor plan generation process starts with laser range data to create a point cloud, followed by floor and ceiling detection using a height histogram. The remaining points are then projected onto a 2D ground plane, where a density histogram and Hough transform are applied to generate 2D line segments forming the floor plan. In projecting to 2D, their method runs the risk of loosing valuable information that may be useful for creating 3D models or detailed floor plans. Similarly, [23] utilizes a histogram-based approach to detect ceilings and floors. Their method involves identifying taller wall segments to create a 2D histogram, and then employing heuristics based on point density within each histogram cell to compute the floor plan. Our approach differs from both [20] and [23] by aligning the mesh with global coordinate axes and not relying on laser data or a point cloud. Working primarily with 3D data throughout the pipeline, our method benefits from enhanced information and generates both a 3D model and a floor plan.

3 Methodology

Our approach for computing floor plans has four main steps (see Algorithm 1). First, a user captures the interior of the building using an augmented reality headset, resulting in a triangle mesh. The data is then oriented to align with primary axes, and the building is divided into stories. Floors and ceilings are removed, and flat walls are detected if desired. Finally, one of two floor plan styles is generated by slicing and projecting the 3D model produced in the previous steps. This section provides a detailed description of these high-level steps.

Data collection Indoor environments can be captured in various formats using different devices. We use a Microsoft HoloLens 2 headset to capture triangle mesh data, annotating the mesh using voice commands.

Capturing the triangle mesh The HoloLens provides both hardware and software to create a 3D representation of the indoor environment using triangles, as shown in Figure 1-left. The headset overlays the triangles on the user's view of the interior of the building. Although the headset captures most of the walls, floors, and ceilings, data may be missing from some regions, as seen in the figure.

Algorithm 1 Methodology

1: $\mathcal{M} \leftarrow CaptureData(\mathcal{E})$;
2: $\mathcal{PM} \leftarrow MeshOrientation\ (\mathcal{M})$;
3: $\mathcal{RM} \leftarrow RemoveCeilingsAndFloors\ (\mathcal{PM})$;
4: $\mathcal{FP} \leftarrow ComputeFloorPlan\ (\mathcal{RM})$;
5: **return** $\mathcal{FP}$.

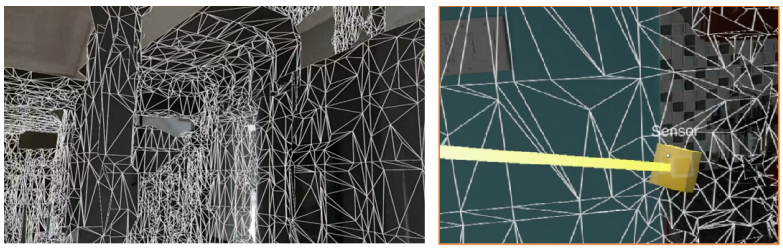

Fig. 1. Left: A triangle mesh is being computed by a HoloLens. Right: The yellow block marks a sensor position. The thick yellow line, known as the *probe*, shows the proposed position and orientation of a sensor or other annotation.

Annotating the mesh To capture object positions such as sensors, thermostats, windows, and doors, we developed an augmented reality user interface. This interface uses eye gaze detection and voice commands to enable users to place synthetic objects at desired locations, as demonstrated in Figure 1-right where a synthetic sensor object is added to the immersive environment and superimposed on a physical sensor.

3.1 Mesh orientation

After capturing the annotated triangle mesh, further processing is required. Initially, the mesh's orientation is based on the user's starting position and gaze direction. However, to generate an accurate floor plan, determining the floor's position and facing direction is necessary. While the augmented reality headset can provide a rough estimation of gravity direction, additional computation is required to improve its precision.

Orienting the floor To determine the orientation of the mesh for floor plan generation, we tested two methods: (1) compute the shortest edges of the mesh's bounding box, and (2) cluster the facing directions of mesh triangles using spherical k-means. While the bounding box method works for buildings with constant altitude and large horizontal area, it fails on other buildings. Thus, we mainly use the second method as described in Algorithm 2.

The spherical k-means method, although more complex than the bounding box method, is applicable to a broader range of meshes, including multi-story building models with vertical dominance. It utilizes the surface normal vector of each triangle Δ in the mesh $\mathcal{M}$ and filters out triangles deviating significantly from the positive y direction, preserving those most likely to represent the floor ($\Delta^{'}$).

We then use a spherical coordinates k-means algorithm with $k = 1$ to find the dominant direction g_m of this collection of triangles. Next, we discard triangles that are more than an angle ϕ from the dominant direction and repeat the k-means algorithm until ϕ_{min} is reached (e.g., starting with $\phi = 30$ degrees and ending with $\phi_{min} = 3$ degrees). This results in an estimate of the true gravity direction g_t.

To orient the mesh, we compute the angle θ between g_t and the negative y-axis and determine the rotation axis $\mathcal{Y}$ by taking their cross product. The mesh is then rotated by θ around the $\mathcal{Y}$ axis, ensuring a horizontal floor. Further details on the spherical k-means method for floor orientation can be found in Algorithm 2.

Figure 2 shows a model where the floor is not level, but tilts down from near to far and from right to left. After Algorithm 2, the floor is corrected to be horizontal.

Finding the height of the floor After orienting the mesh with a horizontal floor, the next step is to determine the altitude of the floor in the y direction. To accomplish this, we calculate the centroid of each mesh triangle whose facing direction is within a small angle of the positive y axis. We consider the y coordinate of each centroid and create a histogram of all these y coordinates, with each bucket representing a small vertical range, such as 2 inches. We consider adjacent pairs of buckets and look for the pair with the highest number of points, such as

Algorithm 2 Orienting the floor using the Spherical k-means method

for each $\Delta \in \mathcal{M}$ **do**
  if N of Δ facing into the room **then**
    $\Delta^{'} \leftarrow \Delta$;
while $\phi < \phi_{min}$ **do**
  $\phi, g_m \leftarrow SphericalKmeans(\Delta^{'}, 1)$;
  for each N *of* $\Delta \in \mathcal{M}$ **do**
    if $angle(N) < \phi$ **then**
      $\Delta^{'} \leftarrow \Delta$;
$\boldsymbol{g_t} \leftarrow$ -$\boldsymbol{y}$;
$\theta \leftarrow \arccos(\boldsymbol{g_t} \cdot \boldsymbol{g_m})$;
$\mathcal{Y} \leftarrow \boldsymbol{g_t} \times \boldsymbol{g_m}$;
$\mathcal{M} \leftarrow rotateMesh(\theta, \mathcal{Y})$;

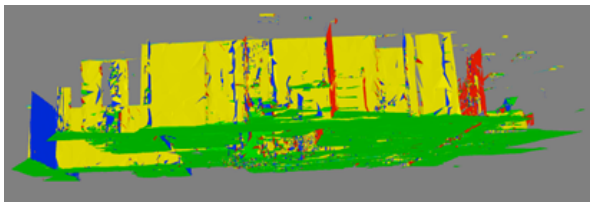

Fig. 2. A model where the floor is not level initially

(0, 1), (1, 2), or (2, 3). For a single-story building, we search for two large bucket pairs representing the floor and the ceiling, one near the bottom and one near the top.

If the building has sunken floors or raised ceilings, the histogram will show spikes at similar but not identical altitudes. To ensure that we locate the true ceilings and floors, we search for a gap of several feet (such as the expected floor-to-ceiling height of a room) between the low and high histogram spikes. The spikes below this gap are probably floors, and those above the gap are probably ceilings.

To generate the floor plan, we can choose the highest of the floor levels and the lowest of the ceiling levels as the computed floor and ceiling levels, respectively. Pairing the buckets rather than considering them individually ensures that we do not overlook spikes in the histogram if the mesh triangles are distributed evenly across two adjacent buckets.

Rotate mesh and associated annotations Our next goal is to align the mesh model's primary wall directions with the axes of euclidean coordinates.

One optional step is to eliminate any mesh triangles whose surface normals are with in some degrees apart from the positive or negative y directions, as these are probably ceiling or floor triangles. This step is not mandatory, but it can speed up processing by decreasing the number of triangles that require processing. Additionally, we can eliminate all triangles below the computed floor altitude and all triangles above the computed ceiling altitude.

We can then examine the surface normals of the remaining triangles. We express each normal in spherical coordinates and use k-means clustering in spherical coordinates to identify the dominant wall directions. Assuming the building has mainly perpendicular walls, there will be four primary wall directions, so we can set $k = 4$ for k-means clustering. If the model still has floor and ceiling triangles, we can set $k = 6$ to account for the two additional primary directions. Figure 3-left illustrates a heat map of surface normal directions in spherical coordinates produced for an office building.

Figure 3-left contains many light blue rectangles that are far from any cluster center (e.g., far from the buckets that are red, orange, and white). These represent triangles whose facing directions do not line up with any of the primary walls, floors, or ceilings. This is true for two reasons: (1) Building interiors contain many objects that are not walls, floors, or ceilings, such as furniture, documents, office equipment, artwork, etc. These objects may be placed at many angles. (2) The augmented reality headset generates triangles that bridge across multiple surfaces (e.g., that touch multiple walls) and hence point in an intermediate direction. To compensate, we use a

modified version of spherical coordinates k-means clustering that ignores triangle directions that are outliers. We modify k-means as follows:

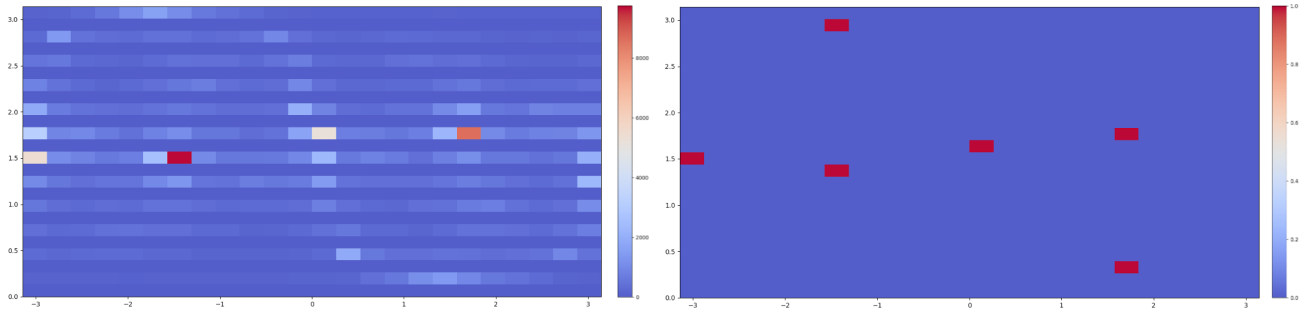

Fig. 3. Left: A heat map of triangle facing directions in spherical coordinates. The horizontal axis is the angle θ around the y-axis. The vertical axis is the angle ϕ from the south pole to the north pole. Warmer colors indicate more triangles per direction bucket. Right: The six cluster centers in spherical coordinates show the directions of the ceiling (top red dot), the floor (bottom red dot) and the four primary wall directions (four dots running left to right at a medium height).

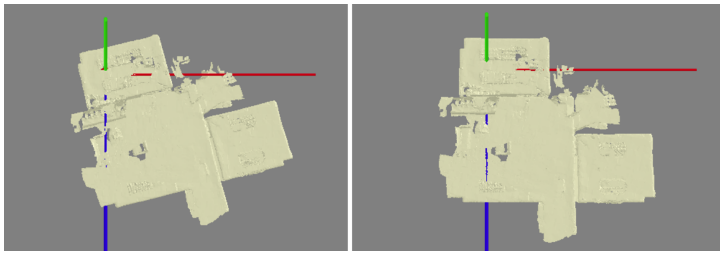

Fig. 4. The mesh model of building B1 (left) that is not aligned initially and the same after wall alignment (right).

After computing spherical k-means in the usual way, we look for all triangles in each cluster whose facing direction is more than a threshold θ_1 from the cluster center. We discard all such triangles. Then we run k-means again, computing updated cluster centers. Then we discard all triangles that are more than θ_2 from each cluster center where $\theta_2 < \theta_1$. We repeat this process several times until we achieve the desired accuracy. For example, in our current implementation, we use this sequence of angles θ_i in degrees: [50, 40, 30, 20, 10, 5, 3]. Once our modified spherical coordinates k-means algorithm completes, we have 4 (or 6) cluster centers. Figure 3-right shows sample results for the $k = 6$ case.

Once the primary wall directions are computed, we pick the cluster with the largest number of triangles, take its direction (cluster center), project that direction onto the $x - z$ plane, and call it θ_{wall}. We rotate the mesh by the angle between θ_{wall} and the x axis. After this operation, the primary walls will be pointing along the x axis. Figure 4-left shows a building that is not aligned with the coordinate axes. Figure 4-right shows the same building after wall rotation. By adding in the x axis (in red) and z axis (in blue), we see that the walls are now well-aligned with the coordinate axes.

Dividing a mesh into separate levels The HoloLens can digitize multi-story buildings. Given a multi-story model, it can compute a floor plan for each story. The process is similar to the one used in 3.1 to find the height of the floor. First, our system splits the model into stories: it computes a histogram, as shown in figure 5 and segments the building into multiple levels, as shown in figure 5-middle and -right.

3.2 Floor plan computation

Our floor plan computation depends on the type of floor plan desired and whether the mesh is oriented with respect to the global axes. If we wish to create a pen-and-ink style floor plan and the mesh is already oriented, we can simply pass the oriented mesh M to the *ComputeAndSuperimposeSlices*() function, as indicated in line 14 of Algorithm 3.

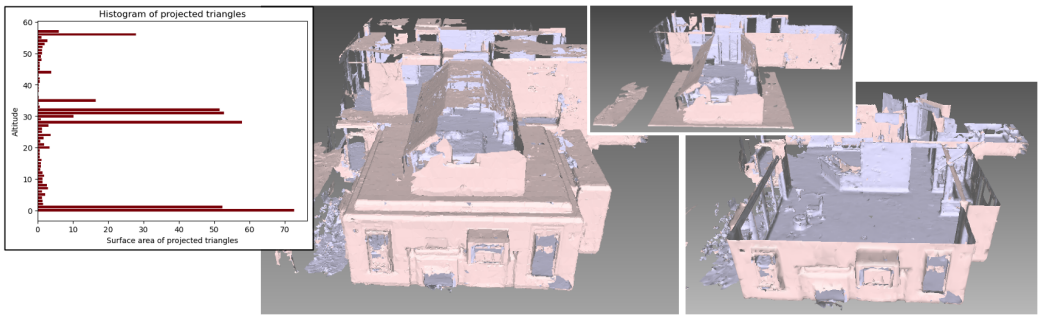

Fig. 5. Left: Using the triangles that face nearly straight up or straight down, find the centroid of each and plot a histogram of the centroid altitudes, weighted by triangle surface area. Histogram spikes are likely altitudes for floors and ceilings. Middle: A two-story building. Right: the two stories of that building.

Algorithm 3 Compute Floor Plan

1: $\Phi \leftarrow \emptyset$
2: **for each** $\Pi_i \in \Pi$ **do**
3: $\quad \Delta_i \leftarrow GetTrianglesFacingThisWay(\mathcal{M}, Pi_i)$
4: $\quad \mathcal{C}_i \leftarrow GetCentroids(\Delta_i)$
5: $\quad CL_i \leftarrow ModifiedDBSCAN(\mathcal{C}_i, l, w, h)$
6: $\quad WS \leftarrow \emptyset$
7: $\quad$ **for each** $CL_{ij} \in CL_i$ **do**
8: $\quad\quad$ **if** CL_{ij} *extends floor to ceiling* **then**
9: $\quad\quad\quad WS \leftarrow WS + CL_{ij}$
10: $\quad$ **for each** $WS_k \in WS$ **do**
11: $\quad\quad \mathcal{R} \leftarrow ComputeRectangles(WS_k)$
12: $\quad \Phi_i \leftarrow \Phi_i + \mathcal{R}$
13: $M' \leftarrow AssembleWalls(\Phi)$
14: $\mathcal{FP} \leftarrow ComputeAndSuperimposeSlices(M')$
15: **return** $\mathcal{FP}$

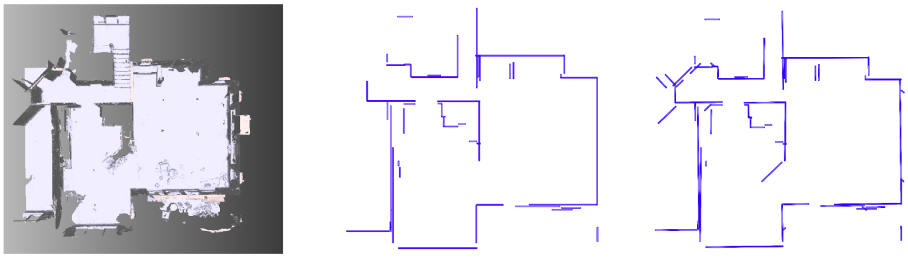

Fig. 6. Left: 3D model. Middle: floor plan obtained via spherical k-means with k=6. Right: floor plan from a modified DBSCAN that considers the arbitrary orientation of walls.

However, if the mesh is not properly oriented, we align it with the global axes before computing the floor plan. If a drafting-style floor plan is desired, we utilize lines 2-13 of Algorithm 3 to compute flat walls. Then we slice the mesh to derive the floor plan by making use of line 14 of Algorithm 3.

Computing flat walls To generate a drafting-style floor plan, we compute flat walls and separate them from other building contents using these steps:

Wall directions Once the floor and ceiling are removed, the remaining walls consist of triangles having surface normals parallel to the xz plane assuming $-y$ is the gravity direction. We can then use these surface normals to divide the triangles into k sets based on their facing directions Π_i, which we refer to as the *wall directions*, denoted as Π. For our initial experiment, we consider $k = 4$ for the principal directions (North, East,

South, and West). With these parameters, we are able to achieve precise floor plans in buildings where walls are along principal directions. However, our approach fails when the walls of the building are facing arbitrary directions. One such case is shown in figure 6. In such cases, we use spherical k-means to find all the wall directions and pass them to our DBSCAN algorithm for further processing.

DBSCAN For each wall direction, we perform a modified DBSCAN algorithm: We compute the centroid $\mathcal{C}$ of each triangle Δ_i. For each centroid point $\mathcal{C}_i$ during DBSCAN, we count the number of other centroid points that are near enough to be considered neighbors. However, instead of looking for neighbors in a sphere around each point as for traditional DBSCAN in 3D, we look for neighbors in a rectangular block of length l, width w and height h centered on the point. This block is tall enough to reach from floor to ceiling in the y direction, a little less wide than a door in the direction parallel to the proposed wall (e.g., 1.5 feet), and a few inches in the wall direction (to allow for walls that deviate slightly from being perfectly flat). Relying on the National Building Code, the wall's minimum height is set at 8 feet, with a thickness of 8 inches. After DBSCAN, the mesh triangles are grouped into wall segments WS.

Filtering We can now discard wall segments that are not good candidate walls. For example, we discard walls that are too small, that don't come near enough to the floor, and that don't come near enough to the ceiling.

Plane fitting For each wall segment, we find a plane that has the same facing direction as the wall direction and that is a good fit to the triangle centroids in that wall segment. Given that the points are tightly collected in this direction, simply having the plane go through any centroid works surprisingly well. However, it is also possible to choose a point more carefully, such as by finding a point at a median position in the wall direction.

Rectangle construction For each remaining wall segment, we construct rectangles $\mathcal{R}$ that lie in the fitted plane and are as wide as the wall segment triangles extend in width and as tall as the wall segment triangles extend in height.

Mesh replacement For purposes of floor plan construction, we can now discard the original mesh triangles and replace them with the new planar wall rectangles to serve as a de-cluttered mesh. If the subsequent steps are based on libraries that expect a triangle mesh, we can use two adjacent right triangles in place of each rectangle.

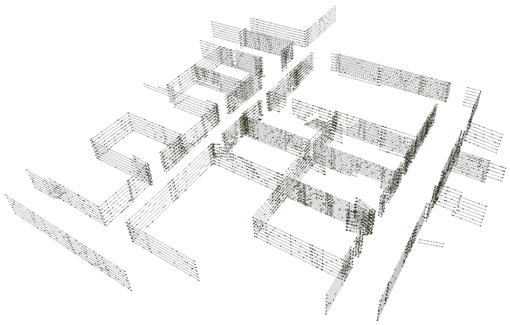

Fig. 7. A building mesh sliced at multiple altitudes.

Slicing the mesh to produce a floor plan If we wish to produce a pen-and-ink style floor plan, we take our mesh after it has been rotated to have a level floor and to have walls aligned with the primary axes, and we slice that mesh at multiple altitudes. As described above, our histogram of y values allows us to identify the height of the floor or floors and the height of the ceiling or ceilings. We pick a floor height, such as the highest floor. Call this floor height y_{floor}. We pick a ceiling height, such as the lowest ceiling. Call its height $y_{ceiling}$. Then we choose a series of y values between y_{floor} and $y_{ceiling}$, such as an even spacing of y values:

$$y_i = y_{floor} + (y_{ceiling} - y_{floor}) * (\frac{i}{n}) \tag{1}$$

for each i such that $0 \leq i \leq n$. For each such y_i, we compute the intersection of the mesh with the plane $y = y_i$. We end up with a stack of slices, as illustrated in Figure 7.

We can use the same method to produce a drafting-style floor plan. In this case, we begin with the flat wall model instead of the full mesh. A model with flat walls has fewer details to capture by slicing at multiple altitudes, so we may choose to slice at a single intermediate altitude to produce the drafting-style floor plan.

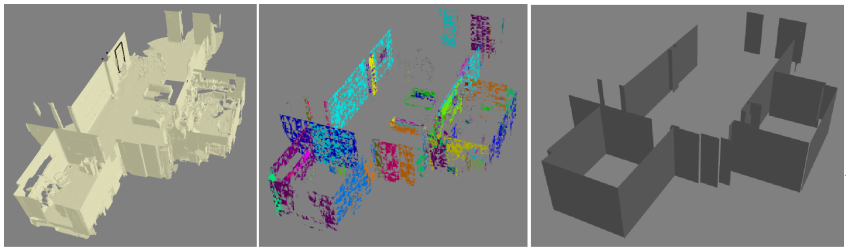

Fig. 8. Left: An oriented mesh from part of a building. Center: The results of DBSCAN. Each color represents a different wall segment. Right: The flat walls that result from plane fitting, rectangle construction, and mesh replacement.

Drawing the floor plan For either style of floor plan, we can produce a two-dimensional floor plan by ignoring the y coordinates of the resulting line segments and plotting the resulting (x, z) coordinates as a two-dimensional image. We have also found it informative and aesthetically pleasing to draw the lines from each slice in a color that is partially transparent, so that features that occur at multiple altitudes appear darker, while features that occur only at a single altitude are less dark.

As an example, Figure 8-left shows a mesh gathered from a commercial building. Figure 8-center shows the result of running our DBSCAN algorithm on that data. Figure 8-right shows the flat walls that result after mesh replacement. Figure 9-right shows the drafting-style floor plan that results from slicing the flat walls. Figure 9-left shows the pen-and-ink floor plan that results from slicing the oriented mesh at multiple altitudes.

Drawing synthetic objects Because our data comes from an augmented reality headset, we are able to add synthetic objects to mark the positions of objects in a room, such as sensors, thermostats, pillars, and windows. We can display these objects in our 3D models and floor plans. For the steps of mesh processing described above, we note the geometric transformations applied to the mesh and apply the same transformations to the synthetic objects, which then appear in the correct places in the 3D views and in the floor plans. For example, the black objects in Figure 8-left represents the objects placed by the user to show the positions of sensors and windows. Likewise, the red objects in the floor plan of Figure 9-left are those same objects, projected onto the same plane as the mesh slices.

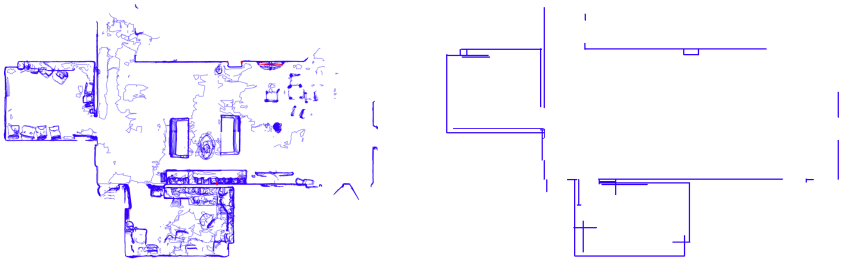

Fig. 9. Pen-and-ink style floor plan (left) and the drafting style floor plan obtained from the model shown in Fig. 8

4 Experiments

In this section, we present the results of our experiments and evaluations on capturing 3D scans of buildings using augmented reality headsets. We compared the floor plan dimensions with actual building dimensions using two different scans, and provide intermediate results such as floor plans and 3D models. We also calculate the time taken for each step in our algorithm. To demonstrate the robustness of our approach, we conducted experiments on various building scans, including commercial buildings B1 and B3, as well as a residential building B3. Additionally, we validated our floor plan generation algorithm using the Matterport2D dataset [22].

Scanned Data Analysis We evaluated the precision of our floor plan generation by comparing the dimensions of the mesh with the computed floor plan. Figure 10(a) illustrates the measurement of the building, which was scanned twice. We call these scans S1 and S2 (shown in figure 10(b) and figure 10(c)) as a case study for the usefulness of this approach. For each scan, floor plans are computed and then the dimensions are computed using

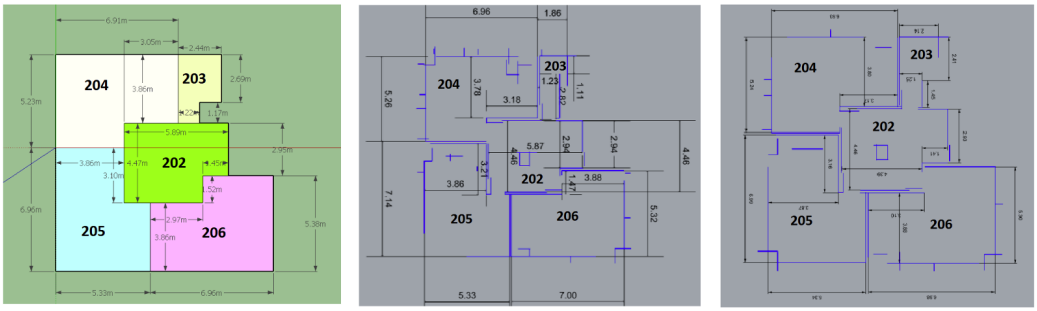

Fig. 10. (a): Building measurement. (b): Scan S1 of the building. (c): Scan S2 of the building.

3D modeling software, Rhino [15]. The accuracy of the floor plan was evaluated by comparing the computed dimensions with the original mesh dimensions. The results are presented in Table 1. These findings demonstrate the applicability of our approach to a wide range of buildings.

Table 1. Scanned Data Analysis: $S1area$ and $S2area$ represent the estimated areas of each room obtained from their respective scans. Additionally, $S1_{err}$ and $S2_{err}$ show the percentage error compared to the actual dimensions of each room.

Room	Area	$S1_{area}$	$S1_{err}$%	$S2_{area}$	$S2_{err}$%
202	24.1243	24.037	0.4	32.7107	1.7
203	7.991	5.5332	30.8	6.9699	12.8
204	31.9608	31.9032	0.2	31.7484	0.7
205	32.5398	33.3375	-2.5	32.6814	-0.4
206	32.9304	32.6536	0.8	32.592	1.0

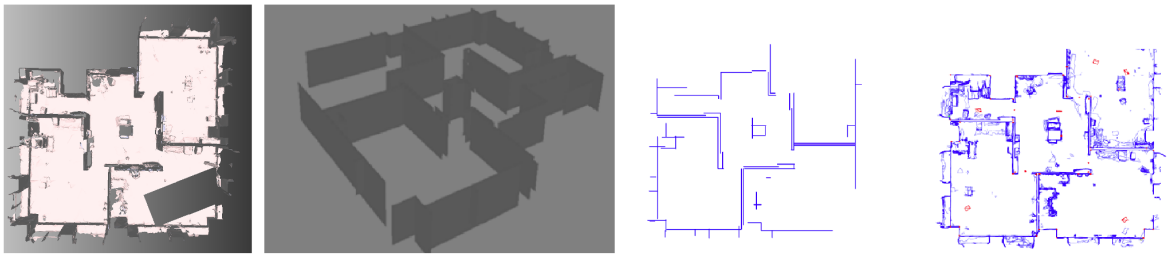

Fig. 11. First: 3D model of building B3. Second: computed flat walls. Third: drafting style floor plan. Fourth: pen-and-ink style floor plan.

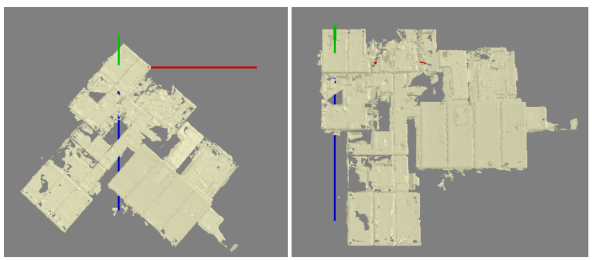

Fig. 12. Mesh model of building B2: Before alignment (left) and after alignment (right).

Our method relies on the quality of the mesh data capture and can accurately compute a floor plan with minimal error. With a higher quality HoloLens scan, the resulting floor plan can be even more precise. Figure 11 displays both types of floor plans and the 3D model obtained from S1.

Orienting the floor and walls We must ensure proper mesh orientation. Spherical k-means is computationally intensive so we optimize it to get good overall performance. In Figure 4, we see the mesh before and after alignment for B1. Mesh orientation for this scene took 12.4 seconds, of which 10.6 seconds were spent aligning the walls using spherical k-means. Similarly, Figure 12 shows the mesh from building B2 before and after alignment.

Partitioning into stories Our pipeline can automatically detect a multi-story building and divide it into individual stories using an additional step. The algorithm projects triangles onto the positive y-axis and creates a histogram, which shows horizontal peaks. By analyzing the peaks in the histogram, the number of stories can be determined. Figures 5 and 13 show examples of a 2-story residential building and a multi-story model from the Matterport3D dataset [22], respectively, that were partitioned into stories.

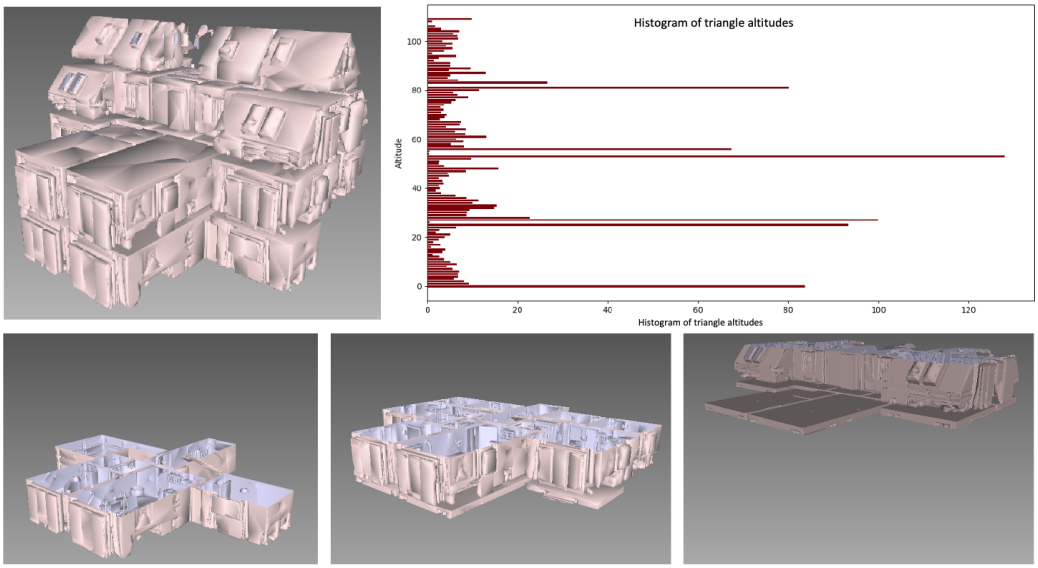

Fig. 13. A three-story building model from the Matterport3D dataset (top-left) and the corresponding triangle altitude histogram (top-right). The bottom figure illustrates the building sliced into three levels.

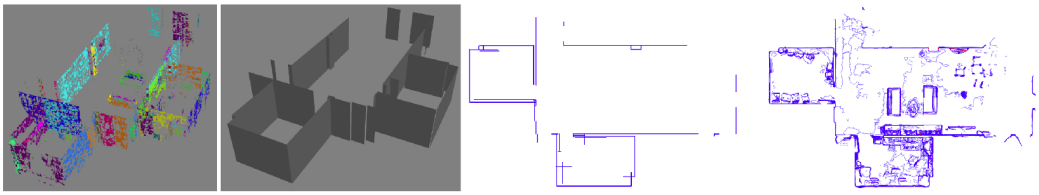

Fig. 14. First: DBSCAN clustering results for building B2. Second: plane-fitted flat walls. Third: drafting-style floor plan. Fourth: pen-and-ink style floor plan.

Finding planar walls To generate a drafting-style floor plan, we eliminate details and identify planar walls. The modified DBSCAN algorithm is the most time-consuming step. In the model of Figure 14, with 79,931 vertices and 134,235 faces, it took 27.4 seconds to prepare the data and run DBSCAN and an additional 3.79 seconds to construct flat walls from the clusters generated by DBSCAN. Similarly, for the residential building of Figure 15, with 173,941 vertices and 285,840 faces, it took 76 seconds to prepare and run DBSCAN and 23.36 seconds to compute flat walls. The results for a Matterport3D model appear in Figure 16.

Generating the floor plan The final step of our floor plan generation is to slice the mesh at different altitudes and superimpose all of the slices. Figure 14, 15, and 16 show floor plans generated using our approach.

We conducted experiments to evaluate the efficacy of our approach for creating floor plan drawings, by investigating different combinations of line segment opacity and slice numbers. We found that an opacity setting of 0.5 produced a floor plan that met our expectations. We also discovered that a floor plan with 100 slices provided a good balance between level of detail and clutter reduction. Optimal numbers will depend on use case.

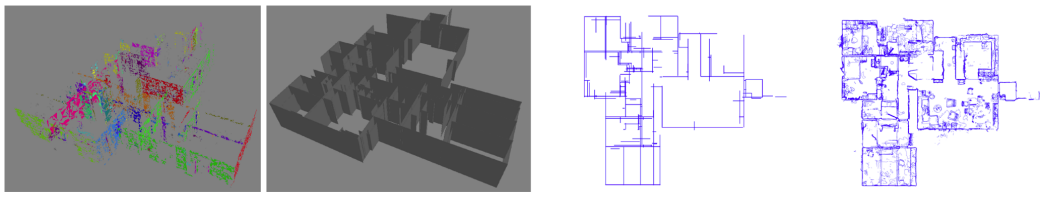

Fig. 15. First: DBSCAN clustering results for B2. Second: plane-fitted flat walls. Third: drafting-style floor plan. Fourth: (c) pen-and-ink style floor plan.

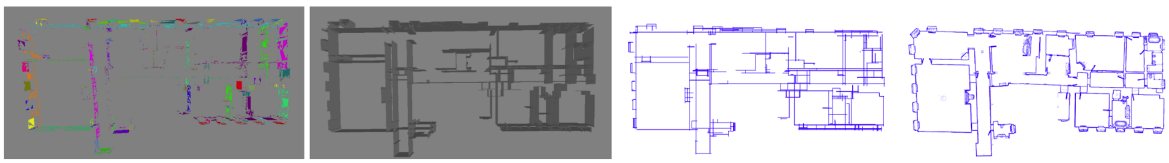

Fig. 16. First: DBSCAN clustering results for a model from the Matterport3D dataset. Second: plane-fitted flat walls. Third: drafting-style floor plan. Fourth: pen-and-ink style floor plan.

Table 2. Computation (Seconds): T_o, T_p, T_{sd} are times for orienting the mesh, finding planar walls, and slicing/drawing.

Building	Type	T_o	T_p	T_{sd}
B1	Commercial	12.1	31.2	0.5
B2	Residential	24.08	81.5	0.9
B3	Commercial	4.63	15.17	0.51

Table 2 lists elapsed time (in seconds) for orienting the mesh T_o, finding flat walls T_p and slicing and drawing T_{sd} for buildings B1, B2 and B3.

5 Conclusion & Future work

In summary, our innovative approach for generating floor plans from triangle mesh data collected by augmented reality headsets has resulted in two unique styles: a detailed pen-and-ink style and a simplified drafting style. Our algorithms align the mesh data with primary coordinate axes to produce tidy floor plans with vertical and horizontal walls, while also allowing for the removal of ceilings and floors and the separation of multi-story buildings into individual stories. Our approach integrates with augmented reality, allowing for the addition of synthetic objects to physical geometry and providing a detailed 3D model and floor plan.

Potential applications include navigation, interior design, furniture placement, facility management, building construction, and HVAC design. Moving forward, we plan to enable support for sloping ceilings, automate wall and door detection, and integrate with other tools such as building energy simulators. Finally, we plan to compare our approach with existing state-of-the-art methods in terms of accuracy and computational time. We are also interested in exploring the applicability of block-based DBScan for 3D reconstruction from incomplete scans. Overall, our approach has the potential to revolutionize the way we generate and visualize floor plans.

References

1. Adan, A., Huber, D.: 3d reconstruction of interior wall surfaces under occlusion and clutter. In: 2011 International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission. pp. 275–281 (2011). https://doi.org/10.1109/3DIMPVT.2011.42
2. Arikan, M., Schwärzler, M., Flöry, S., Wimmer, M., Maierhofer, S.: O-snap: Optimization-based snapping for modeling architecture. ACM Trans. Graph. **32**(1) (feb 2013). https://doi.org/10.1145/2421636.2421642
3. Budroni, A., Boehm, J.: Automated 3d reconstruction of interiors from point clouds. International Journal of Architectural Computing **8**(1), 55–73 (2010). https://doi.org/10.1260/1478-0771.8.1.55

4. Cabral, R.S., Furukawa, Y.: Piecewise planar and compact floorplan reconstruction from images. 2014 IEEE Conference on Computer Vision and Pattern Recognition pp. 628–635 (2014)
5. Cai, R., Li, H., Xie, J., Jin, X.: Accurate floorplan reconstruction using geometric priors. Computers & Graphics **102**, 360-369 (2022). https://doi.org/10.1016/j.cag.2021.10.011
6. Chen, J., Liu, C., Wu, J., Furukawa, Y.: Floor-sp: Inverse cad for floorplans by sequential room-wise shortest path. In: The IEEE International Conference on Computer Vision (ICCV) (2019)
7. Dasgupta, S., Fang, K., Chen, K., Savarese, S.: Delay: Robust spatial layout estimation for cluttered indoor scenes. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 616–624 (2016). https://doi.org/10.1109/CVPR.2016.73
8. Furukawa, Y., Curless, B., Seitz, S.M., Szeliski, R.: Reconstructing building interiors from images. In: 2009 IEEE 12th International Conference on Computer Vision. pp. 80–87 (2009). https://doi.org/10.1109/ICCV.2009.5459145
9. Gao, R., Zhao, M., Ye, T., Ye, F., Wang, Y., Bian, K., Wang, T., Li, X.: Jigsaw: Indoor floor plan reconstruction via mobile crowdsensing. In: Proceedings of the 20th Annual International Conference on Mobile Computing and Networking. p. 249–260. MobiCom '14, Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2639108.2639134
10. Hsiao, C.W., Sun, C., Sun, M., Chen, H.T.: Flat2layout: Flat representation for estimating layout of general room types. ArXiv **abs/1905.12571** (2019)
11. Ikehata, S., Yang, H., Furukawa, Y.: Structured indoor modeling. In: 2015 IEEE International Conference on Computer Vision (ICCV). pp. 1323–1331 (2015). https://doi.org/10.1109/ICCV.2015.156
12. Kruzhilov, I., Romanov, M., Babichev, D., Konushin, A.: Double refinement network for room layout estimation. In: Palaiahnakote, S., Sanniti di Baja, G., Wang, L., Yan, W.Q. (eds.) Pattern Recognition. pp. 557–568. Springer International Publishing, Cham (2020)
13. Lee, C.Y., Badrinarayanan, V., Malisiewicz, T., Rabinovich, A.: Roomnet: End-to-end room layout estimation. 2017 IEEE International Conference on Computer Vision (ICCV) pp. 4875–4884 (2017)
14. Liu, C., Wu, J., Furukawa, Y.: Floornet: A unified framework for floorplan reconstruction from 3d scans. In: ECCV (2018)
15. McNeel, R., et al.: Rhinoceros 3d, version 6.0. Robert McNeel & Associates, Seattle, WA (2010)
16. Microsoft: Spatial mapping. https://docs.microsoft.com/en-us/windows/mixed-reality/spatial-mapping (2022)
17. Monszpart, A., Mellado, N., Brostow, G.J., Mitra, N.J.: Rapter: Rebuilding man-made scenes with regular arrangements of planes. ACM Trans. Graph. **34**(4) (jul 2015). https://doi.org/10.1145/2766995
18. Mura, C., Mattausch, O., Pajarola, R.: Piecewise-planar reconstruction of multi-room interiors with arbitrary wall arrangements. Computer Graphics Forum **35**(7), 179–188 (2016). https://doi.org/https://doi.org/10.1111/cgf.13015
19. Murali, S., Speciale, P., Oswald, M.R., Pollefeys, M.: Indoor scan2bim: Building information models of house interiors. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 6126–6133 (2017). https://doi.org/10.1109/IROS.2017.8206513
20. Okorn, B., Xiong, X., Akinci, B.: Toward automated modeling of floor plans. In: In Proceedings of the symposium on 3D data processing, visualization and transmission. vol. 2 (2010)
21. Pintore, G., Mura, C., Ganovelli, F., Fuentes-Perez, L.J., Pajarola, R., Gobbetti, E.: State-of-the-art in Automatic 3D Reconstruction of Structured Indoor Environments. Computer Graphics Forum (2020). https://doi.org/10.1111/cgf.14021
22. Ramakrishnan, S.K., Gokaslan, A., Wijmans, E., Maksymets, O., Clegg, A., Turner, J.M., Undersander, E., Galuba, W., Westbury, A., Chang, A.X., Savva, M., Zhao, Y., Batra, D.: Habitat-matterport 3d dataset (HM3d): 1000 large-scale 3d environments for embodied AI. In: Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2) (2021), https://openreview.net/forum?id=-v4OuqNs5P
23. Turner, E., Zakhor, A.: Watertight as-built architectural floor plans generated from laser range data. In: 2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization Transmission. pp. 316–323 (2012). https://doi.org/10.1109/3DIMPVT.2012.80
24. Weinmann, M., Wursthorn, S., Weinmann, M., Hübner, P.: Efficient 3d mapping and modelling of indoor scenes with the microsoft hololens: A survey. PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science **89**(4), 319–333 (2021)
25. Xiong, X., Adan, A., Akinci, B., Huber, D.: Automatic creation of semantically rich 3d building models from laser scanner data. Automation in Construction **31**, 325–337 (2013). https://doi.org/10.1016/j.autcon.2012.10.006
26. Zhang, J., Kan, C., Schwing, A.G., Urtasun, R.: Estimating the 3d layout of indoor scenes and its clutter from depth sensors. In: 2013 IEEE International Conference on Computer Vision. pp. 1273–1280 (2013). https://doi.org/10.1109/ICCV.2013.161
27. Zou, C., Colburn, A., Shan, Q., Hoiem, D.: Layoutnet: Reconstructing the 3d room layout from a single rgb image. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2051–2059. IEEE Computer Society, Los Alamitos, CA, USA (jun 2018). https://doi.org/10.1109/CVPR.2018.00219