A. Pérez-Yus, G. López-Nicolás, J. J. Guerrero. Scaled layout recovery with wide field of view RGB-D. Image and Vision Computing, vol. 87, pp. 76-96, 2019.

# Scaled layout recovery with wide field of view RGB-D

Alejandro Perez-Yus*, Gonzalo Lopez-Nicolas, Jose J. Guerrero

*Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad de Zaragoza, Spain*

**Abstract**

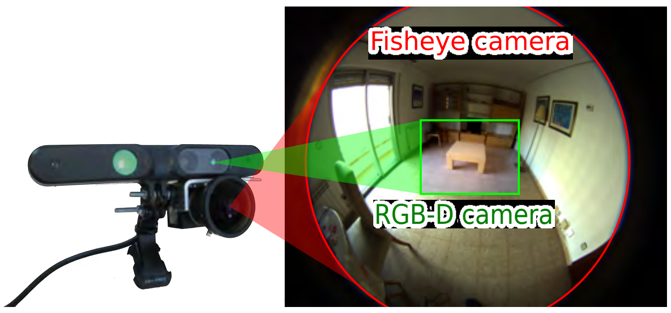
In this work, we propose a method that integrates depth and fisheye cameras to obtain a wide 3D scene reconstruction with scale in one single shot. The motivation of such integration is to overcome the narrow field of view in consumer RGB-D cameras and lack of depth and scale information in fisheye cameras. The hybrid camera system we use is easy to build and calibrate, and currently consumer devices with similar configuration are already available in the market. With this system, we have a portion of the scene with shared field of view that provides simultaneously color and depth. In the rest of the color image we estimate the depth by recovering the structural information of the scene. Our method finds and ranks corners in the scene combining the extraction of lines in the color image and the depth information. These corners are used to generate plausible layout hypotheses, which have real-world scale due to the usage of depth. The wide angle camera captures more information from the environment (e.g. the ceiling), which helps to overcome severe occlusions. After an automatic evaluation of the hypotheses, we obtain a scaled 3D model expanding the original depth information with the wide scene reconstruction. We show in our experiments with real images from both home-made and commercial systems that our method achieves high success ratio in different scenarios and that our hybrid camera system outperforms the single color camera set-up while additionally providing scale in one single shot.

*Keywords:* 3D layout estimation, RGB-D, Omnidirectional cameras, Multi-camera systems

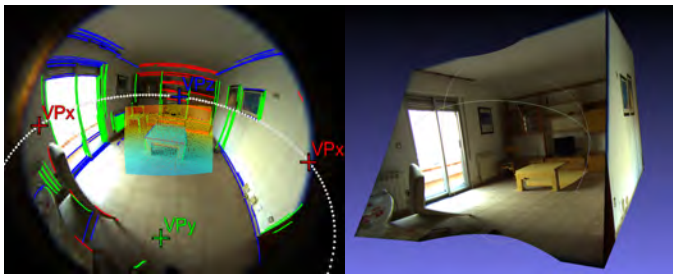
## 1. Introduction

One of the most important topics in computer vision and robotics has been to perceive the 3D information from the scene. The recent advent of consumer RGB-D cameras has caused a great impact in the field, since having color and depth information synchronized in one single shot is very appealing. Unfortunately, these devices usually have a field of view (FOV) too narrow for certain applications, and it is necessary to move the camera in order to capture different views of the scene. That is often not easy to achieve when cameras are attached to systems with limited mobility (e.g. a robot navigating an indoor environment with rigidly attached camera, or a visually impaired aid with a fixed camera pointing towards the front of the user to detect obstacles or other hazards [1]).

Here, we propose to use a color camera with wide FOV to extend the depth information in a hybrid camera configuration composed by a depth and a fisheye camera. The FOV of a fisheye is over 180°, in contrast with the usual FOV of $43° \times 57°$ of consumer depth cameras (Fig. 1a). Once the cameras are calibrated, the system is capable of viewing over a hemisphere of color information where the central part of the image has also depth data (about 8.7% of the total number of pixels, as shown in Fig. 1b). One can think of this configuration inspired in the vision of the human eye, where the central part (fovea) provides richer information than the periphery, and the field of view is



(a)



(b) (c)

**Figure 1:** (a) Fields of view of our proposed system composed by a Fisheye and a RGB-D camera. (b) The depth information in the center is extended to the periphery combining information with the line segments that we use to extract the spatial layout of the scene. (c) As a result we obtain a full-scaled 3D reconstruction of the scene.

*Corresponding author

*Email addresses:* alperez@unizar.es (Alejandro Perez-Yus), gonlopez@unizar.es (Gonzalo Lopez-Nicolas), josechu.guerrero@unizar.es (Jose J. Guerrero)

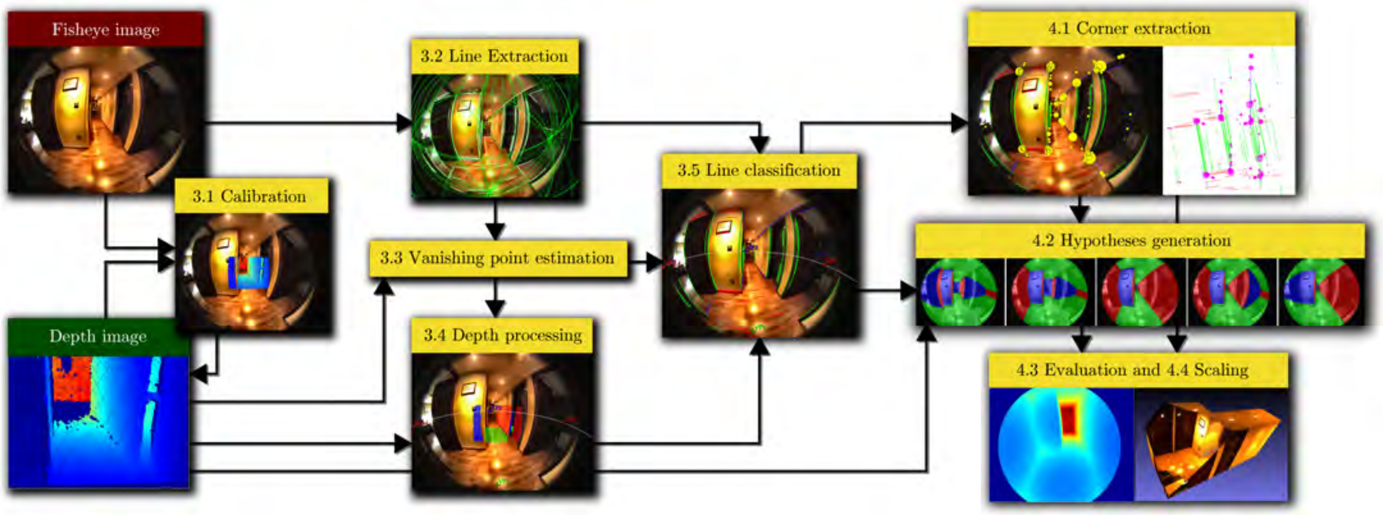


**Figure 2:** Block diagram of the main stages of the algorithm, starting with the initial Fisheye and Depth images and finishing with the result of the scaled layout. In the captions the numbers of the corresponding sections of each stage.

slightly over 180°. Notice that, although our work uses a fisheye camera, the approach could be extended to other kinds of omnidirectional systems. This kind of system could be helpful in limited mobility problems since it provides safe depth observations in front of the moving robot/subject, but at the same time allows to recover additional meaningful scene information with wider context.

In particular, we propose to extend the 3D information in one single shot via spatial layout estimation. Acquiring knowledge about the layout of the room can have advantages in many tasks like mapping, navigation, scene recognition, augmented reality or even object detection (e.g. objects on surfaces, discerning objects from structure). Layout estimation approaches are not new, but most of them are either too restricted (i.e. specific room shapes like boxes) or designed for conventional cameras with small field of view, which weakens the results since the available context is reduced and the possibility of occlusions is much higher. Besides, most of them produce results up to a scale, which is not ideal for the applications mentioned above. Here we propose a method able to overcome all these limitations. Notice that our approach does not only provide geometric reconstruction, but also provides additional contextual information about the scene, such as the distribution of walls, floor and ceiling, and thus the shape of the room. Hence, it is different to methods that register RGB-D frames together that provide just pure geometrical information (including non-structural objects), unless further processing is applied.

Our layout estimation method is based on line segments from the fisheye image, and provides scaled solutions rooted on the depth information. As a result, a final 3D scene reconstruction is provided (see Fig. 1c). The 3D room layout can be seamlessly merged with the original depth information to generate a 3D image with the periphery providing an estimation of the spatial context to the central part of the image, where the depth is known with good certainty. The collaboration between cameras is bidirectional, since the extension of the scene layout to the periphery is performed with the fisheye, but the depth information is used both to enhance the layout estimation algorithm and to scale the solution.

A scheme of the whole algorithm with the corresponding sections in the paper is shown in Fig. 2. In detail, the depth camera provides a region of the image with 3D data, from which an initial estimate of the Vanishing Points (VPs) and 3D planes can be recovered. The VPs are used to retrieve the scene orientation necessary to generate layout proposals. In this work, we assume scenes are from a Manhattan World [2], meaning the world is organized according to three orthogonal main directions. This assumption holds for most human-made scenarios, especially indoors. The 3D planes extracted are used to provide scale, impossible to get otherwise with one single shot and no previous knowledge of the scene. The layout is scaled by detecting the floor plane, however, we include a scaling procedure in case the floor is not found. Having scale has many advantages in this type of methods which usually require several heuristics. For instance, tuning parameters can be grounded in reality. Depth information is also used to filter hypotheses and reward line segments corresponding to intersections of planes.

The line segments from the wide image are classified according to the three Manhattan directions. The horizontal lines are projected either to the floor plane or the estimated ceiling plane to have the 3D segment position in the real world. Structural corners are then looked for, by considering plausible and simple cases of line distribution. The corners of the map are evaluated by our scoring function, and layout hypotheses are proposed by the probability of these corners to occur in the real world, proportional to these scores. Then, layout hypotheses are generated based on geometrically coherent wall distributions that do not contradict the initial depth information and the visible segments. The algorithm is able to work even under high clutter circumstances due to the combination of lines from both floor and ceiling (visible because of using a large FOV camera), but also because of our generation of Manhattan hypotheses that can estimate hidden corners to complete the layout. In order to choose the best hypothesis, we propose and compare four criteria for evaluation of hypotheses. One of them is adapted from the state of the art (Orientation Map from [3]), whereas the other three, newly proposed, have comparable performance while being much more efficient.

Preliminary results were already presented in [4]. To our knowledge, this was the first time this configuration was used, although the interest in such sensor pairing is clear in the recent Google's Tango project (launched in 2014) and recent approaches that also use a similar set-up [5]. Experiments with real images have already shown promising results about both the proposed algorithm and the camera configuration. In particular, our method gets good results even when only a few hypotheses are drawn, and having depth information helps notably in the layout extraction. In this paper, we improve some stages of the algorithm in both performance and efficiency, add some additional features to generalize the method to more environments and extend the experiments showing results from other device. In particular, our improvements with respect to [4] are:

- New evaluation criterion called *Angular Coverage* (AC) which outperforms our Sum of Scores and Sum of Edges and presents similar results to the Orientation Map [3] with much less required computation.
- A scaling procedure has been added for those cases where the floor plane is not in the image.
- The generation of hypotheses now includes a pre-filtering of corners that can be paired with one another, speeding up the process.
- The estimation of the height of the ceiling has been improved using the more robust concept of angular coverage.
- Experiments have been extended considering the new features and evaluation criterion.
- New experiments with images taken from the Google Tango device have been carried out, showing the potential of the method in commercial devices.

## 2. Related work

One of the first attempts to recover 3D layout information of indoor environments with single images was [6], which uses a Bayesian network model to find the floor-wall boundary. In contrast, Lee et al. [3] use line segments to generate layout hypotheses evaluating their fitness to an Orientation Map (OM). Using lines has the advantage of producing results without relying on scene-specific properties such as colors and image gradients. However, while some lines can actually include structural information of the environment (e.g. intersections wall-wall or wall-floor), usually most of them belong to clutter or are useless and misleading. To help with this problem, some assumptions and some set of rules are usually proposed based on geometric coherence. Usually the main assumptions are that all structures in indoor environments are composed by planar surfaces and that these surfaces are oriented according to three orthogonal directions [2]. This assumption holds for most indoor environments, and it is almost unanimously used in the literature.

Other works try to simplify the problem by making assumptions about the structure, e.g. assuming that the room is a 3D box, like Hedau et al. in [7]. This work uses a modified version of Geometric Context (GC) [8] instead of the OM for evaluation of hypotheses which includes a separate clutter category. Similar examples are [9], which applies efficient structured prediction; [10], which defines a catalogue of scene corners; and [11], whose method is based on line consistency. Some other works perform this type of '3D box' reasoning while performing object detection [12, 13, 14, 15, 16, 17]. Doing so can have both tasks help each other (e.g, it is impossible for this room to have a certain layout if there is a bed across the wall, or vice versa). However, using the '3D box' assumption does not generalize well in real world scenes, e.g. in corridors or entrances. Other approaches make use of video sequences instead of single images [18, 19]. While introducing temporal consistency may be beneficial in this task, we focus on getting better layout estimation with single image.

Deep learning has exploded in popularity in the last few years, and recently new interesting approaches in layout estimation are appearing and dominating in challenges such as LSUN Room Layout Estimation [20]. That is the case of [21], which has trained a fully convolutional neural network (FCNN) to extract the informative edges of the scene (i.e. those corresponding to structural boundaries and not coming from clutter), and then use those edges to extract the layout. Another similar proposal with better results is shown in [22]. Dasgupta et al. [23] propose an alternative method that trains a FCNN which returns the heat maps of each surface class (walls, ceiling, floor). RoomNet [24] is an end-to-end neural network which infers the type of room and the main keypoints to build a layout (i.e. corners). These approaches show a great potential of these data-driven techniques. On the other hand, the results are too tied to specific room configurations, not being able to work on complex structures.

The methods mentioned up to this point have in common that all of them use images from conventional cameras. As opposed to that, some recent works use omnidirectional cameras such as catadioptric systems or fisheye cameras. Having greater field of view has many advantages for this task:

- Larger view of the line segments appear in the image, so it is more likely to extract the relevant lines of the scene.
- Allows to perceive better the orientation of the scene and thus a more robust vanishing point estimation.
- Provides better view of the ceiling areas, which usually have less clutter than the lower parts in indoor scenes.
- Larger portion of the room is captured at once, which provides wider or even complete room reconstructions.

For example, in [25], the authors use a fisheye camera to perform layout retrieval, essentially extending the work from [3], but with wider FOV. Lopez-Nicolas et al. in [26] perform the closed layout recovery using a catadioptric system mounted in a helmet. Jia and Li [27] use 360° panorama full-view images, which allows them to recover the layout of the whole scene at once. Fukano et al. [28] propose another method with panoramas, solving the problem as a high order energy minimization. Similarly, PanoContext [29] uses the same type of images to perform layout retrieval along with a whole-room context model in 3D including bounding boxes of the main objects inside the room. Pano2CAD [30] enhances PanoContext by considering not only box-shaped types of room. In [31] they propose an alternative graph-based method for panoramas, combining superpixels and line segments. Cabral et al. [32] gets 3D information from structure from motion with a panoramic image, and then generates the best fitting floor plan reconstruction.

In these approaches the recovered 3D layout is obtained up to a scale, unless previous knowledge about the scene is provided. Modern RGB-D cameras are able to provide depth alongside color, which provides scale information in a single shot. However, most of these cameras have a FOV too narrow for layout estimation. Recently, some alternatives investigate extending the FOV of depth cameras using additional elements. For example, [33] uses two planar mirrors as a catadioptric extension of the RGB-D device to view to the front and to the back of the robot. More generally, [34] proposes a framework for omnidirectional RGB-D camera calibration. A consumer set of wide angle lens is used in [35]. Fernandez-Moral et al. [36] proposed a method to calibrate an omnidirectional RGB-D multi-camera rig from plane observations. These approaches are either expensive to build [36], hard to calibrate [33, 34], or do not provide good enough depth maps [35]. There are commercial systems available such as the Matterport camera that have been used to create the large scale indoor dataset from [37]. However, this camera system is too expensive and unpractical to be used in situations that require mobility.

Here, we propose to use a depth camera alongside a fisheye in the same calibrated system, and perform the depth extension via spatial layout estimation. Our proposal combines the advantages of omnidirectional cameras (recover wider information) and depth cameras (provide 3D certainty and scale). This is also a camera system that already exists in the market (e.g. Google Tango) or can be easily reproduced with state-of-the-art calibration methods [38, 39]. No restrictions about the shape of the room are considered, i.e. not only '3D box' room shapes or layout estimations tied to specific room configurations. No rigid assumption about the camera pose in the scene is considered, since it is found automatically. Besides, no sequences of images or machine learning algorithms are used in our method.

## 3. Depth and fisheye images processing

In this section, we address the initial stages of our method, describing how we extract the information we need to perform the layout recovery (Fig. 2). First we explain the calibration of the system, needed to map information from the depth camera to the fisheye camera (Section 3.1). Then we describe the line segment extraction algorithm in the fisheye image (Section 3.2). To recover the Manhattan directions we use information from depth (for an initial estimate) and lines (for the final values), as described in Section 3.3. Then we perform plane extraction in the depth image to find 3D line intersections and to enable scaled layout extraction (Section 3.4). At the end, the line segments are classified according to its orientation (Section 3.5).

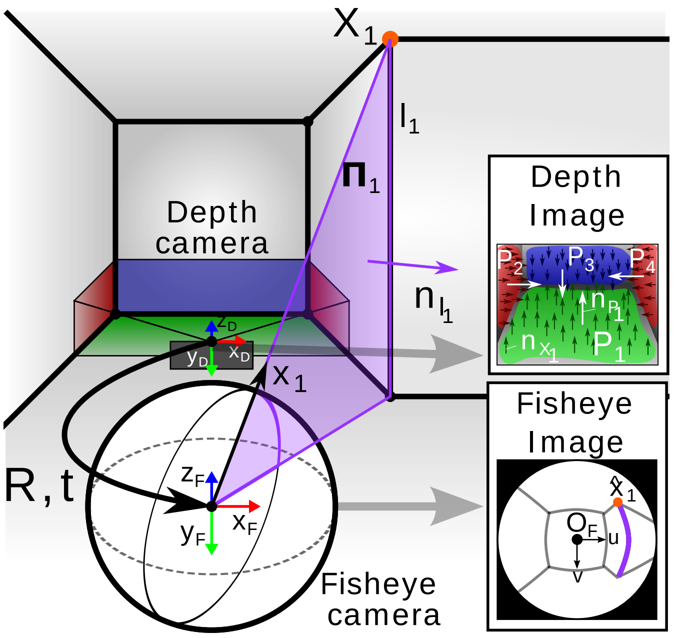


**Figure 3:** Scheme of the system in a 3D world scene with the extrinsic calibration parameters **R** and **t** and the correspondent depth and fisheye images.

### 3.1. System calibration

To map world points $\mathbf{X}$ from the depth camera reference frame $D$ to the fisheye camera reference frame $F$, it is necessary to calibrate the extrinsic parameters $(\mathbf{R}, \mathbf{t})$ and the intrinsic parameters of both cameras (Fig. 3). The extrinsic calibration of range sensors to cameras is not a new issue, but most related works require manual selection of correspondences or do not support omnidirectional cameras [40, 41, 42]. To obtain the intrinsic parameters of the fisheye camera, we need a specific method with an appropriate camera model [43]. In particular, we choose the parametric camera model described by Scaramuzza et al. in [44], which considers the omnidirectional image as a highly distorted image with the distortion modeled as a polynomial. Using this polynomial it is not necessary to provide a specific model of projection and works with all kind of perspective, catadioptric or dioptric cameras. The points in the image $\hat{\mathbf{x}}_i = [u_i, v_i]$ and the vector $\mathbf{x}_i = [x, y, z]$ which points to the world point $\mathbf{X}_i$ are related following:

$$\mathbf{x}_i = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} u_i \\ v_i \\ f(\rho_i) \end{bmatrix} = \begin{bmatrix} u_i \\ v_i \\ a_0 + a_2\rho_i^2 + ... + a_N\rho_i^N \end{bmatrix} \quad (1)$$

where the function $f$ is the polynomial that models the distortion with coefficients $a_0..a_N$ and $\rho_i = \sqrt{u_i^2 + v_i^2}$.

For the extrinsic calibration we have recently proposed two alternative methods suitable for our system [39, 38]. While the one in [39] has the advantages of being more generalizable to other camera systems and does not require to build a calibration device, we use [38] since it is more accurate for our camera system. This method is inspired by [45], adapted to the fisheye camera model from [44]. During the calibration process, the intrinsic parameters of the depth camera are also computed as defined in [45] to improve the default parameters of the system. The depth images as captured by the sensor are transformed to point clouds using these parameters, and they are rotated and translated to the fisheye camera reference frame, following $\mathbf{X}_F = \mathbf{R} \cdot \mathbf{X}_D + \mathbf{t}$. From now on, every computation is done in that frame unless specified. A more detailed analysis of our calibration procedure is presented in [38].

### *3.2. Line extraction in the fisheye image*

For the line extraction in the fisheye camera we use the work from [46], which is compatible with central catadioptric and dioptric systems with revolution symmetry. Unlike conventional cameras, 3D lines in space do not appear as straight lines in the omnidirectional images, but they are projected to curves called line-images. In the schematic scene from Fig. 3 we highlight a vertical line segment on the sphere model and its projection in the fisheye image. The shape of these line-images changes with the type of omnidirectional camera and its specific camera configuration.

The projection of a line $l_i$ in the 3D space can be represented by the normal of the plane $\mathbf{\Pi}_i$ defined by the line itself and the viewpoint of the system, with normal $\mathbf{n}_{l_i} = (n_x, n_y, n_z)^\top$. The direction vector $\mathbf{x}$ of the points $\mathbf{X}$ lying on a 3D line $l$ satisfies the condition $\mathbf{n}_l^\top \mathbf{x} = 0$. From [46] and with (1), the constraint for points on the line projection in image coordinates is:

$$n_x u + n_y v + n_z f(\rho) = 0 \tag{2}$$

The line-images are non-polynomial and do not have conic shape. To extract them is necessary to solve a minimization problem [46]. In the process, each line $l_i$ is associated to a set of contour points, denoted by $c(l_i)$, which are the inliers of the constraint (2).

### *3.3. Estimation of the vanishing points*

As mentioned before, we assume the scenes are organized according to three orthogonal directions, $\{\mathbf{m}_x, \mathbf{m}_y, \mathbf{m}_z\}$, that define the Manhattan reference frame $M$. Parallel lines in the 3D world intersect in one single point in perspective images, called Vanishing Point (VP). In omnidirectional images, line projections result in curved line-images, and parallel lines intersect in two VPs. The directions $M$ are correlated to the VPs in regard that lines along these directions intersect in their corresponding VPs. Thus, we refer indistinctly to the computation of $M$ and the VPs from now on. We estimate the VPs to classify lines and planar surfaces from the depth information according to the three Manhattan directions.

There are previous approaches to obtain the VPs from omnidirectional images [47]. However, we propose a method to extract the VPs taking advantage of both cameras with a two step optimization problem. Depth information is usually more robust, but less accurate than RGB information. Using fisheye images typically obtain a more accurate VP solution, but the problem may be unable to converge if the initial solution is not good enough. Besides that, a joint optimization is problematic as it needs to weight both terms appropriately. Experiments showed that our two-stage optimization procedure performs well without significant extra computational cost.

To compute $M$, we define a $3 \times 3$ matrix $\mathbf{M}$ that has the three Manhattan directions by columns, i.e. $\mathbf{M} = [\mathbf{m}_x, \mathbf{m}_y, \mathbf{m}_z]$. The initial solution of $\mathbf{M}$ is set as a three orthogonal vector base ($\mathbf{M} = \mathbf{I}_{3\times3}$). The variables to optimize are the roll-pitch-yaw angles ($\alpha, \beta$ and $\gamma$) that form the rotation matrix $\mathbf{R}_{\alpha,\beta,\gamma}$ that after the optimization process should orient the vector base according to the Manhattan directions, $\mathbf{M} = \mathbf{R}_{\alpha,\beta,\gamma} \cdot \mathbf{I} = \mathbf{R}_{\alpha,\beta,\gamma}$. All optimizations are performed with the Levenberg-Marquardt algorithm.

#### *3.3.1. Initial estimate with depth information*

The first step is to get the 3D normals of the points in the cloud. The normals $\mathbf{n}_{\mathbf{X}_i}$ of every point $\mathbf{X}_i$ (Fig. 3) can be estimated using the method from [48]. To reduce computation time, the cloud can be previously down-sampled (e.g. with a voxel grid filter). In Manhattan scenes, it is likely for a large amount of points to have normals oriented in these directions. Based on this, the vector base is rotated until the angle between the normals of as many points as possible and one of the three vectors from the base is minimized. Notice that two normals are enough to define the three main directions since the system is orthogonal. Moreover, since this is an initial estimate, a good result may also be eventually obtained having just one normal.

The minimization problem to retrieve $\mathbf{M}$ is formulated as follows:

$$\underset{\alpha,\beta,\gamma}{\arg\min} \sum_{i=1}^{N_x} min\left(\left|arccos(\mathbf{R}_{\alpha,\beta,\gamma}^\top \cdot \mathbf{n}_{\mathbf{X}_i})\right|\right) \tag{3}$$

where $N_x$ is the number of points from the cloud. The product $(\mathbf{R}_{\alpha,\beta,\gamma}^\top \cdot \mathbf{n}_{\mathbf{X}_i})$ gives the cosine of the normal with respect to each one of the directions, from which the *min* function only takes the smallest one of the angles in absolute value. The columns of the final rotation matrix $\mathbf{R}_{\alpha,\beta,\gamma}$ are the three Manhattan directions $\mathbf{M}$. An example where the points have been classified according to their normals and with the correspondent VPs is shown in Fig. 4a.

#### *3.3.2. Final estimate with lines in the fisheye image*

In this second stage, we use as seed the current value $\mathbf{M}$ from the previous optimization. The vector base is now rotated until the angle between the normals of as many lines as possible and one of the three vectors from the base is as close of being orthogonal as possible. This is based in that, by definition, the normal $\mathbf{n}_l$ of every line $l_i$ is orthogonal to the direction of the line in the 3D world, and therefore, if a line follows the Manhattan direction $\mathbf{m}_j$, then $\mathbf{n}_{l_i}^\top \cdot \mathbf{m}_j = 0$. The optimization problem is formulated as follows:

$$\underset{\alpha,\beta,\gamma}{\arg\min} \sum_{i=1}^{N_l} min\left(\left|\mathbf{R}_{\alpha,\beta,\gamma}^\top \cdot \mathbf{n}_{\mathbf{l}_i}\right|\right) \tag{4}$$

where the initial values of $\alpha$, $\beta$ and $\gamma$ are the values returned from the first minimization and $N_l$ is the number of lines in

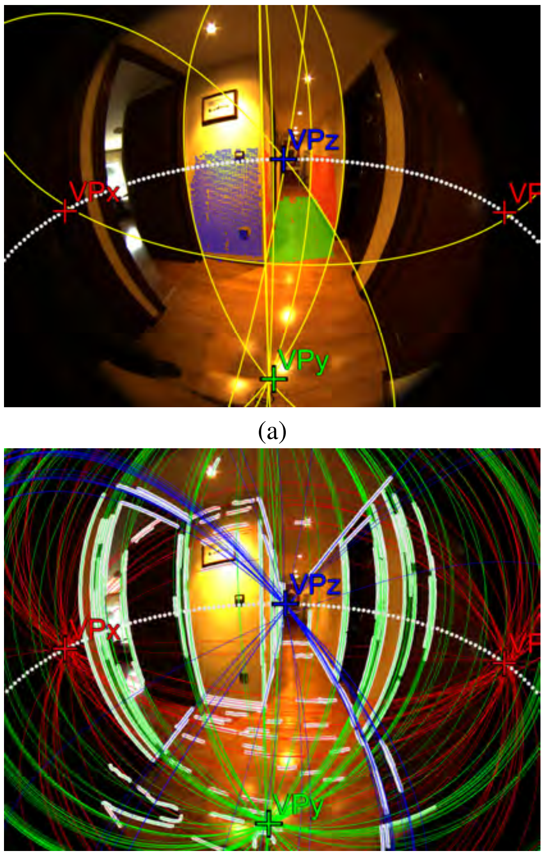


**Figure 4:** (a) Planes from depth classified according to the Manhattan directions (red in $\mathbf{m}_x$, green in $\mathbf{m}_y$, blue in $\mathbf{m}_z$), initial extracted vanishing points, horizon line (white dotted line), and 3D intersections in yellow lines. (b) Line-images classified with their contours in white and the vanishing points after the second optimization.

the fisheye image. In Fig. 4b there is an example where the line-images that support each direction have been colored accordingly.

Our convention is to denote $\mathbf{m}_y$ the column whose vector is closest to the gravity vector given an intuition of how the camera is posed (pointing to the front, slightly downwards). We choose $\mathbf{m}_z$ to be the column pointing to the front and leaving $\mathbf{m}_x$ orthogonal to the previous two. The VPs are the points in the image that result of projecting rays following the Manhattan directions according to the intrinsic parameters (Section 3.1).

### 3.4. *Depth information processing*

In this stage we start from the registered point cloud and extract planes (Section 3.4.1) and we determine if the floor is present in the image and provide a final transformation from camera pose to oriented scene pose (Section 3.4.2).

#### 3.4.1. *Plane extraction*

The points from the point cloud $\mathbf{X}$ are classified depending on the orientation of their normals $\mathbf{n_X}$ in the three orthogonal classes, given a certain angular threshold. For each class we perform a RANSAC for planes to recover its plane equations. It can happen that some of these plane equations have inliers in different surfaces separated in space (e.g. wall planes at each side of an open door, or two separate tables of same height). In this work we recover planar patches instead, i.e. groups of points that belong to the same planar surface and that are also close to each other in Euclidean space. Hence, for each set of inliers, a 3D clustering is performed to recover the planar patches $P$ in the image (Fig. 3). Each plane is defined by its normal $\mathbf{n}_P$ and distance to the origin $X_0$ so that any point $\mathbf{X}$ belongs to a plane if $\mathbf{n}_P^\top \cdot \mathbf{X} + X_0 = 0$.

#### 3.4.2. *Floor detection and scene pose*

In this work we assume the floor and ceiling are unique and symmetric. Among the horizontal planes (i.e. with normal $\mathbf{n}_P = \mathbf{m}_y$), the lowest one (i.e. highest $X_0$ value below the horizon) is initially chosen as *floor plane* ($P_{floor}$). Then we verify if there are a significant amount of points below that plane (considering a threshold due to noise): if there are points below the $P_{floor}$ then it is not a valid floor plane, but other structure (such as a table). When the floor plane is discarded or not found, a virtual $P^*_{floor}$ with normal equal to $\mathbf{m}_y$ and distance to the origin $X_0 = 1$ is created to continue the execution of the algorithm normally. At the end, the rest of the planes are used for scaling (see Section 4.4).

We compute the transformation matrix ${}^M\mathbf{T}_F \in \mathrm{SE}(3)$ that transforms 3D points from the fisheye reference frame to the Manhattan reference frame. To compute ${}^M\mathbf{T}_F$ we create its counterpart ${}^F\mathbf{T}_M$ with rotation part the Manhattan directions ($\mathbf{M}$) and the translation vector $\left[0, X_0^F, 0\right]^\top$, where $X_0^F$ is the height of the camera with respect to the floor. Then, ${}^M\mathbf{T}_F = {}^F\mathbf{T}_M^{-1}$.

### 3.5. *Classification of lines*

Those lines $l_i$ whose minimum angular distance to their closest Manhattan direction $\mathbf{m}_j$ is below a threshold $\theta_1$ are classified as lines in that direction $L_j$:

$$\left|\measuredangle(\mathbf{n}_{l_i}, \mathbf{m}_j) - \frac{\pi}{2}\right| < \theta_1 \rightarrow l_i \in L_j \qquad j = \{x, y, z\} \tag{5}$$

where $\measuredangle$ indicates the angle between its two vector arguments. An example of lines classified is shown in (Fig. 4b).

The *horizon line* is the line-image $l_H$ corresponding to the normal $\mathbf{n}_{l_H} = \mathbf{m}_y$ (drawn in dotted white line in Fig. 4). Lines oriented in $\mathbf{m}_x$ and $\mathbf{m}_z$ are classified as *upper lines* ($\overline{L}$) when they are above horizon, and *lower lines* ($\underline{L}$) when they are below. Lines oriented in $\mathbf{m}_y$ ($L_y$) are classified as *long lines* when they have contour points above and below the horizon.

Some lines correspond to intersections of 3D planes extracted from the depth image. In order to detect such correspondences, we compute the 3D intersection lines of wall planes with the floor plane and between walls, that we call $L^{3D}$. When there are two consecutive wall planes of the same orientation, the line of the border is computed instead. An example is shown in Fig. 4a, where all $L^{3D}$ have been drawn in yellow. Every 3D intersection line $l_j^{3D}$ can be projected to the fisheye image and have its line normal computed ($\mathbf{n}_j^{3D}$). To perform the association, we evaluate the angular distance between their normals, and choose the closest if the angular distance is below a small threshold $\theta_2$:

$$\left|\measuredangle(\mathbf{n}_{l_i}, \mathbf{n}_j^{3D})\right| < \theta_2 \rightarrow l_i \in L^{3D} \tag{6}$$

Those lines supported by 3D evidence have more relevance when generating layout hypotheses. To refer to these lines we use the boolean function $\lambda(l_i)$ defined as:

$$\lambda(l_i) = \begin{cases} 1 & if \;\; l_i \in L^{3D} \\ 0 & otherwise \end{cases} \tag{7}$$

## 4. Layout estimation

To extend the depth information to the periphery, we look for features in the fisheye image that allow us to draw coherent layout hypotheses. We choose *corners*, i.e. points of intersection of three alternatively oriented structural planes in the 3D world, manifested in the image as intersections of lines. In Section 4.1 we describe how the corners are detected and scored for the next stage: the generation of layout hypotheses, explained in Section 4.2. Finally, we deal with the evaluation process in Section 4.3 and the final global scaling (which is to be applied when the floor has not been found) in Section 4.4.

### *4.1. Corner extraction*

We call *corner* ($C$) in this context to the physical intersection of two walls and floor or ceiling. The junctions between these structural planes often produce detectable line segments in an image, whose intersection produces a corner detection. However, not all line intersections are actual corners, and not all actual corners have detectable line segments (e.g. occlusions or not enough contrast in the image).

To address this issue, first we translate the data from pixel space to 2D metric space by creating a floor plan projection of the lower lines (Section 4.1.1). Then we estimate the height of the ceiling and do a similar procedure with the upper lines (Section 4.1.2). We define the types of corners we detect in Section 4.1.3, covering all plausible cases with a minimum set of lines. At the end we describe how we score the corners in Section 4.1.4 to reward corners formed by more lines, longer lines, less distance from the lines to the intersection point and lines coming from 3D plane intersections.

#### *4.1.1. Floor plan projection*

The line segments from Section 3.5 represent just a projection, whose depth is unknown (except for those $l_i \in L^{3D}$). From previous steps, we have the 3D location of at least one structural plane from the depth data (the floor plane $P_{floor}$). We use that plane to project all the lower lines and place them in a scaled 2D floor plan of the scene, we call *XZ-plane*. Notice that, in the cases the floor plane has not been found the algorithm continues with the virtual floor plane $P^*_{floor}$ normally. In those cases, the scale is lost in the process and it is recovered afterwards (Section 4.4).

We can get the ray emanating from the optical center to every contour point of every lower line $\underline{L}_x$ and $\underline{L}_z$ and intersect them with the $P_{floor}$ in 3D (Fig. 5a). With the transformation ${}^{M}\mathbf{T}_F$, we can transform these points from the camera reference frame $F$ to $M$, with the Manhattan directions and origin at the floor level. If we plot the transformed points in the axis $x - z$, we can get a 2D floor plan of the contours with scale (the *XZ-plane* in Fig. 5). Notice also that we naively projected all lower lines, unaware if they actually belong to the wall-floor junction or to clutter, since it is impossible to know with the information we have. Further stages will choose the lines which most likely belong to the real junctions.

With the points now in this 2D projection, we define the *angle of a point*, $\alpha(\mathbf{X}_i)$, as the central angle of the arc between $-\mathbf{z}$ and the radius from the origin to $\mathbf{X}_i$, as shown in Fig. 5a. Similarly, we compute the *angle of a line*, $\overline{\alpha}(l_i)$, as the central angle between its end points. In the Fig. 5a, $\overline{\alpha}(l_1) = \alpha(\mathbf{X}_2) - \alpha(\mathbf{X}_1)$. Note that these angles are defined not only by the value of the angle itself, but also by their starting and ending points. To extract to the value of the angles we define the operation $\langle \bullet \rangle$ that returns a numeric value. For instance, $\langle \alpha(\mathbf{X}_1) \rangle = 100°$ and $\langle \overline{\alpha}(l_1) \rangle = 40°$ in Fig. 5a. Vertical lines $L_y$ are a special case only defined by a single angle $\alpha(L_y)$ and thus $\langle \overline{\alpha}(L_y) \rangle = 0$. These definitions will be helpful in next stages.

#### *4.1.2. Ceiling plane projection*

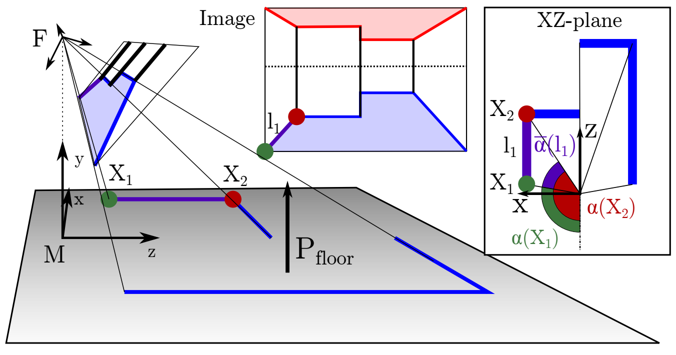
Similarly to the previous section, to get the ceiling plane projection, the rays traced from the optical center to the contour points of the upper lines must be intersected with the $P_{ceil}$. Since we consider floor and ceiling unique and symmetric, we know that the normal of the ceiling plane will be the same as the floor normal, but the distance to the origin is still unknown. To estimate the height of the ceiling ($H_{ceil}$) we assume that, in the XZ-plane view, wall-floor ($\underline{l}_j$) and wall-ceiling ($\overline{l}_i$) intersection segments of the same wall must be coincident. We can generate a $P_{ceil}$ at an arbitrary height, compute the 3D intersections of the projection rays and evaluate how well the contours from upper segments $c(\overline{l}_i)$ coincide with contours from lower segments $c(\underline{l}_j)$ in the XZ-plane. In Fig. 5b there is an example with three different $H_{ceil}$. $H_1$ is too small and $H_3$ too big, so the segments of the floor do not match the segments of the ceiling in the XZ-plane. $H_2$ is the best one as contours from both planes match perfectly. Mathematically, $\forall l_i \in \overline{L}$ and $\forall l_j \in \underline{L}$, we express the overlap in two ways:

- **Contour overlap**. Denoted by $c(l_i) \cap c(l_j)$, determines the number of contours overlapping (i.e. in the 2D plane, contour points from ceiling and floor that are closer than a certain threshold).
- **Angular overlap**. Denoted by $\overline{\alpha}(l_i) \cap \overline{\alpha}(l_j)$, determines the shared angle of the two lines given the definitions from Section 4.1.1.

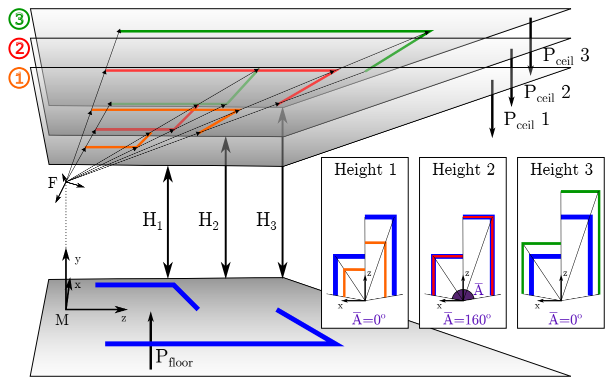
Then we propose the following optimization problem:

$$\underset{H_{ceil}}{\arg\max} \, \langle \overline{A}(H_{ceil}) \rangle \tag{8}$$

where $\overline{A}(H_{ceil})$ is the *Angular Coverage* (AC) of the overlapping contours as a function of $H_{ceil}$. Here we introduce the concept

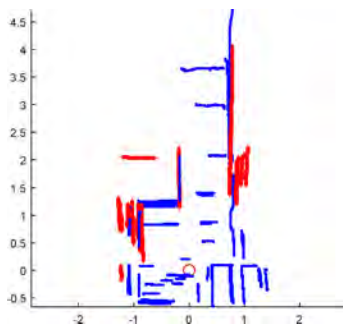


(a)

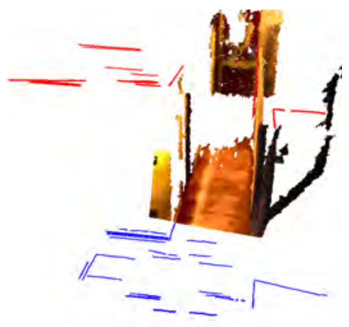


(b)

**Figure 5:** (a) Projection of the lower line segments of the image to the $P_{floor}$ in schematic 3D view and the resulting XZ-plane. Definition of angles of a point, $\alpha(X)$, and angle of a line, $\overline{\alpha}(l)$. (b) Projection of the upper line segments to three virtual ceiling planes ($P_{ceil}$) at different $H_{ceil}$. The chosen $H_{ceil}$ is the one with highest angle coverage of overlapping line segments in the XZ-plane ($H_2$ in the example).



(a)



(b)

**Figure 6:** Real example of contour projection of lower lines (blue) and upper lines (red) to the XZ-plane (a) and 3D point cloud (b). The small circle represents the position of the camera system.

of *Angular Coverage* (AC), denoted by the function $\overline{A}(\bullet)$, which returns the union of central angles around the origin that satisfy certain condition. In this case, the condition is having contour overlap of ceiling and floor line pairs, and $\overline{A}(\bullet)$ is a function of $H_{ceil}$. Mathematically:

$$\overline{A}(H_{ceil}) = \bigcup \left(\overline{\alpha}(l_i) \cap \overline{\alpha}(l_j)\right) \cdot \left(c(l_i) \cap c(l_j) > 0\right) \qquad (9)$$

$\forall l_i \in \overline{L}, \forall l_j \in \underline{L}$, where $\left(c(l_i) \cap c(l_j) > 0\right)$ returns 1 when there is contour overlap and zero otherwise. Higher $\langle \overline{A}(H_{ceil}) \rangle$ means that the lines whose contours are overlapping in the XZ-plane cover a greater angular area. In Fig. 5b, we can see the value of $\overline{A}$ of the three different $H_{ceil}$, where it can be visually appreciated why $H_2$ is the best result.

In [4] we proposed an alternative method considering uniquely the number of contours overlapping. However, we found that the angular coverage method produces better results since they reward a consensus distributed in the scene instead of concentrated areas with many contours.

One of the advantages of working with scaled distances is that we can set reasonable valid ranges of heights to constrain the values of $H_{ceil}$. For example, we can set a default $H_{ceil}$ of 2.5 meters and a span of 2 to 3 meters to look for the $P_{ceil}$, which is very reasonable for indoor environments. If the problem has no solution between the valid range it could be due to clutter, undetected lines or absence of ceiling in the image. Then the algorithm goes on considering the default $H_{ceil}$. When the floor plane has not been found, the range of height values will not be constrained to the default values, which makes the system more prone to mismatches. In Fig. 6a the XZ-plane with the contours of both lower and upper lines from the case from Fig. 4b is shown. Those lines are plotted in 3D in Fig. 6b over the initial point cloud, so we can see how the lines extend beyond the FOV of the RGB-D camera.

### 4.1.3. *Corner definitions*

Line segments are the main piece of information we use to create layout hypotheses. However, we do not know at this point if the lines extracted are informative about the structure of the scene, that is, whether they come from actual intersections between structural planes (e.g. walls, ceiling, floor), or from other elements of the scene. In the literature there are many approaches to tackle this problem. For instance, [3] defines a corner when a minimal set of three or four line segments in certain orientations are detected. This requires having uncluttered environments where most line segments can be perfectly detected. However, in the real world, occlusions or bad lighting conditions may cause some contours to remain undetected. Other works such as [26, 27] tend to give more emphasis to vertical lines and the extension of their segments in their corner definition, which may be problematic for the same reason as before. In a Manhattan World, two line segments are enough to define a corner.

An example of a corner is shown in Fig. 7a, where the corner's 3D point, denoted as **C**, is the intersection of the floor plane ($P_{floor}$) and two walls ($W_x$, $W_z$) with respective planes $P_{W_x}$ and $P_{W_z}$. Another thing to bear in mind is the concept of *visibility* of the corners. The corner from Fig. 7a represents a *fully visible* corner, since all three line segments and corner

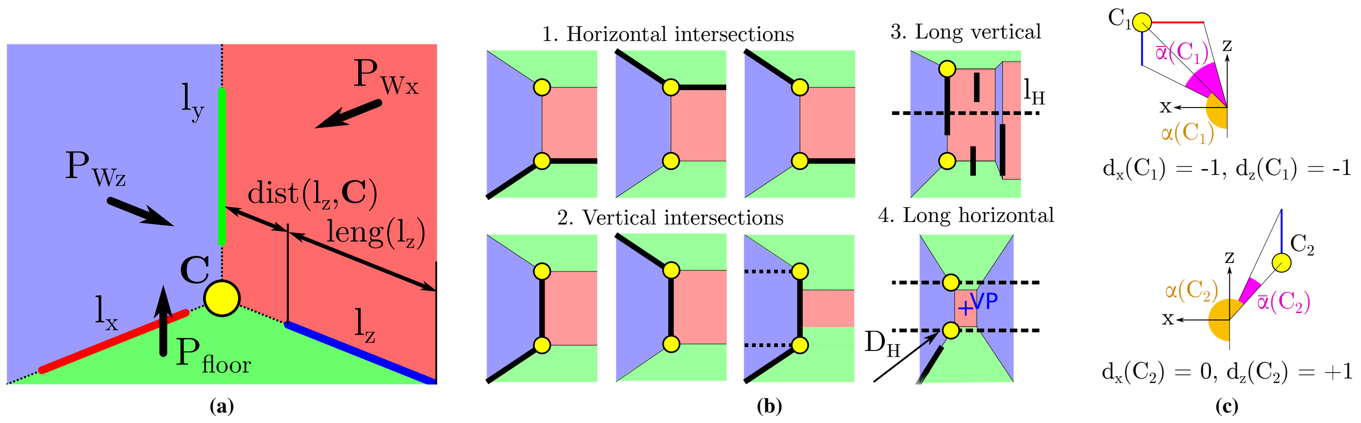


**Figure 7:** (a) Graphical definition of a corner $C$: the corner point **C**, its line segments ($l_x$, $l_y$, $l_z$) and the *dist* and *leng* functions used in our scoring method. (b) Four different types of corners we consider. Detected line segments as black thick lines and corresponding extracted corners as yellow circles. (c) XZ-plane view of two example corners showing their angles $\alpha(C_i)$, angle coverage $\overline{\alpha}(C_i)$ and directions $d_x$ and $d_z$ regarding the position of their line segments with respect to the plane axis.

point can be extracted. In a convex room (e.g. a rectangular room), all corners are fully visible. However, in a concave room (e.g. an L-shaped room), depending on where the camera is located there might be some corners *partially visible*, meaning one of the walls and thus its intersection line segments are occluded by the other visible wall (i.e. the corner point and two segments are visible). There are also *hidden corners*, which have the corner point occluded by the walls of the room, and thus only one or zero segments are observable. Box-shape methods consider that all corners in the scene are fully visible corners (if they are inside the image). Our method aims to find all type of corners. The hidden corners cannot be extracted directly from the image, and will be recovered with the layout estimation process (Section 4.2). In this section we will focus on the visible corners. We propose to use more relaxed requirements to define corners, using just two line segments, or one in special cases of long lines introduced as practical assumptions to compensate for misdetections. Then, we use a scoring function to select the most salient corners and favor their appearance in the layout hypotheses generation.

For the detection of visible corners, we consider four cases depending on the classification of the segments involved (Fig. 7b):

1. **Horizontal intersections** ($L_x - L_z$): These are by definition fully visible corners, formed by two lines in $x$ and $z$ respectively. If there is a $l_y$ passing through the intersection point **C**, the contour points of $l_y$ are scaled by assuming they share the same wall as $l_x$ (i.e. 3D plane $P_{W_x}$) or $l_z$ ($P_{W_z}$). If the scaled 3D contours of $l_y$ have heights between 0 and $H_{ceil}$ the vertical is included to improve the score of the corner.
2. **Vertical intersections** ($L_x - L_y$ **or** $L_y - L_z$): These intersections could represent partially visible corners or fully visible corners with an undetected line. As with the previous case, the segment $l_y$ is scaled to verify plausibility.
3. **Long vertical lines** ($L_y$): We select the end-points of long vertical lines, since those can represent wall-wall intersections. These are added to include cases of misdetections of horizontal lines. Only the ones crossing the horizon are considered as they are more likely to be wall to wall intersections instead of clutter. The projection of their topmost contour point to the $P_{ceil}$ or the bottommost one to the $P_{floor}$ is considered, depending on which one makes the scaled $l_y$ not exceed the height of the ceiling.
4. **Long horizontal lines** ($L_x$ **or** $L_z$): We consider the possibility of long horizontal lines to intersect with the horizon. These are added since sometimes there is no visible or detected corner at the farther end of a corridor or a big room. Horizontal lines are considered *long* if their length is over a threshold (we set 0.5 meters). To keep layouts of reasonable size we restrict the distance of intersection to a maximum of $D_H$ (in particular we set $D_H = 10$ m).

These simple types of corner intersections include all necessary cases to build a layout of any shape. We call *direction* ($d$) of the corner the position of their horizontal segments in the XZ-plane with respect to the corner point. For example, in the $x$ axis, a direction $d_x(C_i) = +1$ means that the corner $C_i$ has $l_x$ defined from the corner point to the positive direction of the $x$ axis. A corner with no $l_x$ has $d_x(C_i) = 0$. Similar definitions for $d_z(C_i)$. We also define the *angle of a corner* as the angle of the corner point $\alpha(\mathbf{C}_i)$ as well as the *angle coverage of a corner* $\overline{\alpha}(C_i)$ as the central angle of the arc between the minimum and the maximum angle among all the contour points from $l_x$ and $l_z$, i.e. $\overline{\alpha}(C_i) = max(\alpha(c(l_x) \cup c(l_z))) - min(\alpha(c(l_x) \cup c(l_z)))$. Two simple examples of corners showing these parameter values are shown in Fig. 7c. In the case of horizontal intersections, to evaluate if a line $l_y$ belongs to the corner, we check if the angle difference $\left|\langle\alpha(\mathbf{C}_i)\rangle - \langle\alpha(l_y)\rangle\right| < \theta_3$.

### 4.1.4. *Corner scoring*

In a natural scene highly populated with line segments, the amount of line intersections and thus corner detections can be very high. Therefore, when generating hypotheses it will be difficult to find the best ones. To avoid excessive amount of cor-

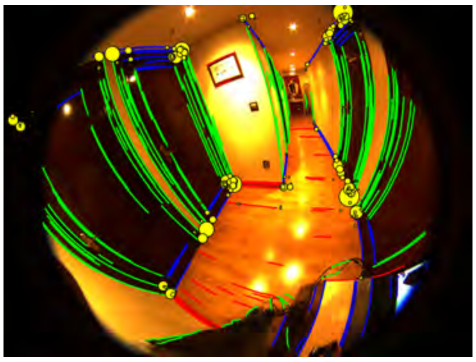


**Figure 8:** Relevant corners in the scene plotted over the fisheye image as yellow circles with diameter proportional to their score.

ners, a solution is applying thresholds, but it is hard to tune the parameters correctly to make it work in all cases. Instead, we perform a scoring of corners to keep those with positive score and make high scored corners more relevant in the generation of hypotheses.

In particular we want to reward corners formed by line segments of great length and low distance from the segments to the corner point. To examine length and the proximity between segment points, instead of using pixel distances, we reason in the 3D world with metric distances. Pixel distance is misleading, as it is affected by how far the points are from the camera, the perspective and the heavy distortion of the fisheye camera. Note that it is also difficult to deal with distances in the 3D world when there is no scale information available. With our system we integrate scale information in the process.

A corner $C$ is defined by a set of $N_l$ line segments, and its score $S_{C_j}$ depends on the number of lines and their respective score value $S_{l_i}$:

$$S_{C_j} = N_{l_j} \cdot \sum_{i=1}^{N_{l_j}} S_{l_i} \tag{10}$$

$$S_{l_i} = (leng(l_i) - dist(l_i, \mathbf{C})) \cdot (1 + \lambda(l_i)) \tag{11}$$

where $\lambda(l_i)$ is defined in (7), $leng(l_i)$ measures the length of the line segment in meters, $dist(l_i, \mathbf{C})$ measures the distance of the closest point of the segment to the actual intersection point $\mathbf{C}$ in meters (Fig. 7a). Note that $S_{C_j}$ computation includes a multiplication by $N_l$ to increase the score of corners supported by more lines. The line score for the corners in the horizon case is modified:

$$S_{l_i} = leng(l_i) \cdot (1 + \lambda(l_i)) \cdot (D_H > dist(l_i, c)) \tag{12}$$

After the extraction of corners we keep those with $S_C > 0$. To avoid redundant corners, we merge those which are close to each other, have similar angle coverage ($\overline{\alpha}(C_i) \approx \overline{\alpha}(C_j)$) and the same corner directions ($d_x(C_i) = d_x(C_j)$ and $d_z(C_i) = d_z(C_j)$). When doing the merging we pick the maximum score among all corners involved. In [4] we took the summation of the scores instead. However, we found that it rewarded too much specific areas crowded with line segments with no necessarily relevant corners. Finally, we assign a probability $\mathcal{P}$ to corners $C$ of occuring in the real world:

$$\mathcal{P}(C_i) = \frac{S_{C_i}}{\sum_{j=1}^{N_C} S_{C_j}} \tag{13}$$

where $N_C$ is the number of corners in the image. In Fig. 8 there is an example of the 100 most probable corners represented as yellow circles with radius proportional to their probability. In Fig. 9 there are corner results of a similar scene in a 3D point cloud, separated by the corner cases defined in Section 4.1.3.

## *4.2. Layout hypotheses generation*

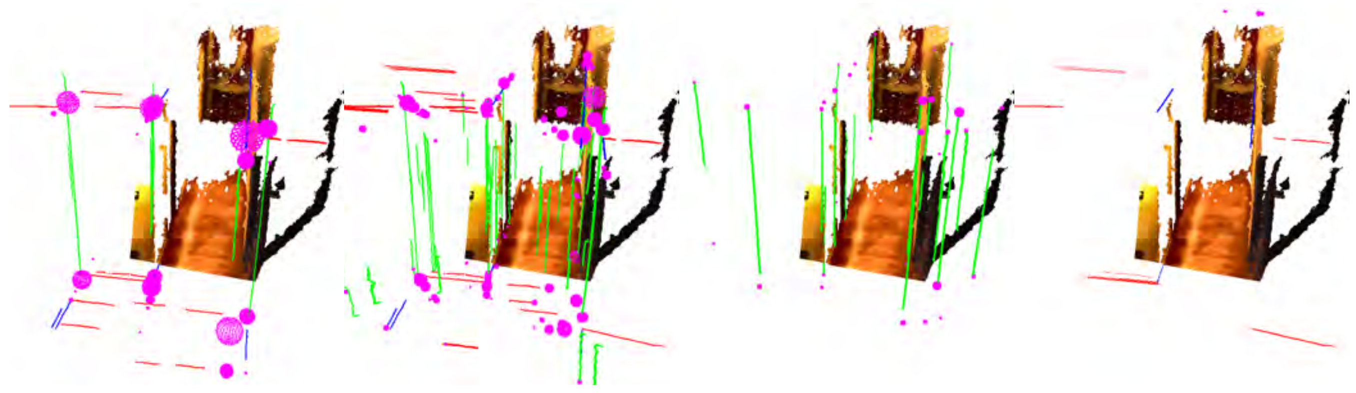
In Section 4.1 we defined corner points as intersections of three structural planes, which can be either intersections of two walls and floor or two walls and ceiling. Here we assume floor-ceiling symmetry, and thus, each corner point at the ceiling has its counterpart at the floor, and vice versa. Since we have recovered the pose of the scene ${}^{M}\mathbf{T}_F$ and the height of the ceiling $H_{ceil}$, we can compute the counterparts of each corner point. Therefore, from now on, when we refer to *corners* we implicitly refer to the pair of opposite corner points in ceiling and floor, and all the points in between that represent the intersection of two planar walls. To simplify notation, in this section we reason in 2D, in the XZ-plane, where the corner is a single point, independently if its detection comes from the ceiling or the floor. This equal consideration of ceiling and floor corners is advantageous, since in most indoor environments the level of clutter is higher in the lower part of the scene. Hence, having corners from the ceiling allows to provide results in difficult environments.

We define completely a layout with their corners and the height of the ceiling, $\mathcal{L} = \{C_1..C_N, H_{ceil}\}$. The walls of the layout, $W$, are implicitly defined as the planes connecting two consecutive corners with a height $H_{ceil}$ (thus, there are $N$ walls). Each individual wall is noted as $w_j \in W$ and they are in practice used as virtual lines with similar properties. To get our layout solution, we generate a set of hypotheses from where a last evaluation process will select the better one. To generate a layout hypothesis, we already have $H_{ceil}$ and we just need to select the corners among the extracted set of corners with $S_C > 0$. Once a hypothesis $\mathcal{L}$ has been generated, with the relative position of the camera to the scene ${}^{M}\mathbf{T}_F$ and the calibration of the system we have enough information to build the complete 3D reconstruction of the scene.

In this section, first we present in Section 4.2.1 the conditions for a layout to be geometrically valid. Then, we generate the layouts as described in Section 4.2.2. Additionally, we introduce a pre-filtering procedure of corner connections to verify its plausible association before start generating layouts (Section 4.2.3).

### *4.2.1. Conditions for a valid layout*

To generate hypotheses we do not impose any condition about the shape of the scene in order to provide valid solutions to any kind of indoor environment. However, we consider a layout valid when it satisfies the next conditions:



**(a)** Horizontal intersections **(b)** Vertical intersections **(c)** Long vertical lines **(d)** Long horizontal lines

**Figure 9:** Projection of the corners to the 3D point cloud as pink spheres with the line segments that form them, for each intersection case defined. Most relevant corners for the layout are outside the depth information range.

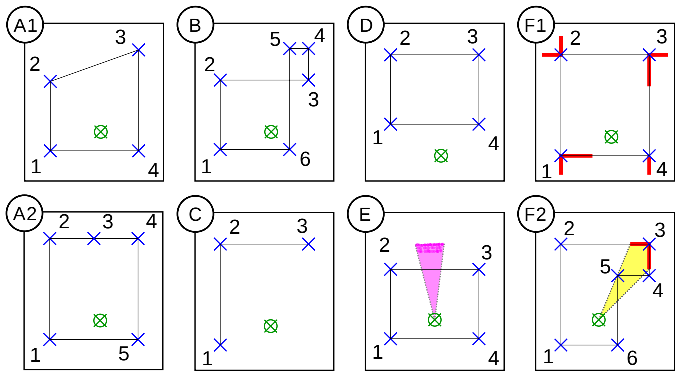


**Figure 10:** Examples of layouts that do not satisfy our conditions for a valid layout described in Section 4.2.1. In each case, corners are numbered blue crosses and the camera viewpoint is the green circled cross. In E the depth points and field of view in pink. In F1 and F2, line segments of the corners in red. In F2, the view of the segments of corner 3 in yellow.

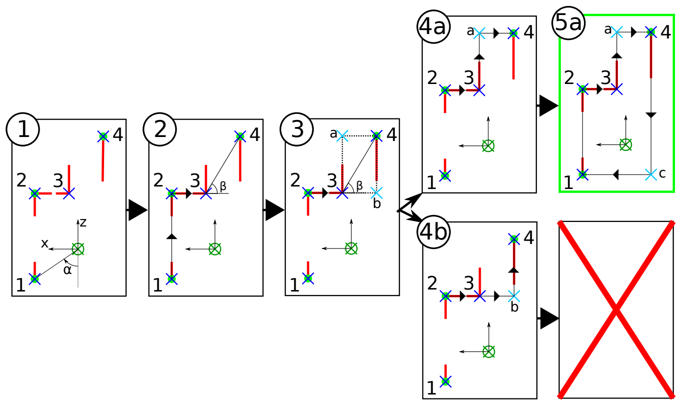


**Figure 11:** Example of layout generation in the XZ-plane given preselected corners from our set (in blue, with their respective red horizontal contours and a green circle when there is a vertical line). The camera position is the green circled cross. Detailed explanation of the procedure is provided in the text.

A. The walls must follow the Manhattan World convention: a wall directed in $\mathbf{m}_x$, must be followed by a wall directed in $\mathbf{m}_z$, and vice versa.
   A1. There must be no walls following other directions.
   A2. Two consecutive walls must not have the same direction.
B. The layout must not have any non-consecutive wall intersecting another.
C. The layout must be closed, i.e. the wall sequence must end in the same point it begins.
D. The camera must be inside the layout.
E. The layout must not contradict the information from the depth camera, i.e. there cannot be a wall in front of the given depth map.
F. The layout must not contradict the information given by the line segments of the corners, since they are considered directly visible.
   F1. Each wall connecting two corners must be *on* their corresponding line segments, if any.
   F2. Walls are opaque, so they must not be in front of any line segment since that would mean it is visible through the wall.

Note that, in Fig. 10 we show respective examples of layout proposals that do not satisfy each condition.

### 4.2.2. *Generation of layout hypotheses*

In the general case, our algorithm looks for a number of hypotheses by iterating following these steps (note that the explanation of this section can be followed using the graphical sample case from Fig. 11):

1. Using the probability from (13), we randomly choose a number of corners from the set to generate a hypothesis. As the view of the scene is not complete and we do not impose any condition about the shape of the room, the number of corners to select cannot be fixed, and therefore it must be randomly chosen every time a hypothesis is generated. We believe a reasonable number of corners to draw is between 2 and 5. In the example, we draw four

corners. The selected corners are ordered clockwise considering their angle $\alpha(C)$ as shown in the Fig. 11 (1).

2. The walls from the layout are defined joining every corner with the following one. The angle $\beta$ between corners is observed to verify if the walls are oriented according to the Manhattan convention (Fig. 11 (2)). If it is closer than an angular threshold to 0, $\frac{\pi}{2}$, $-\frac{\pi}{2}$ or $\pi$, then it is accepted as valid as it is (case between corners 1 and 2 and corners 2 and 3 in Fig. 11).
3. Else (angle between corners 3 and 4 in Fig. 11), two additional corners ($a$ and $b$) are created as shown in Fig. 11 (3).
4. In case two additional corners are defined, the generation of layout goes on with consecutive corners in separate branches, as the cases (4a) and (4b) in Fig. 11. At any point the line segments composing a corner can invalidate a layout generation branch (condition F). For instance, in (4b) the wall from corner 3 to corner $b$ goes in direction $-x$, but there is a line segment that defines corner 3 in direction $-x$ as well that goes in opposite direction. Solution (4a) matches the line segments from corner 3 perfectly.
5. Continue in every branch until the layout is closed (Fig. 11 (5a)) or the solution is invalid (Fig. 11 (5b)). It can be seen how the layout is completed by defining an additional corner $c$ as performed before, and no line segments contradict the wall distribution.

We repeat this process until a predetermined number of hypotheses are extracted. The number of hypotheses is a configuration parameter of the method, whose performance is evaluated in Section 5.3.1. The maximum number of attempts is also limited to prevent the system from getting stuck. A generated hypothesis has to satisfy the conditions mentioned in Section 4.2.1 or could be discarded. To verify conditions D and E we can treat the set of corners as a polygon and verify if the camera and depth points are inside it. For the depth points we set a threshold to determine the minimum percentage of points to be inside the polygon. To verify condition B we check the polygon does not self-intersect.

One of the keypoints of this method is that hidden corners can be estimated using the Manhattan assumption, even if there is no visible evidence of the presence of the corner in the image (e.g. in Fig. 11 corners $a$ and $c$ were not detected but its definition provides a valid closed Manhattan layout). This means that the algorithm can handle heavy occlusions and still provide coherent results. Besides, whenever the information from behind the camera point is not enough to provide closing points, we can assume the walls extend beyond the field of view towards the rear vanishing point (following $-z$) in order to keep our layout closed (condition C). For this operation we need to place additional corners at the horizon, at a previously determined distance ($D_H$ = 10m in our experiments), and watch if these solution do not break other rules. Note that when we introduce additional corners to perform the rear extension it is to keep the model consistent, but the final reconstruction only extends to where the field of view of the fisheye camera reaches.

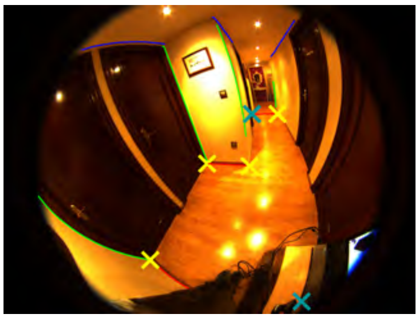
In Fig. 12 there is an example of a layout hypothesis from the scene from Fig. 8, similar to the one from Fig. 11. In Fig. 12a the original corners (in yellow) and the line segments that define them have been displayed in the image along with the additional corners (in light blue). In Fig. 12b the solution has been plotted over the XZ-plane. The resulting wall distribution colored is shown in Fig. 12c. In these labeled images each surface orientation is colored distinctly (Red-Green-Blue for surfaces with normals in $x$-$y$-$z$). As the XZ-plane is scaled and the $H_{ceil}$ have been estimated we can generate a 3D depth map of the scene (Fig. 12d). The depth map can be used to recover the 3D point cloud of the complete layout, as it can be seen in Fig. 12e.

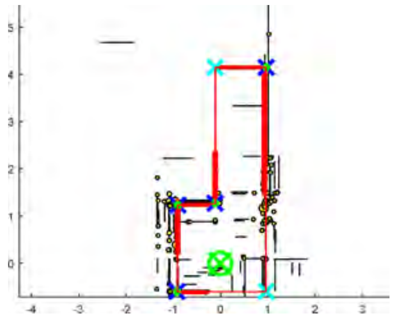
### *4.2.3. Pre-filtering of corner connections*

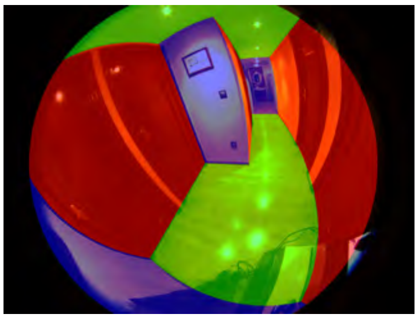
Given the considerations from Section 4.2.1, before start drawing hypotheses we pre-filter the corners that can go with each other in the same layout. This process was not performed in [4], but has proven to help reducing the computation time and providing better hypotheses. At the end of this process we obtain a matrix $\mathcal{M}$ of $N_C \times N_C$ which relates every corner to each other and indicates if they can be connected to each other with one wall, two walls (and thus introducing an additional corner) or cannot be connected by any mean. Besides, if they can be connected with two walls, we compute where should be the additional corner and if there is more than one option. Since we enforce closed layouts, we also verify if a closing strategy in the rear vanishing point can be created between two corners. Our filtering steps are the following:

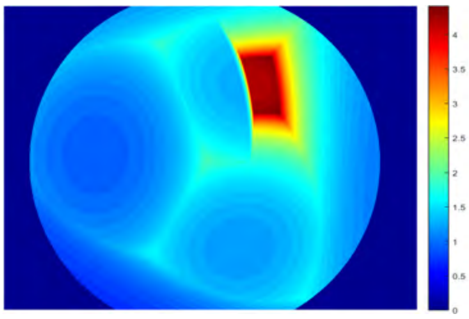
- Considering the quadrant the corners are located on the XZ-plane, there are some detectable cases of impossible corners. For instance, a corner $C_i$ in the $\{+x, +z\}$ quadrant cannot have $d_x(C_i) = -1$ and $d_z(C_i) = +1$ since that would mean $l_z$ would be occluded by the wall of $l_x$, which makes impossible for the segment $l_z$ to be visible and therefore detected. Corners 1 and 3 in Fig. 10 (F1) are impossible corners by this reasoning, and should not have been considered to generate hypotheses.
- Filter those corners close to each other, since we consider rooms with walls relatively large. To check this we compute distances between corner points and apply a threshold (we choose $H_{ceil}/10$).
- Verify if they can be connected clockwise. This is the order we use to generate the hypotheses, so it is only necessary to check corner connections one way.
- Compute the orientation of the walls with respect to the Manhattan directions to verify if they can be connected with a single wall or two (condition 1).
  - Single wall: if line segments are in the wall's direction, they must be facing each other (condition F1).
  - Two walls: no shared angle coverage between corners (condition F2). Get the two positions of the additional corners and discard those cases whose walls do not satisfy conditions F1 and F2.
- Look if there are closing strategies in case they are the first and last corners once ordered with $\alpha$. Check if their corner points have different signs in the $x$ coordinate (otherwise it would not satisfy condition D).

After the pre-filtering process, there might be some corners that cannot be connected to any other, which are discarded. With matrix $\mathcal{M}$, right after performing step 1 from the gen-


**(a)** Extracted corners


**(b)** XZ-plane with layout


**(c)** Labeled image


**(d)** Depth image

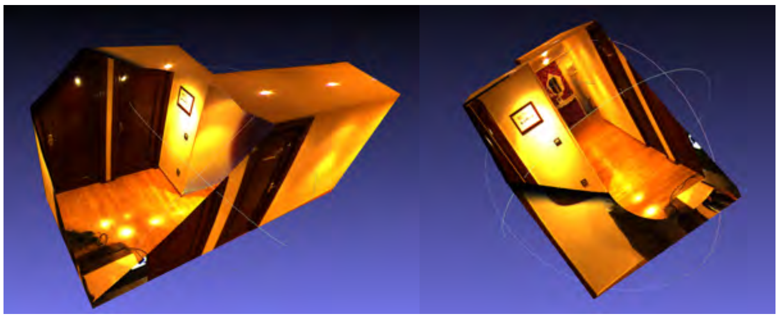

**(e)** 3D views of the scaled reconstruction

**Figure 12:** (a) Layout hypotheses example with its original corners in yellow with their line segments shown and the additional corners in light blue. (b) XZ-plane with the layout overlaid. (c) Colored wall-floor-ceiling distribution of the hypotheses. (d) Corresponding depth map of the hypotheses with scale in meters. (e) Different views of the corresponding 3D point cloud.

eration of hypotheses we can quickly discard those cases of pseudo-randomly chosen corners and pick others, instead of going through all the process avoiding useless steps until finding that some condition is not met.

### 4.3. *Evaluation of the Hypotheses*

The final stage of our method consists in choosing the best hypothesis from the previous stage according to a predefined criterion. For this task, we propose four possible evaluation criteria, whose performance is evaluated in Section 5.3.2. One of the criteria is an adaptation of an existing method (Orientation Map [3]) to our hybrid camera system, whereas the other three are novel approaches. One of these is a new criterion not included in [4] based on the *Angle Coverage* concept introduced in Section 4.1.2. In each evaluation criterion, we compute a score $\mathcal{S}_k$ for each layout $\mathcal{L}_k$, finally selecting as solution the one with higher value. The following sections will describe how we compute the corresponding scores:

***Sum of Scores (SS).*** We define the score of a hypothesis as the sum of scores of the corners that have been used to generate it.

$$\mathcal{S}_k^{SS} = \sum_{i=1}^{N_k} S_{C_i} \tag{14}$$

where $N_k$ is the number of corners in $\mathcal{L}_k$. Notice that the additional corners defined to generate Manhattan layouts have score zero.

***Sum of Edges (SE).*** The polygon defined by the corners of the hypotheses as vertices can be drawn on the XZ-plane in order to choose the hypothesis which overlaps the most with the observed contours, i.e. the layout $\mathcal{L}_k$ with a set of walls $W_k$ such that the sum of contours of the lines in the image $l_i \in L$ that overlap with any wall $w_j \in W_k$. Hence, we select the layout with highest score $\mathcal{S}_k^{SE}$, computed as:

$$\mathcal{S}_k^{SE} = \sum_{i=1}^{N_L} c(l_i) \cdot \Big(c(l_i) \cap c(w_j) > 0\Big) \tag{15}$$

***Angle Coverage (AC).*** In this case we draw the polygon in the XZ-plane and we compute the angular coverage of the layout ($\overline{A}(\mathcal{L})$) similar to the process of Section 4.1.2, but considering all line segments and wall lines instead of floor and ceiling lines. Mathematically:

$$\overline{A}(\mathcal{L}_k) = \bigcup \Big(\overline{\alpha}(l_i) \cap \overline{\alpha}(w_j)\Big) \cdot \Big(c(l_i) \cap c(w_j) > 0\Big) \tag{16}$$

$$\mathcal{S}_k^{AC} = \langle \overline{A}(\mathcal{L}_k) \rangle \tag{17}$$

$\forall l_i \in L, \forall w_j \in W_k$. The hypothesis $\mathcal{L}_k$ with the highest angular coverage value $\langle \overline{A}(\mathcal{L}_k) \rangle$ is selected with this criterion.

***Orientation Map (OM).*** It requires to build a reference image called *Orientation Map* [3], which is an image whose pixels encode the believed orientation given the line segments for perspective cameras. To build that image we create a set of overlapping perspective images from the fisheye image, apply the orientation map algorithm from [3] in each one of them and finally stitch them back together to form an omnidirectional orientation map that we call $\mathcal{I}^{OM}$. The evaluation consists in selecting the layout hypothesis with better fitness between pixels with the same orientation. To compute that fitness, we generate for each hypothesis a labeled image $\mathcal{I}^{LB}$ (such as the one Fig. 12c) where each color represents one of the three orientations $M = \{\mathbf{m}_x, \mathbf{m}_y, \mathbf{m}_z\}$. We do that with the score:

$$\mathcal{S}_k^{OM} = \sum_{m \in M} \sum_{u=1}^{I} \sum_{v=1}^{J} \mathcal{I}_k^{LB}(v,u,m) \ \& \ \mathcal{I}^{OM}(v,u,m) \qquad (18)$$

where $I$ and $J$ are respectively the width and height of the images $\mathcal{I}$, and operation & returns 1 when the pixel value is equal.

### 4.4. Scaling of hypotheses

The layouts can be generated and evaluated as described above without scale information. However, the thresholds and parameters are then harder to tune (e.g. no valid height of the room estimation), and depth cannot be used to discard incoherent layouts. Thus, the normal execution should include floor detection and scaling from the beginning in order to obtain better results. Nevertheless, in the cases the floor cannot be detected, we include a method to scale the layouts once generated, so that the height range and depth information conditions can be verified.

To perform layout scaling from an non-scaled hypothesis $\hat{\mathcal{L}}_k$, first we compute the labeled image $\mathcal{I}_k^{LB}$. Then, we generate a separate label image for each of the walls $w_j \in W_k$, where pixels outside the wall are valued zero, denoted $\mathcal{I}_{w_j}^{LB}$. For each plane $P_i$ extracted in Section 3.4.1, we can compute an equivalent labeled image $\mathcal{I}_{P_i}^{LB}$, whose pixel values inside the plane encode the orientation of the plane, and pixels outside are valued zero. Then we look for the plane with the highest overlap with the hypotheses label to provide the scale, computing an score similar to (18):

$$\mathcal{S}_{i,j} = \sum_{m \in M} \sum_{u=1}^{I} \sum_{v=1}^{J} \mathcal{I}_{P_i}^{LB}(v,u,m) \ \& \ \mathcal{I}_{w_j}^{LB}(v,u,m) \qquad (19)$$

The labeled wall/labeled plane pair $(j, i)$ with highest overlap $\mathcal{S}_{i,j}$ will be used to provide the scale. In particular, we have the distance to the origin $X_0^i$ from Section 3.4.1 and we can compute the equivalent with the non-scaled hypothesis, $\hat{X}_0^j$. The quotient $s = X_0^i / \hat{X}_0^j$ is the scale, and the corner points or obtained point cloud can be simply multiplied to that scale in order to get the full scaled 3D reconstruction, i.e. $\mathcal{L}_k = s\hat{\mathcal{L}}_k$.

## 5. Experiments

In this work, we use a novel camera system with fisheye and depth image. Many datasets for indoor layout retrieval are usually based on conventional images, but not so many on omni-images, and none combining them with depth. For the experimental evaluation we have collected our own set of images with two different devices (Fig. 13):

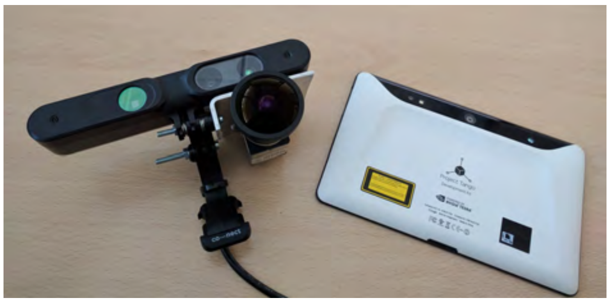


**Figure 13:** The two devices used to collect the data for our experiments. On the left, an RGB-D camera (ASUS Xtion Pro Live) with an adjacent fisheye camera. On the right, the Google Tango Development Kit.

- **Conventional RGB-D system with fisheye camera**: A hybrid camera system built and calibrated by ourselves [38]. With this system we have a dataset with 70 image pairs from indoor scenarios, including 23 from corridors/entrances, 15 from four different bedrooms, 4 from a bathroom, 12 from two living rooms, 4 from a kitchen and 12 from two small cluttered rooms. We have manually labeled the 70 images of the dataset to provide a per pixel label of the three main classes (walls in $\mathbf{m}_x$ or $\mathbf{m}_z$ and floor/ceiling) with the labeled image $\mathcal{I}^{GT}$.
- **Google Tango**: A tablet for developers with built-in depth sensor and fisheye. The Tango technology is now available in commercial phones from well known brands (e.g. Lenovo, Asus). We have taken several images from similar environments as the previous device to test applicability of the method with commercial systems.

Quantitative detailed analysis is provided with the larger dataset from the first device. Unless noted, this is the dataset we use in our experiments. First, we provide implementation details (parameters, time consumption) in Section 5.1. In the following sections, we analyze the performance of the corner extraction (Section 5.2) and the layout estimation (Section 5.3), which are the most important parts of our algorithm. Additionally, we provide some insight about using the proposed camera configuration in Section 5.4, and more results using the Google Tango dataset in Section 5.5.

### 5.1. Implementation details

Here we provide implementation details such as threshold values or computation time, to illustrate the performance of the method and facilitate reproducibility. This method was implemented in Matlab in a computer with processor of 3.4 Ghz, not using any GPU. The method follows the flow diagram represented in Fig. 2, while the datasets were captured using software implemented in ROS [49]. The calibration was performed using [38] and line extraction with [46].

#### 5.1.1. Parameters and thresholds

In the depth processing, we considered a voxel grid size of $V_{size} = 0.02$ meters, and considered planes and clusters of more than 50 points or 10% of the size of the cloud. To validate a floor detection, there shall be no 10% of the points from the point cloud below the floor plane, with a threshold of $3 \times V_{size}$. Lines are classified according to a Manhattan directions with a threshold of $\theta_1 = \pm 15°$, since being too strict could discard lines that are actually useful since not all constructions accurately satisfy Manhattan assumption, and also there might be noise in the measurements. To associate lines to 3D lines, however, we are more strict and use a $\theta_2 = \pm 5°$. We find corners at the horizon when they are based from horizontal lines of at least 0.5 meters, and limit the distance of the horizon to $D_H = 10$ meters. We additionally discard horizontal lines that are close to the horizon and thus their projection to floor or ceiling is farther than $D_H$. Corners are merged when they are within a radius of 0.2 meters and more than 80% of intersecting angles $\alpha(C)$. Vertical lines $L_y$ are included in corners with $\theta_3 = \pm 2°$. In the layout generation, minimum wall length is 10% of the $H_{ceil}$, and deviation with respect to. Manhattan directions of $\pm 10°$. To discard layout hypotheses with depth information, 80% of the cloud must be inside the 3D layout.

### *5.1.2. Time analysis*

In this work, the experiments were performed in single-image and offline. Consequently, we used the images at full resolution: fisheye image of $2560 \times 1920$ pixel and depth image of $640 \times 480$ pixel. While there was no major effort in optimizing the implementation to make it able to run in real-time, it is interesting to see how the system behaves regarding time performance. Table 1 shows the total time consumption of the method in our implementation, and a breakdown of the time consumption of each stage including mean and standard deviation. It can be observed that the two most time-consuming stages are the line extraction and depth processing, whose high value is direct consequence of the high resolution of the fisheye image ($2560 \times 1920$) and the point cloud ($640 \times 480$ with $V_{size} = 2$cm). Nevertheless, more efficient programming languages and libraries (e.g. C++ and PCL [48]) could improve performance up to real-time. The same argument could be made for the VP extraction, since it requires to filter the cloud and obtain the normals previously, which is by far the most expensive part (the optimization itself is very quick). If we count the time spent to perform the processes that end up with a set of corners detected and scored, the times range from 4.67 to 11.26 seconds with an average of 7.32 seconds. It shows a large variance (standard deviation of 1.47s), which shows the performance of the corner extractor depends on the complexity of the scene (i.e. amount of lines and planes).

About the layout hypotheses generation, it is interesting to compare the performance of the current implementation pre-filtering the corners (Section 4.2.3) with respect to our previous work [4], where we simply analyze connectivity between corners as they were selected. In average, pre-filtering makes the method about 4 times more efficient, reducing considerably the standard deviation as well. Notice that both approaches keep selecting corners from the set until a predetermined number of 50 valid hypotheses are generated, or a maximum number of iterations is met. The performance of this part depends heavily on the scene, the corners extracted and how are they distributed in the scene. Then, some cases may take a long time to recover 50 valid hypotheses, producing large standard deviation. This can be improved in both cases reducing the number of maximum iterations: considering the selection of corners is score-based, it might not be necessary to reach 50 valid hypotheses. In the evaluation stage, we did not include Orientation Map criteria since it is much slower than the others (a more extended comparison is shown in Table 3). The other evaluation criteria show that they are very fast, with almost non-existent standard deviation.

**Table 1:** Breakdown of computation time of each stage including mean, $\bar{\mathbf{x}}$, and standard deviation, $\sigma$, in seconds; and the percentage over the total time (%). We have included the hypotheses generation implementation from [4] that did not include pre-filtering for comparison purposes.

| **Stage of the method** | $\bar{\mathbf{x}}$ | $\sigma$ | % |
|---|---|---|---|
| Line extraction (3.2) | 2.060 | 0.288 | 23.907 |
| Vanishing point (3.3) | | | |
| - Voxel and normal estimation | 0.612 | 0.286 | 7.100 |
| - VPs computation | 0.101 | 0.052 | 1.169 |
| Depth processing (3.4) | 1.907 | 0.836 | 22.137 |
| Line classification (3.5) | 0.755 | 0.072 | 8.764 |
| Corner extraction (4.1) | | | |
| - Line projection / $H_{ceil}$ | 0.544 | 0.278 | 6.317 |
| - Corner detection and scoring | 1.342 | 0.790 | 15.578 |
| **Total time for corner extraction** | 7.321 | 1.471 | 84.970 |
| Hypotheses generation (4.2) | | | |
| - Pre-filtering (4.2.3) | 0.587 | 0.097 | 6.8814 |
| - 50 Hypotheses (4.2.2) | 0.659 | 2.237 | 7.651 |
| Hypotheses generation in [4] | | | |
| - 50 Hypotheses | 4.288 | 5.753 | – |
| Hypotheses evaluation (4.3) | | | |
| - SS + SE + CA | 0.049 | 0.008 | 0.564 |
| **Total time** | 8.616 | 2.867 | 100 |

## *5.2. Corner extraction*

In this section we analyze the capacity of the method to extract the relevant corners for the layout estimation, but also its limitations. Ultimately, the success of the system depends on the good extraction of corners. While our layout estimation process allows to introduce additional undetected corners, they are placed between corners that had been extracted beforehand.

In our method, we extract and rank the corners depending on their score, but for the layout extraction we only keep the 100 better ranked corners. The first experiment analyzes how well the corners found with our method correspond to the real world corners. To perform this experiment, we have manually annotated the number of corners that should be found for all the 70 images in the dataset, in order to provide the best layout solution, obtaining 206 corners in total. Then, we visually inspected if these corners are actually among the 100 best ranked corners. The ratio of number of corners found over number of corners to find is of $191/206 \rightarrow 92.7\%$. All the important corners were

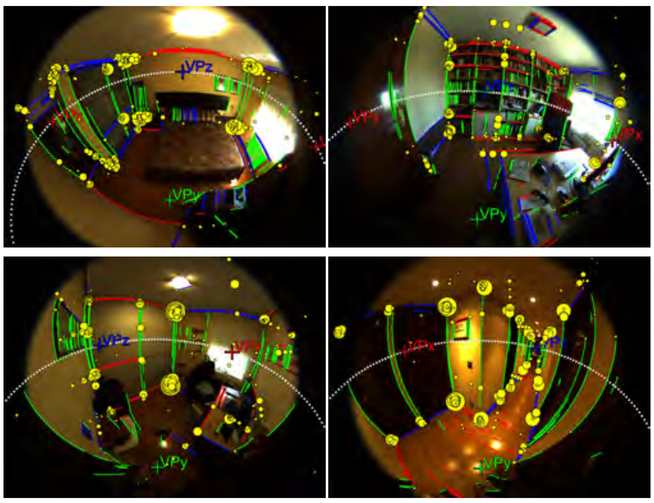
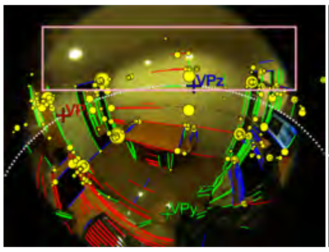


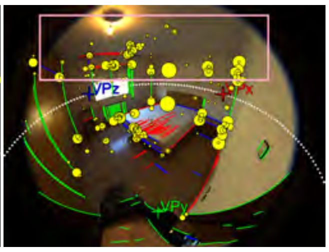
**Figure 14:** Examples of corner detections, represented by yellow circles, in four different scenes. In these examples all the important corners have been found, plus some outliers.

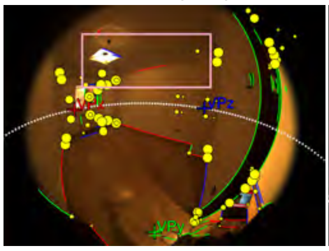
found in 58 of the 70 images. Taking into account that even in cases where relevant corners are missing they could still be recovered as intermediate hidden corners in the layout hypotheses generation, we consider these high percentages a good indicator of performance: in most cases the relevant corners are found, and even when they are not, the algorithm may be able to complete the task. In Fig. 14 there are some cases where all the important corners have been detected. We can also see how very often several corner detections appear around the same real world corner. This multiplicity is due to similar detections from close lines corresponding to different objects (e.g. embellishments). There are also some outliers (i.e. corner detections not corresponding to real world corners) that the layout estimation process has to overcome.

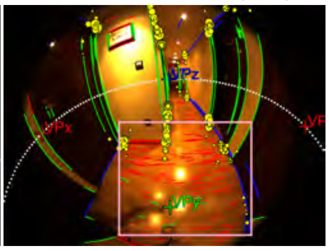
Next, we analyze the main difficulties and causes of failure that the corner extraction problem has, also showing some examples. Given that the basic elements to find corners are the lines, missing some of them may be critical. The line extraction method requires edges to be detected properly, which may not happen when there is low contrast in the image, e.g. because of non-existent color shift, bad lighting conditions or motion blur. Some of the reasons that harmed the results in some experiments are the following: in Fig. 15 (a-c) most lines in the ceiling were not detected and thus, the height of the ceiling could not be obtained properly and some relevant corners are missing. Particularly, in Fig. 15 (c) the illumination from the lamp itself casts a shadow that resembles a wall-ceiling intersection. The opposite problem arises sometimes as well, i.e. lines coming from textures (Fig. 15 (d)) or objects irrelevant for the task (Fig. 15 (e)) provoke accumulation of misleading corners.

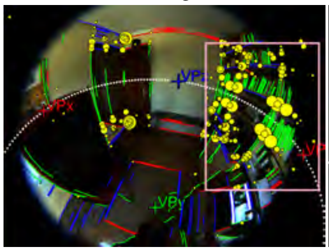
The ceiling plane may be wrongly obtained when fortuitous scene configurations and line distributions occur. For example, in Fig. 15 (f) the rectangular rug resembles the rectangular shape of the ceiling and thus the $H_{ceil}$ is such that makes the contour of the ceiling and the rug overlap in the XZ-plane. In Fig. 15 (g) the wardrobe has parallel lines that deceive the ceiling plane extraction as well. This is a problematic issue, since it affects the layout proposals that combine corners from floor and ceiling (i.e. if the ceiling plane is not right the corner intersection in the ceiling will not be exactly on top of the intersection in the floor). Since we introduce depth information of the process, the ranges for a valid $H_{ceil}$ are restricted to common ones (e.g. from 2 to 3 meters). In Fig. 15 (h) we can see the previous case by removing the input from the depth camera: the lines from the furniture and posters create a situation where the best ceiling plane solution is very inaccurate. Providing scale and restricting the measurements to natural ranges produces that even when the ceiling plane is not properly found, the value it takes is not very far of the real solution. In average, the ratio of success of finding a ceiling plane within a few centimeters error is almost 80%. Comparing this result with our previous method without Angle Coverage [4], the success ratio of the newer ap-

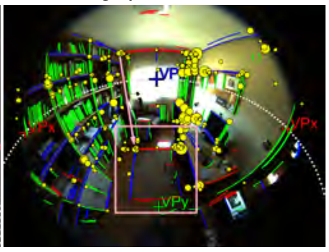

**(a)** Bad lighting

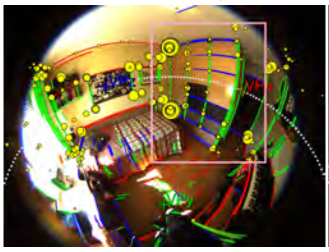

**(b)** No color shift/bad lighting


**(c)** Misleading shadow


**(d)** Highly textured surface


**(e)** Lines from objects


**(f)** Bad ceiling detection with object


**(g)** Bad ceiling detection with parallel lines

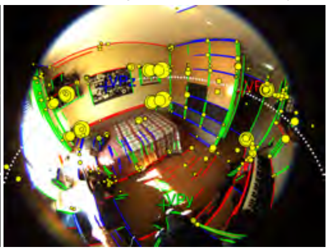

**(h)** No depth information producing bad ceiling detection

**Figure 15:** Examples of corner detections damaged by scene conditions. The pink rectangle points at the region where failure happens.

proach is about 15% higher. Moreover, comparing each case from the dataset, the new $H_{ceil}$ estimation method improves in 54/70 cases, taking in consideration the accuracy with respect to real values and the matchings of lines from ceiling and floor, showing the proposed method better performance.

Despite the aforementioned failure cases, most important corners are generally well extracted in our experiments. The majority of these problems are derived from the line extraction method of our current implementation and not the method itself. More sophisticated approaches of line detection in the vein of [50] could be used to improve the results. Additionally, some filtering methods could be used to remove textures and highlight borders (e.g. [51]), even deep learning methods have been used to detect only structural edges and ignore those from other objects or clutter [21]. However, this line of research was out of scope for this work, and instead we focus on developing a layout estimation method robust enough to overcome the fact that not all corners are always detected.

### *5.3. Layout estimation*

In this section, we analyze quantitatively the results of the system regarding layout estimation: from hypotheses generation to the evaluation and the final result. For this we use the dataset of 70 images for which we have labeled the ground truth. Our ground truth is the labeled images such as the one from Fig. 12c, where each color represents a layout surface of different orientation. Since only the structural information of the scene is to be extracted, in the tagging we ignore all the objects unless they cover entire walls (e.g. wardrobes or bookshelves). We only extract single room layouts, meaning that open doors are ignored during tagging phase as well. The metric employed is the percentage of pixels correctly tagged over the totality of pixels from the ground truth, which we call *Pixel Accuracy* (PA). With that metric, we analyze the quality of our solution depending on the number of hypotheses drawn and the evaluation criterion.

#### *5.3.1. Number of hypotheses*

This experiment analyzes how the PA changes depending on the number of hypotheses to draw with the four evaluation criteria presented. The objective of this experiment is to observe the behavior and determine how many hypotheses we need to have the best results. We have registered the mean PA obtained from 5 to 50 hypotheses by intervals of 5 and from 50 to 200 by intervals of 10. The resulting graph is shown in Fig. 16. At a glance we can see two distinct trends: The Sum of Scores (SS) and Sum of Edges (SE) evaluation criteria present lower score and a small decline through iterations, whereas Angular Coverage (AC) and Orientation Map (OM) rise briefly at the beginning until they reach a steady maximum.

However, in all four cases we can note that the variation of PA through iterations is negligible. This is a consequence of the good performance of the detection and scoring of the corners, which makes higher scored corners more likely to appear in layout hypotheses. In many cases the highest scored corners are the real world corners that we are looking for. Thus, a small

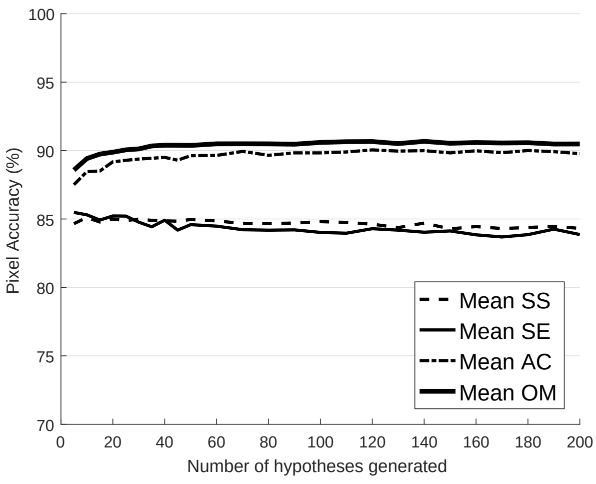


**Figure 16:** Pixel accuracy over the number of hypotheses generated.

number of hypotheses is likely to already provide a good result. Other cases may have a more complex corner distribution, which leads to more variety of layout hypotheses among which the SS and SE evaluation criteria may find one that fits better the criterion while being not very accurate. On the other hand, AC and OM prove to be better for the task, since they look for the best distributed consensus in the scene. Therefore, these criteria tend to improve with a larger variety of hypotheses.

The best number of hypotheses to draw will depend on the criterion. For SS and SE, lower number of hypotheses improves the results, but at least a few should be required (otherwise there is a risk of getting oddly-shaped layouts). Thus, 5-10 hypotheses seem reasonable. For AC and OM, on the other hand, the PA rises until 30-40 hypotheses, and the improvements afterwards are marginal. In all cases, we choose a very reduced number of hypotheses, substantially less than other similar works [29].

#### *5.3.2. Comparison of evaluation criteria*

To provide a more meaningful discussion about the evaluation criteria, the mean PA displayed in Fig. 16 is not enough. In Fig. 17, there is another boxplot-type graph showing the distribution of pixel accuracy in the 70 images with the four evaluation criteria at 50 hypotheses. For each column we show the mean (black line), median (black dotted line), standard error of the mean (SEM) at 95% of confidence (dark rectangle), and standard deviation (SD) (bright rectangle). The values of the mean and median are shown in the Table 2. Additionally, each individual result of the 70 images is also scattered on the graph for visualization purposes. Analyzing only the left part of the graph (the general case, with depth information), we can see how SS and SE are able to tag correctly a median of 86% and 85.7% of the pixels in the image respectively. Both AC and OM reach over 90%, particularly 90.3% and 91.5% of PA. While all criteria perform well, the AC and OM are clearly the best. The OM has the better scoring overall, but the AC has smaller standard deviation and less outliers.

In Fig. 18 and Fig. 19 there are several examples showing

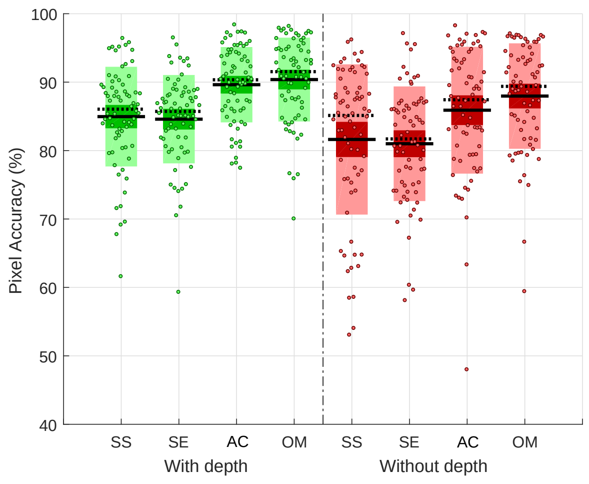


**Figure 17:** Boxplot of the results using the four evaluation criteria with the set of 70 images. The graph is divided in results using depth (on the left, in green) and without depth (on the right, in red). For all cases the black line marks the mean value, the black dotted line the median value, the dark rectangles are the standard error of the mean (SEM) with 95% of confidence and the bright rectangles the standard deviation (SD). The individual values per image are also scattered over each column.

**Table 2:** Mean pixel accuracy of the system with and without depth information (%).

| **Criterion** | **With depth** | | **No depth** | |
|---|---|---|---|---|
| | Mean | Median | Mean | Median |
| SS | 84.96 | 86.05 | 81.62 | 85.13 |
| SE | 84.59 | 85.73 | 81.00 | 81.70 |
| AC | 89.63 | 90.34 | 85.90 | 87.43 |
| OM | 90.38 | 91.53 | 87.97 | 89.38 |

results for each evaluation criterion. In particular, Fig. 18 show corridor scenes (with failure cases), which usually have complex structure and may require to recover six corners. The planes from depth are also displayed, to show how most of these corners are recovered outside the depth field of view, and even some that are *hidden corners* (see second row). While most cases are successful, there are several failure cases representative of the evaluation criteria. For instance, SE is unable to get the full length of the corridor as there are edges on the parquet that fit better the criterion. SS chooses the hypothesis with high valued corners disregarding any other information. The third scene has an open door that misleads OM and eventually causes wrong layout recovery. In Fig. 19 the scenes are more cluttered and thus failure cases are more frequent and harder to analyze. Nevertheless, there is a clear tendency: SS and SE tend to select overly complex hypothesis and are more likely to fail than AC and OM.

However, accuracy is not the only factor to compare methods, so we extend the experiments to test efficiency in terms of computation time. In Table 3 there is a breakdown of the mean times in our current implementation. The first three methods

**Table 3:** Comparison of computation time of each stage of the evaluation for each criterion in our current implementation (in milliseconds).

| **Stage of evaluation** | **SS** | **SE** | **AC** | **OM** |
|---|---|---|---|---|
| Generate orientation map | – | – | – | 18000 |
| Generate 1 labeled image | – | – | – | 50 |
| Evaluate 1 hypotheses | 0.05 | 0.4 | 1.8 | 2.5 |
| Total (1 hypotheses) | 0.05 | 0.4 | 1.8 | 18052.5 |
| Total (50 hypotheses) | 2 | 20 | 90 | 20625 |

(SS, SE and AC) only require simple operations, and thus, are extremely fast (less than 2 milliseconds). On the other hand, the OM is very slow in comparison. Just to generate the map, assuming we have the lines and vanishing points extracted, it takes around 18 seconds. Then it needs to compute the corresponding labeled image per hypothesis, in order to find the best fitting one. To save time we resize the orientation map and labeled images by a scale of 0.25. Then, generating labeled images takes about 0.05 second/hypothesis and selecting the better one takes 2.5 millisecond/hypothesis. Thus, for example, evaluating 50 hypotheses would take about 0.002 seconds to the SS, 0.02 seconds to the SE and 0.09 seconds the AC. The OM method would take $18 + 0.05 \times 50 + 0.0025 \times 50 = 20.625$ seconds. The difference between OM and the other three evaluation criteria is of several levels of magnitude. Thus, considering the small PA value shift between AC and OM, when time is a requirement, AC is much better for the task.

Note that this comparison is performed within the context of our scaled layout recovery method, and the results may not concur when applied to other approaches. For example, AC benefits from having wide FOV and thus it may not perform as well in conventional images, or SS uses Scores particularly defined in this work that require scaled values for computation. Moreover, the criterion in the evaluation stage can be replaced without affecting the rest of the pipeline, allowing to use new criteria as the state of the art evolves (e.g. in [52] they use a deep learning method to extract surface normals [53]).

### 5.3.3. *Performance under different types of scenes*

A breakdown of the results depending on the type of room is provided in Table 4. In general we have experienced better performance in environments where structural lines can be easily seen. For example, corridors often have less objects occluding the important lines. On the other hand, corridors have often more complex shapes. Our method is able to overcome complex shapes in most cases as the high scores in corridors show. In Fig. 18 there are some examples that support those good results.

The rest of the rooms are very scene dependent, and it is harder to establish any correlation in type of room and results. As mentioned in Section 5.2, the results depend on the specifics of each scene, including parameters such as illumination. In Fig. 19 there are some examples of layout retrievals in non-corridor cluttered rooms. The first is a bedroom scene with complex shape, which was recovered successfully and completely closed with AC and OM. The second and third one are

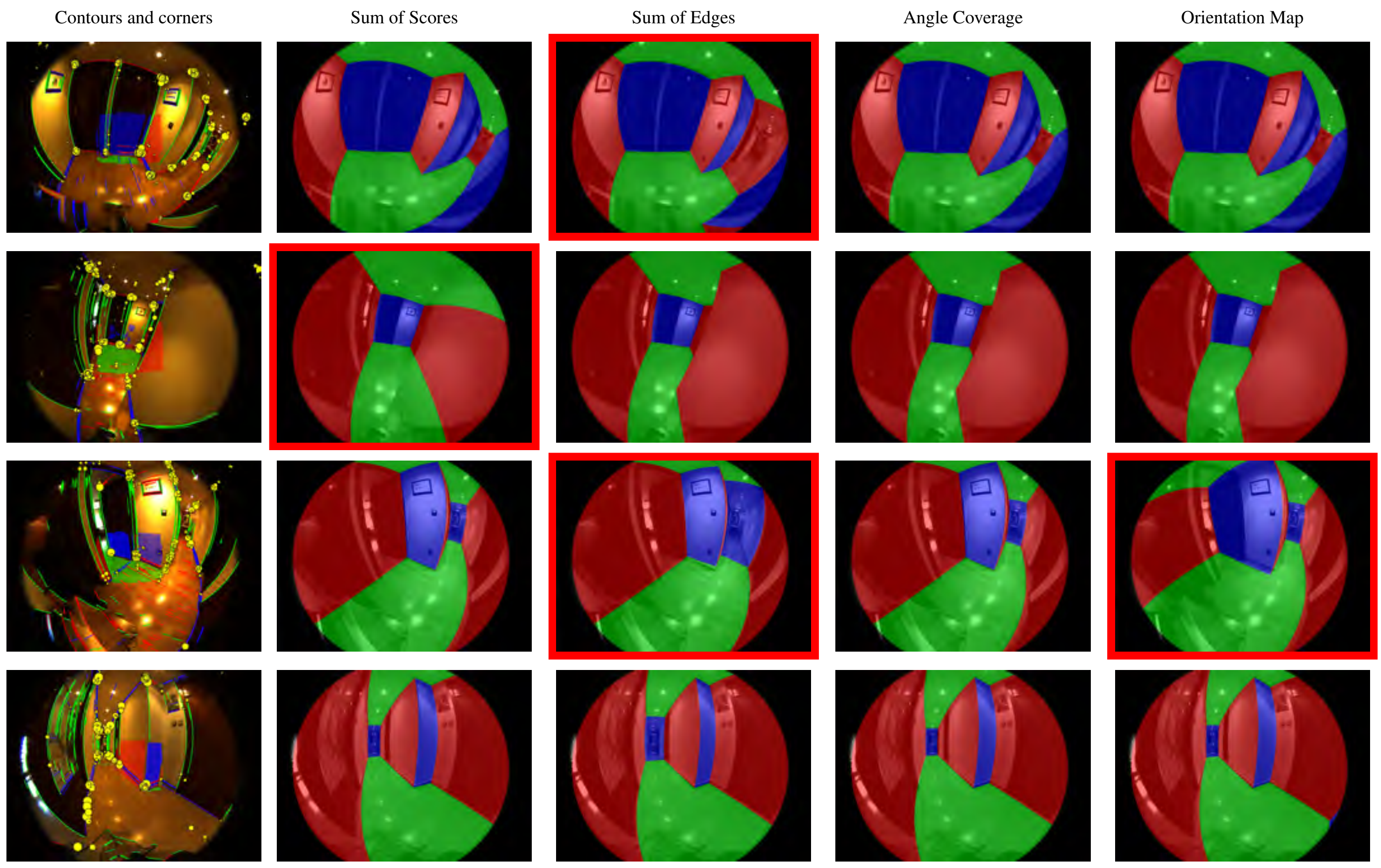


**Figure 18:** Examples of results from corridor scenes of our set with best layout proposal for each evaluation criterion. Those results with red frame are failure cases. Note that in all these results the shapes of the layout are complex, with 6 wall structures.

rooms that include several major occlusions involving chairs, tables, ironing boards. However, as shown in the results from Table 4, bathrooms and kitchens are the most problematic. They are usually crowded with objects and cabinets, even mirrors in the case of the bathroom, which are often problematic in any computer vision algorithm. The two last examples from Fig. 19 are from these types of scenes, with an observable high failure rate and inaccuracies. In particular, the kitchen has a countertop and cabinets that covers the entire left wall, making extremely difficult to recover the structural layout with any computer vision method.

**Table 4:** Mean pixel accuracy depending on the type of scene tested (%).

| **Room** | **SS** | **SE** | **AC** | **OM** |
|---|---|---|---|---|
| Corridor | 89.52 | 85.92 | 93.53 | 91.8 |
| Bedroom | 84.63 | 85.57 | 87.64 | 90.74 |
| Bathroom | 80.10 | 83.87 | 86.84 | 86.42 |
| Living Room | 85.03 | 85.70 | 89.03 | 90.18 |
| Kitchen | 73.02 | 79.77 | 82.87 | 87.61 |
| Other | 82.52 | 82.13 | 88.29 | 89.78 |

### *5.3.4. 3D scaled reconstruction of scenes*

In Fig. 20 there are some examples of 3D reconstructions obtained with our method. We show the fisheye with the depth information that we use as input of the system to visualize how much the depth has been extended. It can be seen that the system is able to reconstruct not only 'box-shaped' rooms, looking at the corridor or bedroom scenes. These results are scaled with the depth information provided, so in a single shot our system is able to get the whole scene at once. We believe this information could be valuable for many tasks.

We have to note that in all cases this is an estimation of the layout, but the only information that is fully reliable all the time is the one that comes from the depth information. Our layout solution can be merged with the initial depth so we can actually use both sources of data at the same time to our advantage. The depth image provides a safe zone where we know for certain what is in front of the camera, but we also have spatial context of the room we are in, enabling many possibilities of higher level reasoning that extends what a conventional depth camera can do, without diminishing its advantages.

## *5.4. Advantages of the camera pairing*

We explore the benefits of using our camera system compared to merely using a fisheye camera. The main one is that the scale of the scene would be lost without the depth camera. Additionally, we repeated the experiments removing all depth

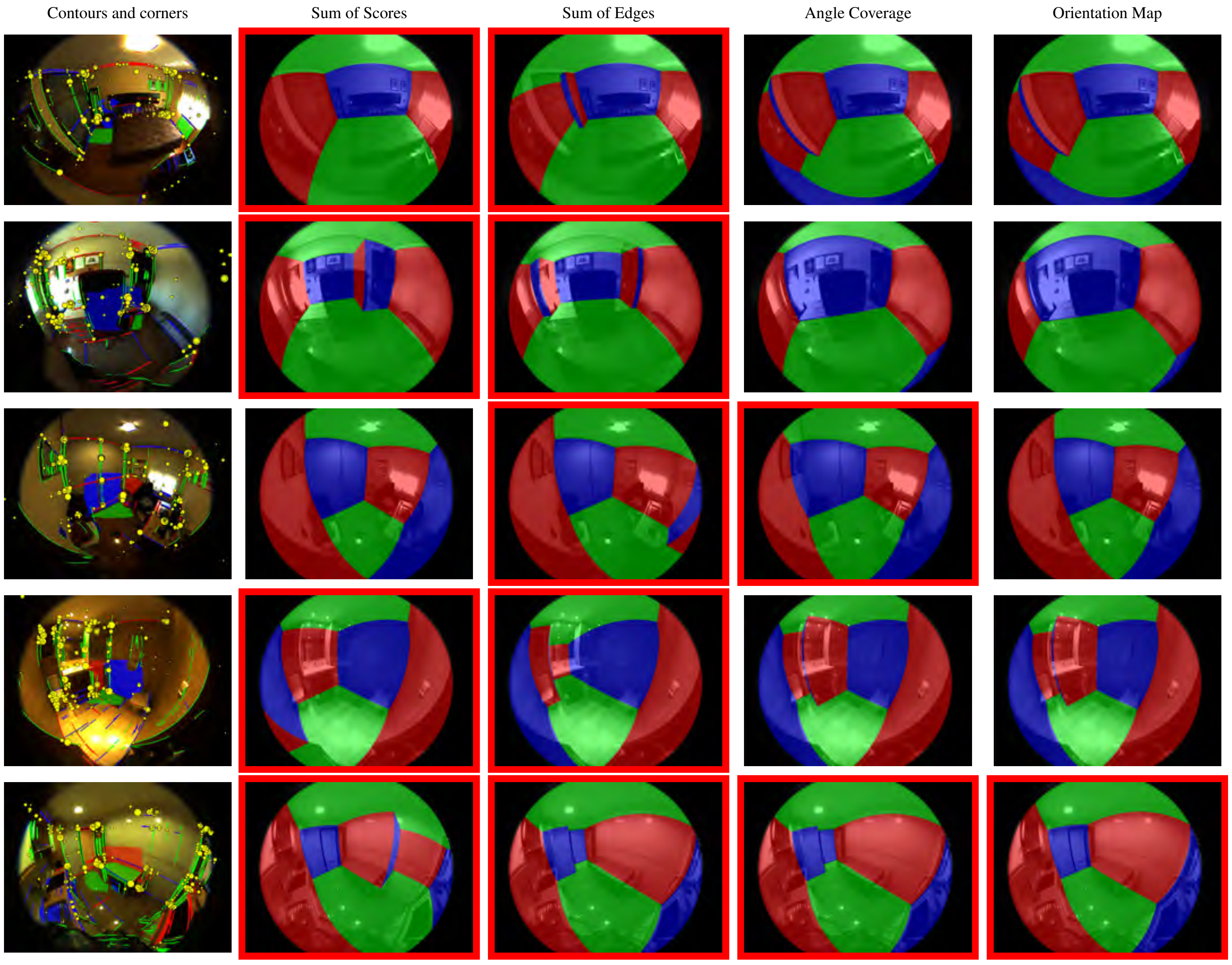


**Figure 19:** Examples of results from non-corridor scenes of our set with best layout proposal for each evaluation criterion. Those results with red frame are failure cases.

information throughout the system in order to numerically observe how the results are affected. The absence of depth affects the computation of the VPs, the scoring of lines, the retrieval of the $H_{ceil}$ and the elimination of contradictory hypotheses. In particular, in Fig. 15h there is an example of corner extraction without depth failing at getting the ceiling plane.

A comparison of results with 50 hypotheses with and without depth information is shown in the Figure 17 and Table 2. It can be observed that the standard deviation of PA increases since many cases result in much lower scores. Consequently, the mean pixel accuracy decreases about 4%. However, observing the per-image values on the Fig. 17, in many cases the results are not affected by the lack of depth, but in some others the results are so bad that it is very noticeable in the mean value. This is also observable on the median value, which does not experiment so much decrease in comparison. Thus, the depth not only provides scale, but it also helps in many individual cases.

### *5.5. Results with Google Tango*

We want to show the applicability of the method with commercial devices, such as the Google Tango (Fig. 13). Even though the device at our disposal is a *Development Kit*, there are several phones with the same technology already in the market. Among other sensors, this device includes: a depth camera which is similar to the other camera system but with less resolution ($320 \times 180$ instead of $640 \times 480$), and a motion-tracking camera which basically is a fisheye of about 170° of field of view and resolution of $640 \times 480$. The simultaneous depth and fisheye image pairs have been captured using Tango ROS Streamer[1].

The decrease in the resolution does not affect the quality of the results notably. Note that one of the first operations with the depth information consists on downsizing the point cloud. In the fisheye camera the loss of information is not significant

[1]`http://wiki.ros.org/tango_ros_streamer`

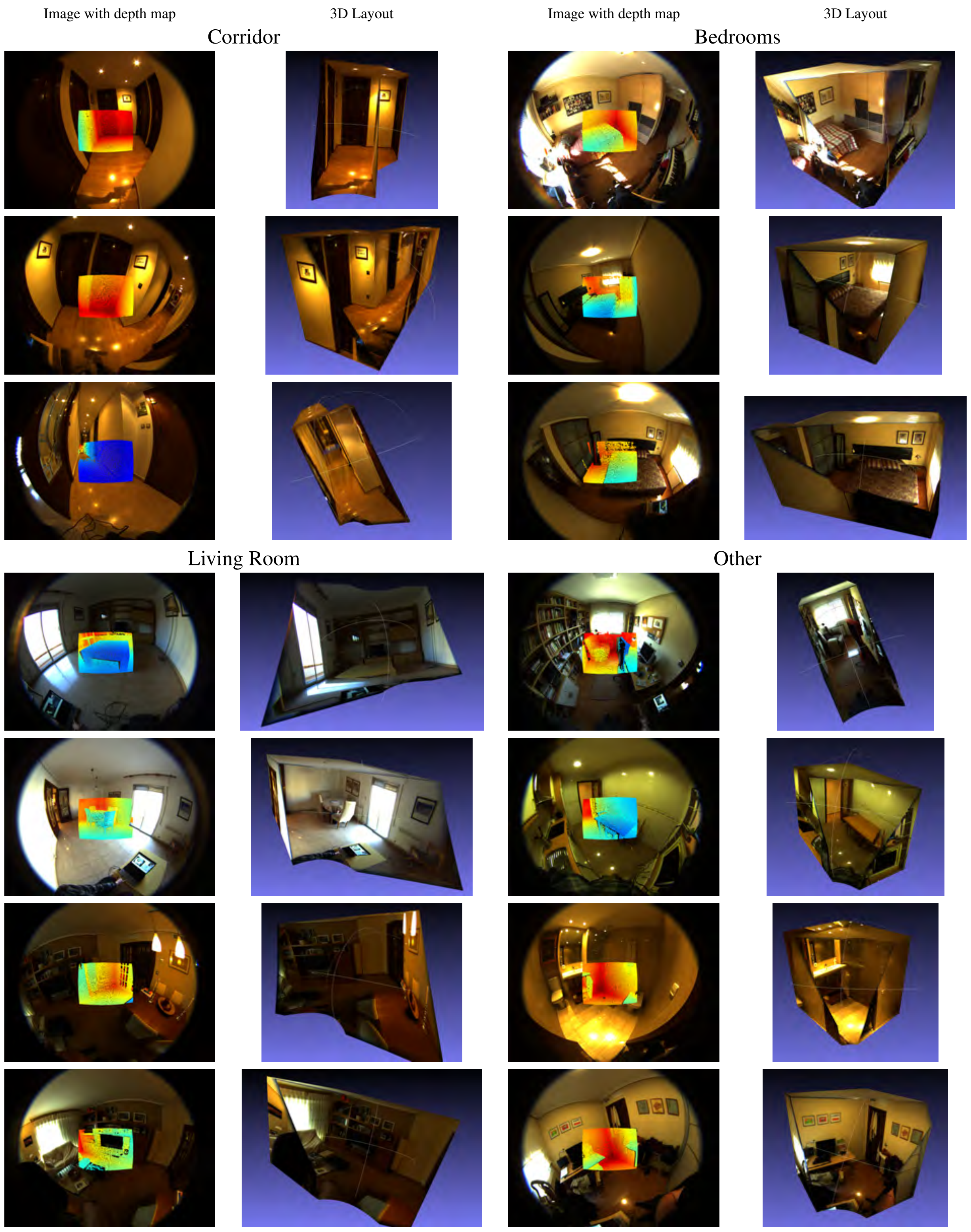


**Figure 20:** Pair of images of fisheye images with the depth information from the depth camera overlaid and the 3D layouts we are able to retrieve corresponding to each case.

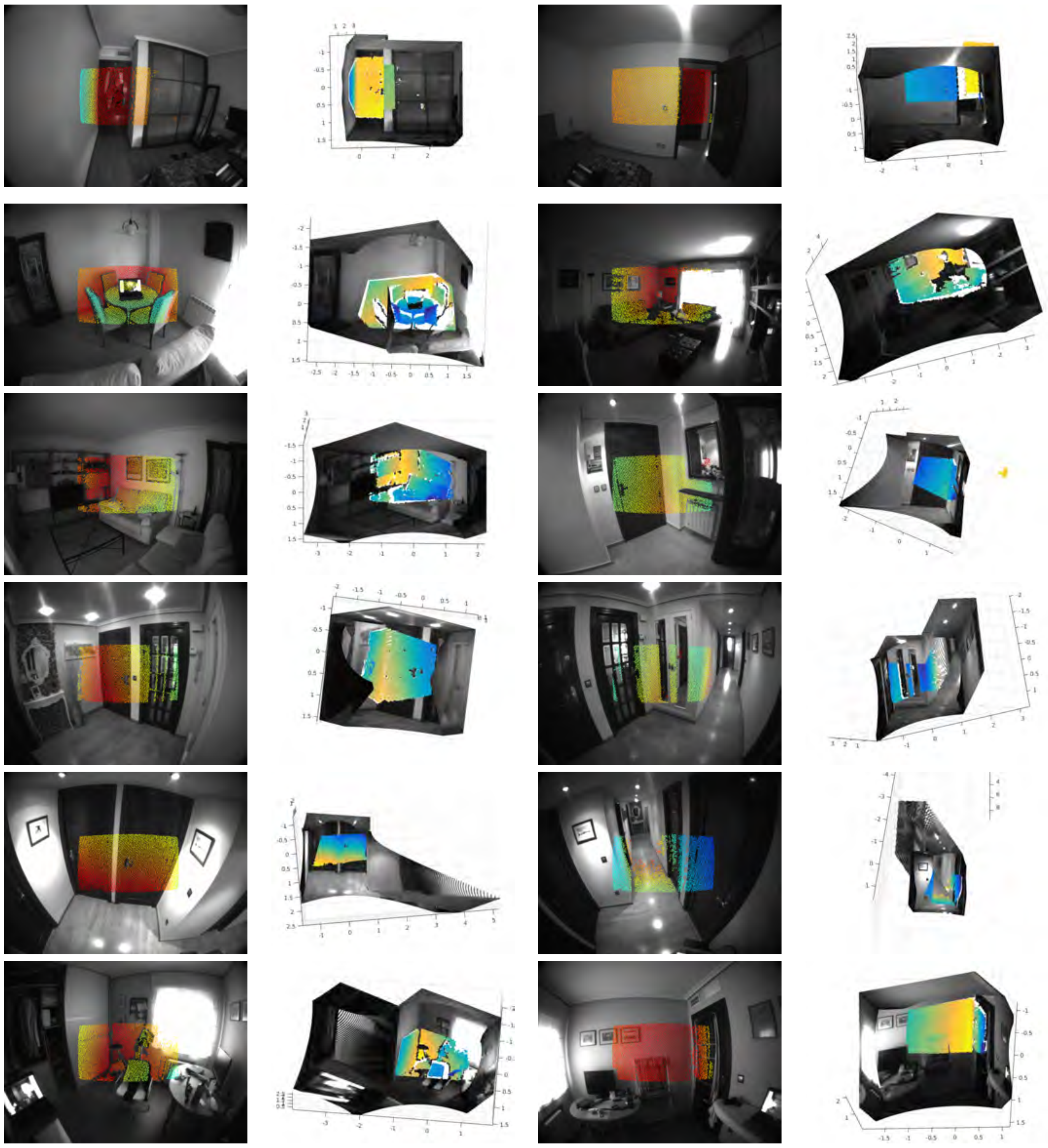


**Figure 21:** Twelve examples of application of our method with a Google Tango device. At the left of each case the fisheye image with the depth points projected in colors (variable color with depth). At the right, the resulting 3D point cloud. For visualization purposes, the initial 3D point cloud is also displayed in color to show how the resulting cloud has been scaled and fits accordingly.

and it actually helps speeding up the algorithm. However, the decrease in field of view is quite relevant for the task, since the lesser spatial view of the environment we have, the lesser likelihood of finding the most useful corners. In order to increase the amount of lines from the ceiling (which usually belong to less cluttered areas) the camera could be pointing slightly upwards than the previous device. This may result in loss of floor plane view, and thus scaled layout recovery. However, we can proceed with the non-scaled layout recovery and apply the scaling procedure presented in Section 4.4 to recover the scaled layout.

In Fig. 21 we show several examples with images from the Tango device in similar type of views to the first dataset, where the non-scaled layout recovery with Angle Coverage method and final scaling have been applied. This shows that our method is able to estimate the scaled layout when the floor is not in the image, and thus with no restrictions about how the camera is posed in the scene. With the previous dataset most images had views of the floor and we could not show this feature. While this alternative approach to solve the problem works for most scenes, it is still useful to have the floor and thus the scale since the beginning. For example, the second example include depth points out of the room through the open door, which breaks condition E (Section 4.2.1). However, since when generating hypotheses we have no scale, we cannot use that condition to discard the hypotheses. Apart from these types of exceptional cases, our method is able to extend the depth information by extracting the correct layout. Notice that here there is also some cases where only one Manhattan direction is observed from depth (e.g. fifth row in first column or first row in second column in Fig. 21). Our vanishing point extraction method combines depth and fisheye (see Section 3.3) and thus is still able to recover the main directions and carry on with the layout estimation process.

## 6. Conclusion

In this work, we have developed a new method to extend the 3D information of a depth camera to a field of view of over 180 degrees. In particular, we propose a novel spatial layout retrieval algorithm, whose main novelty is combining a fisheye and a depth camera. The large field of view helps to use information from both the ceiling and the floor, which is helpful when there is clutter in the scene. The depth information helps by providing scale, necessary for the final 3D reconstruction, and by enhancing the performance of the method. Experimental evaluation with real images of indoor environments shows good results in terms of accuracy, improving the state of the art in functionality: our method has less layout shape restrictions, needs fewer hypotheses and provides full-scaled 3D models of the scene in a single shot. One of the advantages of returning a full-scaled reconstruction is that it complements the information coming from the depth camera: besides the small part of the scene reliably captured by the depth camera, now it is possible to have a good estimation of the surroundings to over 180 degrees. This kind of information could be useful in fields such as robotics or augmented reality. Additionally, the method has been tested with data from a portable consumer device successfully, showing great potential for the future, especially regarding the possibility of using it in a wearable configuration.

**Acknowledgments.**

This work was supported by Projects DPI2014-61792-EXP and DPI2015-65962-R (MINECO/FEDER, UE) and grant BES-2013-065834 (MINECO).

## References

[1] A. Perez-Yus, D. Gutierrez-Gomez, G. Lopez-Nicolas, J. J. Guerrero, Stairs detection with odometry-aided traversal from a wearable RGB-D camera, Computer Vision and Image Understanding 154 (2017) 192–205.

[2] J. M. Coughlan, A. L. Yuille, Manhattan world: Compass direction from a single image by bayesian inference, in: IEEE International Conference on Computer Vision (ICCV), Vol. 2, 1999, pp. 941–947.

[3] D. C. Lee, M. Hebert, T. Kanade, Geometric reasoning for single image structure recovery, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009, pp. 2136–2143.

[4] A. Perez-Yus, G. Lopez-Nicolas, J. J. Guerrero, Peripheral expansion of depth information via layout estimation with fisheye camera, in: European Conference on Computer Vision (ECCV), Springer, 2016, pp. 396–412.

[5] D. R. Walton, D. Thomas, A. Steed, A. Sugimoto, Synthesis of environment maps for mixed reality, in: IEEE International Symposium on Mixed and Augmented Reality (ISMAR), 2017, pp. 72–81.

[6] E. Delage, H. Lee, A. Y. Ng, A dynamic bayesian network model for autonomous 3D reconstruction from a single indoor image, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Vol. 2, 2006, pp. 2418–2428.

[7] V. Hedau, D. Hoiem, D. Forsyth, Recovering the spatial layout of cluttered rooms, in: IEEE International Conference on Computer Vision (ICCV), 2009, pp. 1849–1856.

[8] D. Hoiem, A. A. Efros, M. Hebert, Recovering surface layout from an image, International Journal of Computer Vision 75 (1) (2007) 151.

[9] A. G. Schwing, T. Hazan, M. Pollefeys, R. Urtasun, Efficient structured prediction for 3D indoor scene understanding, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012, pp. 2815–2822.

[10] S. Ramalingam, J. Pillai, A. Jain, Y. Taguchi, Manhattan junction catalogue for spatial reasoning of indoor scenes, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013, pp. 3065–3072.

[11] H.-C. Chang, S.-H. Huang, S.-H. Lai, Using line consistency to estimate 3D indoor Manhattan scene layout from a single image, in: IEEE International Conference on Image Processing (ICIP), 2015, pp. 4723–4727.

[12] V. Hedau, D. Hoiem, D. Forsyth, Thinking inside the box: Using appearance models and context based on room geometry, in: European Conference on Computer Vision (ECCV), Springer, 2010, pp. 224–237.

[13] D. C. Lee, A. Gupta, M. Hebert, T. Kanade, Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces, in: Advances in Neural Information Processing Systems 23, 2010, pp. 1288–1296.

[14] L. Del Pero, J. Guan, E. Brau, J. Schlecht, K. Barnard, Sampling bedrooms, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011, pp. 2009–2016.

[15] L. Del Pero, J. Bowdish, D. Fried, B. Kermgard, E. Hartley, K. Barnard, Bayesian geometric modeling of indoor scenes, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012, pp. 2719–2726.

[16] W. Choi, Y.-W. Chao, C. Pantofaru, S. Savarese, Understanding indoor scenes using 3D geometric phrases, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013.

[17] A. G. Schwing, S. Fidler, M. Pollefeys, R. Urtasun, Box in the box: Joint 3D layout and object reasoning from single images, in: IEEE International Conference on Computer Vision (ICCV), 2013, pp. 353–360.

[18] A. Flint, D. Murray, I. Reid, Manhattan scene understanding using monocular, stereo, and 3D features, in: IEEE International Conference on Computer Vision (ICCV), 2011, pp. 2228–2235.

[19] A. Furlan, S. D. Miller, D. G. Sorrenti, F.-F. Li, S. Savarese, Free your camera: 3D indoor scene understanding from arbitrary camera motion, in: British Machine Vision Conference (BMVC), 2013.

[20] Y. Zhang, F. Yu, S. Song, P. Xu, A. Seff, J. Xiao, Large-scale scene understanding challenge: Room layout estimation, in: Computer Vision and Pattern Recognition Workshop, 2015.

[21] A. Mallya, S. Lazebnik, Learning informative edge maps for indoor scene layout prediction, in: IEEE International Conference on Computer Vision (ICCV), 2015, pp. 936–944.

[22] W. Zhang, W. Zhang, K. Liu, J. Gu, Learning to predict high-quality edge maps for room layout estimation, IEEE Transactions on Multimedia 19 (5) (2017) 935–943.

[23] S. Dasgupta, K. Fang, K. Chen, S. Savarese, Delay: Robust spatial layout estimation for cluttered indoor scenes, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 616–624.

[24] C.-Y. Lee, V. Badrinarayanan, T. Malisiewicz, A. Rabinovich, Roomnet: End-to-end room layout estimation, 2017, pp. 4875–4884.

[25] H. Jia, S. Li, Estimating the structure of rooms from a single fisheye image, in: IAPR Asian Conference on Pattern Recognition (ACPR), IEEE, 2013, pp. 818–822.

[26] G. Lopez-Nicolas, J. Omedes, J. J. Guerrero, Spatial layout recovery from a single omnidirectional image and its matching-free sequential propagation, Robotics and Autonomous Systems 62 (9) (2014) 1271–1281.

[27] H. Jia, S. Li, Estimating structure of indoor scene from a single full-view image, in: IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 4851–4858.

[28] K. Fukano, Y. Mochizuki, S. Iizuka, E. Simo-Serra, A. Sugimoto, H. Ishikawa, Room reconstruction from a single spherical image by higher-order energy minimization, in: IAPR International Conference on Pattern Recognition (ICPR), 2016, pp. 1768–1773.

[29] Y. Zhang, S. Song, P. Tan, J. Xiao, PanoContext: A whole-room 3D context model for panoramic scene understanding, in: European Conference on Computer Vision (ECCV), Springer, 2014, pp. 668–686.

[30] J. Xu, B. Stenger, T. Kerola, T. Tung, Pano2CAD: Room layout from a single panorama image, in: IEEE Winter Conference on Applications of Computer Vision (WACV), 2017, pp. 354–362.

[31] H. Yang, H. Zhang, Efficient 3D room shape recovery from a single panorama, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 5422–5430.

[32] R. Cabral, Y. Furukawa, Piecewise planar and compact floorplan reconstruction from images, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 628–635.

[33] F. Endres, C. Sprunk, R. Kummerle, W. Burgard, A catadioptric extension for RGB-D cameras, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2014, pp. 466–471.

[34] J. Iglesias, P. Mirado, R. Ventura, Towards an omnidirectional catadioptric RGB-D camera, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, pp. 2506–2513.

[35] R. Tomari, Y. Kobayashi, Y. Kuno, Wide field of view Kinect undistortion for social navigation implementation, in: Advances in Visual Computing, Springer, 2012, pp. 526–535.

[36] E. Fernandez-Moral, J. Gonzalez-Jimenez, P. Rives, V. Arevalo, Extrinsic calibration of a set of range cameras in 5 seconds without pattern, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2014, pp. 429–435.

[37] I. Armeni, S. Sax, A. R. Zamir, S. Savarese, Joint 2D-3D-Semantic data for indoor scene understanding, arXiv preprint arXiv:1702.01105 (2017).

[38] A. Perez-Yus, G. Lopez-Nicolas, J. J. Guerrero, A novel hybrid camera system with depth and fisheye cameras, in: IAPR International Conference on Pattern Recognition (ICPR), 2016, pp. 2789–2794.

[39] A. Perez-Yus, E. Fernandez-Moral, G. Lopez-Nicolas, J. J. Guerrero, P. Rives, Extrinsic calibration of multiple RGB-D cameras from line observations, IEEE Robotics and Automation Letters 3 (1) (2018) 273–280.

[40] Q. Zhang, R. Pless, Extrinsic calibration of a camera and laser range finder (improves camera calibration), in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vol. 3, 2004, pp. 2301–2306.

[41] D. Scaramuzza, A. Harati, R. Siegwart, Extrinsic self calibration of a camera and a 3D laser range finder from natural scenes, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2007, pp. 4164–4169.

[42] A. Geiger, F. Moosmann, O. Car, B. Schuster, Automatic camera and range sensor calibration using a single shot, in: IEEE International Conference on Robotics and Automation (ICRA), 2012, pp. 3936–3943.

[43] L. Puig, J. Bermudez-Cameo, P. Sturm, J. J. Guerrero, Calibration of omnidirectional cameras in practice: A comparison of methods, Computer Vision and Image Understanding 116 (1) (2012) 120–137.

[44] D. Scaramuzza, A. Martinelli, R. Siegwart, A toolbox for easily calibrating omnidirectional cameras, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2006, pp. 5695–5701.

[45] D. Herrera C, J. Kannala, J. Heikkilä, Joint depth and color camera calibration with distortion correction, IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (10) (2012) 2058–2064.

[46] J. Bermudez-Cameo, G. Lopez-Nicolas, J. J. Guerrero, Automatic line extraction in uncalibrated omnidirectional cameras with revolution symmetry, International Journal of Computer Vision 114 (1) (2015) 16–37.

[47] J.-C. Bazin, I. Kweon, C. Demonceaux, P. Vasseur, A robust top-down approach for rotation estimation and vanishing points extraction by catadioptric vision in urban environment, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2008, pp. 346–353.

[48] R. B. Rusu, S. Cousins, 3D is here: Point cloud library (PCL), in: IEEE International Conference on Robotics and Automation (ICRA), 2011, pp. 1–4.

[49] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, Ros: an open-source robot operating system, in: ICRA workshop on open source software, Vol. 3, Kobe, Japan, 2009, p. 5.

[50] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, G. Randall, Lsd: A fast line segment detector with a false detection control, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (4) (2010) 722–732.

[51] Q. Zhang, X. Shen, L. Xu, J. Jia, Rolling guidance filter, in: European Conference on Computer Vision (ECCV), Springer, 2014, pp. 815–830.

[52] C. Fernandez-Labrador, A. Perez-Yus, G. Lopez-Nicolas, J. J. Guerrero, Layouts from panoramic images with geometry and deep learning, IEEE Robotics and Automation Letters 3 (4) (2018) 3153–3160.

[53] D. Eigen, R. Fergus, Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture, in: IEEE International Conference on Computer Vision, 2015, pp. 2650–2658.