# 7

# Two Dimensional Floor Plan Synthesis from RGB Images and Location Cues

Understanding indoor spaces and interpreting them in natural language text is one form of interpreting indoor spaces. However, indoor space could also be understood from its RGB images and converted into its pictorial interpretation in the form of its floor plan itself. Understanding the indoor space from its images will take visual information and information that connects the visual information. Hence, the visual information could be its video, panoramic image, $360^o$ spherical image, or RGB images. The information which connects the visuals could be obtained from camera motion by tracking its trajectory. The interpretation of indoor space through its floor plan could help architects and interior designers automatically generate a building floor plan if an older building requires renovation and its floor plan is not available. When the floor plan of a building is not available, the manual way of its recreation is very tedious and time taking. Hence, providing a user with a system to generate a floor plan from its RGB images taken from a conventional mobile phone's camera will save time and effort. The generated plan can also be given to the description generation system for generating textual interpretation. This chapter aims to build a solution to the floor plan's regeneration problem as a compact and simplified representation of the indoor scene representing doors, walls, rooms, and semantic relations.

The current state of the art methods for generating a layout from indoor scenes require specialized hardware such as depth cameras or depth-sensing hardware, which is not very reachable to a common user. The system should take inputs from a device that is commonly available to everyone. Hence, in this chapter, a method, GRIHA (Generating Room Interior of a House using ARCore), is discussed, which uses a simple mobile phone's camera to capture RGB images of the indoor space and generate the layout of the house accordingly. GRIHA uses a stream of RGB indoor space images captured by a user from a conventional mobile phone's camera and using the in-built SLAM in Google ARcore library to track the camera movements. The generated layout agrees well in terms of dimensions of the real layout. Hence, GRIHA also tries to remove the floor plan's explicit requirement to be provided by the user to generate a textual description of the indoor space by interacting directly with its images. The proposed technique GRIHA generates a layout of the indoor environment that can be useful for the housing and architectural industry, indoor navigation for robots, providing an environment for augmented or virtual reality applications. In this chapter, the methodology of generating layout from indoor space images is discussed along with the application developed in ARcore for capturing the images. Also, experiments have been performed using several state-of-the-art Android and iOS applications to generate layouts over area and aspect ratio parameters.

In the rest of the chapter, Sec. 7.1 gives a brief introduction of the proposed system, Sec 7.2 defines the problem statement, and Sec. 7.3 gives the overview of the data collection step for the proposed system. Section 7.4 introduces the system model, the architectural framework of GRIHA, and our assumptions, where Sec. 7.4.2, Sec. 7.4.3 describes the steps taken for depth and boundary estimation for an indoor scene and Sec. 7.4.4 explain the processing steps for the point clouds and the generation of the 2D floor plan from them. Section 7.5 describes the various qualitative and quantitative experiments performed for the proposed algorithm sub-modules, and

Sec. 7.6 concludes with an outlook for the future work.

## 7.1 BRIEF OVERVIEW

Generating Building Information Model (BIM) in 2D/3D from indoor scenes has applications including real estate websites, indoor navigation, augmented/virtual reality, and many more. BIM should include the global layout of the entire floor plan of the space, number of rooms, and arrangements. The most natural way to create the floor plan is by manually measuring each room and integrating everything in Computer-Aided Design (CAD) software to have a global layout. However, such manual measurement is a tedious task and requires a significant amount of work hours. Hence, there is a need to develop a graphics and visualization system that takes a few images or videos of the entire indoor scene and automatically generates a floor plan. The pictures or videos may or may not have depth information since a typical user might not have a depth camera or specialized hardware to measure depth in the scene.

However, estimating an accurate layout from RGB images is a challenging task. It requires precise depth information along with the RGB images and an exact point cloud with real-world scaling. The current state of scholarship, as well as commercial applications for layout estimation, uses RGB-D images or panorama images to solve this problem. Despite good accuracy, these methods require special hardware (depth camera) or a particular photo capture mode (panorama with little to no occlusion). Such requirements restrict widespread adoption. In this work, we propose a framework named GRIHA. This a framework to estimate the 2D floor plan of an indoor scene using RGB images captured through a mobile camera. Figure 7.1 depicts the potential partial input RGB images of indoor locations and their corresponding floor plan of the entire indoor scene. In our proposed method, the user needs to take pictures of all the indoor scene rooms using our developed application. The depth map of the locations is extracted from these pictures adapting the method proposed in Alhashim and Wonka [2018] followed by edge map extraction using the method proposed in Zou *et al.* [2018]. The RGB images and depth maps are used for 3D reconstruction. The point clouds and edge maps are then mapped in 2D for the final layout using a novel regularization technique. We take advantage of Google's ARCore library for pose estimation, thus saving computational time.

In this chapter, our major contributions are: (1) 2D floor plan estimation from RGB images without using panorama or 360° spherical images, (2) Development of an ARCore based Android mobile application for data collection and application of ARCore pose data for 3D reconstruction of the scene, (3) Near accurate layout estimation with a fewer number of images captured through the camera, compared to the existing techniques, and (4) Cross-platform testing of the proposed framework on multiple devices. Currently, we are assuming only Manhattan and weak Manhattan scenes. The proposed method compares well in terms of dimensions measured by the existing commercial applications on the Android platform to generate the layout and measure the dimensions. GRIHA works well in case of occlusion in the indoor scene where existing methods may fail.

## 7.2 PROBLEM STATEMENT

Figure. 7.1 shows the input and output of the proposed system GRIHA. Figure 7.1(a) shows the partial images taken from every corner of the room by mobile phone's camera. Here, $C^1$, $C^2$, $C^3$, $C^4$ are the four locations from which a user captures four room images using a mobile camera.

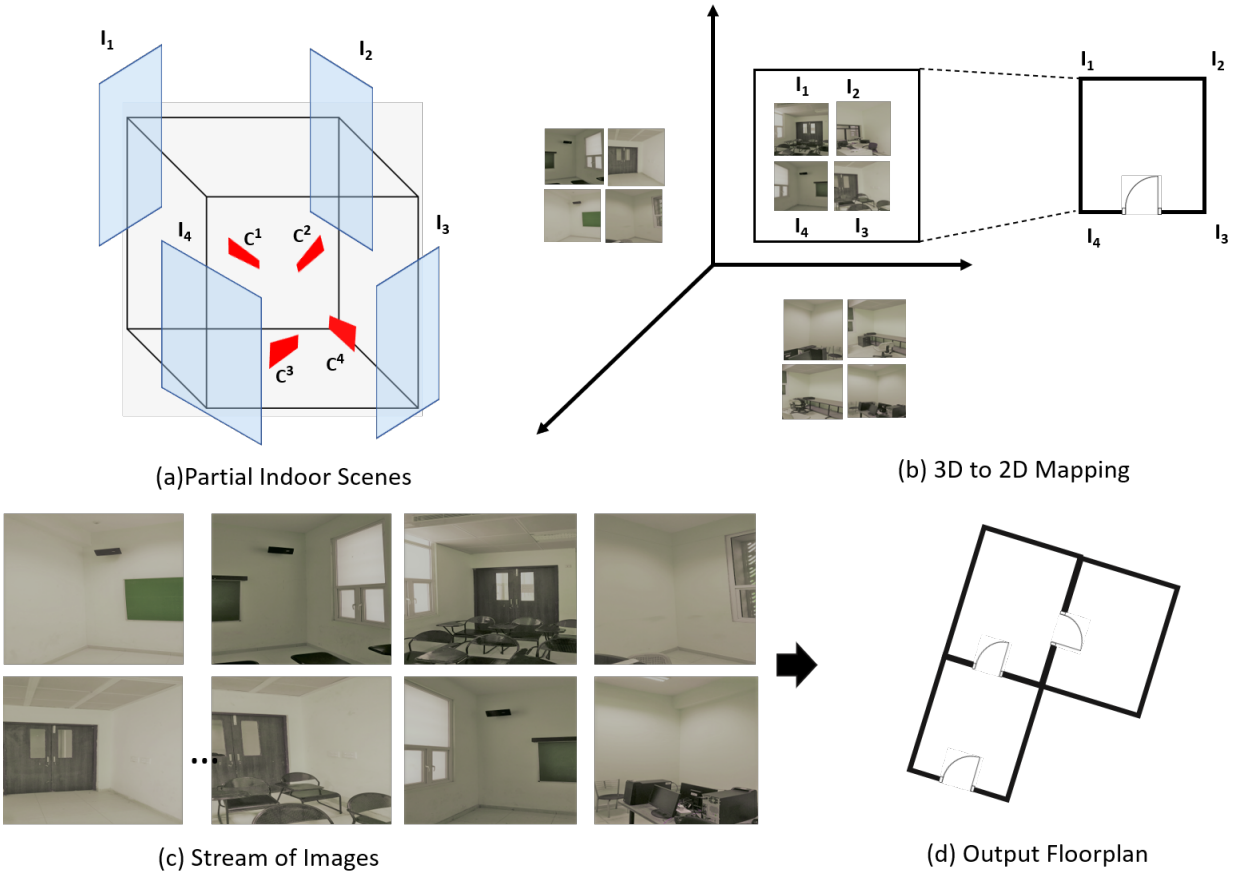



**Figure 7.1 :** A schematic illustration of the entire input and output scenario while using GRIHA.

These captured images are marked as $I_1$, $I_2$, $I_3$, $I_4$, where image $I_k$ is captured from camera pose $C^k$. These are the images that act as inputs to the GRIHA framework. Figure. 7.1 (b) depicts every corner image's mapping to a partial 2D layout. To construct the 2D floor plan, we need to map the images to the corresponding corner points of a room in a Manhattan world and establish a sequence to get the room's complete layout. Figure. 7.1 (c), and (d) show the input stream of indoor scene images for a complete floor and its respective output floor plan. To summarize, the task at hand is *"Given a sequence of images of all the rooms (for every room, images of four corners and the door) in a single floor of a building, generating the 2-D plan of the entire floor".*

## 7.3 IMAGE ACQUISITION PLATFORM

Most of the methods for 3D reconstruction of a room and floor plan estimation require an ample amount of specific hardware such as a depth camera, Kinect camera, LiDAR. Although some methods exist for layout generation from monocular images as proposed by Zou *et al.* [2018], they rely on occlusion-free panoramic photos, which are almost impossible to take in common spaces such as an office or home. Instead, our proposed method relies on multiple regular 2D photos, which are easier to capture in an occlusion free manner. Figure 7.2 (a) shows the panoramic image of a large office space and Fig. 7.2(b) shows multiple 2D images for the same scene.

It can be seen that the panoramic image of a vast space may have several occlusions, which

(a) (b)

**Figure 7.2 :** Panoramic image and multiple 2D images for the same scene.

make their 3D reconstruction difficult and erroneous. Most of their corners and other important edges may get occluded in the panorama because of the furniture and other decors. At the same


**Figure 7.3 :** Highlighted occluded portions of the scene and corresponding 2D images.

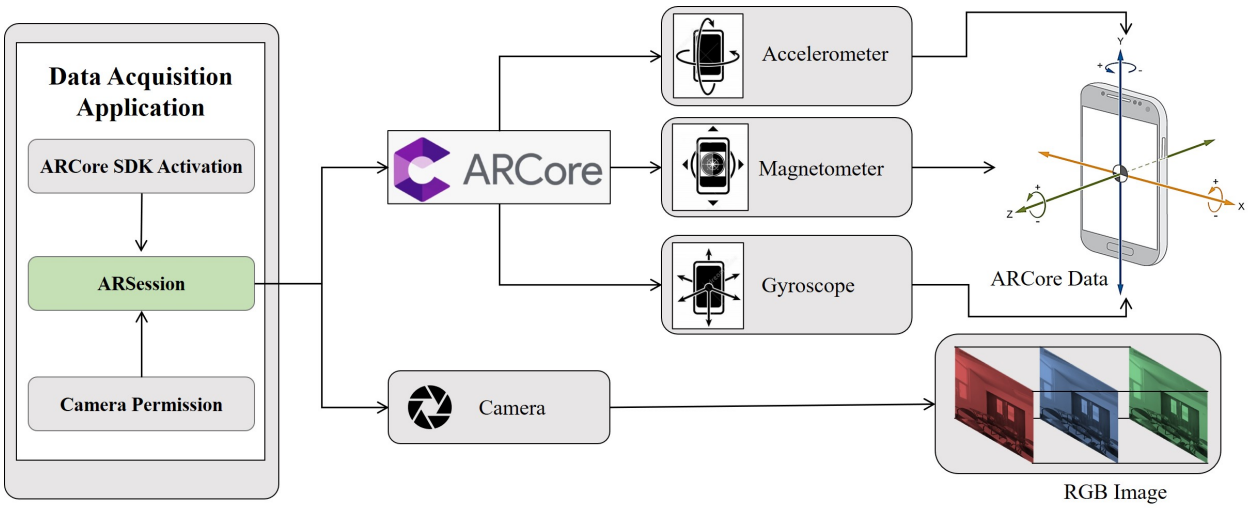



**Figure 7.4 :** A schematic representation of the image acquisition framework developed for GRIHA.

time, they are easier to capture in their multiple 2D images. It's not possible to cover the complete scene (backside of the camera) in a panoramic image without losing important information. While capturing multiple 2D images makes it easier to capture all the corners, edges, and other portions of the scene, without getting occluded by the furniture and without losing any information. Figure 7.3 shows the highlighted portions of the scene in the panoramic image and their corresponding 2D images, which have more information about the scene than their panoramic version. Hence, generating the scene's layout from their separate 2D images gives more information about the scene than generating it from its panoramic image.

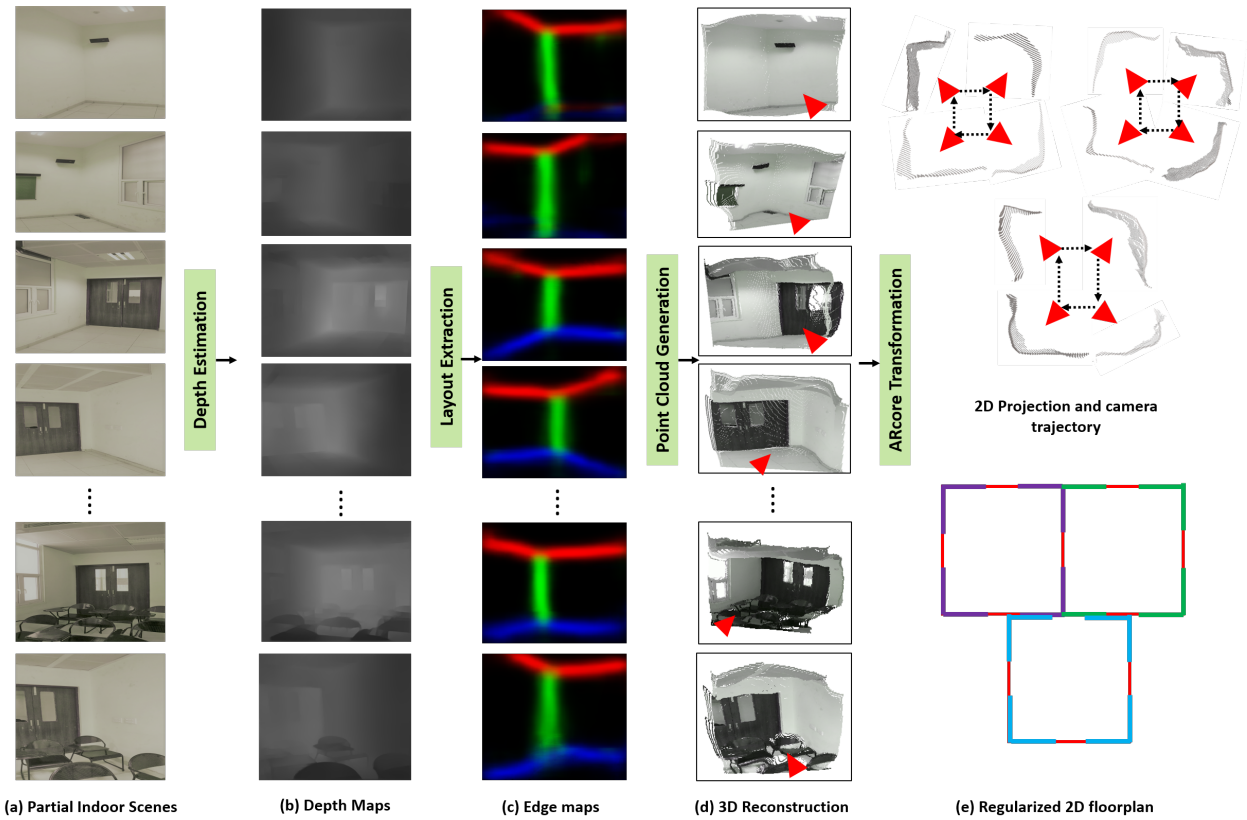



**Figure 7.5 :** Pipeline of the proposed framework.

We used smartphones to capture images and track camera motion for our experiments. The proposed method attempts to address this issue by taking advantage of recent mobile Augmented Reality libraries for pose estimation. Figure 7.4 shows an illustration of how GRIHA collects data using ARCore, which readily available on almost all Android devices. ARcore proposed by Google Developers [2020a], is a library by Google, which uses the phone's IMU sensor's data along with image feature points for tracking the pose of the camera utilizing a proprietary Simultaneous Localization and Mapping (SLAM) algorithm. It performs pose estimation in real-time, challenging to achieve with a third-party SLAM algorithm. To collect the data, an android application was developed in Unity3D environment. The camera's real-world position and orientation were extracted. The proposed method takes partial scene images for each room in the house for floor plan reconstruction.

## 7.4 PROPOSED SYSTEM

### 7.4.1 Overview of GRIHA

Figure 7.5 depicts a step-by-step visual illustration of the working principle of GRIHA. As shown in Fig. 7.5(a), a stream of RGB images are taken for each partial scene in the entire house. Depth estimation is done individually for each RGB image (see Fig. 7.5(b)) and edge/boundary map is estimated by adapting the network proposed in Zou *et al.* [2018] for perspective images (see Fig. 7.5(c)). Mapping the depth-maps to RGB images and edge maps of layout, the point clouds are generated and layout is extracted from the point clouds (see Fig. 7.5(d)). When the pictures are being taken, we utilized ARCore's built-in SLAM algorithm to extract the 3D pose of the camera and saved it alongside the images. The camera poses are utilized to rebuild the camera trajectory, which is utilized for transforming each partial scene point cloud after projecting them into a 2D plane. The point cloud of the layout is extracted from the 3D reconstructed scene using the edge maps generated in the previous steps. All the partial point clouds are projected in 2D and translated using ARcore transformations. Each partial point cloud is then regularized locally and globally to generate the 2D floor plan for the house (see Fig. 7.5(e)). The red markers indicate the trajectory of the camera while capturing the images for each room in the house.

### 7.4.2 Depth Estimation

Depth perception is essential for estimating the correct dimension of the targeted floor plan. Figure 7.6 depicts an illustration of how the depth map is computed from a given image. Traditionally, a device with a built-in depth camera such as Google Tango or Microsoft Kinect is used for capturing point clouds directly from the scene. However, in GRIHA, the input taken is an RGB image taken from a traditional mobile device's camera. Hence, depth estimation is a mandatory step in this context. For depth perception from RGB images, multiple methods exploit feature matching techniques in multiple images of the same scene and reconstruct a 3D model for it. However, such schemes require a trained user to capture the data to ensure correspondence across images. Hence, a feasible solution for estimating depth from a single image is to use a pretrained machine learning model. Depth for RGB images can be learned in a supervised manner from ground truth depth-maps and a trained neural network can be used for estimating the depth for new images. We have adapted the method proposed in Alhashim and Wonka [2018], where an encoder-decoder architecture is used after extracting image features with DenseNet-169, resulting in high-resolution depth maps. The encoder used in this method is a pretrained truncated DenseNet-169. The decoder consists of basic blocks of convolutional layers, concatenated with successive $2\times$ bilinear upsampling blocks, and two $3 \times 3$ convolutional layers, where the output filter is half the size of the input. Figure. 7.7 shows the results for the depth estimation model on our dataset. The depth estimation model's performance analysis is present in Sec. 7.5.

### 7.4.3 Layout Estimation

While analyzing each partial scene, it is required to identify each pixel's classification as a wall or edge pixel. This classification/segmentation helps in the identification of the layout of the indoor scene into consideration. Hence, edge and boundary maps of a scene are a requirement for layout estimation. In our work, we have adapted the technique proposed in Zou *et al.* [2018] to identify the scene's edge/boundary maps. Figure. 7.8 shows the network architecture of LayoutNet and its inputs and respective outputs. LayoutNet network is an encoder-decoder structure. The

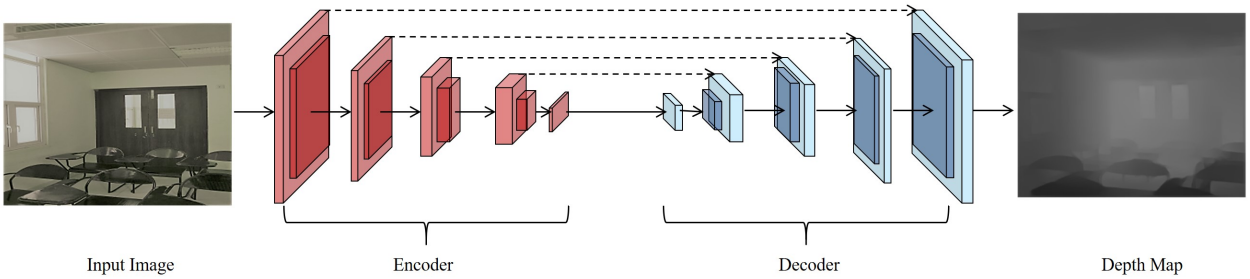



**Figure 7.6 :** An illustration of how depth is estimated by adapting the encoder-decoder model proposed in Alhashim and Wonka [2018].

encoder consists of seven convolutional layers with a filter size of $3 \times 3$ and ReLU (Rectified Linear Unit) function and max-pooling layer follow each convolutional layer. The decoder structure contains two branches, one for predicting boundary edge maps and the other for corner map prediction. Both decoders have a similar architecture, having seven layers of nearest neighbor upsampling operation, each followed by a convolution layer with a kernel size of $3 \times 3$ and the final layer being the Sigmoid layer. Corner map predictor decoder additionally has skip connections from the top branch for each convolution layer. Since their FOV (Field Of View) of the images is smaller, an additional predictor for predicting room type is added to improve corner prediction performance. The RGB image of the scene also takes Manhattan line segments as an additional input that provides other input features and improves the network's performance. Figure 7.9 shows the predicted edge and corner maps for the input image on our dataset. All the annotations for performance evaluation are done using the annotation tool proposed in Dutta and Zisserman [2019]. Our dataset's images also include images that have occluded corners and wall edges, so there is no requirement for the manual addition of corners.

### 7.4.4 3D Reconstruction

To generate a layout of the entire scene, a 3D reconstruction of each partial scene must be done. This step is not required in the methods where depth cameras or specialized hardware are used to capture the image/point clouds. However, in the proposed work, RGB images are taken from a mobile phone's camera, which does not have the depth information in the first place. Hence, a 3D reconstruction of each scene image is required to be done since we have tried to suppress


**Figure 7.7 :** Results of depth generation on our dataset.

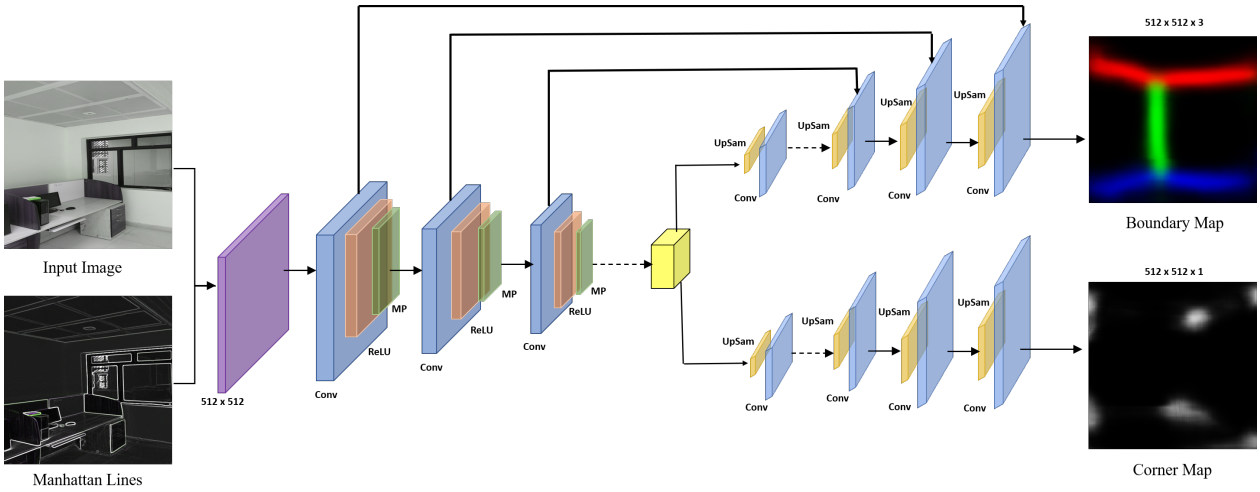



**Figure 7.8 :** An illustration of how LayoutNet is adapted for the layout estimation in GRIHA.

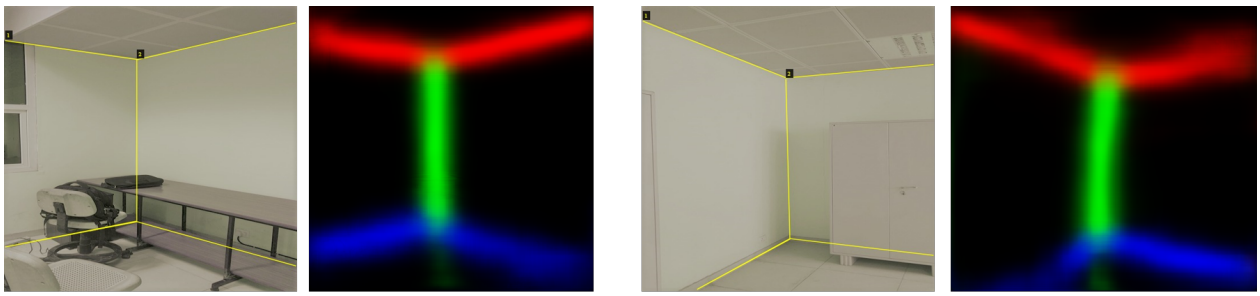
additional hardware requirements for this task. We mapped every pixel of the RGB image to depth maps generated in the previous steps to create a point cloud for the scene. This step is preceded by a camera calibration step to identify the phone's camera's intrinsic parameters: focal length ($f$), center coordinates ($C_x$,$C_y$). Here, coordinates in the point cloud are generated by:

$$Z = \frac{D_{u,v}}{\mathrm{S}} \tag{7.1}$$

$$X = \frac{(u - C_x) * Z}{f} \tag{7.2}$$

$$Y = \frac{(v - C_y) * Z}{f} \tag{7.3}$$

Here, $X,Y,Z$ are coordinates corresponding to the real world and $Z$ is the depth value. $D_{u,v}$ is the depth value corresponding to the $(u,v)$ pixel in the image in the depth map. S is the scaling factor of the scene, obtained empirically and comparing dimensions of real-world objects and point clouds.Here, $f$, $C_x$, $C_y$ are the intrinsic parameters of the camera, generated by calibration. At



(a) (b)

**Figure 7.9 :** Results of layout generation on our dataset.

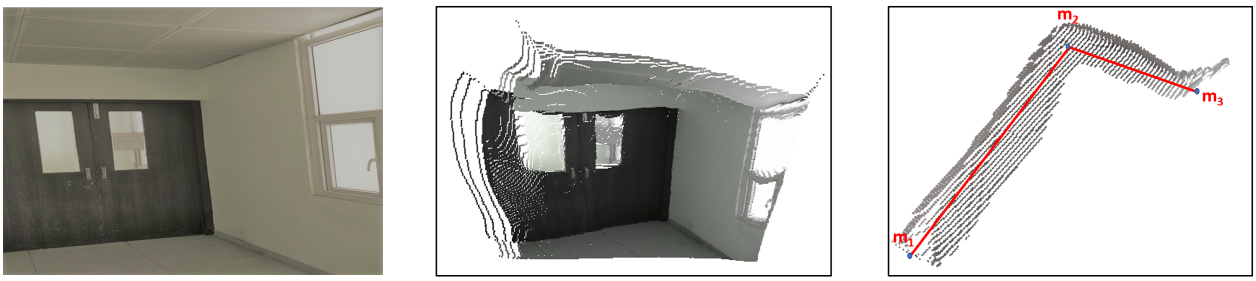
the end of this step, we have 3D reconstruction of each partial scene in the real world scale ( Fig. 7.5(d)). The red marker shows the pose of the camera while capturing the RGB image in the scene.

### 7.4.5 Point cloud transformation and regularization

The generated point clouds in the previous step are then mapped to the edge maps generated to identify the boundary pixels in the point cloud and projected in 2D space (as illustrated in Fig. 7.5(e)). These edge maps (as discussed in Sec. 7.4.3) are used to classify the pixels in point clouds in the wall and edge classes to identify the room's geometry. These point clouds are scattered 3D points of the layout and required to be regularized to reduce the error in the generated 2D layout's geometry (as illustrated in Fig. 7.10 for a given room). We utilized ARCore to extract the 3D pose and generated the camera trajectory from the extracted pose, which is depicted by dotted arrows, as shown in the top panel of Fig. 7.5(e). It returns rotational and translation coordinates for each scene image for the first image taken. The captured images are mapped to the local coordinate system of the image acquisition application discussed in Sec. 7.3. Also, there is no requirement of rotating the coordinate system while considering the transformation.

Algorithm 4 regularizes the local point cloud of each partial scene image for every room($R_j$) in all the room in a scene dataset (**R**). Here, $P_i$ is the point cloud of each $i$-th scene where $n$ is the total number of point clouds. Boundary points for each $P_i$ is extracted in $P_i(K)$. In the $k$-means algorithm, clusters of point set are made for $k = 3$ on the Euclidean distance between them, where $m_1, m_2, m_3$ are the cluster means (line 6). Since we are assuming the Manhattan world for the scene, the line joining means are readjusted to have a right angle (line 7). Each regularized local point cloud ($RP_i$) is transformed ($TP_i$) using rotation angle $\theta_x, \theta_y, \theta_z$, along each $x, y, z$ axis and translation coordinates $[t_x, t_y]$ returned by ARcore (line 9). For global regularization, using each transformed point cloud, polygon ($FP$) is formed (line 10), with $p$ number of sides ($s$). For each pair of sides, the angle between them ($\phi$) is checked and if they are not perpendicular, they are made collinear (line 12) assuming the world to be Manhattan.

GRIHA performs regularization in 3 steps. In step 1, unique layouts are regularized, and in steps 2 and 3, the complete floor plan is regularized, taking everything collectively. Figure 7.10 depicts the partial indoor scene with 2D layout generation. Figure 7.10(a) is the partial indoor scene RGB image in consideration, Fig. 7.10(b) is its 3D reconstruction in the form of point cloud, and Fig. 7.10(c) is the layout extracted from a 3D reconstructed point cloud using the previous steps' layout. It shows the 2D projection of the layout point cloud, where $m_1$, $m_2$, $m_3$ are the means of three clusters extracted. It also shows the lines joining $m_1$, $m_2$, and $m_3$, the regularized point




(a) (b) (c)

**Figure 7.10 :** An illustration of partial scene layout generation from 3-D point cloud.

**Algorithm 4** Regularize point clouds (PC)

$\forall R_j \in \mathbf{R}$ ▷ **R**: Total number of rooms
**for** $i = 1 : n$ **do** ▷ $n$: no of PC
  $P_i = 2DPointClouds$
  $\mathrm{K} = boundary(P_i)$
  $\mathrm{C}(\mathrm{c}_1, \mathrm{c}_2, ..., \mathrm{c}_k) = kmeans(P_i(\mathrm{K}))$ ▷ C: Clusters
  $\mathrm{m}_1, \mathrm{m}_2, \mathrm{m}_3 = mean(\mathrm{c}_1), mean(\mathrm{c}_2), mean(\mathrm{c}_3)$
  $line_1 = line(\mathrm{m}_1, \mathrm{m}_2)$
  $line_2 = line(\mathrm{m}_2, \mathrm{m}_3)$
  **while** $angle(line_1, line_2) <= 90$ **do**
    $Rotate(line_2)$
  **end while**
  $RP_i = (line_1, line_2)$ ▷ $RP_i$: local PC
  $TP_i = (Rot(\theta_x, \theta_y, \theta_z) * Tr(t_x, t_y)) * RP_i$
**end for**
$FP = polygon(TP_1, TP_2, ..., TP_n)$ ▷ $FP$: Final PC
**for** $i = 1 : \mathrm{p}$ **do** ▷ p: no of sides of polygon
  $\phi = angle(s_i, s_{i+1})$ ▷ $s$: sides of polygon
  **if** $\phi > 90$ or $\phi < 90$ **then**
    $\phi(s_i, s_{i+1}) = 0$
  **end if**
**end for**

cloud from the projected set of points.

Figure 7.11 depicts the transformation of partial point clouds and regularization of global layout. Figure. 7.11 (a) shows the coordinate system in real-world $(X_W, Y_W, Z_W)$ and in ARcore with the mobile device $(X_A, Y_A, Z_A)$. ARcore transformations have to be rotated about the $Z_A$ axis to align the coordinate systems. Each partial 2D point set is then rotated and translated with the transformation given by ARcore (Fig. 7.11 (b)). Figure 7.11(c) shows the globally regularized 2D point set for the partial point clouds for a set of rooms, and it agrees with the real world dimensions.

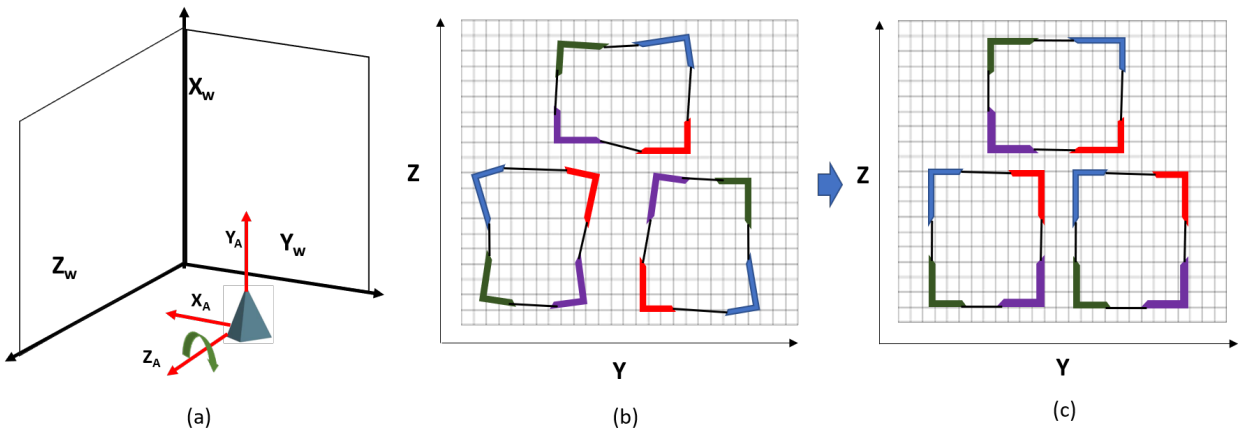
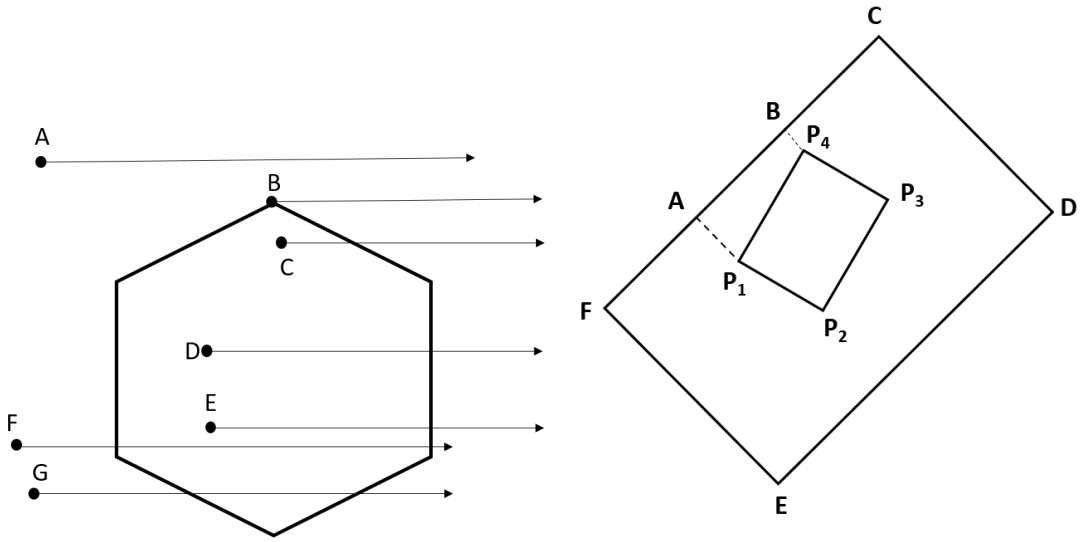



**Figure 7.11 :** Depiction of translation and global regularization.

**Algorithm 5** Finding the points inside the boundary polygon

**for** $i = 1 : n$ **do**
  $L_i = (P_i, \infty)$
  **if** $intersection(L_i, C_{hull}) == even$ **then** ▷ $C_{hull}$ : Convex hull forming boundary
    $P_i \leftarrow P_{outside}$ ▷ $P_{outside}$ : Pool of points outside the boundary
  **else**
    $P_i \leftarrow P_{inside}$ ▷ $P_{inside}$ : Pool of points inside the boundary
  **end if**
**end for**




(a) Finding the points which lie inside the boundary polygon.

(b) Aligning the rooms with the boundary.

**Figure 7.12 :** An illustration of the intermediate stages of the regularization process.

**Algorithm 6** Aligning the points of each polygon to the boundary

$\forall P_i \in polygon$
$L_1 = P_i \perp CF$
$L_2 = CF$
find equation of each line
$y = Y_C + m_{L_1} * (x - X_C)$
$m_{L_1} = \frac{(Y_C - Y_F)}{(X_C - X_F)}$
$m_{L_2} = \frac{(Y_{P_i} - Y_A)}{(X_{P_i} - X_A)}$
$m_{L_1} * m_{L_2} = -1$ ▷ Perpendicularity condition
$X_A = X_{P_i} + \frac{Y_C - Y_A}{X_C - X_F} * (Y_{P_i} - Y_A)$ ▷ Substituting the known values to find unknowns
$Y_A = Y_C + m_{L_1} * (X_A - X_C)$
$X_{P_i} = X_A$ ▷ Replacing the points of polygon with respective points on boundary
$Y_{P_i} = Y_A$

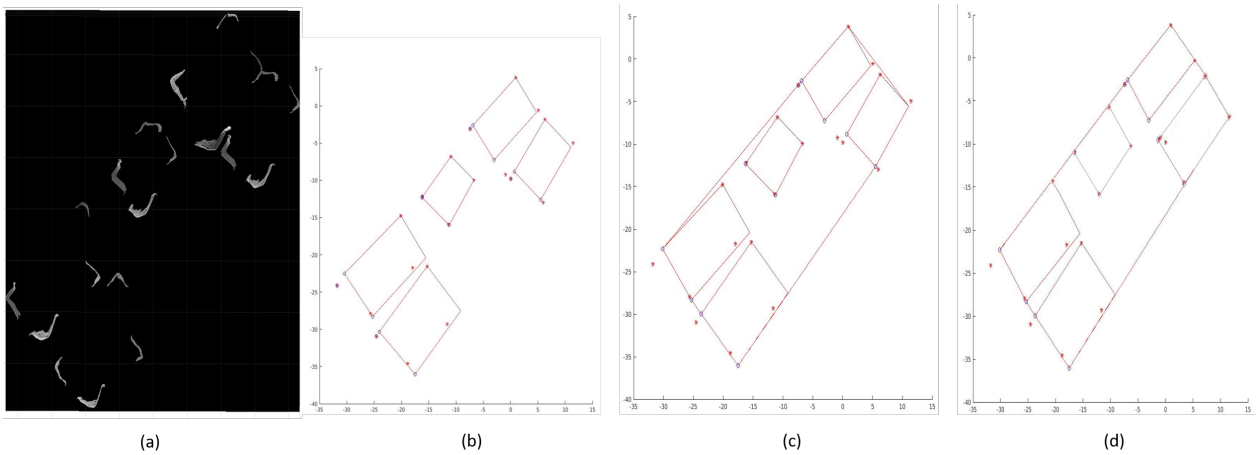



**Figure 7.13 :** Step by step generation of the floor plan of an indoor scene.

The further steps involved in the floor plan's regularization include (1) generating boundaries for all the rooms and (2) a postprocessing step, which aligns all the rooms with the boundaries generated. Algorithm 5 and Alg. 6 depicts the process of finding a boundary for the regularized layouts and their postprocessing to align them along the boundary polygon, respectively. Algorithm 5 identifies the points for each room polygon inside the boundary polygon or on the boundary so that the room polygons could be aligned with the boundary. Points that are supposed to be on the boundary but lying inside are identified using this algorithm. In Alg. 5, line 2, a line $L_i$ is traced for each point $P_i$ to $\infty$, where line 3 checks if the intersection of line $L_i$ with the boundary of Convex hull $C_{hull}$ is an even number of times or an odd number of times. If the intersection has happened 0 or even number of times, then the point is considered to be outside the boundary, otherwise it is considered inside or on the boundary. Figure 7.12 (a) illustrates an example of such process. For each room, polygon, points closer to the boundary line are identified, and a line is drawn from that point to infinity. If the line intersects the boundary polygon 0 or even times (points A, F, G), then that point is outside the polygon. Otherwise, it is inside or on the polygon boundary (point B, C, D, E). The purpose of using Alg. 5 is to find the points which are lying inside the boundary and use them for further post-processing. If the point is identified to be inside the boundary polygon, then using Alg. 6, they are aligned to the boundary line.

Algorithm 6 shows the process of aligning the points of room polygons with the boundary polygons which are found to be inside. Figure. 7.12 (b) shows the example of the polygon $P_1P_2P_3P_4$ which is required to be aligned with the boundary line $CF$. Points $P_1$ and $P_4$ are found to be inside the boundary polygon and needs to be aligned with line $CF$. Hence they need to be replaced with points $A$ and $B$ respectively. Algorithm 6 finds the location of points $A$ and $B$ on line CF and replaces $P_1$ with $A$ and $P_4$ with $B$ by dropping a perpendicular line $P_1A$ on CF and using properties of perpendicular line segments for identifying the coordinates for $A$ and $B$. It checks the slopes of both line segments (Alg. 6, line 6 and line 7) and checks the property of slopes of perpendicular line segments to identify $(X_A, Y_A)$ and $(X_B, Y_B)$, (Alg. 6, line 8). Once identified, it replaces both $P_1$ and $P_4$ with $A$ and $B$ (Alg. 6, line 11 and line 12).

Figure 7.13 depicts the entire phase of generating a floor plan from its 2D projection to the final 2D floor plan. Figure 7.13(a) depicts the 2D projection of each room's partial point clouds on a floor after the transformation taken by ARcore data. Figure 7.13(b) shows the local and global regularized 2D layouts of each room, depicting the global relationship among them. In Fig. 7.13 (c), the boundary of the room is generated by finding a convex hull for the polygons and lines.

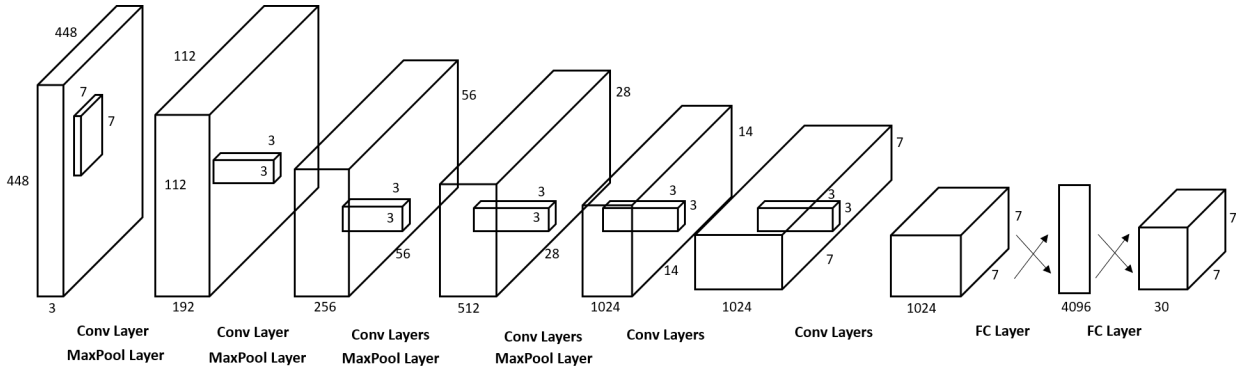



**Figure 7.14 :** A schematic representation of the network architecture of YOLO for door detection.

Figure. 7.13(d) shows the further refined and postprocessed floor plan from the 2D projections. Further on the process of postprocessing of the floor plans generated at this stage, door detection in the scenes and marking them in the floor plans is explained in the coming section.

### 7.4.6 Door Detection and placement

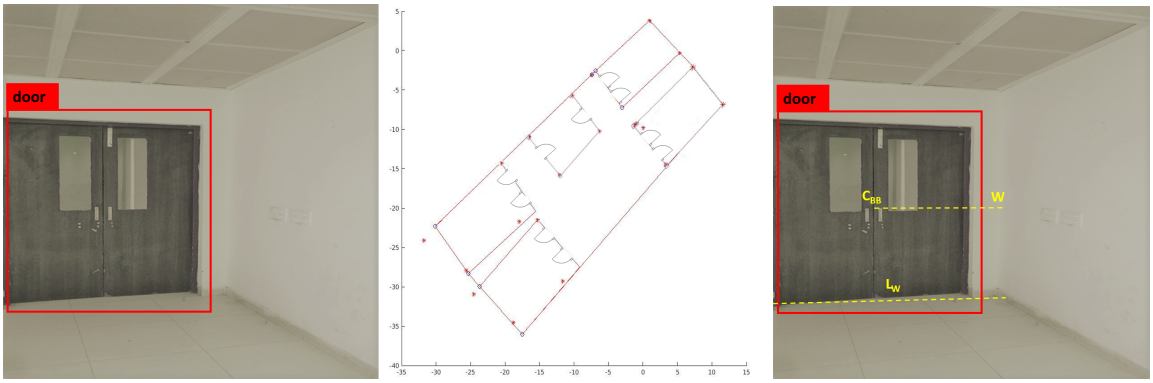
Indoor object detection such as doors, windows, or other objects in the indoor environment, from images or videos is a widely explored problem, solved using object detection networks such as YOLO, Faster-RCNN, SSD, etc. However, a dataset containing doors or windows specific to indoor scenes is not commonly available. In the current scenario, the postprocessing has been limited to the marking of doors, leaving others indoor objects due to very less variation in the other objects in the available dataset. It is challenging to generate a dataset containing doors in an indoor environment with diversity to train/ fine-tune existing networks. Hence, we used the DoorDetect dataset proposed in Arduengo *et al.* [2019]. In our work, we have trained the YOLO object detection network on the DoorDetect dataset to detect doors in the indoor scenes to complete the floor plans. YOLO's detection network has 24 convolutional layers followed by 2 fully connected layers (see Fig: 7.14). Each alternating convolutional layer has a reduction of feature space from its preceding layer. The network is pretrained on ImageNet-1000 class dataset.

The DoorDetect dataset contains 1213 images with annotated objects in an indoor environment. The door images contain various doors such as entrance doors, cabinet doors, refrigerator doors, etc. The mAP on DoorDetect dataset for YOLO came out to be 45%. Figure 7.15(a) shows the door detected in one of the partial scene images in our experiments. Figure 7.15(b) shows the door placement in the floor plan generated in the previous steps. Figure 7.15(c) shows the parameters used for the door placement. The door placement is carried out using the following equations:

$$Ratio_D = \frac{dist(C_{BB_I}, W_I)}{L_{W_I}} \tag{7.4}$$

$$dist(C_{BB_F}, W_{I_F}) = L_{W_F} * Ratio_D \tag{7.5}$$

Where $C_{BB_I}$ is the centroid of the bounding box of door detection (returned by door detection) in the real world image, $dist(C_{BB_I}, W_I)$ is the distance between $C_{BB_I}$ and $W_I$ (wall), $L_{W_I}$ is the distance between two corners of the walls in the real world image and $Ratio_D$ is the ratio




(a) Bounding box around the door. (b) Positioning of the doors on a floor plan. (c) Door centroid placement.

**Figure 7.15 :** An illustration of the performance of the door detection and placement algorithm in floor plan.

between them. $Ratio_D$ is the ratio used for marking the doors in the generated floor plans with the reference of real world images of the scene. For each individual partial scene with a door, is marked with a respective door symbol in its corresponding floor plan. Here, $L_{W_F}$ is the distance between two corners of the walls in the corresponding 2D mapping (floor plan), $dist(C_{BB_F}, W_{I_F})$ is the distance between centroid of the door symbol ($C_{BB_F}$) and wall ($W_{I_F}$) in the corresponding 2D mapping (floor plan) which is the unknown entity and needs to be identified using $Ratio_D$ to mark the doors in the floor plan. The axis of the door is kept perpendicular to the wall it belongs to. $Ratio_D$ is the ratio that is scale invariant for the generated floor plan and will remain same in the partial indoor scene image and its corresponding 2D map. In the next section, the experimental findings will be discussed in detail with their qualitative and quantitative comparison with state-of-the-art methods.

## 7.5 EXPERIMENTAL FINDINGS

### 7.5.1 Setup and Data Collection

For the experiments, we have used two hardware platforms. They are Google Pixel 2 XL and Samsung A50. We have utilized both of these mobile phones to deploy the data collection application and capture the images' for all the locations. For depth estimation accuracy analysis on our dataset, structural similarity, and peak SNR metrics are used. Also, we have used the metrics like pixel error and corner error for layout estimation accuracy analysis on our dataset. For evaluating the proposed layout estimation system's performance, GRIHA, area, and aspect ratio error metrics are used as quantitative analysis. Qualitative analysis is also done to depict the proposed system's robustness over existing Android and iOS-based mobile applications. We have also compared the performance of GRIHA on the two hardware platforms mentioned at the beginning of this paragraph. In GRIHA, we have performed experiments with a 3 set of images. The first dataset is the right wing of the ground floor of the Computer Science Department building in IIT Jodhpur, which are *Classrooms*, the second is the left wing of the same floor, which are *Labs* and the third is the first floor of the same building which are *Offices*. Figure 7.16 shows the sample images from the collected data from each category. It can be seen that the images in the dataset

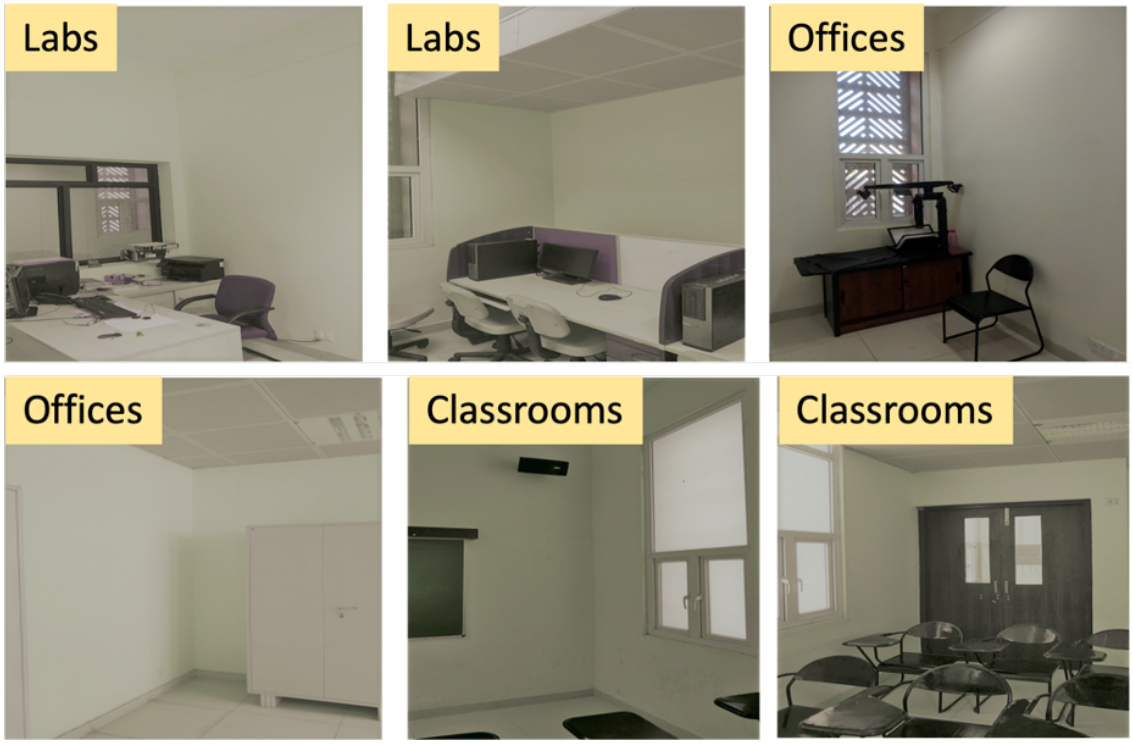



**Figure 7.16 :** Sample images from the dataset.

can contain zero to moderate or heavy occlusion with differently illuminated environments.

### 7.5.2 Depth Estimations analysis

Table 7.1 shows the performance analysis of the depth estimation step in the proposed method. Ground truth depth maps for the images in our dataset were generated using Kinect XBOX 360 depth camera. The performance evaluation is done on two metrics, Structural Similarity (SS) and peak SNR (PSNR) are defined as:

$$SS(x,y) = \frac{(2\mu_x\mu_y + C_1) * (2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_2) * (\sigma_x^2 + \sigma_y^2 + C_2)} \tag{7.6}$$

$$PSNR = 20log_{10} * (MAX_I) - 10log_{10} * (MSE) \tag{7.7}$$

In Eq. 7.6 , $\mu_x$ and $\mu_y$ are the mean intensity terms, while $\sigma_x$ and $\sigma_y$ are the standard deviations in the two image signals $x$ and $y$, $C_1$ & $C_2$ are included to avoid instability when summations of mean intensities are close to zero. Also, for *PSNR*, *MSE* is the mean square error between the reference image and generated image, $MAX_I$ is the maximum possible pixel value of the image. Lower value of SS and PSNR indicates the low quality of the generated images compared to the reference ground truth image. It can be seen that the images in Lab dataset are performing worse than other datasets given its lowest value in terms of Structural Similarity and PSNR because of the presence of a variety of occlusion creating surfaces which creates irregular planes and limited field of view, making depth estimation a challenging task. As shown in Fig. 7.16, for Labs scene images, the corners of partial scenes are highly occluded because of laboratory equipment, fixed sitting spaces, and various other immovable heavy indoor objects.

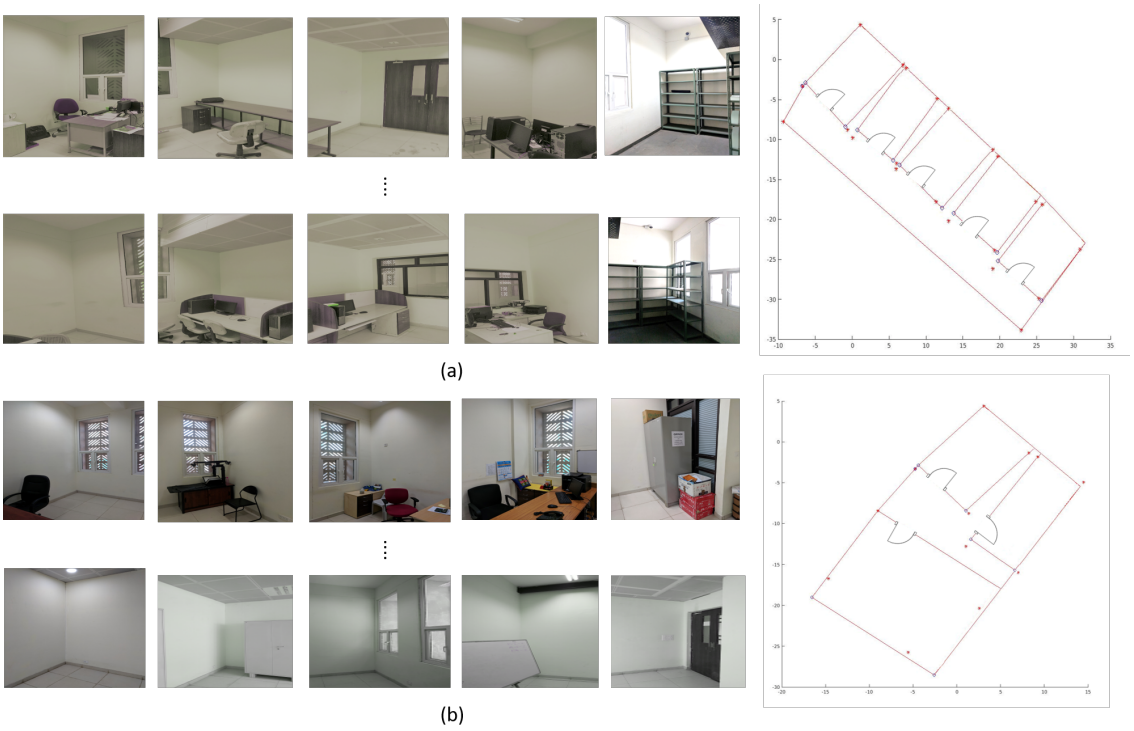



**Figure 7.17 :** Estimated layouts for Labs and office datasets.

**Table 7.1 :** Performance analysis of depth estimation on our dataset

| Method | Classrooms | Labs | Offices |
|---|---|---|---|
| Structural similarity | 0.8433 | 0.7528 | 0.8174 |
| Peak SNR | 22.46 | 17.55 | 20.7808 |

### 7.5.3 Corner and edge estimation analysis

Table 7.2 shows GRIHA's performance on estimating the corners and edges of a room. The annotations for the layouts were generated using the tool proposed in Dutta and Zisserman [2019]. The evaluation is done on two parameters, pixel error, *PE* and corner error, *CE* . Pixel error identifies the classification accuracy of each pixel with the estimated layout and ground truth and

**Table 7.2 :** Corner and edge estimation analysis on our dataset

| Scene | Corner error (%) | Pixel error (%) |
|---|---|---|
| Classrooms | 1.04 | 3.38 |
| Labs | 1.30 | 4.15 |
| Offices | 1.12 | 3.67 |

**Table 7.3 :** Quantitative evaluation of the estimated layouts for different scenes(S) and methods(M)

| **Scene** ⟶ | **Classrooms** | | **Offices** | | **Labs** | |
|---|---|---|---|---|---|---|
| **Methods**↓ | Area (E%) | Aspect Ratio (E%) | Area (E%) | Aspect Ratio (E%) | Area (E%) | Aspect Ratio (E%) |
| GRIHA | **3.12** | **2.21** | **3.25** | 2.65 | 5.59 | 3.07 |
| Magic Plan by Sensopia [2020] | 4.53 | 3.34 | 3.67 | 1.81 | 5.52 | 3.03 |
| Tape Measure by Occipital [2020] | 3.55 | 3.58 | 8.26 | 1.71 | 6.93 | **1.21** |
| Measure by Google Developers [2020b] | 7.27 | 4.06 | 6.65 | 2.93 | 6.02 | 3.07 |
| ARplan 3D by Aptoide [2020] | 3.15 | 5.20 | 4.40 | **1.62** | **4.39** | 2.87 |

averaged over all the images in the dataset.

$$PE = \frac{\sum_{n=1}^{\mathbf{I_n}} Similarity(Pixel_E, Pixel_{GT})}{n} \tag{7.8}$$

$$Similarity = \begin{cases} Pixel \in EdgeMap, & \text{if } Pixel > 0 \\ Pixel \notin EdgeMap, & \text{Otherwise} \end{cases} \tag{7.9}$$

where $n$ is the total number of images in a dataset, $Pixel_E$ and $Pixel_{GT}$ are the pixels in estimated and ground truth images. Corner error $CE$ calculates the $L_2$ distance between the estimated corner and the ground truth corner of a room, normalized by image diagonal and averaged over all the images in a dataset. Here, $Corner_E$ and $Corner_{GT}$ are the estimated and ground truth corners.

$$CE = \frac{\sum_{n=1}^{\mathbf{I_n}} Dist_{L_2}(Corner_E, Corner_{GT})}{n} \tag{7.10}$$

It can be seen that labs and office image dataset is more challenging than other datasets because of more occluded corners and edges with complex design of furniture and other experimental setups.

### 7.5.4 Comparative Result

Figure. 7.17 shows the generated layouts for our collected image datasets. Figure. 7.17 (a) shows the resultant layout for labs dataset and Fig. 7.17(b) shows for office dataset. Layout for labs dataset also includes the formation of corridors in the final layout, where the right panel shows the input stream of RGB images for the respective scenes.

A comparative study was performed with applications such as Magic Plan, presented by Sensopia [2020], Tape Measure, presented by Occipital [2020], Measure, presented by Google Developers [2020b], and AR Plan3D, presented by Aptoide [2020] with the given ground truth measurements for each dataset. For every category of images, the ground truth measurement is done by manually measuring each room's dimension in each dataset and evaluating the area and aspect ratio, respectively. Quantitative evaluation is done on the mean absolute % error for area and aspect ratio for each dataset.

$$\text{Mean Absolute \% Error (E)} = \frac{1}{\mathbf{R}} \sum_{i=1}^{\mathbf{R}} \left| \frac{x_{GT} - x_i}{x_{GT}} \right| \tag{7.11}$$

where $\mathbf{R}$ is the total number of rooms in a dataset, $x_i$ is the area/ aspect ratio of room $R_i$ and $x_{GT}$ is the ground truth area/ aspect ratio for the same.

Table 7.3 depicts the quantitative evaluation for the estimated layout for different scene datasets and other applications of Android and iOS. Results show that GRIHA performs best in terms of mean error % (E) in area and aspect ratio for Class Room dataset and area error for office dataset. For the lab dataset, ARplan3D performs best in terms of area error and Tape measure in aspect ratio error.

Table 7.4 depicts the qualitative comparison between the proposed method and other applications. Here, the number of user interactions and the amount of manual intervention required were considered based on the comparison. In terms of user interaction, our method requires only 4 interactions, i.e., images of 4 corners of a room, while other applications require a continuous scan and movement within the entire room. In terms of manual intervention, the proposed method does not require any after clicking the pictures. Whereas the other apps require manually adding the corners and height of the room. The proposed method's only requirement is to click images, while other apps take some time and manual calibration to understand the environment and features. Due to this continuous scanning and more manual intervention, techniques like Magic Plan, presented by Sensopia [2020] yield more accurate results than ours. However, in the existing apps, if some object occludes the corner, the user has to add the corner themselves. A slight user error can heavily affect the accuracy of the layout. The accuracy of the existing apps also suffers in limited salient features in different scene frames while scanning.

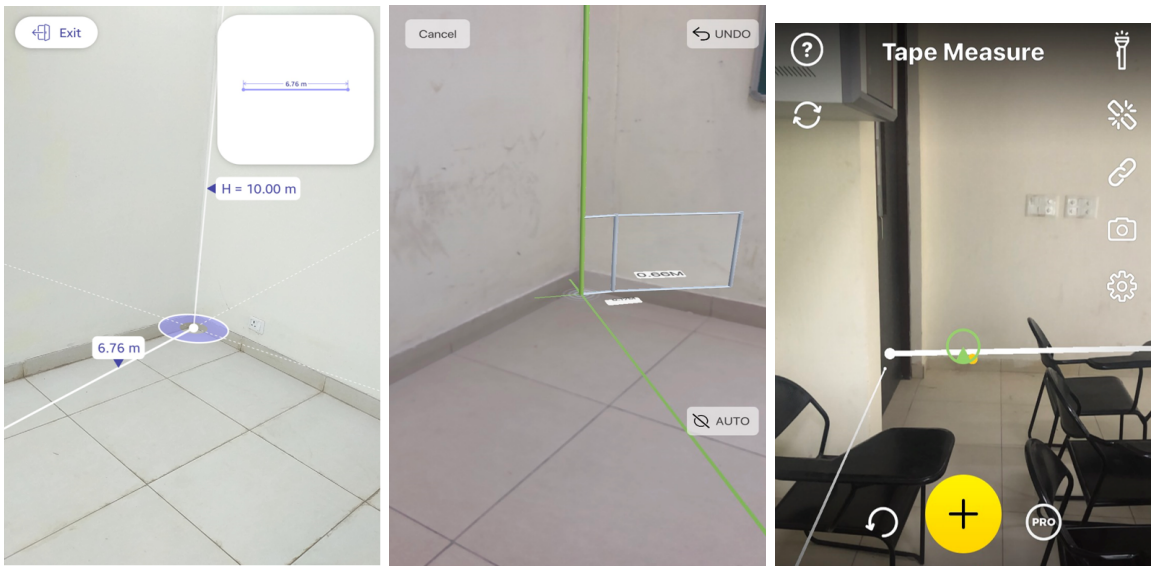
### 7.5.5 Robustness

Figure. 7.18 shows the screenshots of other publicly available mobile applications while collecting the dataset for evaluation. Figure. 7.18(a) is GUI for ARplan 3D, Fig. 7.18(b) is for Magic Plan, and Fig. 7.18(c) is for Tape Measure. There is a lot of manual interruption that requires layout estimation using these apps. For example, for ARplan 3D and Magic Plan, the rooms' corners have to be added manually, which is a very error-prone process. The person holding the mobile device has to be cautious and accurate while adding corners. Otherwise, there will be an error in the measurement of the edges of the room. Also, Fig. 7.18(c) shows that if the edges or corners of the room are occluded by furniture or other room fixtures, measuring the edges with these applications is impossible since the wall edges are invisible and have to be scanned through the furniture only.

However, in the proposed system GRIHA, these issues have been taken care of making it more robust than the existing mobile applications. GRIHA do not require any manual interruption. Hence, the possibility of introducing any manual error is ruled out. Also, it does not require the

**Table 7.4 :** Qualitative comparison of GRIHA and state-of-the-art.

| Method | User Interaction | Manual Intervention |
|---|---|---|
| GRIHA | 4 Nos. | Not required |
| Magic Plan by Sensopia [2020] | Continuous Scan | Add corners |
| Tape Measure by Occipital [2020] | Continuous Scan | Add corners |
| Measure by Google Developers [2020b] | Continuous Scan | Add corners |
| ARplan 3D by Aptoide [2020] | Continuous Scan | Add corners, height |




(a) ARplan 3D (b) Magic Plan (c) Tape Measure

**Figure 7.18 :** Graphical User Interfaces (GUI) for different layout estimation applications.

mobile device to be run through the room edges, making it easy for a user to use and robust in an occluded environment. The existing applications require some time after their launch and need a manual/automatic calibration of AR sensors by rotation and device scanning against the plane ground or wall. The automatic calibration by plane detection becomes difficult or takes longer when the room's lighting condition is not proper or there is no difference in the color of the wall or ground. However, this is not a requirement in the proposed system. The user only requires to click images of the room, making it more robust in different lighting and interior environments. Different light conditions and environments affect the quality of images and final results of layout generation. In state of the art methods, different illuminated environment plays a key role in the functioning of the method. In poor illumination, different applications discussed in the previous section are not able to extract visual features. The applications require scanning of the entire scene with a camera and require high contrast, edges and curved surfaces to detect feature points. If the captured images do not have enough feature points, then different keypoints and features are not detected. In poorly illuminated images there is a lack of contrast between two portions of a scene. Due to inconsistent indoor lighting they are not able to capture feature points and do not start functioning. In contrast, the proposed method GRIHA does not depend upon the illumination or high contrast surfaces in the captured images. Figure 7.19 depicts various scenes taken with mobile applications under different environments. Figure 7.19 (a) is taken with the Measure app, under low illumination, in a smooth surface, which could not be scanned because keypoints could not be detected. Figure 7.19 (b) is taken with Magic Plan app in low light and consistent surface, which could not be scanned because of the same reason. However Fig. 7.19 (c) is a part of dataset collected in GRIHA, which is taken in low illumination and has a consistent surface of smooth walls, and it successfully contributed in the generation of layout.

Figure. 7.20 and Fig. 7.21 shows the mean absolute error comparative analysis for area and aspect ratio across the two devices. These plots infer the robustness and platform independence of the proposed system GRIHA. The relative performance of GRIHA is similar for both devices in terms of area error and aspect ratio error. Figure 7.22 shows the comparative analysis of power consumption in mAh to a growing number of query images across devices. It can be seen that

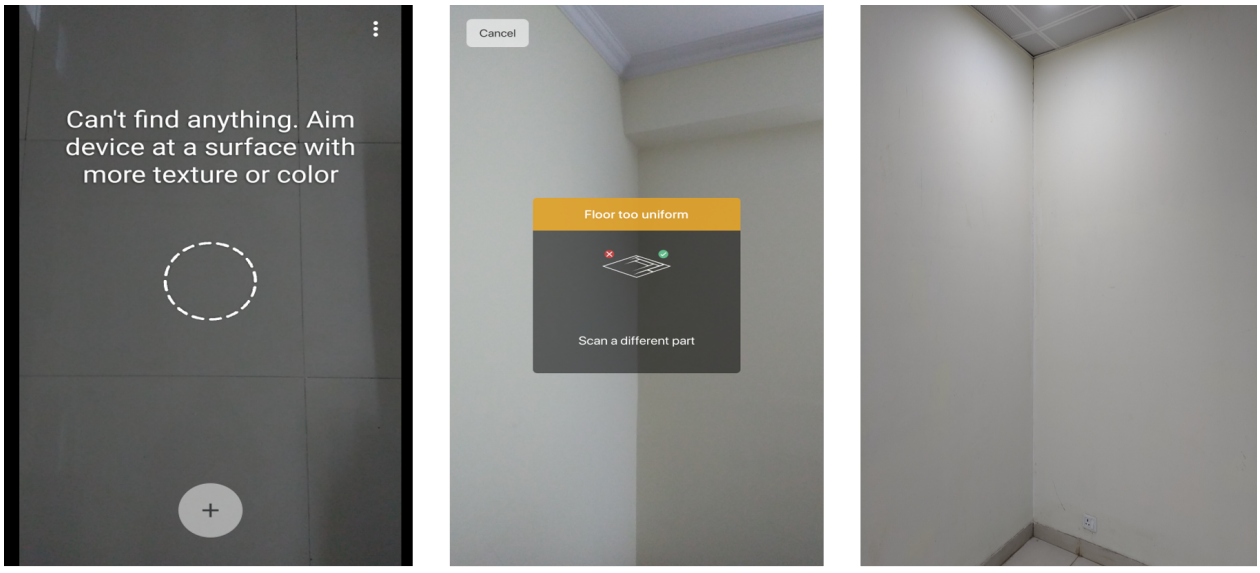



**Figure 7.19 :** Scenes captured in different environments.

Samsung Galaxy A50 is consuming more power for the proposed system and less efficient than Google Pixel 2 XL. Energy consumption is measured in terms of the battery used on each mobile device, which was recorded manually, from the start of data collection, with each query image collected.

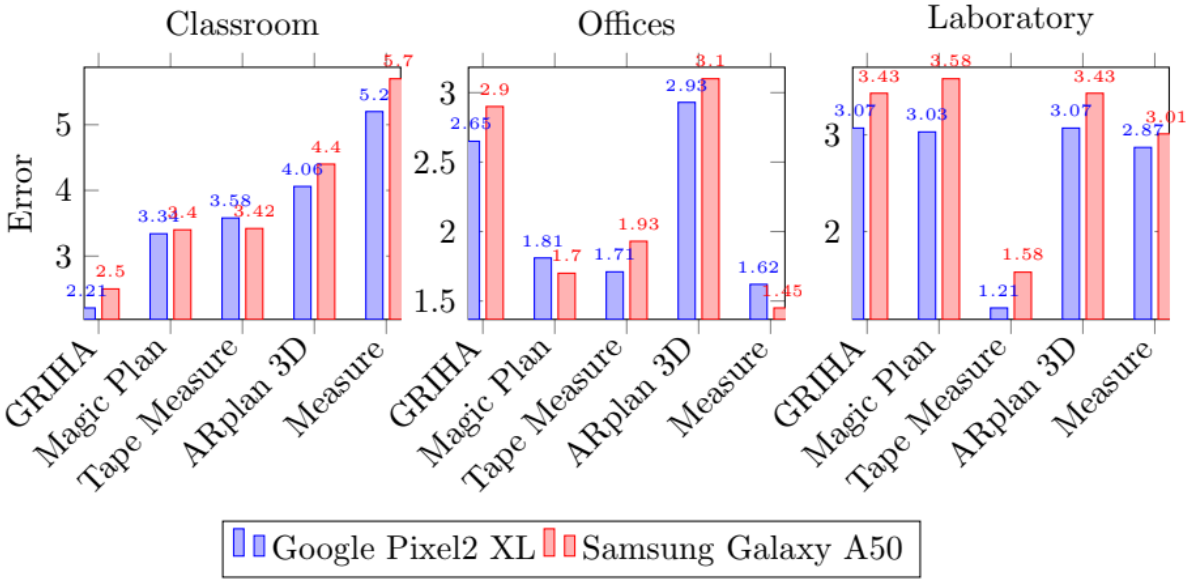



**Figure 7.21 :** Comparison of aspect ratio error across devices.

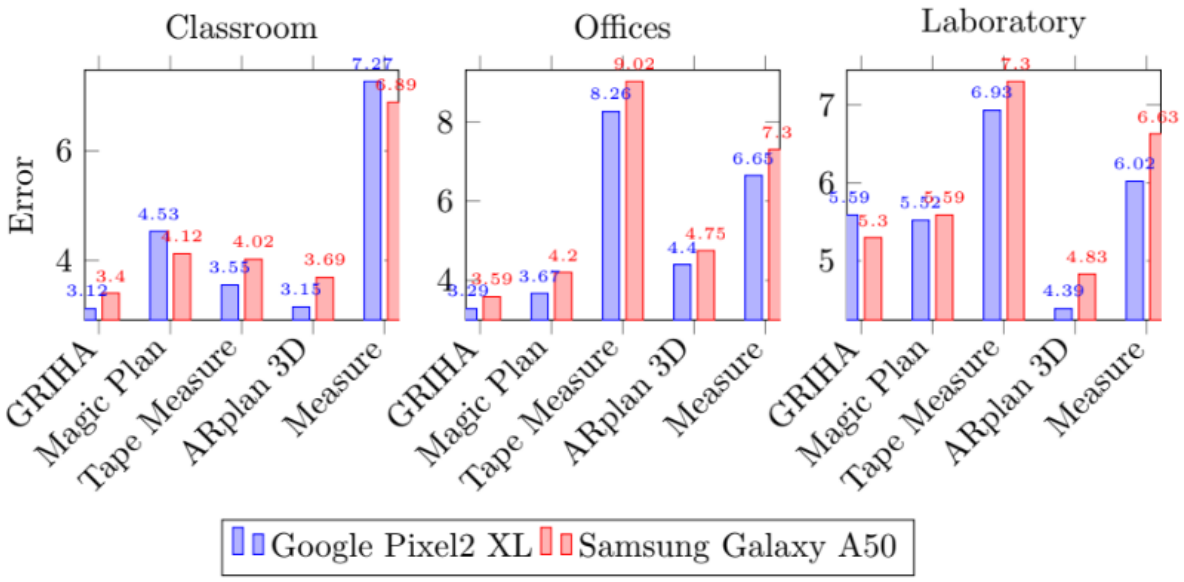



**Figure 7.20 :** Comparative analysis of area error across devices.

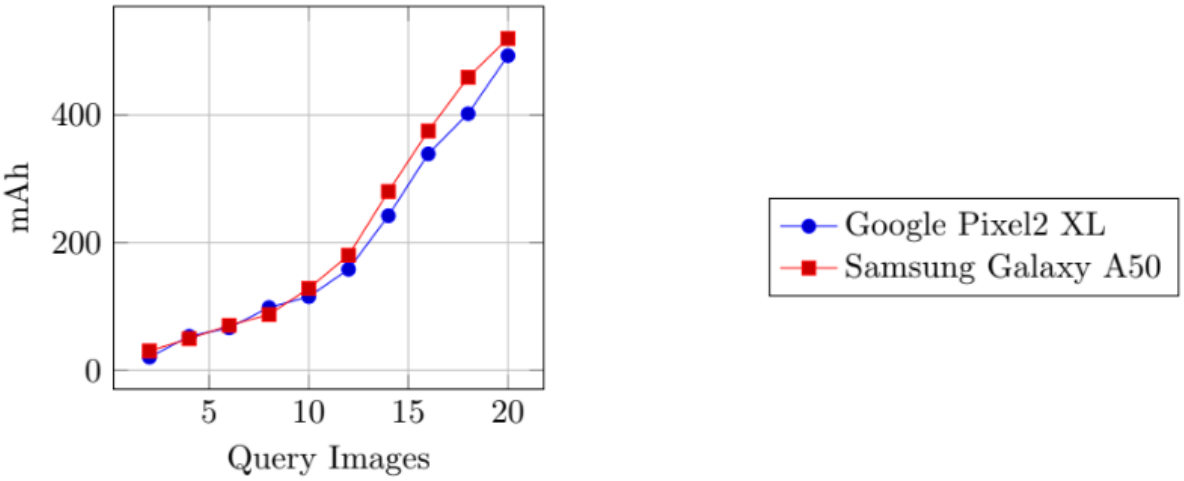



**Figure 7.22 :** Comparison of power consumption across devices.

State-of-the-art methods, such as proposed in Zou *et al.* [2018], are also used to estimate the room's entire floor plan. However, the method proposed in Zou *et al.* [2018] work on the panorama image of the scene. Compared to the proposed approach GRIHA, the method proposed in Zou *et al.* [2018] requires more information to generate a layout. GRIHA works on nonoverlapping pictures of the room, which do not have any shared features. Due to the lack of available datasets for partial scene images, we could not test our method on any publicly available datasets. We tested GRIHA on our dataset and performed experiments using the other apps under the same experimental conditions to ensure fairness. GRIHA can generate a reasonably accurate layout in terms of the error in area, aspect ratio while requiring far less user interaction and intervention.

## 7.6 SUMMARY

In this chapter, we propose a technique, GRIHA, to generate room layout or floor plans using RGB images taken from the mobile phone's camera and camera pose data given by Google ARcore. Depth estimation and 3D reconstruction of the scene are done for the RGB images, and image stitching is done by transformation coordinates given by ARcore, assuming the world to be Manhattan. The estimated layout agrees well in terms of real-world dimensions. The proposed method works well for cluttered indoor settings and occluded walls and corners. The technique requires fewer user interactions and no manual intervention, as compared to other layout generation applications. We will try to estimate the entire floor's layout by relaxing the Manhattan world assumption and for a more generalized scene in future work. The next chapter concludes the thesis, presenting the problem statement, research challenges, proposed solutions, and future directions in brief, for the proposed work in the thesis.