Beyond the Frontier: Predicting Unseen Walls from Occupancy Grids by Learning from Floor Plans

Ludvig Ericson, Patric Jensfelt

Abstract— **In this paper, we tackle the challenge of predicting the unseen walls of a partially observed environment as a set of 2D line segments, conditioned on occupancy grids integrated along the trajectory of a 360° LIDAR sensor. A dataset of such occupancy grids and their corresponding target wall segments is collected by navigating a virtual robot between a set of randomly sampled waypoints in a collection of office-scale floor plans from a university campus. The line segment prediction task is formulated as an autoregressive sequence prediction task, and an attention-based deep network is trained on the dataset. The sequence-based autoregressive formulation is evaluated through predicted information gain, as in frontier-based autonomous exploration, demonstrating significant improvements over both non-predictive estimation and convolution-based image prediction found in the literature. Ablations on key components are evaluated, as well as sensor range and the occupancy grid's metric area. Finally, model generality is validated by predicting walls in a novel floor plan reconstructed on-the-fly in a real-world office environment.**

Index Terms—**Deep Learning Methods, Planning under Uncertainty, Autonomous Agents, Learning from Experience, Map-Predictive Exploration**

I. Introduction

Human and robotic problem-solving approaches differ in dealing with the unknown and predicting the near future. Classical robotic approaches seek exactness at the cost of intuition and foresight, and on the contrary, humans do not meticulously maintain metric maps of their worlds. Even schematics and blueprints, documents explicitly intended to specify technical details, leave room for interpretation. Abstraction seems necessary for our ability to move between the specifics of reality and the generality of ideas and ideals. Replicating this ability to reason abstractly with explicit algorithms has historically proven difficult, but recent advances in learning-based approaches have opened up new avenues and great strides have been made across many subfields of robotics.

In this work, floor plans are used as the medium through which abstract reasoning is made possible. Floor plans are the architectural blueprints of our built environment, a distillate of the real world as a tidy set of shapes and symbols encoding the layouts and purposes of rooms, positions of walls, doorways, and windows. Floor plans obey rules, symmetries, and regularities that are impossible to state explicitly, often driven by aesthetic considerations rather than logic. We have previously shown that recent advances in autoregressive language models can be leveraged to produce a generative model over floor plans as sequences of vector graphic instructions [1]. By contrast to single-shot approaches such as [2], [3], an autoregressive approach casts the floor plan generation task as a series of decisions and their consequences, enabling the model to reason in steps, analogous to the chain-of-thought paradigm in large language models [4].

Autonomous exploration planning is an obvious example of a classical robotics problem where such a prediction model should be immediately applicable; however, we have previously shown that traditional non-predictive exploration planners are not well-suited to using predictions, and predictions can actually have a negative impact on exploration performance [5]. In this article, we have limited the scope to evaluating predictions by the primary variable that they affect in the exploration context: *predicted information gain*.

This paper is structured as follows. In Section IV, an model for predicting floor plans from sensor history, dubbed *Floorist*, is defined. Data modality is the main difference from our previous model [1], which dealt solely with abstract floor plans. Floorist is instead grounded in the real world by taking a partially observed environment as input in the form of 2D occupancy grids from a 360° LIDAR sensor and predicting the unobserved walls of that environment as line segments. In Section V, a dataset generation method is outlined wherein a virtual robot navigates between randomly sampled waypoints in a collection of annotated floor plans, generating input occupancy grids and their target wall segments. In Section VI, cluster-based predicted information gain is defined as in [6], [7], it is the evaluation metric used in this work, suitable for occupancy grid-based prediction models. In Sections VII and VIII, three prediction models are evaluated: Floorist, a baseline convolution-based architecture as in [2], [3], [7]–[9], and a non-predictive approach as in [10]. Floorist performance under ablation of key components is also reported, as well as its sensitivity to sensor range and occupancy grid area. Finally, in Section VIII-D, Floorist is applied to an on-the-fly floor plan reconstruction of a real-world office environment to validate its generality.

The dataset generation software along with training and inference code for Floorist is published in tandem with this work under an open-source license at `lericson.se/floorist`.

II. Related Work

The approach of predicting the unknown from occupancy grids is perhaps most common in the autonomous exploration literature. Autonomous exploration is the task of reconstruct-

This work was supported by the Swedish Research Council. All authors are with the Division of Robotics, Perception and Learning at KTH Royal Institute of Technology, Stockholm, SE-10044, Sweden. For e-mail correspondence, contact `ludv@kth.se`.

Digital Object Identifier (DOI): see top of this page.

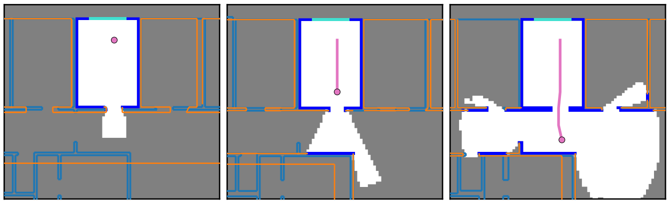

Fig. 1. Three consecutive occupancy grids with Unknown, Free, Occupied, and Window cells; Predicted walls from a Floorist model; Target walls; and Trajectory. Initially (left), few lines match up exactly, apart from the northern exterior wall which is visible, and the predicted rooms do exist though not in the exact locations predicted. In the next step (middle), more information about the corridor is observed, and the predicted segments are also improved, though the adjoining rooms are still misaligned and their doorways misplaced. Finally (right), the adjacent rooms are partially observed, and the predicted doorway alignment is now correct, and both room's widths are correctly adjusted. The images have been cropped for legibility.

ing a map of an environment, typically with no prior information about that particular environment, e.g., [6], [11], [12], though not always [13]. In *frontier-based* exploration planners, the planner considers *frontiers* by spatial clustering of the boundary between free and unknown space. A score is assigned to each frontier, and the planner navigates to the highest-scoring frontier. The score is typically a function of the travel distance and the predicted information gain, an estimate of how many bits of new information will be obtained by visiting that frontier. A popular predictive approach, e.g., [2], [3], [7]–[9], is to derive training data directly from autonomous exploration planning and then train some neural network to predict a 2D occupancy grid, computing the predicted information gain from the occupancy grid prediction. In sampling-based exploration planners such as [10], [14], [15], path and exploration planning are performed simultaneously with some variant of RRT [16]. Some sampling-based works predict information gain directly by a regression formulation, with a deep network [15] or a Gaussian process [14]. In [8], a reinforcement learning-based approach is proposed paired with a convolutional network for occupancy grid prediction; [9] extends the approach to a real-world exploration system in the form of a micro-aerial vehicle.

Another line of inquiry is the explicit reconstruction of floor plans from sensor data for architectural purposes, typically in an offline setting. In [17]–[19] use an attention-based network to approach the task of predicting a cohesive floor plan given a set of point clouds covering the entire environment. [20] solve a similar task, though using sparse multi-views instead of point clouds. Their approach is to model the environment topologically before generating a floor plan suitable to those topological constraints. In [21], the aim is somewhere between offline and online prediction, posing the problem of inferring what is behind closed doors given an occupancy grid of the observable parts of the environment, i.e., after exploration is completed. The authors demonstrate favorable performance in path planning tasks into unknown space, behind closed doors.

Some works concerning autonomous driving take a similar approach to ours in representing and predicting the environment as a set of line segments with a neural network [22], [23]. Emphasis is placed on reconstruction of the visible environment as opposed to reasoning about the unknown, though many of the technical challenges are similar. In the category of learning-based approaches on vector graphics, the approach is typically to embed an entire graphic to enable downstream tasks on that embedding, such as animation and interpolation [24], [25].

In [26], a topological approach to modeling and reasoning about indoor environnments is taken. It also introduces the *KTH floor plan dataset*, which is used for synthesizing training data in the present work. It is a collection of floor plans from a university campus, annotated with positions of walls, doors, and windows.

III. Notation

$\langle\cdot\rangle$ denotes expectation under some probability distribution. $[\cdot]$ denotes the Iverson bracket, sometimes called the indicator function. It maps the truth value of a proposition to one or zero. $\langle[x=1]\rangle$ thus denotes the probability that $x = 1$. $\lfloor x \rfloor$ denotes the floor function, i.e., the greatest integer less than or equal to x.

IV. Probabilistic Model

The problem of predicting floor plans is formulated as an autoregressive prediction task on sequences of line segment vertices. The perspective is an in-situ robot located at the origin having observed some part of its surroundings, and the task is to predict the walls of the floor plan beyond what has already been observed, as illustrated in Fig. 1. The floor plan is represented as a matrix $\boldsymbol{S} \in \mathbb{R}^{N\times 4}$ of the N target line segments from (x, y) to (x', y'), i.e.,

$$\boldsymbol{S} = \begin{pmatrix} x_1 & y_1 & x'_1 & y'_1 \\ x_2 & y_2 & x'_2 & y'_2 \\ \vdots & \vdots & \vdots & \vdots \\ x_\mathrm{N} & y_\mathrm{N} & x'_\mathrm{N} & y'_\mathrm{N} \end{pmatrix} \tag{1}$$

The vector graphic produced by $\boldsymbol{S}$ is invariant to row-wise permutations, and vertex order reversal. However, in an autoregressive regime, order does matter and must be chosen carefully for two reasons. First, any function that models an ordered sequence must by necessity subsume the ordering algorithm. Secondly, autoregression implies that later positions in the sequence are informed by earlier positions. We use a proximity heuristic where the rows are ordered by the distance from the robot to the nearest point on each line segment, as in [1]. The intuition is that segments near the robot are often partially observed, and can be predicted using the occupancy grid as grounding. Later segments that are far away and ungrounded can then be chosen to fit with the nearby grounded segments. For vertex order, a simple strategy suffices. The vertices of each segment are ordered lexicographically, i.e.,

$$(x < x') \lor ((x = x') \land (y \le y')) \tag{2}$$

A. Sequence Tokenization and Factorization

The line segment prediction problem is cast as a sequence prediction problem on the *token sequence* $\boldsymbol{t}$ where the joint distribution is conditioned on some contextual input C, that is

$$p(\boldsymbol{S} \mid C) = p(\boldsymbol{t} \mid C) \tag{3}$$

The sequence $\boldsymbol{t}$ is initialized and terminated by a 'start' and 'end' token respectively, and otherwise consists of vertex pairs, each vertex is quantized by $q : \mathbb{R}\times\mathbb{R} \to \mathcal{V}$ with the index set $\mathcal{V} = \{1, 2, \ldots, HW\}$ and

$$q(x, y) = \lfloor W(\tfrac{1}{2}H - s_y y) + (\tfrac{1}{2}W + s_x x)\rfloor \tag{4}$$

for some $H\times W$ grid at scale s_x, s_y in cells/m. The function from segments $\boldsymbol{S}$ to tokens $\boldsymbol{t}(\boldsymbol{S}) \in \mathcal{T}^{2N+2}$ with the token vocabulary $\mathcal{T} = \{\text{'start'}, \text{'end'}\} \cup \mathcal{V}$ is then defined

$$\begin{aligned}\boldsymbol{t}(\boldsymbol{S}) = (\text{'start'}, q(x_1, y_1), q(x'_1, y'_1),&\\ q(x_2, y_2), q(x'_2, y'_2), \ldots&\\ q(x_\mathrm{N}, y_\mathrm{N}), q(x'_\mathrm{N}, y'_\mathrm{N}), \text{'end'})&\end{aligned} \tag{5}$$

The joint probability is factorized autoregressively as in the recursion

$$\begin{aligned} p(\boldsymbol{t}_{[j\le i]} \mid C) &= p(t_i \mid \boldsymbol{t}_{[j<i]}, C)\, p(\boldsymbol{t}_{[j<i]} \mid C) \\ p(\boldsymbol{t}_{[j\le 1]} \mid C) &= [t_1 = \text{'start'}] \end{aligned} \tag{6}$$

where t_i denotes the ith token, $\boldsymbol{t}_{[j<i]}$ and $\boldsymbol{t}_{[j\le i]}$ denote the subsequence of $\boldsymbol{t}$ up to but excluding (or including) i. The next-token distribution is parameterized by logits from a deep network $\boldsymbol{f}_\theta$, i.e.,

$$p(t_i \mid \boldsymbol{t}_{[j<i]}, C) = \sigma(\boldsymbol{f}_\theta(\boldsymbol{t}_{[j<i]}, C))_i \tag{7}$$

where $\sigma(\cdot)_i$ is the ith element of the normalized logistic function. The function $\boldsymbol{f}_\theta$ assigns probability mass to the true token sequence $\hat{\boldsymbol{t}}$ by gradient descent on $\langle L\rangle$, the expected negative log likelihood under the data distribution, with

$$L_\theta(\hat{\boldsymbol{t}}, C) = -\sum_{\hat{t}_i \in \hat{\boldsymbol{t}}} \log \sigma(\boldsymbol{f}_\theta(\hat{\boldsymbol{t}}_{[j<i]}, C))_{\hat{t}_i} \tag{8}$$

B. Contextual Inputs

The contextual input C consists of two parts: a partial occupancy grid $\boldsymbol{M} \in \mathcal{C}^{H\times W}$, and the *visible line segments* $\boldsymbol{S}(\boldsymbol{M}) \in \mathbb{R}^{N\times 4}$ recovered from the occupancy grid. The cell labels are

$$\mathcal{C} = \{\text{'unknown'}, \text{'free'}, \text{'occupied'}, \text{'window'}\} \tag{9}$$

The label 'window' indicates that a cell contains an *exterior window*, i.e., a window facing outside the building, and hints the floor plan's perimeter. Marching squares [27] is used to find the visible line segments $\boldsymbol{S}(\boldsymbol{M})$. In regions where at least one cell is marked 'unknown', no line segment is produced.

C. Network Architecture

Following success in the language modeling domain, the choice of f_θ in Eq. (7) is a transformer encoder-decoder with multi-head attention (MHA) as in [29], illustrated in Fig. 2. The encoder is computed once per sequence to encode the contextual input, while the decoder is iteratively re-evaluated step-by-step during sampling as the generated sequence is constructed. Unlike [29], each transformer layer is a residual layer with a gated residual as in [28]. The two modalities of the contextual input are tokenized separately. The occupancy grid $\boldsymbol{M}$ is encoded using a ViT network [30], and the tokenized visible line segments $\boldsymbol{t}(\boldsymbol{S}(\boldsymbol{M}))$ are encoded with a discrete embedding with absolute position encoding. The context tokens are concatenated along the sequence dimension and cross-attended to.

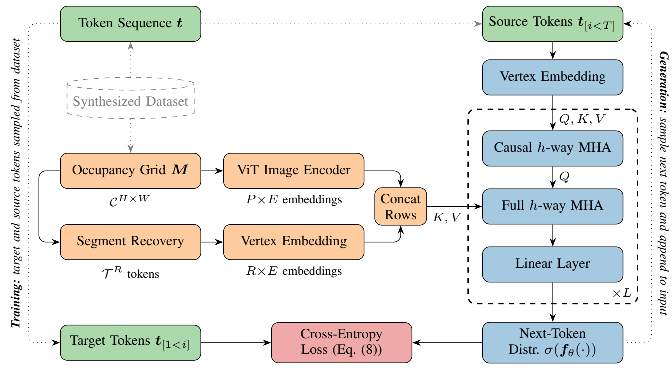

Fig. 2. The Floorist network architecture with training and generation pathways. In training, source and target token sequences are samples from the dataset, where Encoder blocks and Decoder blocks are evaluated in one pass. In generation, the source sequence starts as 'start' and the encoder side is only evaluated once. The decoder is then evaluated to obtain the next-token distribution, and a token is sampled from it and appended to the source sequence before the process repeats again. $R = |\boldsymbol{t}(\boldsymbol{S}(\boldsymbol{M}))|$ is the number of tokens in the line segments visible in $\boldsymbol{M}$, $T = |\hat{\boldsymbol{t}}|$ is the number of target tokens. P is the number of patches from ViT, E is the embedding dimension, Q, K, V are the query, key, and value matrices for a multi-head attention (MHA) layer. $\boldsymbol{t}_{[i<T]}$ denotes removal of the last token (i.e., 'end'), and $\boldsymbol{t}_{[1<i]}$ denotes removal of the first token (i.e., 'start'). This shifts the two sequences so that the target is always the next token. Note that each attention block is a gated residual connection as in [28].

1) Chromatization: Cell labels are mapped to 3-channel pseudo-colors by $k : \mathcal{C} \to \mathbb{R}^3$ with

$$\begin{aligned} k(\text{'unknown'}) &= [-1; -1; -1] & k(\text{'free'}) &= [-1; +1; -1] \\ k(\text{'occupied'}) &= [-1; -1; +1] & k(\text{'window'}) &= [+1; -1; +1] \end{aligned}$$

The exact choice of pseudo-colors is inconsequential as long as they are unique.

2) Image Encoding: The chromatized image is fed through a ViT network that produces a set of image tokens by considering the image as a set of non-overlapping patches which are linearly projected to the embedding dimension E, and then fed through a stack of self-attention layers. The ViT network is not pretrained.

3) Sequence Embedding: Each token value $t_i \in \mathcal{T}$ is mapped to a learnable embedding vector, along with a learned absolute position embedding for each i. The two embeddings are summed to form the final token embedding.

D. Image-Based Formulation

Our baseline is an image-based formulation as in [2], [3], where a chromatized occupancy grid $\boldsymbol{k}(\boldsymbol{M})$ is the input and the target is a rasterized image of the target line segments. The outputs are per-cell occupancy logits Y_{ij}. At inference time, the occupancy state O_{ij} is the maximum likelihood estimate

$$O_{ij} = [Y_{ij}(\boldsymbol{k}(\boldsymbol{M})) < 0] \tag{10}$$

Marching squares is used to recover the predicted line segments. The loss function is per-pixel binary cross-entropy.

V. Data Synthesis

Similar to our previous work, we derive training and test data from the KTH floor plan dataset [26]. Each floor plan is represented as a set of rooms, each room being a polygon of line segments categorized as walls, doors or windows. For our purposes, only two boolean properties of the segment categories are considered: *transparent* and *passable*. Doors are assumed to be open and so are both transparent and passable; windows are transparent but impassable; and, walls are non-transparent and impassable. The segments are collected into two sets, one with impassable segments used to generate paths in the floor plan, and one with non-transparent segments used for simulating sensor occlusions. Note that windows are not only exterior windows, but also glass walls which are common in office environments such as the KTH floor plan dataset.

Waypoints are first sampled by farthest point sampling [31] inside each floor plan. Paths are then generated by Dijkstra's algorithm [32] between every pair of waypoints with a cost based on the truncated distance to the nearest wall, so that some minimum wall clearance is maintained where possible. A virtual sensor is then simulated along each path at a predefined step length, yielding a partial occupancy grid at each step of the path built from the scans up to that step by ray marching. Incident cells are first marked 'free', and the terminal cells are then marked 'occupied' or 'window' if the ray is a *hit*, i.e., terminated before the sensor's maximum range. Each occupancy grid is accompanied by its target line segments from the floor plan. The target segments are filtered by removing subsegments that lie inside 'occupied' cells.

A. Axis Alignment

The occupancy grid axes are automatically rotated to coincide with the visible line segments, which adds invariance in the world-to-robot rotation up to an integer multiple of $90°$. Alignment also improves fidelity, as straight lines in the environment are rotated to coincide with either the row or column axis of the occupancy grid which minimizes discretization error, which is important for recovering the visible line segments. Following [33], a histogram is computed over the angle of the line passing through each pair of neighboring LIDAR points, modulo $90°$. The robot-to-surroundings rotation α is then computed by an average of the histogram's mode and the two adjacent bins, weighted by frequency. Finally, the LIDAR scans and target line segments are rotated by $-\alpha$ to align them to the occupancy grid axes before the occupancy grid is rendered.

B. Subdividing Segments

Subdividing segments serves to simplify the prediction problem by allowing long segments to be predicted piece by piece. The subdivision algorithm must be reproduced by the model function f_θ, so it should be simple and *predictable*. Furthermore, increasing the sequence by padding tokens improves model performance [34], [35], suggesting that shorter segments should improve performance, ceteris paribus. In practice, it becomes a compromise between model performance versus inference speed and memory usage.

Our solution is to subdivide the segments by superimposing a 21×21 regular grid and cutting the segments where they intersect this grid. The effect is that from the model's perspective, the segments are always subdivided at the same predefined coordinates.

VI. Predicted Information Gain

Evaluating generative models is notoriously difficult as there is typically no way to quantify the quality of novel generations, and the measure of quality is often subjective. A common approach to this issue is to evaluate the generation quality by measuring feature statistics from a different neural network [36], [37]. As argued by [38], generative models are best evaluated by an intended application, which in our case is to predict information gain in frontier-based autonomous exploration. To define information gain, a probabilistic interpretation must be made of the occupancy grid $\boldsymbol{M}$. For this definition, independence between cells is assumed, and a categorical probability distribution is defined for the label of each cell by

$$p(M_{ij}=k)=\begin{cases}|\mathcal{C}|^{-1} & \text{if } M_{ij}=\text{`unknown'}\\ [M_{ij}=k] & \text{otherwise}\end{cases} \tag{11}$$

If a cell is unknown, its label distribution is uniform, otherwise, the distribution is an indicator function of the label, with $M_{ij}\in\mathcal{C}$ as defined in Eq. (9). Let $\boldsymbol{M}'$ denote the occupancy grid $\boldsymbol{M}$ after sensor data integration at some frontier. Information gained in $\boldsymbol{M}'$ given $\boldsymbol{M}$ is then defined as

$$I(\boldsymbol{M}'\mid\boldsymbol{M})=\sum_{ij,k}p(M'_{ij}=k)\log\frac{p(M'_{ij}=k)}{p(M_{ij}=k)} \tag{12}$$

Notice that terms where $M_{ij}=M'_{ij}$, or $M'_{ij}=$ 'unknown', become zero. Assuming known cells in $\boldsymbol{M}$ remain the same label, then the only non-zero terms will be those where M_{ij} is unknown and M'_{ij} is known. In other words, Equation (12) is proportional to the number of unknown-turned-known cells, which is the definition of information gain used in this and other works, e.g., [7].

Each occupancy grid cell is classified as either a frontier or non-frontier based on its 4-connected neighbors. Specifically, a cell is deemed a frontier cell if it is free, and amongst its four neighboring cells there exists both free and unknown cells. Following this classification, the DBSCAN algorithm [39] is used to group these frontier cells into frontier clusters. The frontier location is the location of the cell closest to the average cell location.

VII. Experimental Setup

Since the KTH floor plan dataset represents rooms as closed polygons, there are no connecting line segments where rooms connect, e.g., doorways, making it possible to traverse and see inside the walls at these doorways. A heuristic is used to insert wall segments in such situations: find a bijection from each door segment uv to its nearest door segment $u'v'$ and insert two wall segments uu' and vv' so that a

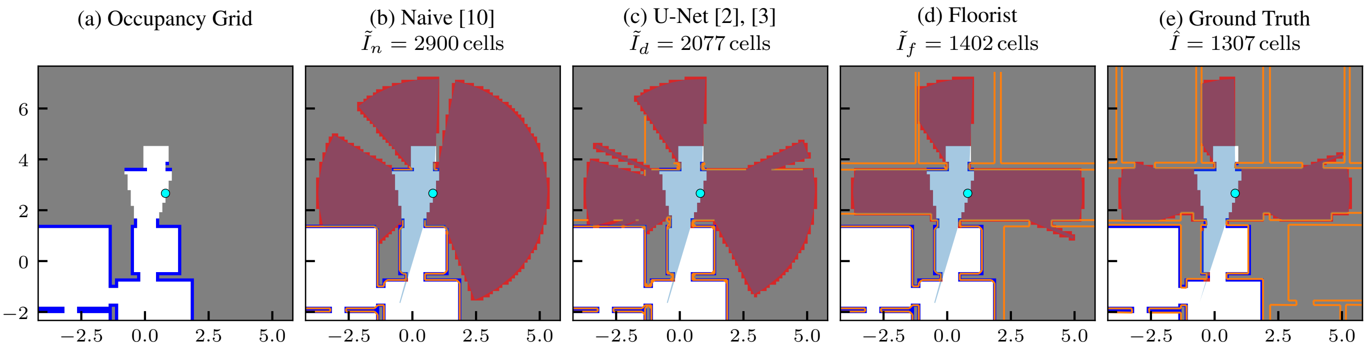

Fig. 3. Illustration of how information gain is computed for a Frontier location found along Trajectory. The initial occupancy grid $\boldsymbol{M}$ is shown in (a), while (b) to (e) show the occupancy grid $\boldsymbol{M}'$ after a simulated sensor scan in the predicted environment has been integrated into $\boldsymbol{M}$, with Information gain cells, Sensor scan, and Walls. Occupancy grid colors as in Fig. 1. In (b), walls are extracted from (a), corresponding to the typical way information gain is estimated for a frontier in non-predictive autonomous exploration [10]. In (c), walls are predicted by a model using (a) as input to a U-Net predictor as in [2], [3], and in (d) with the method proposed in this paper. In (e), the ground truth walls are used. In this example, the naive $\tilde{I}_n$, convolutional $\tilde{I}_d$, and our approach $\tilde{I}_f$ differ from the ground truth $\hat{I}$ by 1593, 770, and 95 cells (relative difference 122 %, 58.9 %, and 7.26 %) respectively.

quadrilateral is formed, if $\|uu'\| + \|vv'\| < 2\,\mathrm{m}$. After this, each set of segments is canonicalized by first merging nearly identical vertices (within $1\,\mathrm{mm}$), then joining overlapping line segments, and finally removing vertices that are not corners; i.e., vertices v with exactly two neighbors u, w that form a straight line uw through v.

The generated paths are filtered by length and number of turns, only keeping paths between $5\,\mathrm{m}$ to $100\,\mathrm{m}$ long and having at least 3 turns. The virtual sensor returns 720 points per scan, and a scan is obtained every $80\,\mathrm{cm}$ along the path. A cell is 'window' if a hit inside it is within $10\,\mathrm{mm}$ from the nearest exterior window segment. A window segment is classified as exterior if both its vertices are within some threshold distance ($100\,\mathrm{mm}$) from the building perimeter.

Frontier clusters with less than 3 cells are excluded from evaluation, as are those whose center lies within 5 cells of the edge of the occupancy grid. Cluster larger than 30 cells are divided into subclusters by k-means [40].

There are on average 75 context line segments and 165 target line segments per sample in 6 392 919 samples from 164 floor plans. Each occupancy grid represents a $15 \times 15\,\mathrm{m}^2$ area in 121×121cells. The vertex quantization function Eq. (4) is identical in scale and size to the occupancy grid, i.e., $\frac{H}{s_y} = \frac{W}{s_x} = 15\,\mathrm{m}$ and $H = W = 121$. The maximum sensor range is $r = 4.5\,\mathrm{m}$ unless otherwise stated.

A. KTH Dataset Considerations

It is important to note that the KTH floor plan dataset contains duplicated floor plans, and that there are strong similarities between floors of a single building. It would therefore be an error to simply shuffle the entire dataset, as this would contaminate the training set with test data. As in [1], we deduplicate the KTH floor plan dataset and split it into training and test *before* shuffling, and adjust the splitting point such that a single building's floor plans are all exclusively training or test data, preventing any cross contamination.

B. Model Hyperparameters

In all experiments, the embedding dimension $E = 512$. AdamW [41] is used with weight decay 10^{-2}, learning rate 10^{-4}, and batch size 6. A $10\,\%$ dropout is also applied. Since the dataset is large compared to the model size, the network can be trained indefinitely without overfitting; training was stopped when the validation loss stopped decreasing. The occupancy grids, visible line segments, and target line segments are jittered by one of the eight symmetries of the square, i.e., a random combination of mirroring and rotating. Performance was not improved by relaxed regularization.

Top-p sampling [42] is used for all generation, with $p = 80\,\%$. Other choices of p were evaluated but yielded worse results. No temperature scaling or repetition penalty is applied.

VIII. Experimental Results

In Fig. 4, examples of the generated wall segments are shown for six continuous steps of randomly sampled trajectories in the test set. Frontier locations used when evaluating predicted information gain are also shown. Results from a model with larger area but same network size is also shown, showing less coherent output, as the prediction task is harder; e.g., there are segments passing through free space, a mistake that the smaller model does not tend to make. Finally, results from the image-based predictor are also presented, illustrating the difficulty in a pixel-wise approach to occupancy regression.

A. Predicted Information Gain

Predicted information gain is evaluated using the following different sets of wall segments, illustrated in Fig. 3:

1) Naive: Assume that only what has been observed to be occupied is occupied, i.e., no segments are predicted and only the walls visible in the occupancy grid occlude the sensor. This is a common approach in non-predictive exploration planning [10], and is equivalent to assuming that unknown space can be considered sensor transparent.

2) U-Net: U-Net [43] is a fully-convolutional image segmentation network, chosen as a baseline as it has been used in previous work on occupancy grid prediction for autonomous exploration [2], [3]. The network was not pretrained. Several network architectures were evaluated for image-based prediction [8], [30], [43]–[46]. All architectures reached convergence at the same loss value, suggesting that convergence is caused by the intrinsic difficulty of image-based prediction.

3) Floorist: A $121{\times}121$ vertex grid with $L = 6$ attention layers, 8-way MHA, 4096 GeLU units, and $E = 512$ embedding dimensions. The image encoder is a three-layer ViT [30] with patch size 6 (i.e., $P = 400$), 8-way MHA, and 4096 hidden units.

4) Ground Truth: Using the actual walls of building, i.e., true information gain.

In Fig. 5, the cumulative distribution function of absolute errors in predicted information gain is presented for the three evaluation targets on test data. Image-based prediction provides substantial improvements over naive non-predictive estimation, and our sequence-based approach in turn provides substantial improvements over image-based prediction. $95\,\%$ confidence intervals for the median absolute error (MAE) are 1195 ± 2, 452 ± 1, and 236 ± 1 cells. The intervals are computed by bootstrapping with 1000 trials. A *two-sample Kolmogorov-Smirnoff test* (KS) indicates whether a pair of CDFs differ significantly by measuring the largest vertical gap. Image-based predictions are significantly more accurate than naive estimation (ΔMAE $745\,\text{cells}$, KS $34.5\,\%$), and Floorist predictions are in turn significantly more accurate than image-based prediction (ΔMAE $217\,\text{cells}$, KS $16.2\,\%$).

B. Scale and Sensor Range

To assess the impact of scale, the information gain evaluation is performed on a model with twice the area, $30{\times}30\,\mathrm{m}^2$. The occupancy grid size is limited by GPU memory, and remains $121{\times}121$ cells. The resolution is therefore four times lower ($16.27\,\text{cells}/\mathrm{m}^2$ vs $65.07\,\text{cells}/\mathrm{m}^2$). The larger area resulted in $75\,\%$ longer token sequences, requiring significantly more GPU memory; batch size was therefore reduced to 2. Information gain is evaluated as in Section VI, with two sensor range settings: $r = 4.5\,\mathrm{m}$ and $r = 9\,\mathrm{m}$. To ensure the results are comparable, information gain is computed at the base resolution. The MAE of each model at each setting is reported in Table I, with the frequency of overestimating and underestimating of the information gain. The first three rows are the results reported in Section VIII-A. The performance of the larger-scale model is slightly worse than the base model in the shorter range setting (ΔMAE $40\,\text{cells}$). The large-scale model is significantly better than the base model in the 9-meter case (ΔMAE $679\,\text{cells}$), because the longer sensor range often reaches outside the $15{\times}15\,\mathrm{m}^2$ prediction area of the base model, giving an overestimating effect similar to naive estimation. This is reflected in the sharp increase in frequency of overestimation.

C. Model Ablations

We ablate some key components of our model and evaluate the impact on its performance on the test set in terms of the loss and the accuracy of the maximum likelihood estimate. The following ablations are reported in Table II: 'window' labels replaced by 'occupied' in training and test occupancy grids; embedding weights of visible segments is shared with

TABLE I
EVALUATION OF SCALE AND SENSOR RANGE

Range (m) r	Area (m^2)	Median (cells) $\lvert\tilde{I} - \hat{I}\rvert$	Under (%) $\langle[\tilde{I} < \hat{I}]\rangle$	Over (%) $\langle[\tilde{I} > \hat{I}]\rangle$
4.5	Naive	1195	0.00	98.0
	U-Net	452	11.3	85.9
	$15{\times}15$	**236**	28.5	66.5
	$30{\times}30$	276	34.0	58.7
9.0	$15{\times}15$	1475	19.0	80.1
	$30{\times}30$	**796**	31.5	66.0

TABLE II
EVALUATION OF MODEL ABLATIONS ON TEST SET

Ablation	Loss (bits) $\langle L\rangle$	Accuracy (%) $\langle[\hat{t} = t^*]\rangle$
Floorist $15{\times}15\,\mathrm{m}^2$	1.13	81.7
No Window Labels	1.17 (+0.043)	81.3 (-0.38)
Shared Vertex Embedding	1.20 (+0.065)	80.8 (-0.87)
No Context Segments	1.23 (+0.103)	80.1 (-1.55)
No Image Encoder	1.40 (+0.267)	78.1 (-3.54)

Values in parentheses are relative to the first row. L is defined as in Eq. (8), and t^* is the maximum likelihood estimate.

embedding weights of the target segments; without visible segments as cross-attended tokens, i.e., only using the occupancy grid as context; and, without the image encoder, i.e., only using visible segments as context. We see that each ablation degrades performance, and that the occupancy grid context is most important. This may be because the occupancy grid is the only volumetric representation of the environment, and indicates where free versus unknown space is.

Window labels have the smallest impact on the reported metrics since windows are relatively uncommon in the dataset. Window labels were included as a cue for the building exterior, so the effect of those labels should be evaluated by evaluating if they prevent the model from predicting walls outside the building exterior. We specifically look at instances where at least one occupancy grid cell has the 'window' label, and compute the length of predicted line segments that fall outside the building perimeter as a proportion of the total predicted segment length. Window labels reduced the length of segments outside the building perimeter by $22.4\,\%$, from $10.93\,\%$ to $8.48\,\%$, indicating that the model does respond to the window cues.

D. Reconstructed Floor Plans

The software library *RoomPlan* [47] enables online reconstruction of floor plans from real-world sensor data in cluttered environments. As a test of the generality of our approach, we show that our method can use such reconstructions. RoomPlan runs on a modern smartphone using an image sensor and a solid-state LIDAR sensor, and produces a parametric 3D model of the environment as a set of objects of some predefined types: walls, doors, windows, and miscellaneous furniture. From this model, 2D line segments are derived by the orthogonal projection of the walls and windows onto the ground plane. A LIDAR sensor is simulated along the

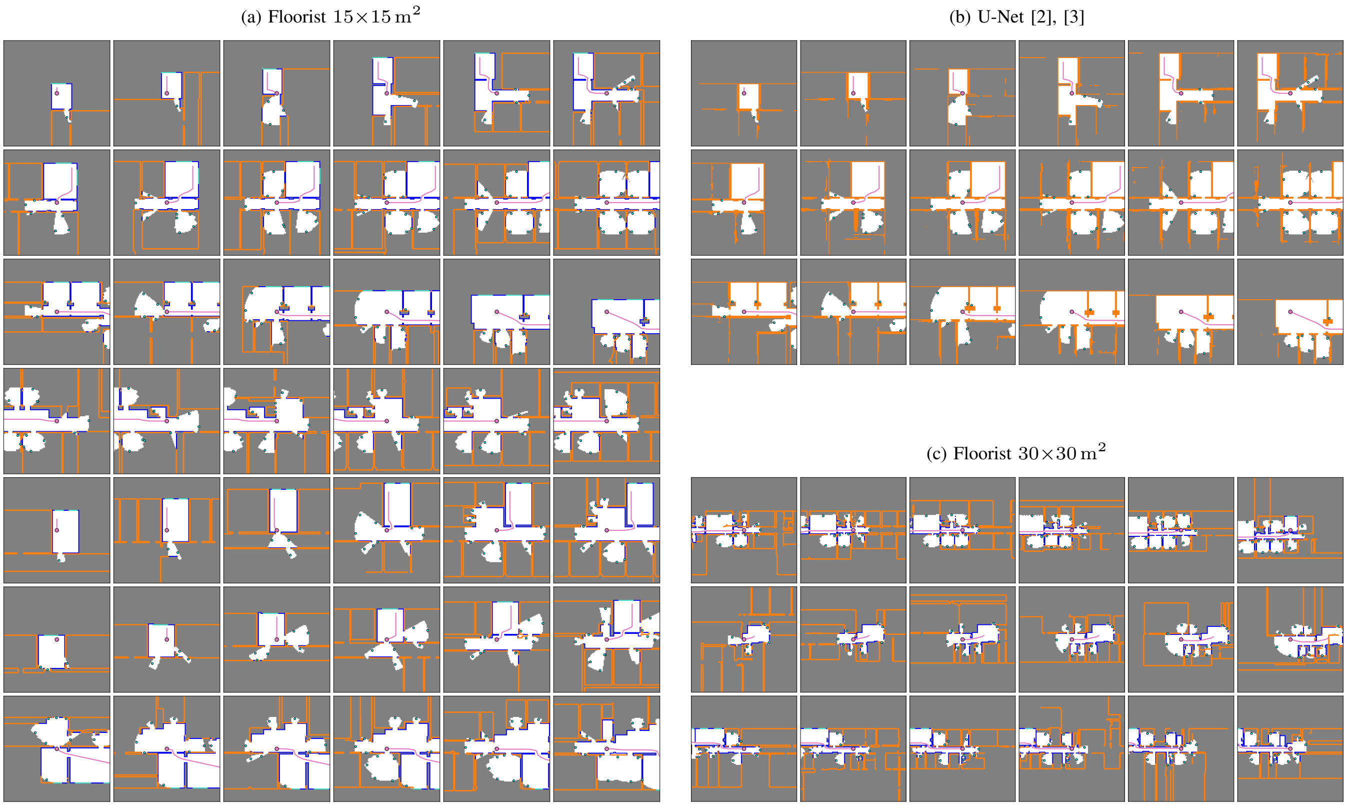

Fig. 4. Examples of predictions on random samples from the test set. Each row is a section of a single trajectory, in sequence from left to right. Frontier locations used in the information gain evaluation are also shown. Other conventions as in Fig. 3. Note that in (b), the predicted occupancy grid is shown with Occupied cells. It is advisable to use a digital document viewer to zoom the vector graphics.

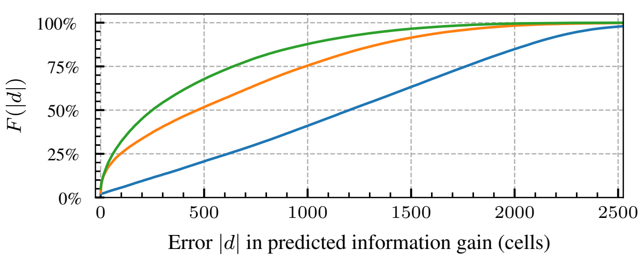

Fig. 5. Cumulative distribution function F of absolute error $|d| = |\tilde{I} - \hat{I}|$ in predicted information gain $\tilde{I}$ from the true information gain $\hat{I}$ using line segments from Naive, U-Net, and Floorist. $N = 1\,464\,140$.

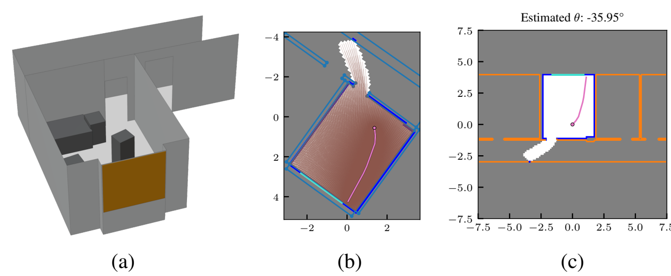

Fig. 6. Illustration of (a) a reconstructed floor plan from [47], with walls, a window, and furniture; (b) simulated Trajectory inside the floor plan derived from (a) with sensor scans; and (c) the axis-aligned occupancy grid, and an example of Predicted walls from Floorist.

trajectory, and the floor plan is predicted from the resulting occupancy grid. The reconstructed 2D floor plan, the trajectory, and a wall prediction is shown Fig. 6. The model correctly infers that there is a corridor outside, and that there is an adjoining room; however, it predicts the room to be in the middle of the corridor, not the end, as is the actual case.

IX. Conclusion

In this work, we have presented an attention-based generative model for floor plans conditioned on realistic sensor data. We have shown that our model can cope with the high dimensionality of a time-dependent occupancy grid representation with a realistic sensor model and make competent predictions as quantified by evaluating the predicted information gain. The approach offers advantages over traditional image-based predictions at the cost of longer inference time, though autoregressive sampling performance is an active area of research. We have also shown that the approach is robust enough to be used in real-world environments, demonstrated by using an off-the-shelf floor plan mapping solution as the source of floor plan data. In the future, we aim to extend our real-world demonstration to create a real-time floor plan prediction system from sensor data. Another interesting direction is adversarial multi-agent contexts where intuiting the surrounding is important, such as pursuit-evasion and other search games.

References

[1] L. Ericson and P. Jensfelt, "FloorGenT: Generative vector graphic model of floor plans for robotics," in *International Conference on Intelligent*

Robots and Systems. IEEE, 2022.

[2] Y. Tao, Y. Wu, *et al.*, "SEER: Safe efficient exploration for aerial robots using learning to predict information gain," in *International Conference on Robotics and Automation*. IEEE, 2023.

[3] K. Katyal, K. Popek, C. Paxton, P. Burlina, and G. D. Hager, "Uncertainty-aware occupancy map prediction using generative networks for robot navigation," in *International Conference on Robotics and Automation*. IEEE, 2019.

[4] J. Wei, X. Wang, *et al.*, "Chain-of-Thought prompting elicits reasoning in large language models," *Advances in Neural Information Processing Systems*, vol. 35, 2022.

[5] L. Ericson, D. Duberg, and P. Jensfelt, "Understanding greediness in map-predictive exploration planning," in *European Conference on Mobile Robots*. IEEE, 2021.

[6] B. Zhou, Y. Zhang, X. Chen, and S. Shen, "FUEL: Fast UAV exploration using incremental frontier structure and hierarchical planning," *Robotics and Automation Letters*, 2021.

[7] R. Shrestha, F.-P. Tian, W. Feng, P. Tan, and R. Vaughan, "Learned map prediction for enhanced mobile robot exploration," in *International Conference on Robotics and Automation*. IEEE, 2019.

[8] E. Zwecher, E. Iceland, *et al.*, "Integrating deep reinforcement and supervised learning to expedite indoor mapping," in *International Conference on Robotics and Automation*. IEEE, 2022.

[9] Y. Tao, E. Iceland, *et al.*, "Learning to explore indoor environments using autonomous micro aerial vehicles," *arXiv 2309.06986*, 2023.

[10] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon "next-best-view" planner for 3D exploration," in *International Conference on Robotics and Automation*. IEEE, 2016.

[11] D. Duberg and P. Jensfelt, "UFOExplorer: Fast and scalable sampling-based exploration with a graph-based planning structure," *Robotics and Automation Letters*, 2022.

[12] J. Yu, H. Shen, J. Xu, and T. Zhang, "ECHO: An efficient heuristic viewpoint determination method on frontier-based autonomous exploration for quadrotors," *Robotics and Automation Letters*, 2023.

[13] M. Luperto, M. Antonazzi, F. Amigoni, and N. A. Borghese, "Robot exploration of indoor environments using incomplete and inaccurate prior knowledge," *Robotics and Autonomous Systems*, 2020.

[14] M. Selin, M. Tiger, D. Duberg, F. Heintz, and P. Jensfelt, "Efficient autonomous exploration planning of large-scale 3D environments," *Robotics and Automation Letters*, 2019.

[15] L. Schmid, C. Ni, Y. Zhong, R. Siegwart, and O. Andersson, "Fast and compute-efficient sampling-based local exploration planning via distribution learning," *Robotics and Automation Letters*, 2022.

[16] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, 2011.

[17] Y. Yue, T. Kontogianni, K. Schindler, and F. Engelmann, "Connecting the dots: Floorplan reconstruction using two-level queries," in *Conference on Computer Vision and Pattern Recognition*, 2023.

[18] J.-W. Su, K.-Y. Tung, *et al.*, "SLIBO-Net: Floorplan reconstruction via slicing box representation with local geometry regularization," in *Conference on Neural Information Processing Systems*, 2023.

[19] J. Chen, Y. Qian, and Y. Furukawa, "HEAT: Holistic edge attention transformer for structured reconstruction," in *Conference on Computer Vision and Pattern Recognition*. IEEE/CVF, 2022.

[20] A. Gueze, M. Ospici, D. Rohmer, and M.-P. Cani, "Floor plan reconstruction from sparse views: Combining graph neural network with constrained diffusion," in *International Conference on Computer Vision*. IEEE/CVF, 2023.

[21] M. Luperto, F. Amadelli, M. Di Berardino, and F. Amigoni, "Mapping beyond what you can see: Predicting the layout of rooms behind closed doors," *Robotics and Autonomous Systems*, 2023.

[22] B. Liao, S. Chen, *et al.*, "MapTR: Structured modeling and learning for online vectorized HD map construction," in *International Conference on Learning Representations*, 2023.

[23] Q. Li, Y. Wang, *et al.*, "HDMapNet: An online hd map construction and evaluation framework," *arXiv 2107.06307*, 2021.

[24] A. Carlier, M. Danelljan, A. Alahi, and R. Timofte, "Deepsvg: A hierarchical generative network for vector graphics animation," *Advances in Neural Information Processing Systems*, 2020.

[25] P. Reddy, M. Gharbi, M. Lukac, and N. J. Mitra, "Im2Vec: Synthesizing vector graphics without vector supervision," in *Conference on Computer Vision and Pattern Recognition*. IEEE, 2021.

[26] A. Aydemir, P. Jensfelt, and J. Folkesson, "What can we learn from 38 000 rooms? Reasoning about unexplored space in indoor environments," in *Conference on Intelligent Robots and Systems*. IEEE, 2012.

[27] C. Maple, "Geometric design and space planning using the marching squares and marching cube algorithms," in *International Conference on Geometric Modeling and Graphics*. IEEE, 2003.

[28] T. Bachlechner, B. P. Majumder, H. Mao, G. Cottrell, and J. McAuley, "ReZero is all you need: Fast convergence at large depth," in *Uncertainty in Artificial Intelligence*. PMLR, 2021.

[29] A. Vaswani, N. Shazeer, *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017.

[30] A. Dosovitskiy, L. Beyer, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv 2010.11929*, 2020.

[31] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, 2017.

[32] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, 1959.

[33] G. Weiß, C. Wetzler, and E. Von Puttkamer, "Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans," in *International Conference on Intelligent Robots and Systems*, vol. 1. IEEE, 1994.

[34] G. Weiss, Y. Goldberg, and E. Yahav, "Thinking like transformers," in *International Conference on Machine Learning*. PMLR, 2021.

[35] S. Goyal, Z. Ji, *et al.*, "Think before you speak: Training language models with pause tokens," *arXiv 2310.02226*, 2023.

[36] T. Salimans, I. Goodfellow, *et al.*, "Improved techniques for training GANs," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[37] K. Pillutla, S. Swayamdipta, R. Zellers, *et al.*, "Mauve: Measuring the gap between neural text and human text using divergence frontiers," *Advances in Neural Information Processing Systems*, 2021.

[38] S. Barratt and R. Sharma, "A note on the inception score," *arXiv 1801.01973*, 2018.

[39] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *International Conference on Knowledge Discovery and Data Mining*, vol. 96, no. 34. AAAI Press, 1996.

[40] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Symposium on Mathematical Statistics and Probability*, vol. 1, no. 14. University of California Press, 1967.

[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.

[42] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," in *International Conference on Learning Representations*. Curran Associates, Inc., 2019.

[43] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015.

[44] Z. Liu, H. Mao, *et al.*, "A convnet for the 2020s," in *Conference on Computer Vision and Pattern Recognition*. IEEE/CVF, 2022.

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Conference on Computer Vision and Pattern Recognition*. IEEE/CVF, 2016.

[46] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv 1706.05587*, 2017.

[47] Apple Inc., "3D parametric room representation with RoomPlan," 2022. [Online]. Available: https://machinelearning.apple.com/research/roomplan