IEEE Robotics and Automation Letters (RAL) paper presented at
2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Spatial Graph-based Localization and Navigation on Scaleless Floorplan

Zu Lin Ewe[1], Fu-Hao Chang[2], Yi-Shiang Huang[3] and Li-Chen Fu[4]

***Abstract*— Effective navigation in unfamiliar environments remains a critical challenge for successful deployment. Current navigation methods, which rely on autonomous or teleoperated exploration and map building, pose technical difficulties for inexperienced end-users. In contrast, humans can effectively navigate using abstract floorplans, suggesting the potential for service robots to leverage similar techniques. The practical application of floorplan-based navigation, however, is currently limited by methods that require pre-exploration or floorplan with accurate measurements or scale. This paper aims to address the aforementioned challenges and investigate the feasibility of floorplan-based navigation in unfamiliar environments. Specifically, we propose a novel scale-invariant floorplan localization method, enabling navigation without relying on precise scale information. Furthermore, we introduce an incremental graph augmentation approach that enriches the floorplan representation with traversability information derived from robot observations. Finally, we develop an efficient navigation framework capable of utilizing both the inherent structure of the floorplan and real-time observations. Experimental results demonstrate that our scale-invariant floorplan localization method outperforms baseline methods in most cases when floorplan scale information is unavailable, and our graph-based navigation system exhibits superior success and efficiency as compared to grid-based counterparts. The outcomes of this research contribute to the advancement of service robot deployment in unfamiliar environments, particularly in scenarios where extensive exploration and map building may be impractical or technically challenging for end-users.**

I. INTRODUCTION

This confluence of technological progress and societal needs has promoted the widespread adoption of robots in our daily life. Yet, to ensure the successful deployment of robots, one fundamental requirement stands out: navigation capability. Generally, these autonomous systems need an accurate internal representation of their environment to execute the navigation tasks effectively. To meet this, it necessitates mapping, which is the creation of a spatial representation of the environment. Simultaneous Localization and Mapping (SLAM) [1] plays a pivotal role in achieving this, primarily in unknown or unexplored environments. The traditional methods of map building in these environments typically either involve autonomous or teleoperated exploration, which, while being effective, can be technically challenging and time-consuming.

This work is supported by the National Science and Technology Council of the Republic of China under Grant MOST 111-2223-E-002

[1]Zu Lin Ewe, [2]Fu-Hao Chang, and [3]Yi-Shiang Huang are with the Department of Electrical Engineering, National Taiwan University, 100 Taipei, Taiwan, R.O.C. [1]r10921107@ntu.edu.tw, [2]d10921027@ntu.edu.tw, [3]r12921015@ntu.edu.tw

[4]Li-Chen Fu is with NTU Center for Artificial Intelligence and Advanced Robotics, National Taiwan University, 100 Taipei, Taiwan, R.O.C. lichen@ntu.edu.tw

Interestingly, when we compare these methods with human navigation strategies, we find a stark contrast. Humans have a remarkable capability to navigate using abstract floorplans. They intuitively understand and navigate spaces by associating the tasks with the high-level spatial information provided by the floorplans, thus eliminating the need for extensive exploration. Research in cognitive science [2] indicates that humans maintain cognitive maps, *i.e.* abstract representations of spatial environments, which enable them to effectively navigate based on the high-level information such as semantics or verbal instructions.

However, despite the compelling nature of this approach, the current state-of-the-art floorplan-based localization and navigation for robots falls short in a few significant ways. Most existing methods [3], [4], [5], [6] assume the availability of floorplans with exact geometry details or scale, which may be difficult or impossible to obtain in many real-world scenarios, thus inevitably requiring pre-exploration. These limitations inhibit the practical applicability and usability of these techniques.

Furthermore, traditional floorplans often lack interior details such as furniture locations and object semantics, which can be crucial for effective navigation in real-world environments. Additionally, most existing techniques rely on metric representations for navigation, which are computationally intensive and don't scale well with environment size. Hierarchical navigation methods, particularly graph-based ones [7], on the other hand, could offer an efficient alternative by abstracting the environment into essential spatial relationships and related properties. However, such potential remains largely unexplored in the literature on floorplan-based navigation. Consequently, there is an imperative need for innovative strategies that utilize floorplans effectively, augment them with detailed interior information, and adopt efficient hierarchical navigation methods, without relying on precise scale information. This unmet need serves as the fundamental motivation for this work.

II. RELATED WORK

Robotic navigation predominantly requires a precise environmental representation, typically obtained after autonomous exploration or human supervision while running a SLAM algorithm. As we know, SLAM, a commonly employed technique in robotics for mapping and localization in unfamiliar environments, can be computationally cumbersome and necessitate complex sensor arrangements. Furthermore, autonomous exploration in large environments tends to be inefficient and time-consuming and often requires the intervention of a human operator with technical expertise.

In contrast, floorplans offer a structured representation of indoor spaces, providing a bird-eye-view (BEV) of the environmental layout. Instead of pre-exploring the environment, floorplan-based navigation capitalizes on the inherent structure and spatial information and potentially delivers more reliable navigation outcomes.

So far, several studies have proposed using floorplans for robot localization, based on RGB images [5], [6] and range/depth measurements [3], [4], [8], where floorplans were employed directly as the 2D metric maps. Mendez *et al*. [8] proposed a sensor model for Monte Carlo Localization (MCL) [9] that capitalizes on the environment's semantics, specifically doors, walls, and windows, obtained by processing the RGB images with a Convolutional Neural Network (CNN). Chu *et al*. [5], on the other hand, used MCL to estimate the 3D pose of a camera in an extruded floorplan. This model leverages a 3D metrical point cloud obtained based on monocular visual SLAM, which uses a single moving camera to construct a 3D environment. However, floorplans may not always accurately represent actual on-ground conditions, particularly in typical indoor spaces with furniture and cubicles. The discrepancies between floorplans and the real environments can lead to localization errors that the existing methods may not adequately address. So high consistency between floor plans and environments is still required for the aforementioned approaches. To counter the discrepancy issue, researchers like Boniardi *et al.* [6] have explored vision-based robot localization using floorplans, relying on edge-based room layout matching. Moreover, some studies have also tackled the floorplan navigation problem as a whole, where the floorplan is used for both localization and navigation. Li *et al.* [10], for instance, proposed a cognitive floorplan navigation method utilizing 2D floorplans as the initial topometric map. Still, the deficiency is that human operators need to pre-explore the environments to first determine the critical nodes and then collect visual features.

Furthermore, there is a need for a floorplan navigation algorithm that can handle arbitrary floorplan images, where the scale and precise measurements are unavailable. Our proposed research work holds the following contributions:

- Extract the scale-invariant geometric features from the floorplan to enable localization on floorplans without actual scale.
- Eliminate the need to collect the visual features by pre-exploring the environment as described by the floorplan, *i.e.*, rely on the floorplan for localization.
- Develop a graph-based navigation framework capable of utilizing both the inherent structure of the floorplan and real-time observations.

III. METHODOLOGY

As illustrated in Fig. 1, the proposed method consists of five main components. Floorplan Interpreter extracts geometric features and graph representation from the floorplan image. Perception processes the robot's sensor data, extracting geometry and traversability information. Localization aligns the geometric features observed by the robot with those present in the floorplan to estimate the robot's pose. Graph Builder dynamically integrates traversability information with the floorplan into a graph. Finally, Navigation leverages this graph to plan paths towards goals. Each component will be explained in detail in the following section.

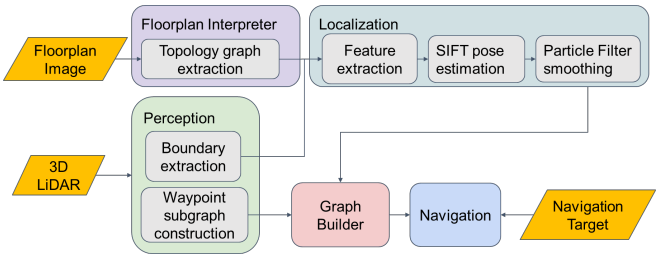


Fig. 1: System Overview

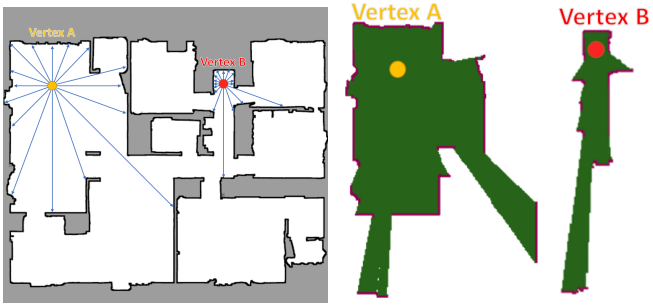
A. Floorplan Interpreter

Our work focuses on abstract floorplans without a specific scale, as shown in Fig. 2(a). These floorplans typically contain two essential types of spatial information: one for topology and another for geometry. The floorplan interpreter aims to identify locations of spatial significance, establish spatial relationships (such as traversability and relative direction) among various locations (topology), and capture the observed spatial geometry. The extracted information is integrated into a graph-like abstract map to achieve this goal, where each node refers to a specific geometric representation. Our approach here is based on a skeleton of the floorplan image, called Generalized Voronoi Diagram (GVD) [11], where nodes are extracted locations such as critical intersections of the skeleton. While these nodes establish a graph, each of them is embedded with a 2D image of a simulated laserscan (*i.e.*, a 360° laser scan of the layout with a laser sensor being placed at the node position, as shown in Fig. 3) to characterize the geometry representation of the corresponding location.

1) Floorplan Topology Graph (FTG) Extraction: First, we obtain the binary image of the floorplan via simple thresholding and then follow the work of Lau *et al.* [12], which employs a dynamic variant of the brushfire algorithm to update the cells that are affected by changes in the environment. Next, the FTG, denoted as $G^f = (V^f, E^f)$, can be extracted from the GVD, as illustrated in Fig. 2(b).

(a) (b)

Fig. 2: (a) A binary image from the Floorplan image; (b) GVD construction from the binary image. Red lines represent the one-pixel thick GVD of the floorplan.



(a) Given Floorplan (b) (c)

Fig. 3: Virtual laserscans are generated from the floorplan image. The blue lines demonstrate the virtual scan pattern. Figure (b) and (c) show the simulated laserscans from vertex A and vertex B respectively.

Given a set of points on the GVD diagram, denoted as P_{gvd}, and let $N_4(x)$ be the 4-connected neighborhood of point x on the image, then $W(x)$ is used to denote the set of neighbors of x that are also GVD points. Thus, the set of vertices V^f defining key locations is extracted from the GVD points based on the following criteria:

$$W(x) = \{y \mid y \in N_4(x) \text{ and } y \in P_{gvd}\}$$

$$V^f = \{x \mid x \in P_{gvd} \text{ and } |W(x)| \geq 3\}$$

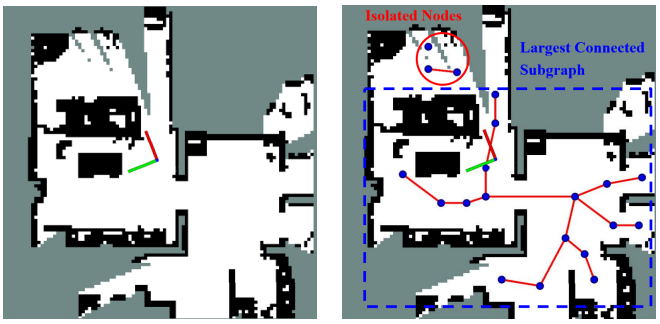
Such criteria ensure that the extracted locations represent intersections of the GVD, symbolizing critical spatial points in the floorplan. For each vertex point, we find connections to other vertices using Breath-First Search on the GVD. Edges between identified vertex pairs are created, ensuring that the graph reflects the topology of the floorplan and captures the spatial relationships between the identified vertices. We further sparsify the graph by pruning unnecessary vertices using the radius-based approach as in [7].

2) Laserscan Simulation: For each vertex in the FTG, we need a geometric description that provides sufficient information for localization. One valuable source of geometric information available on the floorplan image is the boundary or walls of the environment, which can be captured through laserscan observations. Laserscan typically measure distances on a horizontal plane by calculating the travel time of laser beams. We use the fast voxel traversal algorithm [13], a method based on raycasting, to obtain a virtual laserscan for each vertex, as illustrated in Fig. 3.

B. Perception

The floorplan image alone cannot provide a complete representation of the actual environment, as it lacks important real-time information about interior details such as traversability. In this subsection, we introduce our approach for extracting information from the onboard LiDAR sensors.

1) Egocentric Waypoint Subgraph Extraction: We aim to extract a traversability graph, referred to as the egocentric waypoints subgraph, denoted as $G^{\hat{w}} = (V^{\hat{w}}, E^{\hat{w}})$, from the 3D LiDAR observations. However, directly extracting the subgraph from a single frame of observation is highly challenging due to the viewpoint/occlusion issues. This results in inconsistencies in the extracted waypoint subgraph across frames, which will ultimately affect the usability of the traversability graph for navigation. To overcome this, our approach involves building a fixed-size egocentric 3D occupancy map that integrates and maintains information from multiple frames. The map consists of evenly spaced 3D grids that store the probability of the occupancy of the space represented by the grid. Following the common approach in [1], the grids are updated probabilistically by using raycasting of the 3D LiDAR points. Additionally, as the robot moves in the environment, we shift the data in the occupancy map such that the robot is always kept centered on the map. The grids are updated probabilistically by using raycasting of the 3D LiDAR points. The waypoint subgraph can be extracted from the map by first flattening the 3D map to 2D, referring to Fig. 4(a). Similar to the process in III-A.1, the subgraph is extracted from the GVD of the 2D map. Due to its nature, the egocentric map might contain a disconnected area that has isolated nodes/subgraphs. As shown in Fig. 4(b), we traversed over the subgraph and removed the isolated nodes by determining the largest connected subgraph.



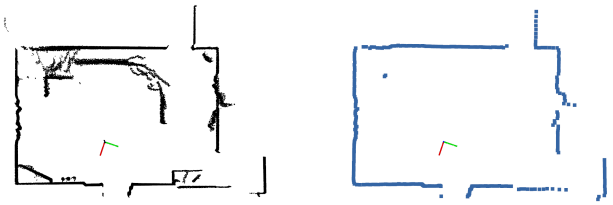
(a) Flattened 2D Gridmap (b) Waypoint Subgraph

Fig. 4: Local Map and Waypoint Subgraph. The red-green axes represent the location of the robot.

2) Boundaries Extraction: In III-A, we have extracted the environment boundary from the floorplan by simulating a virtual laserscan on the floorplan image. For subsequent localization purpose, we thus need to obtain comparable observations, and hence we extract similar laserscans from the LiDAR point clouds, which reflect the boundaries of the environment while overlooking the objects or furniture in the interior, referring to Fig. 5. We define these boundaries as the "outer boundary". The outer boundary is extracted by projecting the 3D LiDAR point cloud onto the XY plane and recording the largest distance at each angle.

C. Localization

Our localization approach can be divided into three distinct steps: *1)* Deep learning is used to determine the similarity between virtual and observed laserscans, helping identify nodes (or vertices) on the floorplan whose geometric features are similar to those of the observed ones. *2)* Using the Scale Invariant Feature Transform (SIFT) [14], relative transformations are calculated between the laserscan observed by the



(a) Bird Eye View (b) Outer Boundary

Fig. 5: Outer boundary extraction.

robot and those virtual ones associated with some floorplan nodes sharing similar geometric features. *3)* A particle filter is utilized to integrate global information derived from the estimated transformation with relative information from the robot's odometry.

1) Geometric Features Similarity via Contrastive Learning: Determining the robot's location on a floorplan based on its surrounding geometric features is challenging due to spatial arrangement and scale variations. To overcome this, we adopt a data-driven approach that involves training an image encoder to generate a 128-dimensional latent feature representation of the laserscan data. This representation captures the geometric structure while invariant to dimensions and orientations, enabling similarity prediction.

Our training approach consists of two stages: Autoencoder, and Contrastive Learning. In the first stage, an Autoencoder is trained to learn a compact representation that captures the overall geometric structure of the current laserscan observation. We employ a modified ResNet-18 [15] architecture as the encoder and decoder. It is trained using pixel-wise L1 loss. In the second stage, the encoder is further fine-tuned using contrastive learning techniques to ensure that observations with similar geometries, regardless of scale and orientation, are encoded into embeddings that are close to each other in the latent space. Conversely, observations from different geometries, such as different rooms, should be encoded into embeddings that are far apart.

During data collection, positive pairs are defined as pairs of observations that satisfy two conditions: a) their locations are close to each other, and b) they exhibit similar geometric structures. The similarity is approximated by transforming the pairs to a common coordinate system (available during data collection) and calculating the intersection over union (IoU) of the pair. Negative pairs consist of any pair of data that are not positive. The encoder is fine-tuned to maximize the cosine similarity between positive pairs and minimize the cosine similarity between negative pairs, referring to Fig. 6. The similarity scores are optimized using InfoNCE [16] loss.

2) SIFT Pose Estimation: Based on the similarity scores, we obtain the top-K similar nodes and further estimate the robot pose by finding the affine transformation between the observed laserscan and the virtual laserscan of those nodes. In our case, we are interested in the scale, rotation, and translation components of the transformation. Note that the scale here is the same for the X-axis and the Y-axis. Therefore, we assume the X:Y aspect ratio of the floorplan is close to the

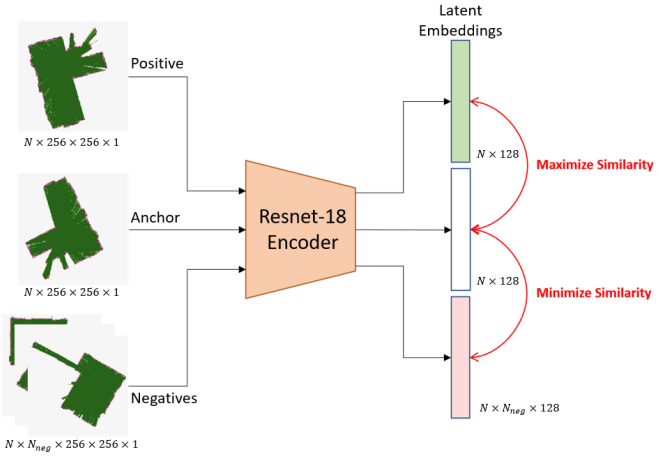


Fig. 6: Contrastive Learning Network

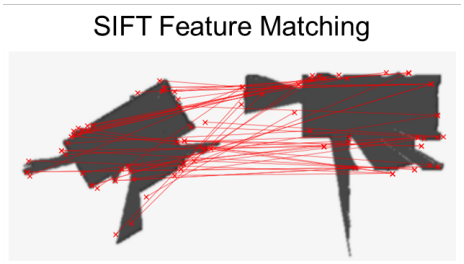


Fig. 7: Each SIFT feature is denoted by a red cross. When two SIFT features are connected by a red line, it signifies a match.

ratio in a real environment. For each observed-virtual pair of laserscan, we compute the SIFT keypoints and descriptors and perform feature matching (as illustraed in Fig. 7), where the affine transformation can be solved analytically from the matched feature pairs. The affine transformation can be expressed as:

$$\begin{bmatrix} {}^{n_i}x \\ {}^{n_i}y \\ 1 \end{bmatrix} = {}^{n_i}_{obs}\mathbf{H} \cdot \begin{bmatrix} {}^{obs}x \\ {}^{obs}y \\ 1 \end{bmatrix}, \quad {}^{n_i}_{obs}\mathbf{H} = \begin{bmatrix} a & -b & c \\ b & a & d \\ 0 & 0 & 1 \end{bmatrix}$$

where the parameters of the transformation are given by:

$$a = \text{scale} \cdot \cos\theta, \ b = \text{scale} \cdot \sin\theta, \ \text{scale} = \sqrt{a^2 + b^2}$$
$$c = \text{translation}_x, \ d = \text{translation}_y$$

Lastly, Random Sample Consensus (RANSAC) [17] is applied to eliminate outliers and compute the final transformation. To validate the estimated transformations and account for false positives, we employ a simple test. The observation is transformed to the node frame using the estimated transformation, and the Intersection-over-Union (IoU) between the observation and the node laserscan is computed. The estimated transformation is considered valid only if the IoU exceeds a certain threshold to ensure the accuracy and reliability of the localization results. The estimated robot pose can be obtained from the transformation from the robot frame to the floorplan frame ${}^{\text{fp}}_{\text{robot}}T$ as follows:

$${}^{\text{fp}}_{\text{robot}}T = {}^{\text{fp}}_{n_i}T \cdot {}^{n_i}_{obs}\mathbf{H} \cdot {}^{obs}_{\text{robot}}T$$

where the transformation ${}^{obs}_{\text{robot}}T$ represents the pose of the sensor relative to the robot which is given by its prior physical setting; the transformation ${}^{\text{fp}}_{n_i}T$ represents the pose of virtual laserscan at node i, as shown in Fig. 3(b) on the floorplan.

3) Particle Filter Smoothing: While we can identify nodes with similar features to the current observation, ambiguity may arise when there are two or more rooms with the same geometric structure. To address this issue, we employ MCL [9], which is a widely used algorithm, to integrate the global position estimate with the robot's relative odometry. However, in our case, we cannot employ the laserscan beam model or likelihood model [1] due to the unavailability of the exact floorplan's scale. To tackle this challenge, we propose a GPS-like sensor model that leverages the estimated global pose from the SIFT features to determine the weight of the particles in the localization process. The true robot pose $\mathbf{x}$ is modeled as a multivariate Gaussian distribution of a 3-dimensional random vector $\mathbf{X}$. The mean of the distribution corresponds to the estimated pose derived from the SIFT features, while the standard deviation is a predefined parameter. Since there can be multiple estimated poses (up to K), each pose contributes to its Gaussian distribution. These individual Gaussian distributions are then combined to form a multivariate multimodal Gaussian distribution, with a probability density function $p(\mathbf{x})$ that is used to update the importance weight of the particles in the particle filter. Finally, we resample the particles with probability proportional to the weight. The mean of these particles will be the final output of the localization system.

$$p(\mathbf{x}) = \sum_{k=1}^{K} \frac{1}{K} \frac{1}{\sqrt{(2\pi)^3 * \det(\mathbf{\Sigma}_k)}} * e^{-\frac{1}{2}*((\mathbf{x}-\boldsymbol{\mu}_\mathbf{k})^T \cdot \mathbf{\Sigma}_k^{-1} \cdot (\mathbf{x}-\boldsymbol{\mu}_\mathbf{k}))}$$
$$\mathbf{X}_k \sim \mathcal{N}_3(\boldsymbol{\mu}_k, \mathbf{\Sigma}_k)$$
$$\text{where } \mathbf{x} = (x, y, \theta)^\mathbf{T}$$

D. Graph Builder

This section focuses on the construction of a hierarchical graph to represent the environment, serving as a memory that captures the evolving states based on observations over time.

1) Graph Overview: As illustrated in Fig. 8, the hierarchical graph H consists of two layers: Floorplan layer graph, G^f and Waypoints layer graph, G^w. The hierarchical graph can be expressed as:

$$G^f, G^w \in H = (V, E)$$
$$V = \{V^f, V^w\},\ E = \{E^f, E^w, E^l\}$$
$$\text{where } E^l \text{ is the interlayer edges.}$$

Each layer has its unique role in capturing different aspects of the environment:

- The Floorplan layer G^f represents the graph obtained from the Floorplan Interpreter, as described in III-A, where nodes correspond to spatially significant locations and edges symbolize unverified or virtual information determined only based on the floorplan image.

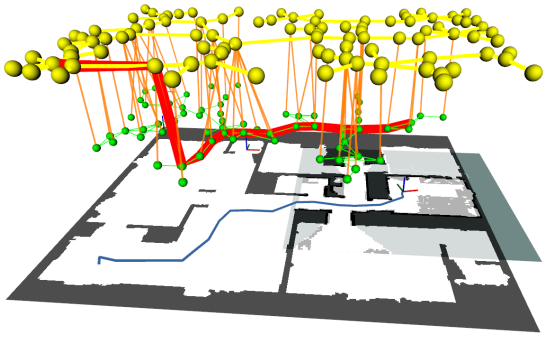


Fig. 8: Graph Overview. The orange lines represent interlayer edges. Note that green nodes and yellow nodes respectively are waypoint nodes and floorplan nodes. The red trajectory illustrates the planned navigation path, revealing that the planner prioritizes waypoint nodes over floorplan nodes unless there are no other alternatives.

- The Waypoints layer G^w incorporates data from the robot's sensors and preserves extracted traversability information within the waypoint subgraph $G^{\hat{w}}$, as mentioned in III-B.1.

TABLE I: Node attributes for each layer of the graph.

Layer	Node Attributes	Remarks
Floorplan	$(x, y) \in \mathbb{R}^2$	coordinates
	$f^{learn} \in \mathbb{R}^{128}$	learned latent embeddings
	$((u_1, v_1), \ldots, (u_N, v_N)) \in \mathbb{Z}^{+2\times N}$	SIFT keypoints in image space
	$(f_1^{SIFT}, \ldots, f_N^{SIFT}) \in \mathbb{R}^{128\times N}$	SIFT descriptors
Waypoints	$(x, y) \in \mathbb{R}^2$	coordinates

Table I provides an overview of the attributes associated with the nodes in each layer. To establish spatial relationships between the layers, interlayer edges E^l are introduced, connecting the corresponding nodes from different layers.

2) Incremental Graph Augmentation: Integrating information from observation across different time steps into the floorplan graph poses challenges due to the inconsistencies and uncertainties associated with the collected data. We propose an approach that involves adding, removing, and merging Waypoint nodes. This incremental graph augmentation process enables us to effectively incorporate new observations while preserving the integrity of the graph.

We update the Waypoint layer G^w based on the information extracted from the waypoint subgraph $G^{\hat{w}}$. An example of this process is shown in Fig. 9. To begin with, we transform $G^{\hat{w}}$ from the robot frame to the floorplan frame using the robot pose. We then use nearest neighbor algorithm [18] to identify the nearest Waypoint node $v^{w*}(v_i^{\hat{w}})$ of each subgraph node $v_i^{\hat{w}}$. For each $v_i^{\hat{w}}$, we consider adding it as a new node or merging it with the existing Waypoint node

based on the distance threshold d_{min}.

If $d(v^{w*}(v_i^{\hat{w}}), v_i^{\hat{w}}) \geq d_{min}$, add $v_i^{\hat{w}}$ as a new node to G^w.
Else, merge $v_i^{\hat{w}}$ with $v^{w*}(v_i^{\hat{w}})$.

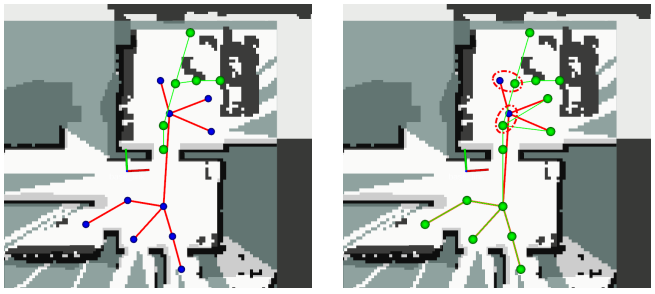


Fig. 9: Waypoints update. The left and right column shows the before and after the waypoints update respectively.

When adding new nodes to the Waypoints layer, an interlayer edge E^l is created between each new node and the Floorplan layer's node $v^f \in V^f$ that is closest to the new node. We can denote E^l as a function of v^w as follows. Thus, this spatial relationship denotes the adjacency of Waypoint nodes and Floorplan nodes, providing traversability information between the Waypoint layer and the Floorplan layer.

$$E^l(v^w) = \left(v^w, \underset{v^f}{\operatorname{argmin}}\, dist(v^w, v^f)\right), v^f \in V^f$$

Furthermore, considering the occupancy information obtained from the egocentric map in III-B.1, we remove Waypoints nodes that are positioned on occupied grid cells.

E. Navigation

Our navigation approach focuses on the point goal navigation task using a hierarchical graph. Specifically, we identify the start and end nodes on the graph through the nearest neighbor algorithm and then utilize the A* path planning algorithm [19] to determine the optimal path between these nodes. The cost and heuristic are calculated based on the Euclidean distance between nodes, promoting the selection of shorter paths for efficient navigation. Additionally, we introduce a weight factor w^f to influence node preference. For instance, if the next node on the path is a floorplan node, we adjust the cost and heuristic by multiplying them with the weight factor. The weight factor is set to a significantly large constant value, encouraging the prioritization of waypoint nodes over floorplan nodes unless there are no other viable alternatives, as illustrated in Fig. 8.

To ensure adaptability and responsiveness, our system periodically performs global path replanning at regular time intervals. Additionally, if the next node on the path is a floorplan node, it suggests the possibility of new information being available in the graph. This triggers a replanning event to incorporate any updated data and provide the most up-to-date path guidance. Once the global path is planned, it is passed to the low-level path planner for obstacle avoidance, which is not the focus of this paper.

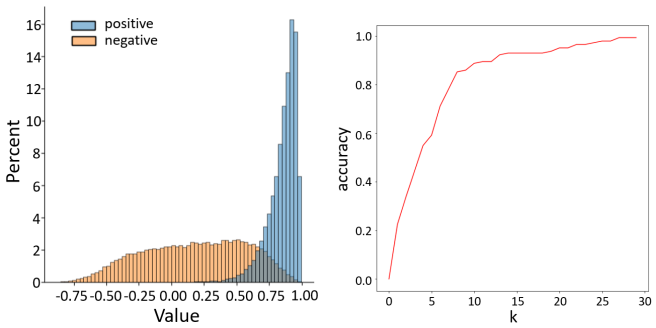
IV. EXPERIMENTS

A. Experiment Setup

To evaluate our approach, we run the simulation on Ubuntu 20.04 and an i7-12700 CPU, utilizing the Habitat Simulator [20] and the Habitat-Matterport 3D Research Dataset (HM3D) [21]. The dataset includes BEVs of the environments, which we employ as floorplans with minimal image processing. From the dataset, we randomly select a subset of scenes that cover a range of different environments and enable us to assess the performance and generalization capabilities of our approach across diverse scenarios.

B. Localization

1) Contrastive Learning for Geometric Features Similarity: We assess our deep neural network's ability to generate geometrically descriptive features using laserscan data from 27 scenes in the HM3D. We collect and preprocess 500 laserscan data points from each scene, then label each data pair as positive (IoU $>$ 0.75) or negative (IoU $<$ 0.50). After augmenting the data with random rotations and crops, we train the network on 25 scenes and validate it on 2 scenes. Analysis of similarity value distributions (Fig. 10) confirms the network's ability to encode geometric features and accurately distinguish between positive and negative pairs. Acknowledging that similarity values may not offer precise node identification, we further evaluate its ability to identify the closest nodes within a floorplan from the current location. We assess this on scenes No. 9, 48, and 569 by computing top-K accuracy, which measures how often the actual closest node ranks within the top-K nodes according to similarity values. This flexible K parameter allows us to gauge the network's node retrieval performance and guide subsequent stages. Fig. 10(b) presents our findings, revealing an approximate 90% retrieval rate at K = 10 (also the selected value for future system stages).



(a) Similarity Histogram (b) Top-K Accuracy

Fig. 10: Geometric Features Similarities Result

2) Overall Localization Result: We evaluate our approach's overall localization performance, which integrates geometric similarity prediction, SIFT pose estimation, and a particle filter-based localization system. We aim to measure the accuracy of robot localization within the environment using two key metrics: absolute trajectory error (ATE) and

relative pose error (RPE). To establish a baseline for comparison, we test various adaptations of the AMCL algorithm, which typically relies on scale information and range sensor measurements. In scaleless scenarios, we introduce inaccurate scales as perturbations, as it's challenging to estimate scale through heuristics accurately. We also explore an approach inspired by Li *et al.* [10], which uses AMCL for floorplan-based localization and navigation. They collect visual features during manual exploration and use them for pose correction when AMCL encounters discrepancies. We simulate this by manually providing accurate poses to AMCL at selected locations, mainly junctions.

Table II indicates that AMCL performs well with scale information but struggles in scaleless scenarios, accumulating significant errors over time. In contrast, our approach excels in scaleless environments, handling situations without precise measurements or scale information. While Li *et al.* (s = 2.0) shows a slightly better average ATE than our approach, closer inspection reveals its trajectories (in blue-green) are jagged with abrupt corrections, as seen in Fig. 11.

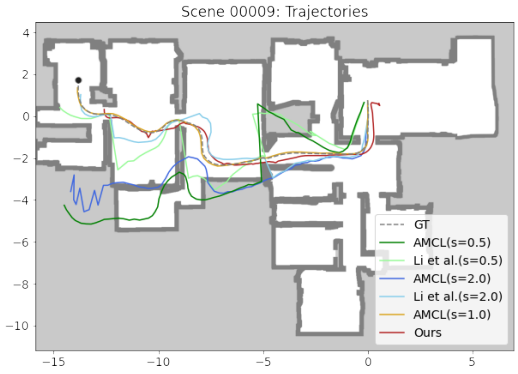
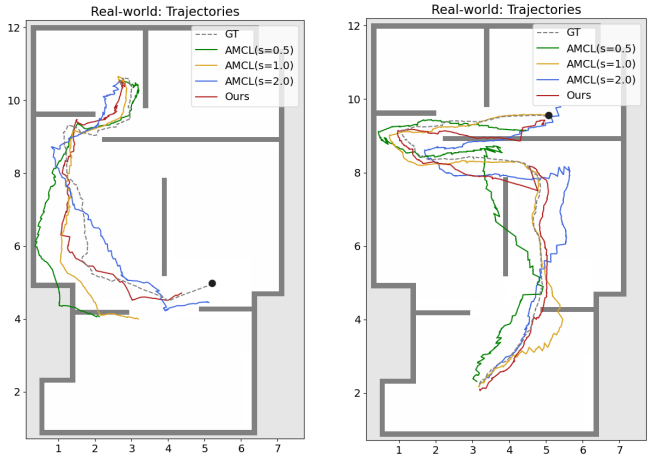


Fig. 11: Localization Results for Scene No.9. The endpoint is marked by a black dot.

3) Real-world Localization Result: We conduct a real-world experiment in Room 412 at YONGLIN Bldg. at National Taiwan University, an indoor environment with a floor space of about $77.5m^2$. The robot used in the experiment is equipped with Velodyne VLP-16 3D Lidar and i7-6700TE CPU. Fig. 12 and Table II respectively present qualitative and quantitative localization results obtained in the real-world environment. The ground-truth data were generated referencing the LiDAR odometry method of Chen *et al* [22]. From the result, it is evident that our proposed method demonstrates comparability with AMCL (s = 1.0), which possesses accurate scale information. Surprisingly, our method even outperforms this baseline in terms of the ATE metric. This discrepancy might arise due to the substantial noise present in odometry data from the real world, making it challenging for AMCL, even with a precise scale, to correct its localization. On the other hand, it also indicates that our feature-matching approach exhibits a superior ability to address these challenges and bring about improvements.



(a) Route No.1 (b) Route No.2

Fig. 12: Localization Results in a real environment. The endpoint is marked by a black dot.

C. Navigation

We evaluate our graph-based navigation in three simulated scenes, each with 8 tested routes, using success rate (SR) and Success Weighted by Path Length (SPL) [23]. The map or graph is empty initially and allowed to grow incrementally when the robot navigates through these 8 paths. Besides, to evaluate the performance within the unexplored environment, we also show the SPL of the first route, which is $\text{SPL}_{\text{first}}$. We compare our method with three alternative global planners, all using the same local planner, Model Predictive Path Integral (MPPI) controller [24] with parameters mostly consistent across methods, and ground truth odometry:

- SLAM: Build a grid map and use A* for planning. In each scene, the SLAM starts with an empty map.
- Floorplan Grid: Use the floorplan image as a grid map and use A* for planning.
- Floorplan Node Only: Similar to our approach, but only use the floorplan nodes for planning.

In Table III, our method consistently performs well even with a sparse environment representation. Though it is not as efficient as SLAM in fully explored settings, it performs better in unexplored environment, as indicated by SPL_{min}. The "Floorplan Node Only" method often fails due to paths crossing obstacles.

V. CONCLUSION

In this work, we have introduced a floorplan navigation system based on graph algorithms, addressing the challenges posed by environments lacking accurate floorplan measurements or scale information. Looking ahead, we foresee potential enhancements through the addition of semantic information, such as objects and room label, as well as the application of graph neural networks (GNNs), which could enable spatial embeddings, long-range dependencies, and advanced reasoning. This paves the way for higher-level applications like natural language-based navigation queries, facilitating human-robot interaction for navigating

TABLE II: Overall Localization Results

Method	Scene No.9 ATE↓	Scene No.9 RPE↓	Scene No.48 ATE↓	Scene No.48 RPE↓	Scene No.569 ATE↓	Scene No.569 RPE↓	AVERAGE ATE↓	AVERAGE RPE↓	Route 1 ATE↓	Route 1 RPE↓	Route 2 ATE↓	Route 2 RPE↓	AVERAGE ATE↓	AVERAGE RPE↓
Scaled														
AMCL (s=1.0)	0.056	0.031	0.079	0.033	0.114	0.039	0.083	0.034	0.640	**0.003**	0.264	**0.002**	0.452	**0.003**
Scaleless														
AMCL (s=0.5)	2.813	0.159	2.160	0.092	1.278	0.183	2.084	0.145	1.020	0.003	0.742	0.002	0.881	0.003
Li *et al.*(s=0.5)	1.062	0.312	0.756	0.269	0.611	0.372	0.810	0.317	-	-	-	-	-	-
AMCL (s=2.0)	1.768	0.162	3.926	0.137	1.334	0.109	2.343	0.136	0.765	0.003	0.602	**0.002**	0.683	0.003
Li *et al.*(s=2.0)	**0.352**	0.104	0.362	0.138	0.486	0.136	**0.400**	0.126	-	-	-	-	-	-
Ours	0.610	**0.055**	**0.220**	**0.051**	**0.389**	**0.087**	0.406	**0.064**	**0.458**	0.003	**0.327**	0.003	**0.393**	0.003

Note: The units of ATE and RPE are both meters. Values with a gray background indicate the results of real experiments.

TABLE III: Navigation Results

Method	Scene No.9 SR↑	Scene No.9 SPL↑	Scene No.9 SPL_{first}↑	Scene No.48 SR↑	Scene No.48 SPL↑	Scene No.48 SPL_{first}↑	Scene No.569 SR↑	Scene No.569 SPL↑	Scene No.569 SPL_{first}↑	AVERAGE SR↑	AVERAGE SPL↑	AVERAGE SPL_{first}↑
SLAM	**1.00**	**0.96**	0.91	**1.00**	0.95	0.85	**1.00**	0.89	0.52	**1.00**	0.93	0.76
Floorplan Grid	0.75	0.72	0.00	**1.00**	**0.97**	**0.92**	**1.00**	**0.97**	**0.95**	0.92	0.89	0.62
F Node Only	0.25	0.22	0.00	0.75	0.69	0.00	0.63	0.58	0.00	0.54	0.50	0.00
Ours (F + W Node)	**1.00**	0.95	**0.93**	**1.00**	0.92	0.76	**1.00**	0.94	0.85	**1.00**	**0.94**	**0.84**

Note: The value being 0.00 represents a failure case.

complex environments. In conclusion, our graph-based approach offers a scalable, adaptable, and efficient solution for floorplan localization and navigation, overcoming limitations of traditional methods reliant on accurate measurements.

REFERENCES

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, Aug. 2005.

[2] R. A. Epstein, E. Z. Patai, J. B. Julian, and H. J. Spiers, "The cognitive map in humans: spatial navigation and beyond," *Nature Neuroscience*, vol. 20, no. 11, pp. 1504–1513, Nov. 2017, number: 11 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/nn.4656

[3] S. Song and H. Myung, "Floorplan-based Localization and Map Update Using LiDAR Sensor," in *2021 18th International Conference on Ubiquitous Robots (UR)*, July 2021, pp. 30–34, iSSN: 2325-033X.

[4] X. Wang, R. J. Marcotte, and E. Olson, "GLFP: Global Localization from a Floor Plan," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 1627–1632, iSSN: 2153-0866.

[5] H. Chu, D. K. Kim, and T. Chen, "You are here: Mimicking the human thinking process in reading floor-plans," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2210–2218.

[6] F. Boniardi, A. Valada, R. Mohan, T. Caselitz, and W. Burgard, "Robot Localization in Floor Plans Using a Room Layout Edge Extraction Network," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 5291–5297, iSSN: 2153-0866.

[7] H. Oleynikova, Z. Taylor, R. Siegwart, and J. Nieto, "Sparse 3D Topological Graphs for Micro-Aerial Vehicle Planning," July 2018, arXiv:1803.04345 [cs]. [Online]. Available: http://arxiv.org/abs/1803.04345

[8] O. Mendez, S. Hadfield, N. Pugeault, and R. Bowden, "SeDAR - Semantic Detection and Ranging: Humans can Localise without LiDAR, can Robots?" in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 6053–6060, iSSN: 2577-087X.

[9] D. Fox, S. Thrun, W. Burgard, and F. Dellaert, "Particle Filters for Mobile Robot Localization," in *Sequential Monte Carlo Methods in Practice*, ser. Statistics for Engineering and Information Science, A. Doucet, N. de Freitas, and N. Gordon, Eds. New York, NY: Springer, 2001, pp. 401–428. [Online]. Available: https://doi.org/10.1007/978-1-4757-3437-9_19

[10] J. Li, C. L. Chan, J. Le Chan, Z. Li, K. W. Wan, and W. Yun Yau, "Cognitive Navigation for Indoor Environment Using Floorplan," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept. 2021, pp. 9030–9037, iSSN: 2153-0866.

[11] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, "Spatial tessellations: concepts and applications of voronoi diagrams," 2009.

[12] B. Lau, C. Sprunk, and W. Burgard, "Improved updating of Euclidean distance maps and Voronoi diagrams," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2010, pp. 281–286, iSSN: 2153-0866.

[13] J. Amanatides, A. Woo, *et al.*, "A fast voxel traversal algorithm for ray tracing." in *Eurographics*, vol. 87, no. 3, 1987, pp. 3–10.

[14] D. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, Sept. 1999, pp. 1150–1157 vol.2.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," Dec. 2015, arXiv:1512.03385 [cs]. [Online]. Available: http://arxiv.org/abs/1512.03385

[16] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation Learning with Contrastive Predictive Coding," Jan. 2019, arXiv:1807.03748 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1807.03748

[17] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, June 1981. [Online]. Available: https://dl.acm.org/doi/10.1145/358669.358692

[18] J. L. Blanco and P. K. Rai, "nanoflann: a C++ header-only fork of FLANN, a library for nearest neighbor (NN) with kd-trees," https://github.com/jlblancoc/nanoflann, 2014.

[19] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, July 1968.

[20] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra, "Habitat: A Platform for Embodied AI Research," Nov. 2019, arXiv:1904.01201 [cs]. [Online]. Available: http://arxiv.org/abs/1904.01201

[21] K. Yadav, R. Ramrakhya, S. K. Ramakrishnan, T. Gervet, J. Turner, A. Gokaslan, N. Maestre, A. X. Chang, D. Batra, M. Savva, A. W. Clegg, and D. S. Chaplot, "Habitat-Matterport 3D Semantics Dataset," Dec. 2022, arXiv:2210.05633 [cs]. [Online]. Available: http://arxiv.org/abs/2210.05633

[22] K. Chen, B. T. Lopez, A.-a. Agha-mohammadi, and A. Mehta, "Direct lidar odometry: Fast localization with dense point clouds," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2000–2007, 2022.

[23] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, and A. R. Zamir, "On evaluation of embodied navigation agents," 2018.

[24] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information theoretic mpc for model-based reinforcement learning," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 1714–1721.