Renderable Neural Radiance Map for Visual Navigation

Obin Kwon Jeongho Park Songhwai Oh *
Department of Electrical and Computer Engineering, ASRI, Seoul National University
obin.kwon@rllab.snu.ac.kr, jeongho.park@rllab.snu.ac.kr, songhwai@snu.ac.kr

Abstract

We propose a novel type of map for visual navigation, a renderable neural radiance map (RNR-Map), which is designed to contain the overall visual information of a 3D environment. The RNR-Map has a grid form and consists of latent codes at each pixel. These latent codes are embedded from image observations, and can be converted to the neural radiance field which enables image rendering given a camera pose. The recorded latent codes implicitly contain visual information about the environment, which makes the RNR-Map visually descriptive. This visual information in RNR-Map can be a useful guideline for visual localization and navigation. We develop localization and navigation frameworks that can effectively utilize the RNR-Map. We evaluate the proposed frameworks on camera tracking, visual localization, and image-goal navigation. Experimental results show that the RNR-Map-based localization framework can find the target location based on a single query image with fast speed and competitive accuracy compared to other baselines. Also, this localization framework is robust to environmental changes, and even finds the most visually similar places when a query image from a different environment is given. The proposed navigation framework outperforms the existing image-goal navigation methods in difficult scenarios, under odometry and actuation noises. The navigation framework shows 65.7% success rate in curved scenarios of the NRNS [21] dataset, which is an improvement of 18.6% over the current state-of-the-art. Project page: https://rllab-snu.github.io/projects/RNR-Map/

1. Introduction

In this paper, we address how to explicitly embed the visual information from a 3D environment into a grid form and how to use it for visual navigation. We present *renderable neural radiance map (RNR-Map)*, a novel type of a grid map for navigation.We point out three main properties of RNR-Map which make RNR-Map navigation-friendly. First, it is *visually descriptive*. Commonly used grid-based maps such as occupancy maps [10, 11, 17] and semantic maps [9, 19, 36], record obstacle information or object information into grids. In contrast, RNR-Map converts image observations to latent codes which are then embedded in grid cells. Each latent code in a grid cell can be converted to a neural radiance field, which can render the corresponding region. We can utilize the implicit visual information of these latent codes to understand and reason about the observed environment. For example, we can locate places based on an image or determine which region is the most related to a given image. RNR-Map enables image-based localization only with a simple forward pass in a neural network, by directly utilizing the latent codes without rendering images. We build a navigation framework with RNR-Map, to navigate to the most plausible place given a query image. Through extensive experiments, we validate that the latent codes can serve as important visual clues for both image-based localization and image-goal navigation. More importantly, a user has an option to utilize the renderable property of RNR-Map for more fine-level of localization such as camera tracking.

RNR-Map is *generalizable*. There have been a number of studies that leverage neural radiance fields (NeRF) for various applications other than novel view synthesis. The robotic applications of NeRF are also now beginning to emerge [1, 15, 28, 35, 42]. However, many of the approaches require pretrained neural radiance fields about a specific scene and are not generalizable to various scenes. This can be a serious problem when it comes to visual navigation tasks, which typically assume that an agent performs the given task in an unseen environment [16]. In contrast, RNR-Map is applicable in arbitrary scenes without additional optimization. Even with the unseen environment, the RNR-Map can still embed the useful information from images to the map and render images. The neural radiance fields of RNR-Map are conditioned on the latent codes. A pair of encoder and decoder is trained to make these latent codes from images of arbitrary scenes and reconstruct images using neural radiance fields. These pretrained encoder and decoder enable the generalization to unseen environments.

Third, RNR-Map is *real-time capable*. The majority of the present NeRF-based navigation methods require a significant time for inference because of required computation-heavy image rendering and rendering-based optimization

*This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2019-0-01190, [SW Star Lab] Robot Learning: Efficient, Safe, and Socially-Acceptable Machine Learning). *(Corresponding author: Songhwai Oh)*

steps. The RNR-Map is designed to operate fast enough not to hinder the navigation system. By directly utilizing the latent codes, we can eliminate the rendering step in mapping and localization. The mapping and image-based localization frameworks operate at 91.9Hz and 56.8Hz, respectively. The only function which needs rendering-based optimization is camera tracking, which can localize under odometry noises, and it operates at 5Hz.

To the best of our knowledge, the RNR-Map is the first method having all three of the aforementioned characteristics as a navigation map. The RNR-Map and its localization and navigation frameworks are evaluated in various visual navigation tasks, including camera tracking, image-based localization, and image-goal navigation. Experimental results show that the proposed RNR-Map serves as an informative map for visual navigation. Our localization framework exhibits competitive localization accuracy and inference speed when compared to existing approaches. On the image-goal navigation task, the navigation framework displays 65.7% success rate in curved scenarios of the NRNS [21] dataset, where the current state-of-the-art method [37] shows a success rate of 55.4%.

As RNR-Map finds a place based on the visual information of the map and the query image, we also consider a variant version of image-based localization. In real-world scenarios, there can be partial changes in the target place (changes in furniture placement, lighting conditions, ...). Also, the user might only have images from similar but different environments. We test the proposed localization framework in both cases. We find that the RNR-Map is robust to environmental changes and is able to find the most visually similar places even when a novel query image from a different environment is provided.

The contributions of this paper can be summarized as follows:

- We present RNR-Map, a novel type of renderable grid map for navigation, utilizing neural radiance fields for embedding the visual appearance of the environment.
- We demonstrate efficient and effective methods for utilizing the visual information in RNR-Map for searching an image goal by developing RNR-Map-based localization and navigation framework.
- Extensive experiments show that the proposed method shows the state-of-the-art performance in both localization and image-goal navigation.

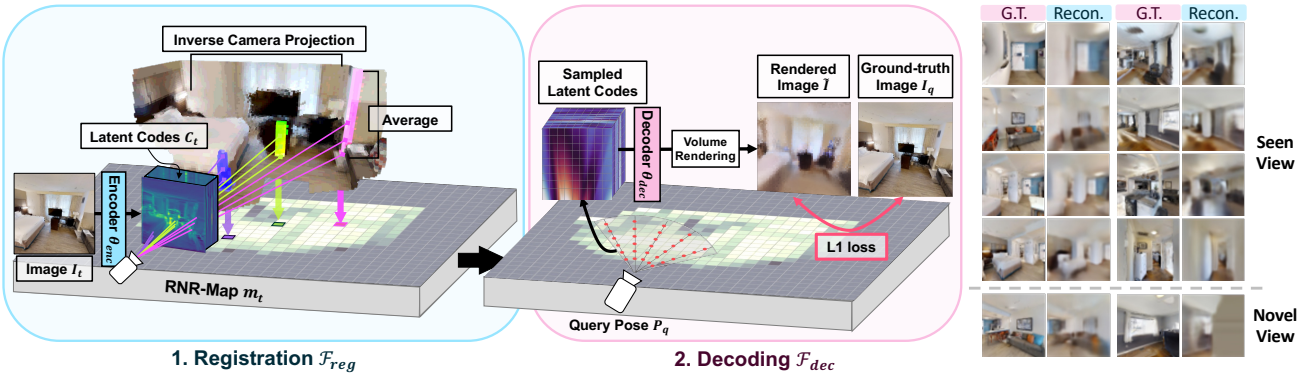
2. Related Work

Embodied AI with spatial memories. One of the central issues in recent embodied AI research is how to construct a useful memory for the embodied agent [16]. A memory that contains the navigation history, as well as information about the observed environment, is required for successful task execution. There is a large body of works using occupancy maps for visual navigation [10, 11, 17, 34]. An occupancy map expresses the environment in a grid form, and each grid cell has obstacle information about the corresponding region. An occupancy map represents the overall structure of the environment and can guide a robot to navigate the environment safely. There have been various kinds of research about building a spatial memory which contains additional information more than obstacles. The additional information can be object classes of the observed objects [7, 9, 19, 36], or implicitly learned useful information for a specific task [20, 22, 32, 33]. MapNet [22], SMNet [7] and ISS [33] have a similar mapping architecture with our method. Like our approach, they learn how to convert RGBD observations into useful latent information, and record it in the spatial memories using 3D inverse camera projection. Using the recorded latent information, MapNet [22] learns to localize the agent pose, and SMNet learns to generate a semantic map. ISS [33] is more related to ours since this method addresses scene generation and novel view image synthesis from a grid map. Our research is focused on how we can utilize such latent information for visual navigation. We develop localization and navigation frameworks which actively utilize the embedded visual information in RNR-Map.

Robotics with neural radiance fields. The neural radiance field (NeRF) [30] has gained significant popularity in various AI tasks. Not only in computer vision or graphics tasks, but NeRF is also adopted for robot applications in recent years. NeRF predicts the RGB color and density of a point in a scene so that an image from an arbitrary viewpoint can be rendered. This property enables pose estimation [1, 27, 28, 35] based on the photometric loss between the observed image and the rendered image, or manipulation of tricky objects [5, 12, 14, 23, 26]. A pretrained NeRF can also work as a virtual simulator, in which a robot can plan its trajectory [1] or can be used to train an action policy for the real-world [6]. Among the existing approaches, NICE-SLAM [42] is relevant to our work because it performs mapping and localization in arbitrary environments. NICE-SLAM builds a 3D implicit representation of the environment from image observations. The camera pose is inferred from optimizing the photometric loss between the observation image and the rendered image. Our method, on the other hand, places less emphasis on mapping quality, and it is designed for successfully completing navigation tasks. We focus on how the RNR-Map can be efficiently used for visual navigation, in terms of both speed and performance. The mapping and the target-searching function of RNR-Map are designed to operate fast enough to not hinder the other parts of the system. Also, the proposed RNR-Map method is generalizable in various environments without additional fine-tuning.

3. RNR-Map

A robot agent has an RGB-D camera, and also knows its odometry information. Here, the odometry means how



(a) **Reconstruction Framework** (b) **Examples of the rendered images**

Figure 1. (a) **Illustration of the reconstruction framework.** Two neural networks, encoder θ_{enc} and decoder θ_{dec} are used in this reconstruction framework. (b) **Examples of the rendered images.** Odd columns are the given observations, and even columns are reconstructed results. The proposed method can reconstruct the images from the novel view (the last row).

much the agent has moved from its previous position and we consider 3-DoF pose in this paper. At time t, the robot observes an RGBD image I_t and its relative pose $\Delta p_t = (\Delta x_t, \Delta y_t, \Delta a_t)$ from the previous pose (xy position and heading angle). By cumulating pose differences, the agent can determine its relative pose p_t from the start pose $p_0 = (0, 0, 0)$.

A pair of the pretrained encoder and decoder is used when building a RNR-Map. The training process of these encoder and decoder resembles the autoencoder method. However, unlike 2D images, autoencoding a 3D environment is not a trivial problem. We build a reconstruction framework as shown in Figure 1a. The encoder encodes an image and embeds the pixel features into the RNR-Map. We denote each embedded feature in a grid of RNR-Map as a latent code. A query pose is then provided, and the decoder samples latent codes along each camera ray corresponding to each pixel and renders the corresponding images. We present details of each part in the following section.

Registration $\mathbf{F}_{\text{reg}}, \mathbf{F}_{\text{map}}$. When an RGBD image $I_t \in \mathbb{R}^{H\times W\times 4}$ comes in, the encoder encodes the image into a same-sized feature $C_t \in \mathbb{R}^{H\times W\times D}$, where H and W refers to height and width of the image, respectively, and D refers to the channel size. First, each pixel $c_{h,w} \in \mathbb{R}^D$ in C_t is mapped to its corresponding 3D world position $[q_x, q_y, q_z]^T$ using the known camera intrinsic $\mathbf{K}$, the extrinsic matrix $[\mathbf{R}|\mathbf{t}]$ based on the agent pose, and the depth information $d_{h,w}$ of the pixel. The world position of each pixel is calculated using inverse camera projection, as follows:

$$\begin{bmatrix} q_x(h,w) \\ q_y(h,w) \\ q_z(h,w) \end{bmatrix} = d_{h,w}\mathbf{R}^{-1}\mathbf{K}^{-1}\begin{bmatrix} h \\ w \\ 1 \end{bmatrix} - \mathbf{t}\ . \tag{1}$$

Then we digitize each position by normalizing with the map resolution s, and get map position (u, v) as shown in (2). We aggregate the pixel features that belong to the same 2D map position, and average the aggregated features into a single feature vector. The pixel features are registered in the corresponding grid in the RNR-Map $m \in \mathbb{R}^{U\times V\times D}$, where U and V refer to the height and width of the RNR-Map, respectively. The number of averaged features at each 2D map position is also recorded in mask $n \in \mathbb{R}^{U\times V}$. We denote the element of m at the map position (u, v) by $m(u, v)$ and the mask at (u, v) by $n(u, v)$. The registered feature $m(u, v) \in \mathbb{R}^D$ is the latent code which contains visual information of the region corresponding to the map position (u, v). This process can be written as follows:

$$X_{(u,v)} = \left\{ c_{h,w} \in C_t \middle| u = \left\lfloor \frac{q_x(h,w)}{s} \right\rceil, v = \left\lfloor \frac{q_y(h,w)}{s} \right\rceil \right\}$$
$$m(u,v) = \frac{1}{n(u,v)} \sum_{c_i \in X_{(u,v)}} c_i, \quad n(u,v) = |X_{(u,v)}|. \tag{2}$$

The registration process F_{reg} includes the encoding of C_t, inverse projection, and feature registration. The F_{reg} is represented as:

$$m_t^l, n_t^l = F_{\text{reg}}(I_t, p_t; \theta_{\text{enc}}), \tag{3}$$

where θ_{enc} refers to the network parameters of the encoder. The registration process F_{reg} outputs a local map m_t^l and a local mask n_t^l. The local map m_t^l only contains the information from the image I_t, and this will be integrated with other local maps to form the global map m^g.[1] When multiple observations are given, we can use n to compute the average value from the original and new latent codes. We name this integration process F_{map}, which operates F_{reg} over multiple observations. F_{map} at time t with previous m_{t-1}^g is formulated as follows:

$$(m_t^g, n_t^g) = F_{\text{map}}(I_t, p_t, m_{t-1}^g, n_{t-1}^g; \theta_{\text{enc}})$$
$$m_t^g(u,v) = \frac{m_t^l(u,v)\cdot n_t^l(u,v) + m_{t-1}^g(u,v)\cdot n_{t-1}^g(u,v)}{n_t^l(u,v) + n_{t-1}^g(u,v)}$$
$$n_t^g(u,v) = n_t^l(u,v) + n_{t-1}^g(u,v)\ . \tag{4}$$

[1] For simplicity, m without any superscript refers to the global map (m^g) in the rest of the paper.

Decoding $\mathbf{F}_{\text{dec}}$. To make these latent codes contain visual information, we reconstruct the image observation from the latent codes. We use a decoder which has a structure similar to the generative scene network (GSN) [13] for rendering an RGBD image from the 2D latent map. Originally, GSN generates a random indoor floorplan from random variables. Then GSN proposed how to render images from the generated floorplan, based on locally conditioned radiance fields. Our approach involves designing the encoder θ_{enc} and the registration process F_{reg} to transform image observations into latent codes, which can be converted to the locally conditioned radiance fields. We utilize the locally conditioned radiance fields from GSN to render an image from m. Given the camera parameters, we can sample latent codes on points along the camera ray, corresponding to the pixel location which will be rendered in the camera. The sampled latent codes are converted into modulation linear layer-based locally conditioned radiance fields [4, 13]. The decoder is trained to render an RGBD image from the latent codes to be close to the image observations I_t. The reader can refer to [13] for a more detailed explanation about the rendering procedure.

By training the rendered RGBD images $\tilde{I}_t$ to be close to the original observation I_t, the encoder is trained to extract the visual information from the image. This mechanism is similar to training an autoencoder, to extract and summarize useful information from an image into a latent vector. We sample N images in a random scene and embed them into the RNR-Map. Then, we render each image from the final RNR-Map and compare them with the original images. The encoder and the decoder are trained using the following loss:

$$
\begin{aligned}
m_i^g, n_i^g &= F_{\text{map}}(I_i, p_i, m_{i-1}^g, n_{i-1}^g; \theta_{\text{enc}}), \, i{=}1{:}N \\
\text{Loss}(\theta_{\text{enc}}, \theta_{\text{dec}}) &= \frac{1}{N} \sum_{i=1}^{N} ||I_i - F_{\text{dec}}(m_N^g, p_i; \theta_{\text{dec}})||_1,
\end{aligned}
\tag{5}
$$

where θ_{enc} and θ_{dec} are weight parameters of the encoder and decoder, respectively.

Since the rendering process is conditioned on the latent codes from the image observation, our proposed reconstruction framework is capable of embedding and rendering arbitrary images from unseen environments. This leads to the generalizable property of RNR-Map. Also, the decoder can synthesize a novel view, different from the observed direction, based on m. Examples of reconstructed observations are shown in Figure 1b. The decoder can render images from novel viewpoints in the observed region, based on the latent codes. Note that the rendering boundary is limited to the observed 3D points. We can see that the decoder generates grey color for the unobserved regions, in the last row of Figure 1b. More examples of reconstructed images are provided in the supplementary material A, as well as the network architectures of the encoder and decoder B.1.

Mapping. After training, we can use F_{map} for logging visual information from the environment. Note that the rendering function F_{dec} is not needed for mapping. The RNR-Map is built incrementally during navigation, based on the odometry information and the known camera intrinsics. At time t, the agent obtains m_t and n_t using F_{map}, as formulated in (4). If the same 3D point in the environment is observed multiple times, F_{map} averages the latent codes based on the number of observation instances.

4. Localization

One of the crucial components of the navigation task is localization. In the following sections, we describe how RNR-Map is used for two localization tasks: *image-based localization* and *camera tracking*. Here, we consider 3-DoF pose (x, y, a) for localization.

4.1. Image-Based Localization

The implicit visual information of the latent codes in RNR-Map can provide useful clues for finding out which grid cell is the closest to the given target image I_{trg}. Inspired by the fast localization method in [22], we directly compare latent codes from the target image I_{trg} and the RNR-Map m for localization. We denote this image-based localization function as F_{loc}. Suppose that the target image is taken at the pose $p_{\text{trg}} = (x_{\text{trg}}, y_{\text{trg}}, a_{\text{trg}})$, and the corresponding map position is $(u_{\text{trg}}, v_{\text{trg}})$. F_{loc} outputs a heatmap $E \in \mathbb{R}^{U \times V}$ and the predicted orientation of the target a_{trg}. E highlights which grid cell corresponds to the target location $(u_{\text{trg}}, v_{\text{trg}})$, among the observed area in m.

The localization process F_{loc} is shown in Figure 2. The RNR-Map m is constructed from a partial observation about the environment. The query image is transformed into m_{trg} using $F_{\text{reg}}(I_{\text{trg}}, p_0; \theta_{\text{enc}})$. Here, we use origin $p_0 = (0, 0, 0)$ as an input to F_{reg} since the agent does not know the position of the target location. F_{loc} includes three convolutional neural networks, F_k, F_q, and F_E. F_k and F_q are for proccessing m and m_{trg} into m'_k and m'_q, respectively. We found it helpful to introduce neural networks (F_k, F_q) for better localization results. Then, we search query m'_q by convolving (cross-correlation, denoted as Conv) with m'_k. The query map m'_q is rotated into R different angles $\{0°, ..., 360° \times \frac{R-1}{R}\}$. $(m'_q)_r$ denotes $Rot_r(m'_q)$, where Rot_r represents the r-th from the R possible rotations. Each rotated query $(m'_q)_r$ works as a filter, and the output from the Conv is forwarded to F_E. F_E outputs $E \in \mathbb{R}^{U \times V}$ which highlights the most query-related region in m. Each pixel $e_{u,v} \in E$ in the heatmap represents the probability of the query image being taken at (u, v). Also, F_E has an additional head which predict the orientation of the target observation.

The overall localization process can be put together as:

$$
F_{\text{loc}}(m, m_{\text{trg}}; \theta_{loc}) = F_E(\text{Conv}(F_k(m), \{F_q(m_{\text{trg}})_r\}_1^R)),
\tag{6}
$$

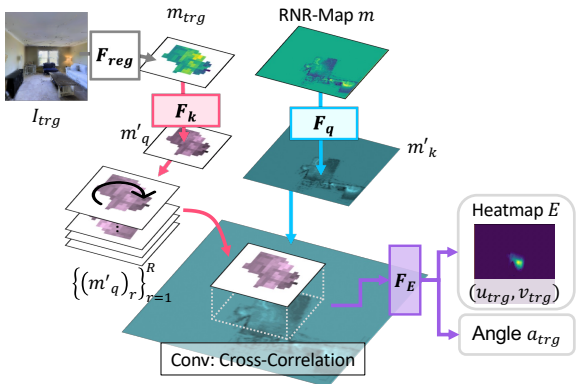


Figure 2. **Localization using $\mathbf{F}_{\mathrm{loc}}$.** A target observation can be localized by cross-correlation (Conv) between m and m_{trg}. Before the cross-correlation, each RNR-Map is forwarded to the CNN F_k and F_q. After Conv, F_E takes the output of the Conv and outputs a heatmap E which highlights the most plausible target area.

where θ_{loc} denotes the weight parameters of F_k, F_q, and F_E. Detailed explanations about the network architecture are provided in the supplementary material B.2.

We make a ground-truth Gaussian heatmap E_{gt} which highlights the ground-truth map location $(u_{\mathrm{trg}}, v_{\mathrm{trg}})$ of the query image. The representation of an orientation a_{trg} is discretized into 18 bins, and F_{E} is trained to select the appropriate bin using cross-entropy loss. The framework is trained to generate a distribution similar to the ground-truth heatmap E_{gt} and predicts the a_{trg}. The following loss term is used to train F_{loc}:

$$\begin{aligned}(\hat{E}, \hat{a}_{\mathrm{trg}}) &= F_{\mathrm{loc}}(m, m_{\mathrm{trg}}; \theta_{loc}) \\ \mathrm{Loss}(\theta_{\mathrm{loc}}) &= D_{KL}(E_{\mathrm{gt}}, \hat{E}) + CE(a_{\mathrm{trg}}, \hat{a}_{\mathrm{trg}}),\end{aligned} \tag{7}$$

where D_{KL} refers to KL divergence, and CE refers to cross-entropy loss. We provide quantitative and qualitative experimental results of F_{loc} in the supplementary material C.2.

4.2. Camera Tracking

During the navigation, the agent needs to be aware of its own pose to accurately record the observed visual observation. By cumulating odometry readings, the agent can calculate its relative pose to the start pose. However, in the real world, it is difficult to determine the accurate pose of a robot due to noises in odometry readings. The differential rendering function of F_{dec} can be used for adjusting the rough estimation of the agent pose p_t. The pose optimization is based on the photometric loss between the rendered image and the current observation. As the rendering process F_{dec} is differential, the pose of the agent p_t can be optimized with the gradient descent method. We name this camera tracking function as F_{track}. At time t, the agent has the previously estimated pose $\hat{p}_{t-1}$, and takes the odometry data Δp_t. The rough estimation of the current pose p_t can be calculated by simply adding Δp_t to the previous pose $\hat{p}_{t-1}$. $\bar{p}_t$ denotes such roughly estimated pose: $\bar{p}_t = \hat{p}_{t-1} + \Delta p_t$. Using

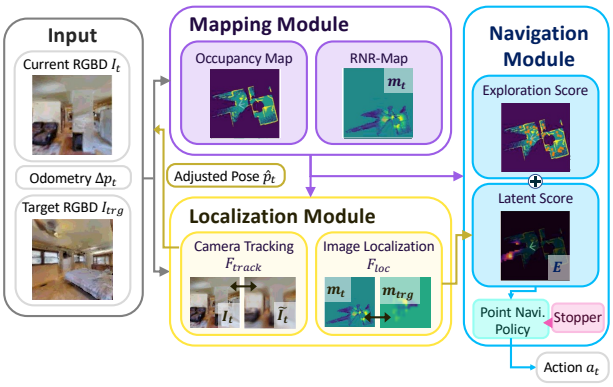


Figure 3. **Navigation System Overview.**

F_{track}, $\bar{p}_t$ is optimized to $\hat{p}_t$. The output of pose optimization can be derived by the following equation:

$$\hat{p}_t = F_{\mathrm{track}}(m_{t-1}, \bar{p}_t) = \arg\min_{\delta p_t} |F_{\mathrm{dec}}(m_{t-1}, \bar{p}_t + \delta p_t) - I_t|, \tag{8}$$

which minimizes the error between the current observation and the rendered image from the latent map. $\bar{p}_t$ is the initial value of the pose in the optimization process. By sampling a small subset of pixels from the image, we can make this optimization process fast enough to use it in navigation. We provide the accuracy and inference speed of the proposed camera tracking method in the supplementary material C.1.

5. Navigation

Now we describe how the RNR-Map and RNR-Map-based localization functions can be utilized in a navigation system. We consider the visual navigation task, especially image-goal navigation. The objective of image-goal navigation is to find the target location given the image taken from the target location. We build a visual navigation framework which includes F_{map} for mapping, F_{track} for localization, and F_{loc} for target searching. Figure 3 shows an overview of the proposed navigation system. The system consists of three modules, *mapping*, *localization*, and *navigation*. In the following section, we describe how each module works during navigation.

5.1. Mapping Module

The mapping module builds RNR-Map using the pre-trained encoder and F_{map}. At the start of the episode ($t = 0$), the mapping module transforms the target image I_{trg} into an RNR-Map m_{trg}. While maintaining m_{trg}, the module updates the current RNR-Map m_t using each image observation I_t with F_{map}. Also, the mapping module builds an occupancy map using depth information. This occupancy map is used for collision avoidance in a point navigation policy. The navigation module also uses this occupancy map to add more exploration property to the navigation system.

5.2. Localization Module

The localization framework described in Section 4 works as a localization module in the proposed navigation system. This localization module has two objectives during navigation. First, the localization module finds the most probable area which is similar to the target location. Second, considering a noisy odometry sensor, the localization module is needed to figure out the precise current pose of the agent. The image-based localization function F_{loc} and camera tracking function F_{track} are actively used in this module. With high probability, the target location may not be in the current RNR-Map m_t in the navigation scenario. Hence, F_{loc} for the navigation task is trained to find the region in m_t which is closer to the target location.

5.3. Navigation Module

The navigation module consists of three submodules: exploration, point navigation, and stopper.

Exploration. The objective of the exploration module is to select the most plausible region to explore. The module decides where to visit in order to search the target location, based on the probabilty heatmap E from F_{loc} in the localization module. We have adopted the concept from robot exploration [24, 31, 41, 43], which builds a generalized Voronoi graph on the occupancy map. We draw a Voronoi graph on the occupancy map and calculate visitation priority scores for each node of the created graph. Based on the scores, the exploration module selects the nodes to explore. Two types of scores are used for selecting exploration candidates, the latent score and the exploration score. The latent score is based on the heatmap E and represents how probable the node is near the target location. The exploration score of each node is simply calculated based on the values in the occupancy map. The occupancy map has three types of value: occupied, free and unseen. The exploration score of a node is proportional to the number of unseen pixels in the neighborhood of a node. The visitation priority of a node is determined based on the sum of the latent score and the exploration score.

Point navigation policy and Stopper. The point navigation policy is a simple occupancy-map-based path-following algorithm. When the exploration target node is selected, the point navigation module draws the shortest path to the target position. Following the path, the point navigation policy heuristically avoids obstacles using the occupancy map. The stopper module determines the arrival at the target location and calculates the relative pose from the target location. We employ a neural network F_{stop} which decides whether the agent is near the target location. This neural network is trained to output a binary value (1 if the target location is reached and 0, otherwise) based on the m_{trg} and m_t. For efficient target reaching, we adopted the recent last-mile navigation method [37] in stopper module. Based on keypoint matching, the relative pose between the target location and the current location is calculated using Perspective-n-Point [25] and RANSAC [18]. After F_{stop} detects the target location, the point navigation policy navigates to the target using the estimated relative pose. We provide detailed explanations about graph generation and exploration planning in the supplementary material G.

5.4. Implementation Details

The modules that require training are the encoder and decoder, and the neural networks used in F_{loc} and F_{stop}. We trained them using the same dataset. We have collected 200 random navigation trajectories from each scene in 72 Gibson [39] environments with the Habitat simulator [29]. The pair of encoder and decoder is first trained, then F_{loc} and F_{stop} are trained based on the pretrained encoder. Further implementation details (network architectures and training details) are provided in the supplementary material B.

6. Experiments

We have evaluated RNR-Map in both localization and navigation tasks. However, as the main objective of the proposed RNR-Map is visual navigation, we focus on analyzing the experiment results of the navigation task in the main manuscript. For localization tasks, we summarize the experimental results here and provide detailed information and analyses in the supplementary material C.

6.1. Localization

We have tested F_{track} and F_{loc} with other related baselines [22, 42]. In **camera tracking task**, the RNR-Map F_{track} shows high speed (5Hz) and accuracy (0.108m error) which are adequate for a real-time navigation system. More than localizing the current pose of the agent, the RNR-Map F_{loc} is able to locate the previously seen images (**image-based localization task**) with a high recall rate (inliers less than 50cm with 99%) in the recorded map. We can leverage this RNR-Map for searching the most similar place to the query image even if the exact place is not in the current environment. Two scenarios can be considered: (1) (Object Change) There have been large changes in the environment so that the object configuration of the environment is different from the information in the RNR-Map. (2) (Novel Environment) The user only has a query image from a different environment but wants to find the most similar place in the current environment. We have tested the RNR-Map in both scenarios and observed that F_{loc} robustly localizes query images even if some object configuration changes. When 33.9% of the observed images have changed, F_{loc} localizes the query image with less than 50cm error in 97.4% of cases, and 20° error in 97.5% of cases. Also, when given a query image from a different scene, the localized location achieves 94.5% of visual similarity compared to the best possible visual similarity in the given scene. The examples of the image-based localization tasks are shown in Figure 4. We can see that F_{loc} finds visually similar places based on

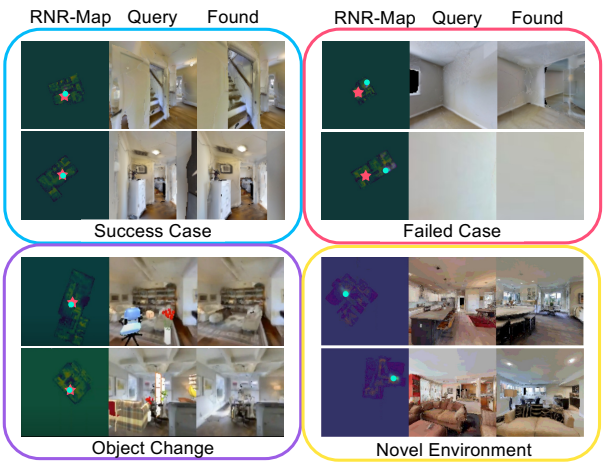


Figure 4. **Examples of image-based localization**. ★ location of the query image on RNR-Map, and ● is the location found by F_{loc}. More examples are provided in the supplementary material C.2 and C.3.

the RNR-Map, even when the environment has changed or given in a novel environment. Additionally, we found that the suggested method F_{loc} mislocates when there are multiple, similar locations in the environment or when the query has poor visual information. We provide more detailed experiments with baselines (MapNet [22], NICE-SLAM [42]) and examples in the supplementary material C.

6.2. Image-Goal Navigation

6.2.1 Baselines

We compare RNR-Map with learning-based agents using behavior cloning (**BC+RNN**) and reinforcement learning (**DDPPO** [38]). Also, to figure out the advantages of the RNR-Map over an occupancy map, we include the occupancy-map-based coverage method [10] as a baseline. The objective of this method is to visit all possible area in the given scene. We modified this method with the target distance prediction from [21], to make the agent reach the target when it is detected while exploring the environment (**ANS [10]+Pose Pred**). We also compare our method with the recent state-of-the-art image-goal navigation methods. **ZSEL** [2], and **OVRL** [40] are reinforcement learning-based methods which learn image-goal navigation task with specially designed rewards and the pretrained visual representation. **NRNS** builds a topological map and select the nodes to explore by predicting the distances between the target image and the node images with a distance prediction network. **SLING** is a last-mile navigation method which predict the relative pose of the target based on keypoint matching, after the target is detected. This method needs to be integrated with the exploratory algorithm such as NRNS or OVRL. Note that our method adopted this method, for efficient target reaching. The digits of the baseline DDPPO, OVRL, (NRNS,OVRL)+SLING are from [37] and their open-sourced code[2], and NRNS, ZSEL are from the original papers [21], [2], respectively.

6.2.2 Task setup

We have tested each method in the public image-goal navigation datasets from NRNS [21] and Gibson [39] with Habitat simulator [29]. Gibson dataset consists of 72 houses for training split, and 14 houses for the validation split. NRNS dataset consists of three difficulty levels (easy, medium, hard) with two path types (straight and curved). Each difficulty level has 1000 episodes for each path type, except for the hard-straight set (806). The objective of the image-goal navigation is to find the target place given the image of the target location. The agent only has access to the current RGBD observations and an odometry reading. We consider a noisy setting [10] where the odometry sensor and the robot actuation include noises as in the real world. The RGBD image observation comes from a directional camera with 90° of HFOV. A discrete action space is used in this paper with four types of actions: move forward $0.25m$. turn right 10°, turn left 10°, and stop. The maximum time step of each episode is set to 500. An episode is considered a success when the agent takes a stop action within $1m$ from the target location. Two evaluation metrics are used: success rate (SR) and success weighted by (normalized inverse) path length (SPL) [3], which represents the efficiency of a navigation path.

6.2.3 Image-goal navigation Results

RNR-Map helps efficient navigation. Table 1 shows the average SR and SPL of each method. We can see that the proposed navigation framework with the RNR-Map shows competitive or higher performance compared to the baselines, on image-goal navigation tasks. Many of the baselines (DDPPO, ZSEL, OVRL, OVRL+SLING) include reinforcement learning which is sample inefficient and computationally heavy, while having relatively simple representation about the environment. In contrast, the RNR-Map shows higher performances while only requiring an offline trajectory dataset for training neural networks. Based on this result, we argue that having a good internal representation of the environment and finding how to extract exploratory signals from such representation are important. An agent with an informative environmental representation can navigate well without the numerous inefficient interactions often required in reinforcement learning. Compared to baselines which have their own internal representation of the environment (NRNS, NRNS+SLING, ANS), our method shows a much higher performances in curved scenarios. From this result, we can infer that the RNR-Map indeed provides useful information for searching targets, more than coverage signals, and better than the existing methods. The ablation study shown in Table 2 also displays a similar trend. We ablated the main functions of the navigation framework F_{loc} and F_{track}, as well as noises. Without the latent score from

[2] https://github.com/Jbwasse2/SLING

	Straight								Curved							
	Easy		Medium		Hard		Overall		Easy		Medium		Hard		Overall	
	SR	SPL	SR	SPL	SR	SPL	SR	SPL	SR	SPL	SR	SPL	SR	SPL	SR	SPL
BC + RNN	39.4	27.9	25.7	15.9	13.3	9.0	26.1	17.6	26.4	12.6	20.3	10.5	8.4	4.8	18.4	9.3
DDPPO [38]	43.2	38.5	36.4	34.8	7.4	7.2	29.0	26.8	22.2	16.5	20.7	18.5	4.2	3.7	15.7	12.9
ANS + Target Pred [10]	68.8	55.1	54.0	30.3	42.4	22.9	55.1	36.1	48.0	21.0	46.0	20.5	31.3	14.6	41.8	18.7
NRNS [21]	64.1	55.4	47.9	39.5	25.2	18.1	45.7	37.7	27.3	10.6	23.1	10.4	10.5	5.6	20.3	8.8
ZSEL [2]	-	-	-	-	-	-	-	-	41.0	28.2	27.3	18.6	9.3	6.0	25.9	17.6
OVRL [40]	53.6	34.7	48.6	33.3	32.5	21.9	44.9	30.0	53.6	31.8	47.6	30.2	35.6	22.0	45.6	28.0
NRNS + SLING [37]	**85.3**	**74.4**	66.8	**49.3**	41.1	28.8	64.4	**50.8**	58.6	16.1	47.6	16.8	24.9	10.1	43.7	14.3
OVRL + SLING [37]	71.2	54.1	60.3	44.4	43.0	29.1	58.2	42.5	68.4	47.0	57.7	39.8	40.2	25.5	55.4	37.4
RNR-Map (ours)	76.4	55.3	**73.6**	46.1	**54.6**	**30.2**	**68.2**	43.9	**75.3**	**52.5**	**70.9**	**42.3**	**51.0**	**27.4**	**65.7**	**40.8**

Table 1. **Image-goal Navigation Result**. SR: Success Rate. SPL: Success weighted by Path Length.

			Easy		Medium		Hard		Overall	
Noise	F_{loc}	F_{track}	SR	SPL	SR	SPL	SR	SPL	SR	SPL
✘	✘	-	60.3	41.0	57.7	36.8	39.9	23.4	52.6	33.7
✘	✔	-	72.8	61.1	67.7	48.9	48.8	31.1	63.1	47.0
✔	✔	✘	74.6	53.6	64.0	40.9	40.8	23.6	59.8	39.4
✔	✔	✔	75.3	53.9	72.2	44.2	52.8	28.9	66.9	42.3

Table 2. **Ablation study**. The values are the average of the straight and curved scenarios.

Mapping F_{reg}	Tracking F_{track}	Localization F_{loc}	Rendering F_{dec}
91.9 Hz (10.9 ms)	5.0 Hz (200 ms)	56.8 Hz (17.6 ms)	14.7 Hz (68.0 ms)

Table 3. **Runtime analysis of RNR-Map.**

F_{loc}, the success rate and SPL dramatically drop. We also have tested the proposed method in MP3D [8] dataset, and observed similar results with Gibson dataset. We provide the results in the supplementary material E, and additional ablation study about the submodules of the navigation module is also provided in D.

F_{track} makes RNR-Map robust to noise. Comparing the third row and the last row of Table 2, we can infer that the pose adjusting function F_{track} helps the agent to find the image goal more effectively, even with the noisy odometry. The proposed method shows higher success rates in noisy settings, while SPL values are low. We believe this is from the randomness of noises, which helps the local navigation policy to escape from being stuck in a single location.

RNR-Map is real-time capable. We analyzed the runtime for each feature of RNR-Map and report them in Table 3.

The runtimes are measured on a desktop PC with a Intel i7-9700KF CPU @ 3.60GHz, and an NVIDIA GeForce RTX 2080 Ti GPU. We can see that each function of RNR-Map operates fast enough for real-time navigation, even when including NeRF-based rendering.

Navigation example. Figure 5 provides an example of an image-goal navigation episode. F_{loc} highlights two types of places, a place that looks similar to the target image and an explorable place like the end of the aisle, or doorway. We can see that the highlighted area at the start of the episode has a similar observation to the target image. First, the agent navigates to a similar-looking place but decided not to stop at the location. While navigating to another place that shows a high latent score, the agent has found the target location, and the latent score also changed to highlight the target. Additional qualitative results of navigation episodes are provided in the supplementary material F and video.

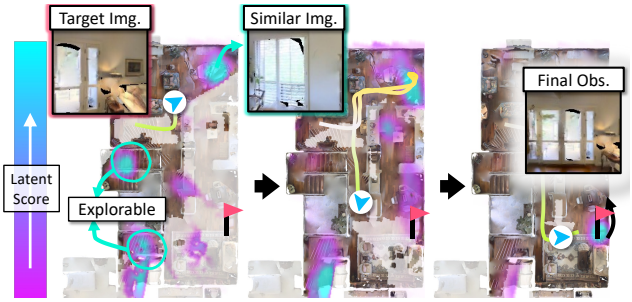


Figure 5. **Example of image-goal navigation episode.** The heatmap value of E from F_{loc} is highlighted on the map according to the colorbar on the left.

7. Conclusion

In this paper, we have proposed RNR-Map for visual navigation, which captures visual information about the environment. The RNR-Map is helpful for visual navigation in two ways: (1) The latent codes in the RNR-Map can provide rich signals to find the target location, given a query image. (2) The rendering property of the RNR-Map helps the agent to accurately estimate its pose based on a photometric error, in an unseen environment. The proposed method has outperformed other methods in image-based localization. Also, we have found that the image-based localization of RNR-Map is robust to environmental changes. In image-goal navigation tasks, the proposed method outperforms the current state-of-the-art image-goal navigation methods. Furthermore, the fast inference time of the RNR-Map shows its potential for real-world applications. However, the proposed method still has limitations. Once the observation images are embedded in grids, RNR-Map is hard to correct the past odometry error. We can consider applying loop closure using the proposed localization framework. Inspired by existing graph-based SLAM methods, a pose graph with local RNR-Maps can be leveraged to optimize poses, leading to consensus in the pixel renderings from the global RNR-Map.

References

[1] Michal Adamkiewicz, Timothy Chen, Adam Caccavale, Rachel Gardner, Preston Culbertson, Jeannette Bohg, and Mac Schwager. Vision-Only Robot Navigation in a Neural Radiance World. *IEEE Robotics and Automation Letters*, 7(2):4606–4613, 2022. 1, 2

[2] Ziad Al-Halah, Santhosh K. Ramakrishnan, and Kristen Grauman. Zero Experience Required: Plug & Play Modular Transfer Learning for Semantic Visual Navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 7, 8

[3] Peter Anderson, Angel X. Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir Roshan Zamir. On Evaluation of Embodied Navigation Agents. *arXiv preprint arXiv:1807.06757*, 2018. 7

[4] Ivan Anokhin, Kirill Demochkin, Taras Khakhulin, Gleb Sterkin, Victor Lempitsky, and Denis Korzhenkov. Image Generators with Conditionally-Independent Pixel Synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 4

[5] Valts Blukis, Taeyeop Lee, Jonathan Tremblay, Bowen Wen, In So Kweon, Kuk-Jin Yoon, Dieter Fox, and Stan Birchfield. Neural Fields for Robotic Object Manipulation from a Single Image. *arXiv preprint arXiv:2210.12126*, 2022. 2

[6] Arunkumar Byravan, Jan Humplik, Leonard Hasenclever, Arthur Brussee, Francesco Nori, Tuomas Haarnoja, Ben Moran, Steven Bohez, Fereshteh Sadeghi, Bojan Vujatovic, et al. NeRF2Real: Sim2real Transfer of Vision-guided Bipedal Motion Skills using Neural Radiance Fields. *arXiv preprint arXiv:2210.04932*, 2022. 2

[7] Vincent Cartillier, Zhile Ren, Neha Jain, Stefan Lee, Irfan Essa, and Dhruv Batra. Semantic mapnet: Building allocentric semanticmaps and representations from egocentric views. *arXiv preprint arXiv:2010.01191*, 2020. 2

[8] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D Data in Indoor Environments. *International Conference on 3D Vision (3DV)*, 2017. 8

[9] Devendra Singh Chaplot, Dhiraj Gandhi, Abhinav Gupta, and Ruslan Salakhutdinov. Object Goal Navigation using Goal-Oriented Semantic Exploration. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 1, 2

[10] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning To Explore Using Active Neural SLAM. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020. 1, 2, 7, 8

[11] Tao Chen, Saurabh Gupta, and Abhinav Gupta. Learning Exploration Policies for Navigation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019. 1, 2

[12] Qiyu Dai, Yan Zhu, Yiran Geng, Ciyu Ruan, Jiazhao Zhang, and He Wang. GraspNeRF: Multiview-based 6-DoF Grasp Detection for Transparent and Specular Objects Using Generalizable NeRF. *arXiv preprint arXiv:2210.06575*, 2022. 2

[13] Terrance DeVries, Miguel Angel Bautista, Nitish Srivastava, Graham W. Taylor, and Joshua M. Susskind. Unconstrained scene generation with locally conditioned radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 4

[14] Danny Driess, Zhiao Huang, Yunzhu Li, Russ Tedrake, and Marc Toussaint. Learning Multi-Object Dynamics with Compositional Neural Radiance Fields. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2022. 2

[15] Danny Driess, Ingmar Schubert, Pete Florence, Yunzhu Li, and Marc Toussaint. Reinforcement learning with neural radiance fields. *arXiv preprint arXiv:2206.01634*, 2022. 1

[16] Jiafei Duan, Samson Yu, Hui Li Tan, Hongyuan Zhu, and Cheston Tan. A Survey of Embodied AI: From Simulators to Research Tasks. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2022. 1, 2

[17] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989. 1, 2

[18] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 6

[19] Georgios Georgakis, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, and Kostas Daniilidis. Learning to Map for Active Semantic Goal Navigation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022. 1, 2

[20] Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive Mapping and Planning for Visual Navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[21] Meera Hahn, Devendra Chaplot, Shubham Tulsiani, Mustafa Mukadam, James Rehg, and Abhinav Gupta. No RL, No Simulation: Learning to Navigate without Navigating. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 1, 2, 7, 8

[22] João F. Henriques and Andrea Vedaldi. MapNet: An Allocentric Spatial Memory for Mapping Environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 4, 6, 7

[23] Jeffrey Ichnowski*, Yahav Avigal*, Justin Kerr, and Ken Goldberg. Dex-NeRF: Using a neural radiance field to grasp transparent objects. In *Conference on Robot Learning (CoRL)*, 2020. 2

[24] Jonghoek Kim, Fumin Zhang, and Magnus Egerstedt. A provably complete exploration strategy by constructing voronoi diagrams. *Autonomous Robots*, 29(3):367–380, 2010. 6

[25] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate O(n) solution to the PnP problem. *International Journal of Computer Vision*, 81(2):155–166, 2009. 6

[26] Yunzhu Li, Shuang Li, Vincent Sitzmann, Pulkit Agrawal, and Antonio Torralba. 3D Neural Scene Representations for Visuomotor Control. In *Conference on Robot Learning*, 2021. 2

[27] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. BARF: Bundle-Adjusting Neural Radiance Fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 2

[28] Dominic Maggio, Marcus Abate, Jingnan Shi, Courtney Mario, and Luca Carlone. Loc-NeRF: Monte Carlo Localization using Neural Radiance Fields. *arXiv preprint arXiv:2209.09050*, 2022. 1, 2

[29] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 6, 7

[30] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 2

[31] Keiji Nagatani and Howie Choset. Toward robust sensor based exploration by constructing reduced generalized voronoi graph. In *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No. 99CH36289)*, volume 3, pages 1687–1692. IEEE, 1999. 6

[32] Emilio Parisotto and Ruslan Salakhutdinov. Neural Map: Structured Memory for Deep Reinforcement Learning. In *International Conference on Learning Representations*, 2018. 2

[33] Benjamin Planche, Xuejian Rong, Ziyan Wu, Srikrishna Karanam, Harald Kosch, YingLi Tian, Jan Ernst, and Hutter Andreas. Incremental Scene Synthesis. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 2

[34] Ziad Al-Halah Santhosh Kumar Ramakrishnan and Kristen Grauman. Occupancy Anticipation for Efficient Exploration and Navigation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 2

[35] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew Davison. iMAP: Implicit Mapping and Positioning in Real-Time. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 1, 2

[36] Saim Wani, Shivansh Patel, Unnat Jain, Angel X. Chang, and Manolis Savva. MultiON: Benchmarking Semantic Map Memory using Multi-Object Navigation. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 1, 2

[37] Justin Wasserman, Karmesh Yadav, Girish Chowdhary, Abhinav Gupta, and Unnat Jain. Last-Mile Embodied Visual Navigation. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2022. 2, 6, 7, 8

[38] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. DD-PPO: Learning near-perfect pointgoal navigators from 2.5 billion frames. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020. 7, 8

[39] Fei Xia, Amir R. Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson Env: Real-World Perception for Embodied Agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 6, 7

[40] Karmesh Yadav, Ram Ramrakhya, Arjun Majumdar, Vincent-Pierre Berges, Sachit Kuhar, Dhruv Batra, Alexei Baevski, and Oleksandr Maksymets. Offline Visual Representation Learning for Embodied Navigation. *arXiv preprint arXiv:2204.13226*, 2022. 7, 8

[41] Qiwen Zhang, David Whitney, Florian Shkurti, and Ioannis Rekleitis. Ear-based exploration on hybrid metric/topological maps. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014. 6

[42] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. NICE-SLAM: Neural Implicit Scalable Encoding for SLAM. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2, 6, 7

[43] Xinkai Zuo, Fan Yang, Yifan Liang, Zhou Gang, Fei Su, Haihong Zhu, and Lin Li. An Improved Autonomous Exploration Framework for Indoor Mobile Robotics Using Reduced Approximated Generalized Voronoi Graphs. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 1:351–359, 2020. 6